DATA-DRIVEN MATHEMATICAL AND STATISTICAL MODELS OF ONLINE SOCIAL NETWORKS

EDITED BY: Shudong Li, Zhen Wang, Chengyi Xia, Lin Wang and Daihai He
PUBLISHED IN: Frontiers in Physics

frontiers Research Topics

ISSN 1664-8714
ISBN 978-2-88974-596-8
DOI 10.3389/978-2-88974-596-8

About Frontiers

Frontiers is more than just an open-access publisher of scholarly articles: it is a pioneering approach to the world of academia, radically improving the way scholarly research is managed. The grand vision of Frontiers is a world where all people have an equal opportunity to seek, share and generate knowledge. Frontiers provides immediate and permanent online open access to all its publications, but this alone is not enough to realize our grand goals.

Frontiers Journal Series

The Frontiers Journal Series is a multi-tier and interdisciplinary set of open-access, online journals, promising a paradigm shift from the current review, selection and dissemination processes in academic publishing. All Frontiers journals are driven by researchers for researchers; therefore, they constitute a service to the scholarly community. At the same time, the Frontiers Journal Series operates on a revolutionary invention, the tiered publishing system, initially addressing specific communities of scholars, and gradually climbing up to broader public understanding, thus serving the interests of the lay society, too.

Dedication to Quality

Each Frontiers article is a landmark of the highest quality, thanks to genuinely collaborative interactions between authors and review editors, who include some of the world's best academicians. Research must be certified by peers before entering a stream of knowledge that may eventually reach the public - and shape society; therefore, Frontiers only applies the most rigorous and unbiased reviews.
Frontiers revolutionizes research publishing by freely delivering the most outstanding research, evaluated with no bias from both the academic and social point of view. By applying the most advanced information technologies, Frontiers is catapulting scholarly publishing into a new generation.

What are Frontiers Research Topics?

Frontiers Research Topics are very popular trademarks of the Frontiers Journals Series: they are collections of at least ten articles, all centered on a particular subject. With their unique mix of varied contributions from Original Research to Review Articles, Frontiers Research Topics unify the most influential researchers, the latest key findings and historical advances in a hot research area! Find out more on how to host your own Frontiers Research Topic or contribute to one as an author by contacting the Frontiers Editorial Office: frontiersin.org/about/contact

DATA-DRIVEN MATHEMATICAL AND STATISTICAL MODELS OF ONLINE SOCIAL NETWORKS

Topic Editors:
Shudong Li, Guangzhou University, China
Zhen Wang, Northwestern Polytechnical University, China
Chengyi Xia, Tianjin University of Technology, China
Lin Wang, University of Cambridge, United Kingdom
Daihai He, Hong Kong Polytechnic University, SAR China

Citation: Li, S., Wang, Z., Xia, C., Wang, L., He, D., eds (2022). Data-Driven Mathematical and Statistical Models of Online Social Networks. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-88974-596-8

Table of Contents

frontiers in Physics

ORIGINAL RESEARCH
published: 04 August 2021
doi: 10.3389/fphy.2021.715465

Audio Information Camouflage Detection for Social Networks

Jiu Lou, Zhongliang Xu, Decheng Zuo, Zhan Zhang and Lin Ye*

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

Sending camouflaged audio information for fraud in social networks has become a new means of social networks attack. The hidden acoustic events in the audio scene play an important role in the detection of camouflaged audio information. Therefore, the application of machine learning methods to represent hidden information in audio streams has become a hot issue in the field of network security detection. This study proposes a heuristic mask for empirical mode decomposition (HM-EMD) method for extracting hidden features from audio streams. The method consists of two parts: First, it constructs heuristic mask signals related to the signal's structure to solve the modal mixing problem in intrinsic mode function (IMF) and obtains a pure IMF related to the signal's structure. Second, a series of hidden features in environment-oriented audio streams is constructed on the basis of the IMF. A machine learning method and hidden information features are subsequently used for audio stream scene classification. Experimental results show that the hidden information features of audio streams based on HM-EMD are better than the classical mel cepstrum coefficients (MFCC) under different classifiers. Moreover, the classification accuracy achieved with HM-EMD increases by 17.4 percentage points under the three-layer perceptron and by 1.3% under the depth model of TridentResNet. The hidden information features extracted by HM-EMD from audio streams revealed that the proposed method could effectively detect camouflaged audio information in social networks, which provides a new research idea for improving the security of social networks.

Keywords: social networks, machine learning, audio information camouflage, audio scene classification, emd

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Chengyi Xia,
Tianjin University of Technology, China
Longfei Wu,
Fayetteville State University, United States
Chao Gao,
Southwest University, China

****Correspondence:***
Decheng Zuo
zuodc@hit.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *27 May 2021*
Accepted: *20 July 2021*
Published: *04 August 2021*

Citation:
Lou J, Xu Z, Zuo D, Zhang Z and Ye L (2021) Audio Information Camouflage Detection for Social Networks. Front. Phys. 9:715465. doi: 10.3389/fphy.2021.715465

INTRODUCTION

Getting hot topics through social networks [1] and sharing news based on communities [2] have become the life style of modern people. Especially with the rise of we media technology in recent years, audio information has gradually become one of the main forms of information exchange in social networks. But it also brings a lot of security risks [3]. Using speech synthesis, interception audio stream for reediting and other methods to generate camouflage audio for fraud has become a new means of social networks attack. As a result of the rapid development of modern speech processing technology, people can easily edit "false" audio that cannot be easily distinguished by hearing, which makes it more difficult for people to distinguish the true and false audio in social networks. Therefore, the application of machine learning methods to mine hidden information in acoustic signals to identify authenticity and monitor risks [4, 5] has become a research hotspot in the field of acoustic signal processing. The special complex noise and sudden acoustic events in ambient background sounds are some of the important factors to judge audio authenticity. Furthermore, the frequency aliasing caused by complex noise and abrupt frequency changes caused by sudden acoustic

events increases the difficulty of classifying environmental audio streams. Therefore, this paper proposes an audio environment hidden information feature extraction method based on HM-EMD and aims to use the key factors contained in the special environment information to detect the authenticity of audio and solve the problem of audio fraud in social networks.

In the current deep learning framework, audio environment hidden information feature extraction methods are mainly divided into two categories: traditional feature representation [6–8] and automatic learning audio feature representation based on deep network [9, 10]. MFCC [11], spectrograms, acoustic event histograms [12], and gradient histograms based on time–frequency learning [13] are the most commonly used traditional methods for acoustic feature representation. Acoustic event classification and detection is generally based on the spectral features of the MFCC [14]. The first team proposed a deep network for the TridentResNet series and used the snapshot method to filter the network classification results [15]. In addition to the aforementioned classical feature representation methods, deep neural networks (DNNs) can automatically learn audio features. The typical network models include the time-delay neural network [16], VGGISH-based embedding model [17], and the mixed feature extraction model based on DNN and convolutional neural network proposed by [18]. However, this end-to-end feature was not used by the first 30 teams in the DCASE tournament. This is because the end-to-end feature learning approach using deep networks directly requires a large number of evenly distributed data sets; however, in the real scene, the environmental sound source is complex, and the occurrence time and frequency of hidden acoustic events in environmental audio streams are not fixed and are often random and unpredictable. These scenarios enhance the mutagenicity and nonstationary properties of environmental acoustic signals and indirectly lead to the uneven distribution of various hidden acoustic events in datasets [19]. The result of the MFCC processing of acoustic signals is the mel-frequency. The mel-filter is designed according to the sensitivity of the human ear to frequency. Therefore, the MFCC achieves good results in speech and speaker recognition. However, in a nonspeech environment, ambient sound signals are nonstationary and hidden, and the sequence features of the MFCC lose a large amount of high-frequency hidden information and hidden information outside the hearing threshold range. Therefore, to improve the accuracy of hidden information mining in environmental audio streams, it is necessary to establish a more accurate time–frequency feature representation system that can further locate and analyze hidden acoustic events and ultimately improve the classification accuracy of environmental audio streams.

Environmental audio streams are typical nonlinear and nonstationary time-varying signals. Thus, they require time-varying filtering and decomposition technologies. Proposed by Norden E. Huang in 1998, empirical mode decomposition (EMD) is a signal processing method suitable for nonlinear unsteady time-varying signals [20].

However, the traditional EMD method has a few disadvantages, including mode aliasing and the inconsistency of IMF dimensions after signal decomposition. These drawbacks limit the application of EMD to acoustic signal processing. Modal aliasing causes the frequency distribution of some IMFs after signal decomposition to overlap. Hence, accurately estimating the IMF range of a certain frequency distribution is difficult. Dimension inconsistency may cause variations in the number of IMFs obtained from the decomposition of source signals with the same frame length; these variations will lead to the mismatch of the required eigenvector dimensions and hinder the subsequent signal analysis and processing [21]. In 2005, R. Deering and J. E. Kaiser proposed the ensemble empirical mode decomposition (EEMD) decision method [22], which attempts to solve the problem of mode aliasing by introducing Gaussian white noise into the signal to be decomposed. In EEMD, the attributes of Gaussian white noise should be adjusted artificially. However, the Gaussian white noise leaves traces in the IMF decomposed from the signal, thereby resulting in low signal restoration accuracy and extensive calculations. Time-varying filtering-based empirical mode decomposition (TVF-EMD) uses the b-spline time-varying filter for mode selection and thus solves the problem of mode aliasing to a certain extent. However, TVF-EMD must calculate the cutoff frequency first, thus leaving the problem of dimension inconsistency unsolved [23].

By taking advantage of the time–frequency analysis of EMD, the problems of modal aliasing and frequency inconsistency in EMD must be resolved. Therefore, the current study consists of two parts. First, based on the traditional EMD, an improved heuristic empirical mode decomposition (HM-EMD) method is proposed. This method improves the purity of IMFs by adopting adaptive mask signals. With this method, the frequency domain distribution and IMF dimension can be stabilized and the inconsistency of the IMF dimension can be improved. The mask signals introduced in EMD can be obtained through heuristic learning and provide technical support for the feature extraction of hidden acoustic signals. On the basis of the mask signals, the hidden audio component features (HACFs) for audio stream recognition are constructed. According to the classification dataset Task-A [19] of the environmental audio stream in DCASE, hidden acoustic events, such as 'birdsong' and 'footsteps' in environmental audio streams, can be located and analyzed. The analysis results can be applied for multiple levels and multiple time scales of environment safety certification in audio streams. They can also be applied to other complex acoustic analyses and processing.

This work is divided into five parts. The second part mainly introduces the principle of HM-EMD. The signal processing flow and existing problems of the classical EMD algorithm are analyzed, and the principle of the HM-EMD algorithm is presented in detail. The third part describes the mining of hidden information in audio streams on the basis of the proposed HM-EMD. Specifically, environmental audio stream data are analysed, and the HACFs for hidden information in audio streams are designed according to the analysis results. The fourth part presents the classification of audio streams on the basis of HM-EMD. The experimental dataset is obtained from the low-complexity acoustic scene

classification task provided by DCASE in 2020. The experimental results show that the proposed method can accurately extract and locate hidden acoustic events, thus improving the accuracy of audio stream classification. The fifth part summarises the characteristics of the proposed method and presents future research directions.

HEURISTIC MASK FOR EMPIRICAL MODE DECOMPOSITION

Empirical Mode Decomposition Method

Empirical Mode Decomposition

EMD can decompose the original signal x(t) ($t \in N, N = \{0, 1, ...n\}$) into a series of IMFs whose upper and lower envelopes have a mean value of 0. This decomposition method does not need to preset basis functions (such as Fourier transform or wavelet analysis), but the IMFs should satisfy the following formulas:

$$|Num_{extream} - Num_{cross}| \leq 1 \tag{1}$$

$$\sum\nolimits_{t\in N} B_{\max}(t) + \sum\nolimits_{t\in N} B_{\min}(t) = 0 \tag{2}$$

where $Num_{extream}$ is the number of extreme points of the data sequence and Num_{cross} is the number of zero crossings; $B_{\max}(t)$, $B_{\min}(t)$ are the upper and lower envelopes by cubic spline interpolation with the maximum and minimum points as the control points, respectively. **Equation 1** represents the narrow-band constraint condition of the IMF, and **Eq. 2** represents the local symmetry constraint condition. Algorithm description in **Algorithm 1**.

Modal Aliasing of EMD

The most significant disadvantage of EMD is mode aliasing. In mode aliasing, a single IMF contains signals of different frequencies or signals of the same frequency that appear in different IMF components. The typical mode aliasing phenomena are described as follows:

1) For multiple single-frequency signals, a mixed signal is an amplitude modulation (AM)–frequency modulation (FM) signal if the energy levels of the source signals are similar. When the frequency ratio $is \epsilon [0.5, 2]$, the FM signal and AM signal overlap and the amplitude between the extreme values changes excessively. In such a case, the ordinary cubic spline function cannot easily and accurately fit any signal, resulting in the loss of local scale. This condition also leads to the mixing of multiple frequency domains in the IMF composition, such as signal $x_1(t) = sin2\pi{*}2.4t + \sin2\pi{*}3.5t + \sin2\pi{*}7t$, the frequency ratio between two mixed signals is 1.45,2 and 2.91. There are two frequency ratio $is \epsilon [0.5, 2]$. The EMD Results for $x_1(t)$ is shown in **Figure 1**. **Figure 1A** shows the IMFs of signal, which the corresponding FFT transform spectrum shows in **Figure 1B**. Mode aliasing can be seen in the FFT spectrum IMF1 and IMF2 in **Figure 1B** and abnormal mutation of the instantaneous frequency occurred in IMFs in **Figure 1C**.

2) Several different frequency signals are superimposed at different times, and the maximum value of the nonaliased part is missing, resulting in modal aliasing, such as EMD Results for signal $x_2(t) = u(t, 0, 2){*}sin2\pi{*}2.4t + sin2\pi{*}6t + u(t, 8, 10){*}sin2\pi{*}15t$ shown in **Figure 2** where

$$ut, t_1 t_2 = \begin{cases} 1 & t \in \{t_1, t_2\} \\ 0 & others \end{cases}.$$

The frequency ratios in $x_2(t)$ are all out of [0.5, 2], but there sin (2π*2.4t) start at time 2 s and the sin (2π*15t) end at time 8 s which lead to the mode aliasing as shown in **Figure 2B**. The abnormal mutation of the instantaneous frequency occurred in IMFs in **Figure 2C**.

3) Mode aliasing also occurs when the distribution of extreme points in a window is not uniform even if Eqs **1**, **2** are satisfied, such as signal $x_3(t) = ut, 0, 2{*}sin2\pi{*}\ 2.4t + sin2\pi{*}3.5t + ut, 8, 10{*}sin2\pi{*}7t$. The model aliasing can be seen in FFT spectrum of IMFS in **Figure 3B**. The negative frequency in **Figure 3C** is caused by the large fluctuation of the IMF amplitude.

When the frequency interval between multiple signals is too small or there is noise, the local extremum will jump many times in a very short time interval. The local extremum as control points for cubic b-spline interpolation in the process of EMD. The cubic b-spline interpolation resulting in the spectrum envelope will be fluctuated if the extreme value loss or extreme value distribution is inconsistent. This condition can adversely affect the spectral envelope. At this time, the time-domain signal does not meet the narrow-band requirements of IMF decomposition, resulting in mode aliasing. Therefore, the absence of extremum is an important cause of mode aliasing in EMD calculation. The different causes of the lack of extreme values require different processing methods. The causes of missing extreme values can be divided into two categories. One is the uneven distribution of extreme values in the analysis window caused by signal concealment at a certain time (**Figure 2**). The key to dealing with this type of modal aliasing is to

ALGORITHM 1 | EMD decomposition to obtain an IMF

Empirical Mode Decomposition
Input: Original signal x(t), Supposed IMF number i
output: Intrinsic Mode Functions, IMF
1 i = 1, $x^1(t)$= x(t)
2 Get the extremum points $\{u_1^{max}, u_1^{min}, u_2^{max}, ...\}$ of signal $x^i(t)$, calculate the upper and lower envelope $B_{\max}(t), B_{\min}(t)$ by cubic spline interpolation with the maximum and minimum points as control points, get the average value of upper and lower envelope $B_{mean}(t)$ at every points;
3 $r(t) = x^i(t) - B_{mean}(t)$. if $r(t)$ satisfies **Eqs. 1**, **2**, then $r(t)$ is taken as the i th IMF signal $r_{IMF}^i(t)$, i = i+1; if not, repeat step 2 and 3 for signal $r(t)$.
4 $x^i(t) = x^{i-1}(t) - r_{IMF}^{i-1}(t)$, return to step 1 until the termination condition is satisfied;

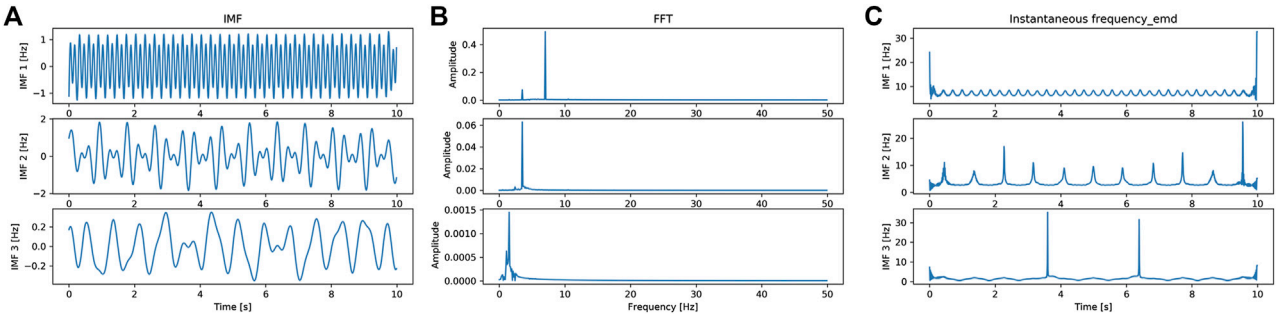


FIGURE 1 | EMD results for signal $x_1(t)$, **(A)** IMFs of $x_1(t)$; **(B)** FFT spectrum of each corresponding IMF; **(C)** Instantaneous frequency of each IMF.

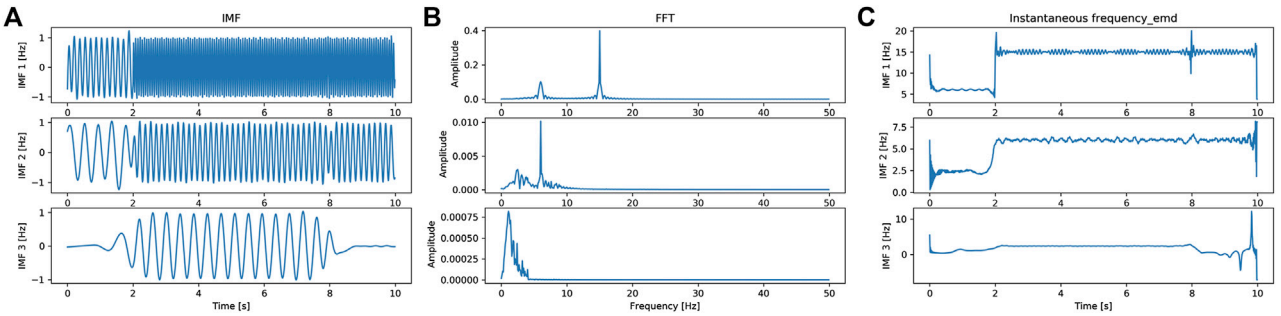


FIGURE 2 | EMD results for signal $x_2(t)$, **(A)** IMFs of $x_2(t)$; **(B)** FFT spectrum of each corresponding IMF; **(C)** Instantaneous frequency of each IMF.

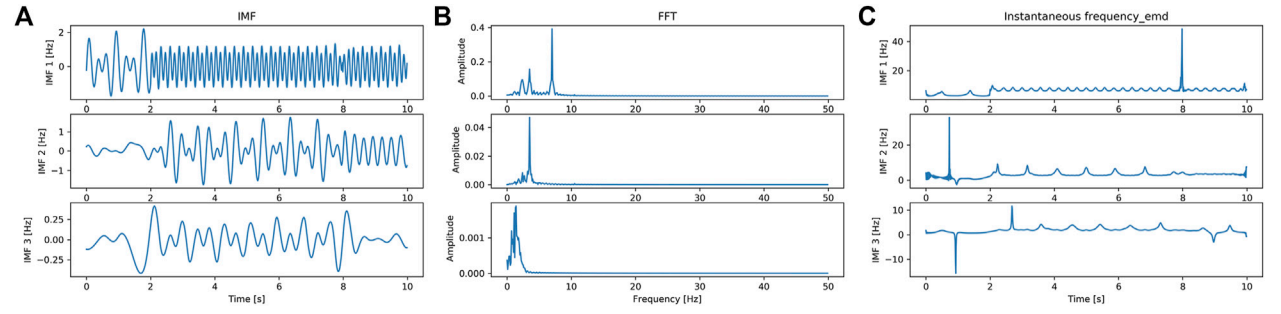


FIGURE 3 | EMD results for signal $x_3(t)$, **(A)** IMFs of $x_3(t)$; **(B)** FFT spectrum of each corresponding IMF; **(C)** Instantaneous frequency of each IMF.

determine the time when the signal concealment occurs. If the analysis is conducted according to the time point of concealment, then aliasing will not occur. The other category involves signal spectrum aliasing, which can be addressed by adding mask signals; that is, by creating a mask signal s (T), we can derive the following:

$$x_+(t) = xt + st \tag{3}$$

$$x_-(t) = xt - st \tag{4}$$

For $x_-(t)$ and $x_+(t)$, EMD is performed to obtain the natural mode functions $r_{IMF-}(t)$ and $r_{IMF+}(t)$, respectively. The final IMF is defined as follows:

$$r_{IMF}(t) = \frac{r_{IMF+}(t) + r_{IMF-}(t)}{2} \tag{5}$$

However, signal mode aliasing in practical applications cannot be attributed to a single factor (**Figure 3**). It usually includes hiding and spectrum aliasing. Therefore, it can be considered to determine the time of signal concealment. Then, a mask signal is introduced to the period of concealment to perform mode decomposition. Therefore, the current work proposes the HM-EMD method. This method maximises the use of the intrinsic properties of signals to construct variable analysis windows and mask signals that can adapt to a variety of signal

contents. The principle and implementation process are described herein.

Heuristic Mask Signals

Basic Principle Analysis

The signal properties need to be established prior to EMD. A time-varying FM/AM model can be used to express any nonstationary signal; that is,

$$x(t) = Atsin(\omega(t)) \tag{6}$$

where a (T) is the envelope function and ω (T) is the phase function. The analytical signal is

$$z(t) = \mathrm{x}t + jH[x(t)] \tag{7}$$

Here, $H[\cdot]$ denotes the Hilbert transform. We calculate the instantaneous phase $\omega(t) = arctan\frac{H[x(t)]}{x(t)}$ and instantaneous frequency $f_{IF}t = \frac{1}{2\pi}\frac{d[\omega(t)]}{dt}$. Using Hilbert transform, we can separate the AM and FM components of the IMF to achieve the purpose of modal separation.

For the single component mode, the instantaneous frequency $f_{IF}t$ should be nearly linear, while the variation range of ω(t) should be considerably small. When mode aliasing occurs, $f_{IF}t$ should clearly change without consideration of the end points. Especially, for hidden components, a jump of $f_{IF}t$ occurs at the time point of concealment, as shown in **Figures 2D**, **3D**. We constructed a variable analysis window according to the time–frequency characteristics of instantaneous frequency. Then, we divided the signal into several parts.

If $f_{IF}t$ of the segmented signal is still unstable, then the modal separation problem can be transformed into the $\frac{d[\omega(t)]}{dt}$ minimisation problem, in which the bandwidth of $sin(\omega(t))$ is minimised. The bandwidth calculation method for nonstationary signals can be obtained by the Carson rule:

$$BW_{AM-FM} = 2\left(\Delta f + f_{FM} + f_{AM}\right) \tag{8}$$

where Δf is the deviation of the instantaneous frequency from its mean value and f_{AM} and f_{FM} denote the frequencies of the AM and FM signals, respectively. We can make $\Delta f = 0$ to minimise the bandwidth. In other words, the decomposition frequency of each IMF is expected to be equal to the centre frequency of the instantaneous frequency, that is, equal to the mean value of the instantaneous frequency $\overline{f_{IF}}t$. Then, a mask signal with the same frequency as $\overline{f_{IF}}t$ can be selected and the number of IMFs required can be determined.

Algorithm Description

The HM-EMD algorithm comprises the following steps: variable analysis window construction and mask signal construction.

1) Variable Analysis Window Construction.

The jump point t_i should be picked such that **Eq. 9** is satisfied:

$$\left(\left|f_{IF}(t_i) - f_{IF}(t_{i+1})\right| + \left|f_{IF}(t_{i-1}) - f_{IF}(t_i)\right|\right) > \mu_{\Delta f_{IF}(t)} + \rho\varepsilon_{\Delta f_{IF}(t)} \tag{9}$$

where $\Delta f_{IF}(t)$ is the difference in instantaneous frequencies at t_i, $\mu_{\Delta f_{IF}(t)}$ is the mean value of $\Delta f_{IF}(t)$ at all time points, $\varepsilon_{\Delta f_{IF}(t)}$ is the variance and ρ is the variable parameter. The original signal is divided into two parts by the time division points t_i and decomposed by EMD independently.

2) Mask Signal Construction.

The sine signal is a common form of a mask signal, and its amplitude and frequency should be determined. As analyzed in *Empirical Mode Decomposition Method*, the frequency is determined as the average instantaneous frequency $\overline{f_{IF}}$. Hence, the amplitude is also determined as the mean value $\overline{A_{IF}}$ of the instantaneous amplitude. Then, the mask signal s*t* is defined as

$$st = \overline{A_{IF}}sin2\pi\overline{f_{IF}}t \tag{10}$$

where $\overline{A_{IF}} = \frac{1}{n}\sum_{t=1}^{n}\sqrt{r_{IF}(t)^2 + H(r_{IF}(t)^2)}$ and $\frac{1}{n}\sum_{t=1}^{n}\frac{d}{dk}\left(arctan\frac{H(r_{IF}(t))}{r_{IF}(t)}\right)$.

Then, the IMFs can be refreshed by **Eqs. 3**–**5**, in which the number of IMFs are determined by $\overline{f_{IF}}$ and f_c is the sampling frequency. The algorithm flow is as follows **Algorithm 2**.

HM-EMD-BASED ACOUSTIC SCENE CLASSIFICATION

Acoustic Scene Signal Analysis

When processing the original signal with HM-EMD, the variable analysis window and mask signal are used to intervene the decomposition of the original signal. The frame length is selected according to the frequency structure of the signal itself, while the frequency domain components corresponding to each IMF are relatively independent, which provides higher interpretability of the features. The instantaneous frequency and amplitude of each IMF also contain all information of IMF components, which means that the instantaneous frequency and amplitude of all IMF components contain most of the information of the signal to be analyzed, and can be directly used as the basic characteristics of the signal. **Figure 4** shows the time-domain waveforms of some typical IMFs with hiding acoustic events in the ambient audio stream, in which only the most significant one of all IMF waveforms is shown. It can be seen that the time-domain waveform characteristics of these events are very obvious, the extreme value and over average rate are very different, and they are distributed in low, medium and high frequency bands, such the airport luggage roller in **Figure 4A** and Metro rail joint collision in **Figure 4B** are low frequency, steps in **Figure 4D** and tram acceleration in **Figure 4F** are medium frequency, chirm in **Figure 4C**, vehicles from far to near in **Figure 4E** are high frequency. Therefore, this paper proposes a full band IMF hiding component features, which can distinguish them well, to effectively improve the effect of ambient audio stream recognition algorithm, the feature calculation method is shown in *Mutagenic Component Features*.

Mutagenic Component Features

Figure 4 shows various hidden components in the acoustic scene data. On the one hand, the hidden components cause a significant interference to the signal spectrum, thereby greatly affecting the ambient audio stream recognition effect based on traditional spectrum features (such as MFCC). On the other hand, the types and characteristics of hidden components corresponding to different ambient audio streams also exhibit significant differences. These hidden components are closely related to the types of acoustic events. The features constructed on the basis of hidden

Algorithm 2 | Heuristic empirical mode decomposition with a masking signal

Heuristic Empirical Mode Decomposition with a Masking Signal
Input: Signal, Supposed IMF number, input: Signal x(t), Supposed IMF number i
output: Intrinsic Mode Function, IMF
1 $x_1(t) = x(t)$, i = 1;
2 Get the first IMF of the signal residual $x_i(t)$, calculate the mean and variance of $\Delta f_{IF}t$, use **Eq. 8** to determine whether there is a hiding jump point, variable analysis window is constructed according to the hiding jump point and $x_i(t)$ is segmented.
3 Construct mask signal for each IMF_i: $s_i t = \overline{A_{IF_i}} sin 2\pi \overline{f_{IF_i}} t$;
4 Do EMD on $x_{i+}t = x_i(t) + s_i t$ and $x_{i-}t = x_i(t) - s_i t$, get the first IMF $r_{IMF_i+}(t)$ and $r_{IMF_i-}(t)$;
5 Let $r_{IMF_i}(t) = (r_{IMF_i+}(t) + r_{IMF_i-}(t))2$, and splice all the divided pieces.
6 i = i+1, $x_i(t) = x_{i-1}(t) - r_{IMF_i}(t)$, return to step2, until $\overline{f_{IF_i}}t < \frac{f_c}{2i}$, or no new IMF is required;

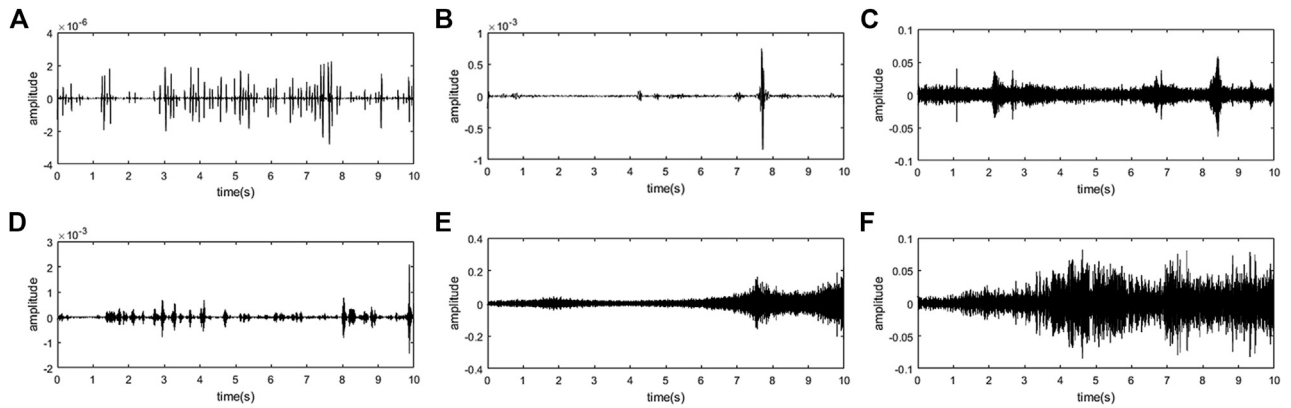


FIGURE 4 | IMF waveforms with significant hidden components in different environments of the audio stream, **(A)** Airport luggage roller–IMF16, low freq; **(B)** Metro rail joint collision–IMF14, low freq; **(C)** Chirm–IMF1, high freq; **(D)** steps–IMF10, medium freq; **(E)** Vehicles from far to near–IMF1, high freq; **(F)** Tram acceleration–IMF4, Medium and high freq.

components can help to distinguish ambient audio streams. For a hidden component, its frequency, amplitude and change mode information can effectively reflect its essential attributes. Almost all of such information can be reflected by the envelope shape of the IMF obtained by decomposition. Therefore, we design a set of HACFs. Based on the IMF decomposed by HM-EMD, the features extract the relevant information of hidden components, including the shock intensity feature SH and over-average feature average crossing rate (ACR).

1) Shock intensity feature (SH):

$$SH_{maxj} = max\left(r_{IMFj}^{up}(t)\right) \text{and } SH_{\min j} = \min\left(r_{IMFj}^{up}(t)\right) \tag{11}$$

where max $(r_{IMFj}^{up}(t))$ is the upper limit of the signal amplitude in the jth IMF and min $(r_{IMFj}^{up}(t))$ is the lower limit. Both limits represent the change intensities of the hidden components relative to the steady components for measuring the changes in signal amplitude. As the sum of the mean values of the upper and lower envelopes of the IMF is 0, the signal is symmetrical along the time axis, and the information carried by the upper and lower envelopes are almost the same. Therefore, a one-sided envelope is enough to ensure the consistency of the symbols of the two values. The superscript means that the upper envelope is used for calculation.

2) ACR feature:

$$ACR_i = \frac{1}{2T}\sum_{i=2}^{T}\left|sgn\left[r_{IMFj}^{up}(t) - \overline{r_{IMFj}^{up}(t)}\right] - sgn\left[r_{IMFj}^{up}(t-1) - \overline{r_{IMFj}^{up}(t)}\right]\right| \tag{12}$$

ACR features can express the number of times the upper envelope of an IMF passes through its mean point, that is, the number of times the IMF's upper envelope (time domain amplitude) fluctuates significantly. If the value is large, the IMF amplitude frequently fluctuates near the mean value. For ambient audio stream recognition application scenarios, if the value is greater than a certain threshold (10 Hz or above), the data may not have obvious and meaningful hidden components and the change of the upper envelope near the mean value is only the normal fluctuation of the acoustic signal itself. If the value is less than the threshold, the data may contain significant hidden components, and one-half of the zero crossing frequency is the frequency of the hidden components.

Ambient Audio Stream Classification

The ambient audio stream classification process based on HM-EMD is shown in **Figure 5**. HM-EMD is used to obtain the IMF set of the

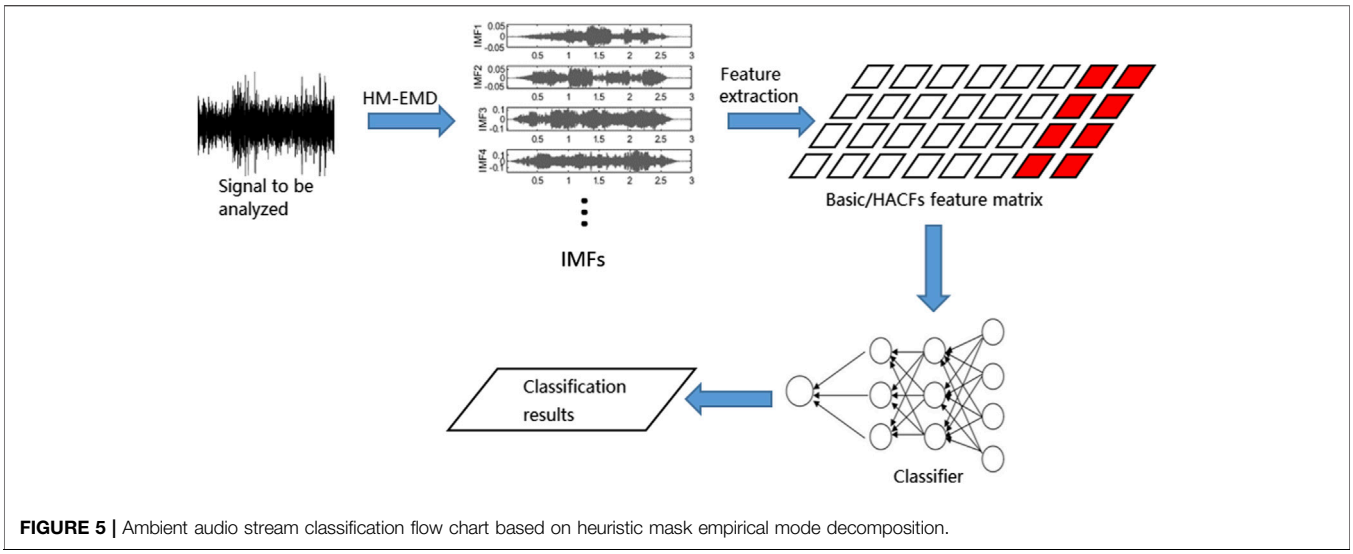


FIGURE 5 | Ambient audio stream classification flow chart based on heuristic mask empirical mode decomposition.

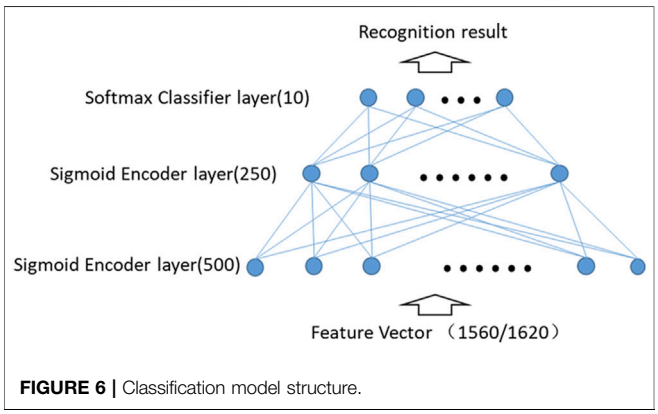


FIGURE 6 | Classification model structure.

signal to be analyzed. Then, the following basic features are extracted: instantaneous frequency, instantaneous amplitude and the HACFs proposed in this work. Organised as a feature matrix, the features are input into the classifier to obtain the final recognition result. We select the neural network model for the classifier. The network structure is shown in **Figure 6**. To prove the effectiveness of the features, we select a three-layer neural network model. The first two layers use a sigmoid function as the activation function. First hidden layer has 500 and second hidden layer has 250 neurons. The output layer uses a softmax classifier and has 10 neurons. The experimental results show that the feature system still shows satisfactory results even with the use of a simple classification model. The specific experimental results and analysis are presented in the next section.

EXPERIMENTS AND RESULTS

Experimental Setup

We verify the results of this work from two aspects: the validity of modal separation and the validity of the HM-EMD features for environmental audio stream classification. The experiments use Python language, the deep learning framework uses the PyTorch framework, and the data set uses the task 1A audio scene classification dataset in DCASE competition.

Validation of Modal Separation

A nonlinearity index is defined in **Eq. 13**, and it measures the stability of the decomposition results. The larger the DN is, the greater the nonlinear degree is, indicating the more unstable components; the verification data are the mixed signals of the three modes in **Figures 1–3.**

$$DN = \left[\frac{1}{n}\sum_{t=1}^{n}\left(\frac{f_{IF}t - \overline{f_{IF}t}}{\overline{f_{IF}t}}\right)^2\right]^{1/2} \tag{13}$$

Validation of the Features of HM-EMD for the Classification of Ambient Audio Streams

The data used in the experiment come from the TASK1A dataset of DCASE [19]. Task1A that is to classify the acoustic scene with multiple devices. The dataset contains data on ten cities and nine devices, that is, three real devices (A, B, C) and six simulated devices (S1–S6). The dataset has good annotation, including three different types of indoor, outdoor and traffic. It also has ten different ambient audio streams, namely, airport, shopping mall, metro, metro station, pedestrian, street traffic, tram, park and public square and bus. The acoustic data span a total of 64 h, with 40 h used in dataset training and with 24 h used in verification. Each audio segment is 10 s long, and the sampling rate is 44.1 kHz.

To verify the effectiveness of designing a series of features based on HM-EMD, we use a basic HM-EMD feature matrix and a basic features + HACF matrix as the input parameters of the classifier. Specifically, the number of mask EMD reference IMFs is 20, HM-EMD basic feature is 2D and HACFs is 3D, which number of dimension is 20 × 3. The audio frame length is 0.5 s, and the interframe overlap is 0.25 s, the total number of dimensions is 39 × 20 × 3 = 2340. The classical MFCC are selected as the contrast features; they include 13 dimensional MFCCs and delta features. The

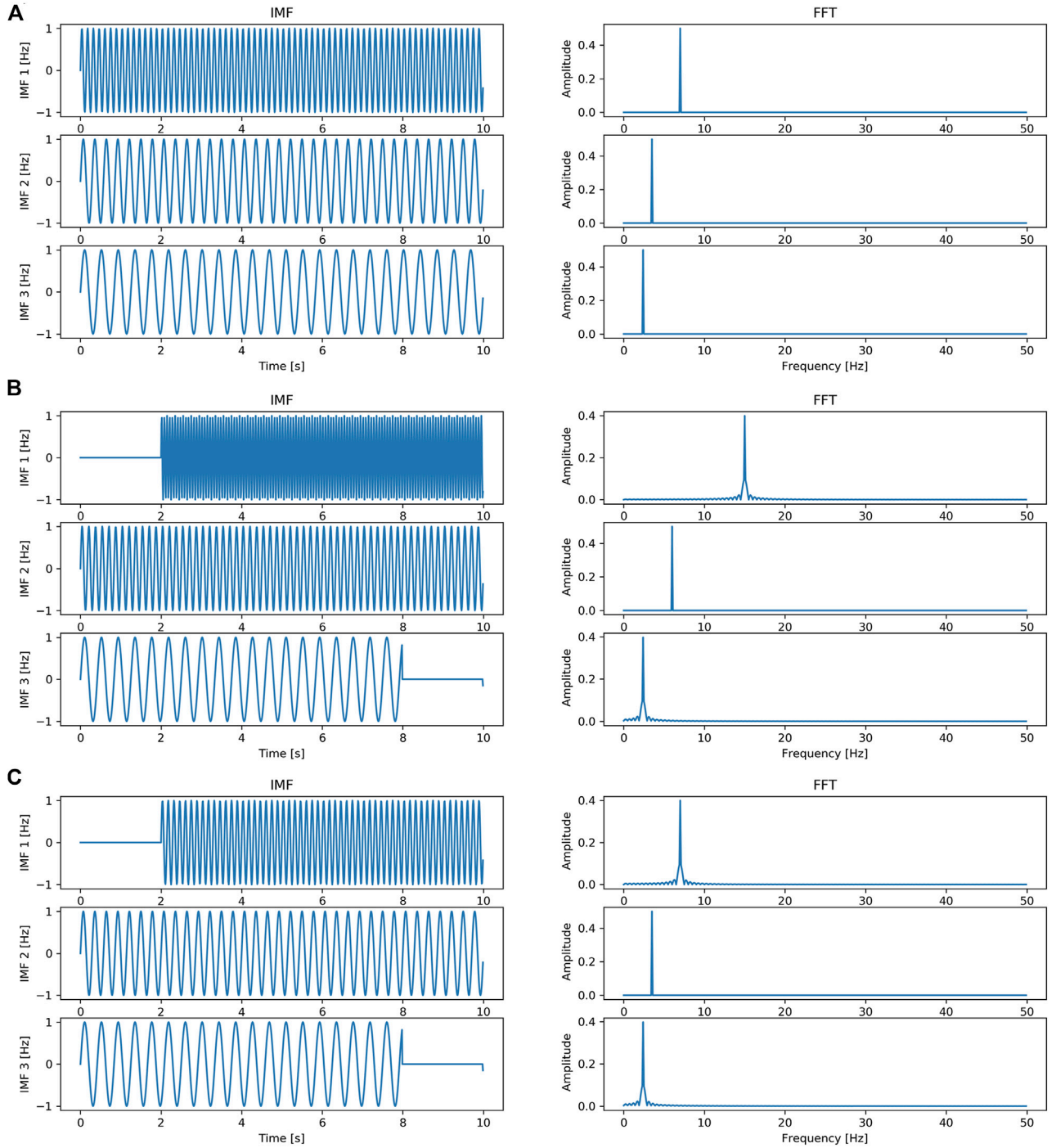


FIGURE 7 | HM-EMD results of $x_1(t)$, $x_2(t)$, $x_3(t)$, **(A)** IMFs of $x_1(t)$ and the FFT spectrum of each corresponding IMF; **(B)** IMFs of $x_2(t)$ and the FFT spectrum of each corresponding IMF; **(C)** IMFs of $x_3(t)$ and the FFT spectrum of each corresponding IMF.

total number of dimensions is 39, and the audio frame length is 40 ms. The specific experimental results are described herein.

After setting the characteristic parameters, we conducted the test according to the process designed in **Figure 5**. We trained the classifier parameters with the training dataset and tested them with the test set.

Results and Analysis

Effectiveness Analysis for Modal Separation

By comparing the traditional EMD results, we can see DNHM-EMD/DNEMD <1 for any given case. Hence, the IMF processed by the HM-EMD method has the lowest

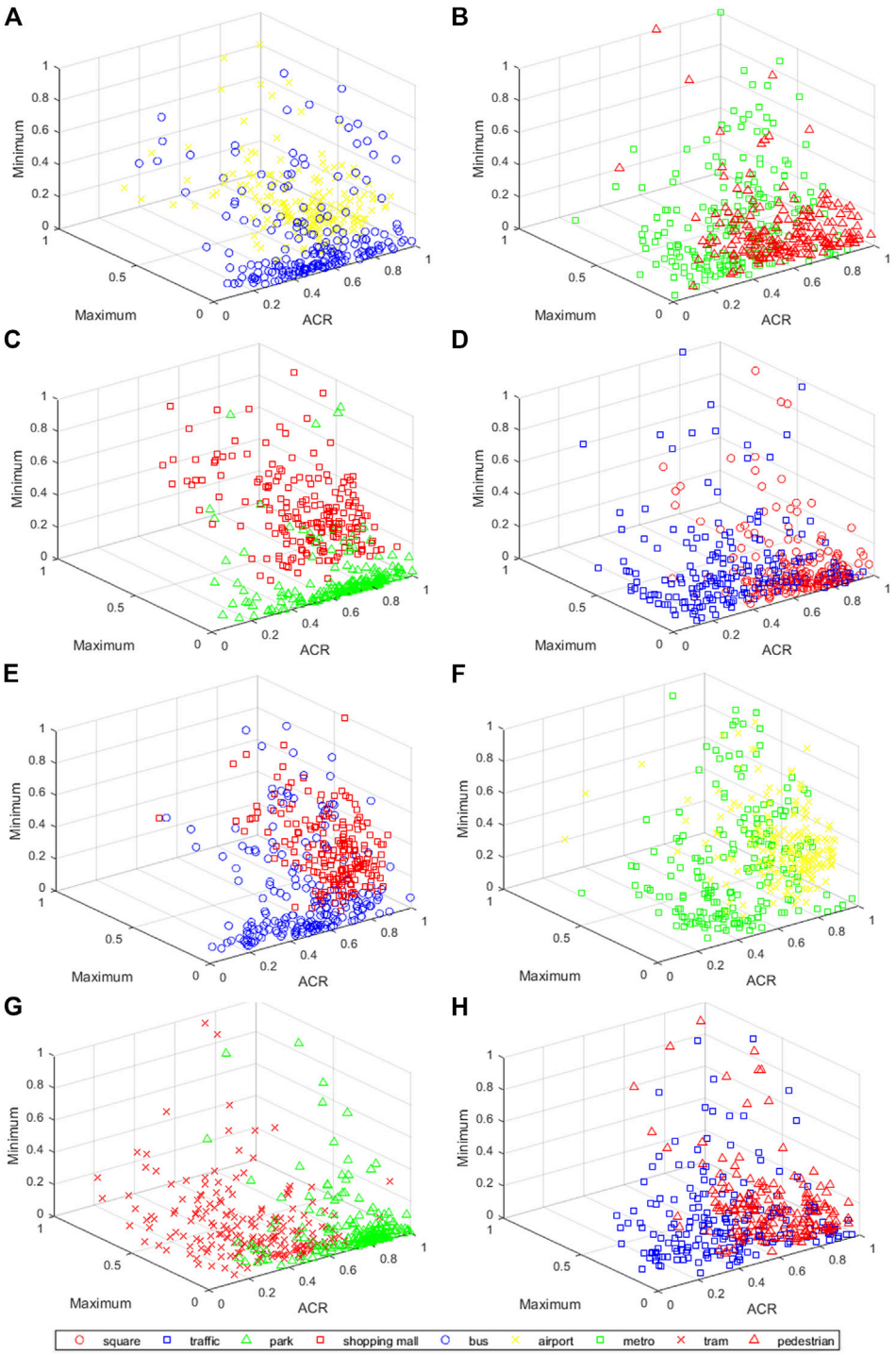


FIGURE 8 | HACFS feature distribution in 3D space, **(A)** IMF1 of bus & airport; **(B)** IMF1 of metro & pedestrian; **(C)** IMF1 of park & shopping mall; **(D)** IMF1 of square & traffic; **(E)** IMF2 of bus & shopping mall; **(F)** IMF2 of met; **(G)** IMF2 of tram & park; **(H)** IMF2 of traffic & pedestrian.

nonlinearity; that is, the IMF has a high purity and is close to the blind separation result under an ideal state. The separation result is shown in **Figure 7**. From the FFT spectrum corresponding the IMFs of $x_1(t), x_2(t)$ and $x_3(t)$ in **Figures 7A–C,** we can see that the mode aliasing was solved and each IMF was pure. The features based on this high-purity

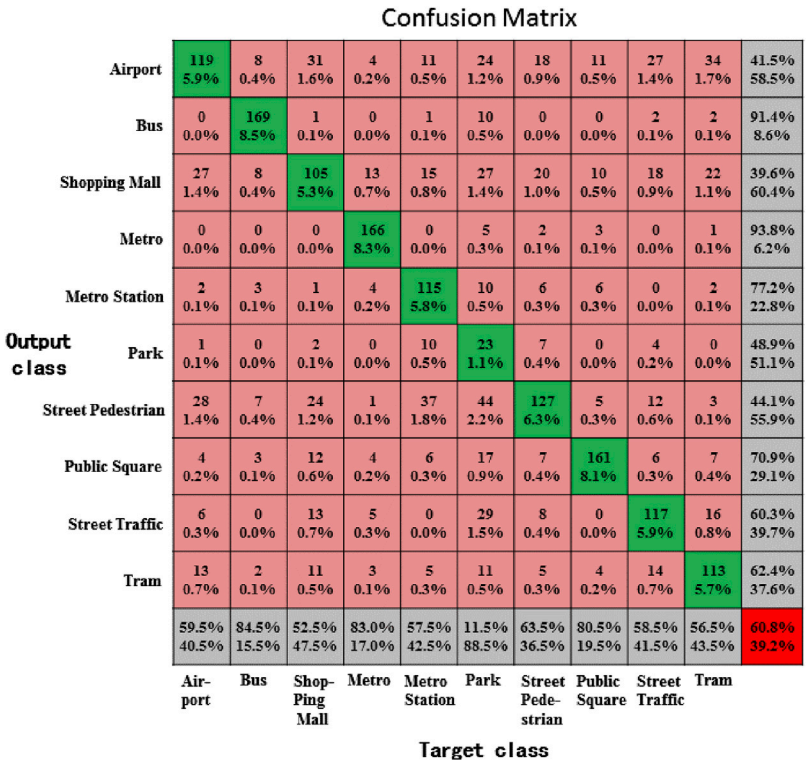


FIGURE 9 | Recognition results of basic instantaneous frequency and instantaneous amplitude features.

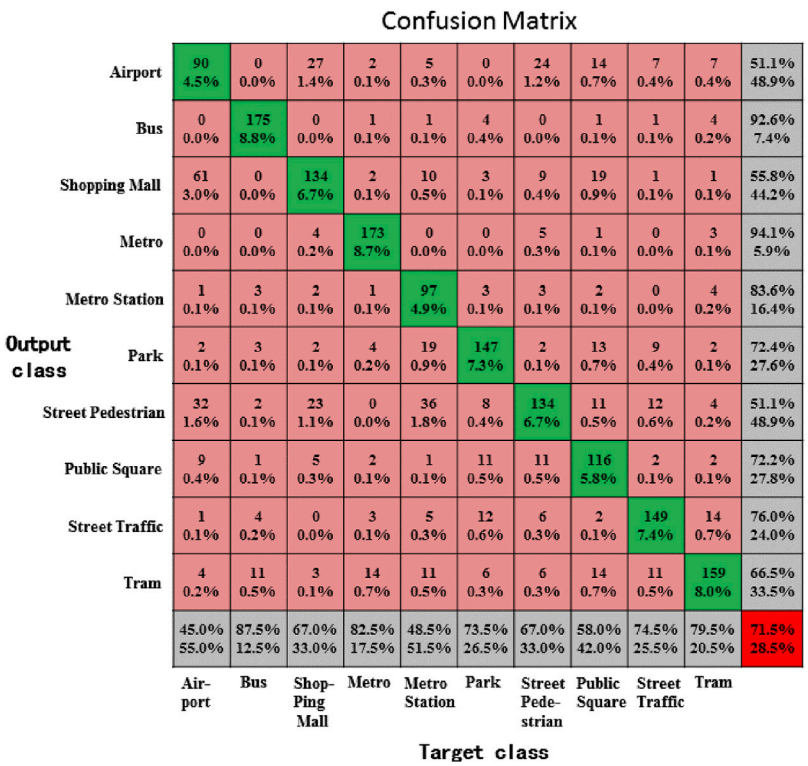


FIGURE 10 | Recognition results of basic features and HACFs features.

IMF signal can effectively characterize the subtle changes in the signal components in the time and frequency domains. Hence, the method is suitable for all types of acoustic correlation analyses and recognition, especially for the recognition of ambient audio streams with hidden acoustic events.

Based on HM-Feature Validity of EMD

HACFs can be used to identify the hidden components in IMFs and are thus of great significance for ambient audio stream recognition. We verified the discrimination ability of HACFs in different scenarios in **Figure 8**. The figure shows the scatter projection of some hidden component features in the three-dimensional space. Even the three-dimensional features in a single IMF have a strong scene discrimination ability. HACFs show good discrimination ability among different ambient audio stream categories and thus provide technical support for subsequent ambient audio stream classification.

We use the simple classifier shown in **Figure 6** to classify and recognize the environmental audio streams in different scenes based on HM-EMD basic feature and basic + HACFs feature respectively, and then use the confusion matrix to represent the recognition accuracy in each scene. The vertical axis represents the classification output, and the horizontal axis represents the annotation result, as shown in **Figures 9**, **10**. As can be seen from **Figure 9**, the average recognition rate for all scenes is 60.8%, and the average recognition rate increases to 71.5% in **Figure 10** after adding HACFs feature. The main improvements are achieved in the airport, shopping mall, metro station, park, pedestrian, street traffic and tram scenes. Special hidden events often occur in these scenes, and these acoustic events are highly related to the background. Therefore, the proposed HACFs can effectively represent the hidden information to improve recognition rate.

We use the basic classifier and complex classifier in each classifier, according to HM-EMD basic characteristics, HM-EMD basic features + HACFs features and classic MFCC features are used to classify and recognize the environmental audio stream. The basic classifier is the simple three-layer perceptron shown in **Figure 6**, while the complex classifier adopts the optimal classifier used in the DCASE competition [24], the classification results are shown in **Tables 1**, **2**. From these two tables, it can be seen that the HM-EMD feature is superior to the MFCC feature with different classifiers: Given the basic classifier, $f_{IF} + A_{IF}$ is 6.7 percentage points higher than that in the MFCC series; after the addition of HACFs, the recognition rate increases by 17.4 percentage points. This result is close to the classification accuracy of the RESNET network with a 32 m model size in the DCASE competition, while the simple model we used is only 225K. In a complex classification model, the improvement of model classification can make up for the lack of features to some extent. However, in this case, $f_{IF} + A_{IF} + AHCFs$ still improves the accuracy by 1.3%, and the recognition result reaches 75.7%. This result indicates that the HM-EMD feature provides complete and pure time–frequency domain information, which helps improve the accuracy of the classification of environmental audio streams and proves the effectiveness of the proposed features.

TABLE 1 | Comparison of environmental audio stream classification results based on DCASE dataset.

Scene label	MFCC (%)	$f_{IF} + A_{IF}$	$f_{IF} + A_{IF} + HACFs$
Airport	45.0	41.5%	51.1%
Bus	62.9	91.4%	92.6%
Metro	53.5	93.8%	94.1%
Metro Station	53.5	77.2%	83.6%
Park	71.3	48.9%	72.4%
Public Square	44.9	70.9%	72.2%
Shopping Mall	48.3	39.6%	55.8%
Street Pedestrian	29.8	44.1%	51.1%
Street Traffic	79.9	60.3%	76.0%
Tram	52.2	62.4%	66.5%
Average	54.1	60.8%	71.5%

TABLE 2 | Comparison of classification results of DCASE dataset based on complex classifiers.

Classifier	MFCC (%)	$f_{IF} + A_{IF}$	$f_{IF} + A_{IF} + HACFs$
TridentResNet_DevSet	73.7	74.5%	75.0%
TridentResNet_EvalSet	73.7	74.5%	75.0%
TridentResNet_Ensemble	74.2	75.0%	75.5%
TridentResNet_Weighted_Ensemble	74.4	75.2%	75.7%

CONCLUSION

The processing requirements of hidden acoustic signals in ambient audio stream classification are analyzed in this work. Specifically, an ambient audio stream feature extraction method based on HM-EMD is proposed. With the construction of an adaptive mask signal, the frequency domain distribution and IMF dimension are stabilised, and the instability of the time–frequency domain feature system in the acoustic signal processing of the classical EMD algorithm is solved. Then, the time–frequency analysis characteristics of EMD in nonlinear and nonstationary signal processing are fully exploited. Through the Hilbert transform spectrum of each IMF, the hidden components in ambient audio stream signals are analyzed and located to construct the related HACFs. The experimental results show that HM-EMD-based features exhibit greater capability in hidden acoustic event representation than MFCC. Therefore, in our future work, we will study the methods to improve the representation ability of ambient audio streams by exploring the relationship between HM-EMD feature systems and different hidden acoustic events. Attempts will also be made to achieve the accurate labelling of hidden acoustic events in multilevel and multitime scale ambient audio streams with HACFs, such as hidden acoustic event location and hidden acoustic event recognition. In general, the ambient audio stream feature extraction based on HM-EMD represents an effective effort toward ambient audio stream classification. By improving the time–frequency resolution for the analysis of nonstationary environmental acoustic signals, capturing the hidden features of the environment and enhancing the local feature representation, the proposed method can effectively improve the

efficiency and performance of the classification modelling of the hidden information of ambient audio streams, which provides technical support for camouflaged audio information detection. Hence, it helps to reduce the risk of audio camouflage attacks in social networks.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: http://dcase.community/challenge2020/task-acoustic-scene-classification.

AUTHOR CONTRIBUTIONS

JL and DZ conceived and designed the study. ZX performed the simulations. ZZ and LY reviewed and edited the manuscript. All authors read and approved the final manuscript.

FUNDING

This work was supported in part by the N National Key Research and Development Program (2018YFC0830602), and the Sichuan Science and Technology Program (No. 2019YFSY0049).

REFERENCES

1. Li S, Zhao D, Wu X, Tian Z, Li A, and Wang Z. Functional Immunization of Networks Based on Message Passing. *Appl Mathematics Comput* (2020). 366: 124728. doi:10.1016/j.amc.2019.124728
2. Han W, Tian Z, and Huang Z, Topic Representation Model Based on Microblogging Behaviour Analysis. *World Wide Web*. (2020). 23. p. 11–2. doi:10.1007/s11280-020-00822-x
3. Chen T, Kumar A, Nagarsheth P, Sivaraman G, and Khoury E. *"Generalization of Audio Deepfake Detection," Odyssey 2020 the Speaker and Language Recognition Workshop*. Tokyo, Japan (2020). p. 132–7. doi:10.21437/Odyssey.2020-19
4. Jati A, Hsu C-C, Pal M, Peri R, AbdAlmageed W, Narayanan S, et al.Adversarial Attack and Defense Strategies for Deep Speaker Recognition Systems. *Computer Speech Lang* (2021). 68:101199. doi:10.1016/j.csl.2021.101199
5. Al-Turjman F, and Salama R. "Cyber Security in Mobile Social Networks," *Security IoT Soc Networks*. (2021). p. 55–81. doi:10.1016/b978-0-12-821599-9.00003-0
6. Lin ZD, Di CG, and Chen X. Bionic Optimization of MFCC Features Based on Speaker Fast Recognition. *Appl Acoust* (2020). 173:107682. doi:10.1016/j.apacoust.2020.107682
7. Sudo Y, Itoyama K, Nishida K, and Nakadai K. Sound Event Aware Environmental Sound Segmentation with Mask U-Net. *Adv Robotics* (2020). 34(20):1280–90. doi:10.1080/01691864.2020.1829040
8. Waldekar S, and Saha G. Two-level Fusion-Based Acoustic Scene Classification. *Appl Acoust* (2020). 170:107502. doi:10.1016/j.apacoust.2020.107502
9. Baltrusaitis T, Ahuja C, and Morency LP. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Trans Pattern Anal Mach Intell* (2018). 41(2): 423–43. doi:10.1109/TPAMI.2018.2798607
10. Zhang T, and Wu J. Constrained Learned Feature Extraction for Acoustic Scene Classification. *Ieee/acm Trans Audio Speech Lang Process* (2019). 27(8): 1216–28. doi:10.1109/taslp.2019.2913091
11. Chandrakala S, and Jayalakshmi SL. Generative Model Driven Representation Learning in a Hybrid Framework for Environmental Audio Scene and Sound Event Recognition. *IEEE Trans Multimedia* (2019). 22(1):3–14. doi:10.1109/TMM.2019.2925956
12. Xie J, and Zhu M. Investigation of Acoustic and Visual Features for Acoustic Scene Classification. *Expert Syst Appl* (2019). 126:20–9. doi:10.1016/j.eswa.2019.01.085
13. Lostanlen V, Lafay G, and Anden J. Relevance-based Quantization of Scattering Features for Unsupervised Mining of Environmental Audio. *EURASIP J Audio Speech Music Process* (2018). 15. doi:10.1186/s13636-018-0138-4
14. Li SD, Jiang LY, and Wu XB. A Weighted Network Community Detection Algorithm Based on Deep Learning. *Appl Mathematics Comput* (2021). 401: 126012. doi:10.1016/j.amc.2021.126012
15. Suh S, Park S, and Jeong Y. (2020).Designing Acoustic Scene Classification Models with CNN Variants. *Dcase2020 Challenge, Tech Rep*. Available at: http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Suh_101.pdf.
16. Garcia-Romero D, and McCree A. Stacked Long-Term TDNN for Spoken Language Recognition. In 17th Annual Conference of the International-Speech-Communication-Association (INTERSPEECH 2016). San Francisco, CA, USA (2016). p. 3226–30.
17. Cramer J, Wu HH, and Salamon J. Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). United Kingdom: Brighton (2019). p. 3852–6.
18. Khan MA, Javed K, Khan SA, Saba T, Habib U, Khan JA, et al. Human Action Recognition Using Fusion of Multiview and Deep Features: an Application to Video Surveillance. *Multimed Tools Appl* (2020). doi:10.1007/s11042-020-08806-9
19. Heittola T, Mesaros A, and Virtanen T. TAU Urban Acoustic Scenes 2020 Mobile, Development Dataset. *Dcase2020 Challenge, Tech Rep* (2020). doi:10.5281/zenodo.3819968
20. Barbosh M, Singh P, and Sadhu A. Empirical Mode Decomposition and its Variants: A Review With Applications in Structural Health Monitoring. *Smart Mater Struct* (2020). 29(9):093001. doi:10.1088/1361-665X/aba539
21. Kim D, Kim KO, and Oh H-S. Extending the Scope of Empirical Mode Decomposition by Smoothing. *EURASIP J Adv Signal Process* (2012). 2012: 168. doi:10.1186/1687-6180-2012-168
22. Deering R, and Kaiser JE. The Use of a Masking Signal to Improve Empirical Mode Decomposition. In ICASSP 2005-2005 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Philadelphia, PA, USA (2005). p. 485–8.
23. Li H, Li Z, and Mo W. A Time Varying Filter Approach for Empirical Mode Decomposition. *Signal Process.* (2017). 138:146–58. doi:10.1016/j.sigpro.2017.03.019
24. Shim HJ, Kim JH, and Jung JW. Audio Tagging and Deep Architectures for Acoustic Scene Classification: Uos Submission for the DCASE 2020 Challenge. *Dcase2020 Challenge, Tech Rep* (2020). Available at: http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Shim_120.pdf.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

frontiers in Physics

ORIGINAL RESEARCH
published: 09 August 2021
doi: 10.3389/fphy.2021.687883

Critical Terrorist Organizations and Terrorist Organization Alliance Networks Based on Key Nodes Founding

Jun Hu[1], *Chengbin Chu*[1], *Ling Xu*[1], *Peng Wu*[1]* *and Hui-jia Lia*[2]*

[1]School of Economics and Management, Fuzhou University, Fuzhou, China, [2]School of Science, Beijing University of Posts and Telecommunications, Beijing, China

The past years have witnessed increasingly widespread terrorism, violently destroying world peace and regional prosperity. Therefore, uncovering terrorist plots has become the most crucial step for eliminating terrorist attacks. However, with the terrorist scheme being disguised under the huge amount of data flow on the internet, identifying terrorist organizations still remains challenging. Since many terrorist organizations are prone to launch terrorist attacks together, here, we model their relationships as a Terrorist Organization Alliance (TOA) network and propose a novel method to identify the key terrorist organizations in the TOA network. The TOA network utilizes existing key nodes in order to extract useful information, and, with the help of the entropy weight method, the new solution to the TOA network is effective and precise. The experiments are performed on the dataset from the Global Terrorism Database, and the results are statistically validated through t-tests and convergence analysis. Compared with the traditional methods, our method is proven to be superior in terms of measure the harm of terrorist attack organizations and find the key terrorist organizations.

Keywords: terrorist organization alliance, key terrorist, complex network, centrality, social network

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Gui-Quan Sun,
North University of China, China
Chao Gao,
Southwest University, China

***Correspondence:**
Peng Wu
wupeng88857@gmail.com
Hui-jia Lia
Hjli@amss.ac.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *30 March 2021*
Accepted: *07 June 2021*
Published: *09 August 2021*

Citation:
Hu J, Chu C, Xu L, Wu P and Lia H-j (2021) Critical Terrorist Organizations and Terrorist Organization Alliance Networks Based on Key Nodes Founding. Front. Phys. 9:687883. doi: 10.3389/fphy.2021.687883

1 INTRODUCTION

In recent years, terrorist attacks have happened frequently around the world. With the rapid and widespread data flow on the internet and media contents, terrorist attacks are becoming increasingly serious. Terrorism has a significant and lasting impact on the social security, political process, and social ecology of all countries [1, 2]. The task of counter-terrorism and terrorism prevention is urgent and arduous; however, the cost is extremely expensive.

Globally, with terrorist organizations such as Al-Qaeda (and the extremist group Islamic State in 2017) suffering heavy blows, terrorist organizations have begun to change their operational strategies. Their activity areas have begun to show a diffuse expansion from the center to periphery, illustrating a new trend of organizational terrorism to individual terrorism, cyber terrorism, and so on. Compared with the government, terrorist organizations are generally relatively weak, and they often cooperate with each other in order to enhance their strength. This is becoming a new trend in the current development of terrorism, manifesting in international cooperation.

In the research of terrorist activity, the traditional studies mostly focus on the forecast of the terrorist attack event based on the terrorist activity characteristic. Xue A. et al (2011) proposed a

prediction algorithm PBCS based on context subspace, and it aims to predict terrorist behavior [3]. Nurudeen. M et al. (2018) proposed a hybrid neural fuzzy model in order to predict criminal behavior in a wide range of areas through simulating crime indication events extracted from wide-area surveillance networks [4]. Li Z. et al.(2018) proposed a comprehensive framework that combines social network analysis, wavelet transform, and the pattern recognition approach to investigate the dynamics and eventually predict the attack behavior of terrorist group [5]. According to the new characteristics of terrorist organization cooperation, scholars introduce network analysis methods to the investigation of terrorist organizations and terrorist attacks. Carly et al (2002) analyzed the terrorist network and suggested that the corresponding terrorist attack prevention strategy should be formulated according to the time of the terrorist attack [6]. Li G. et al.(2019) analyzed the construction process of the terrorist attack alliance network and adopted a new dynamic interactive clustering algorithm to analyze the subgroups of tourist organizations [7, 8]. Hakim et al.(2020) studied the role social contexts played in the link between interpersonal networks and social identity dynamics of a mujahid, found that constraints for the participation in different interpersonal networks. The constraints influenced the process of identity negotiation as a mujahid versus alternative identities of a family member and belonging to a neighborhood [9].

There exist various methods to evaluate the importance of nodes in networks, and many are essentially derived from graph theory [10–11] and graph-based data mining [13, 14]. The research on the importance of nodes in complex networks originates from the field of sociological network analysis [15–18]. Freeman and other scholars have done a lot of research on sociological networks in the early stage. Since then, the fields of system science research, information search, and document retrieval have raised similar problems independently and explored the important sections in networks. The importance of nodes in networks has become a basic problem in various research fields of complex networks.

In this paper, we proposed a new way to analyze the key terrorist organizations. In **section 2**, a definition of the TOA network is provided, and a figure is given to explain the construction process of this network. In **section 3**, some traditional methods are given to find the key terrorist organizations, and an entropy method to find critical organizations is also given base on these traditional methods. In **section 4**, we calculate the importance of nodes by using the traditional method and our method, respectively; In **section 5**, we use the *t*-test and convergence analysis to test the results, which found that the accuracy of our method is better than that of traditional methods.

2. MODEL

Up to now, various countries have been attacked by terrorist organizations, as shown in **Figure 1**, The frequencies of attacks occurring in South Asia, the Middle East, North Africa, and Sub-Saharan Africa amount to 81%, which means that these places are frequently affected by terrorist attacks. In the future, some countries might also be attacked, and it is thus of great significance to analyze the regular pattern of the terrorist attack is very meaningful. In this section, we first propose a Critical Nodes Finding Model for Terrorist Organization Alliance Networks and present a solution.

2.1 Problem Definition

We know that there exist certain social relationships between terrorist groups, including but not limited to sectarian, blood relatives, ethnic relations, etc. At the same time, within the same regions and similar terrorist organizations of related skill fields, resources, and tasks, there is often cooperation. In terms of terrorist attacks, there will be the phenomenon of coalition, and these coalitions will often make some terrorist attacks escalate.

Figure 2 illustrates the construction process of the terrorist groups network. When a terrorist attack event occurs, some groups usually claim that this attack was launched by them. Based on this fact, we suppose there is a relationship between these groups, and these groups will be linked. Thus, a complex network $G = (V, E)$ of terrorist organizations is built. The point set $V(G)$ is the terrorist groups, and the link in E indicates that they make the same attacks at the same time.

3 PRELIMINARIES

3.1 Centrality Measures of the Terrorist Organizations Graph

This is a terrorist organizations graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges. Various methods are proposed to measure the importance of nodes, such as Degree centrality (DC), closeness centrality (CC), and betweenness centrality (BC) [19]. The TOA network as shown in the **Figure 3**.

3.1.1 Degree Centrality

The degree centrality (short for DC) of terrorist organization alliance network's node i, being denoted as $C_D(i)$, is defined as

$$C_D(i) = \sum_{j}^{N} x_{ij}, \tag{1}$$

where i is the focal node, j represents certain node, N is the total number of nodes, and x_{ij} represents the connection between node i and node j. The value of x_{ij} is defined as 1 if node i is connected to node j, and 0 otherwise.

3.1.2 Betweenness Centrality

The betweenness centrality (short for BC) of terrorist organization alliance network's node i, being denoted as $C_B(i)$, is defined as

$$C_B(i) = \sum_{j,k \neq i} \frac{g_{jk}(i)}{g_{jk}}, \tag{2}$$

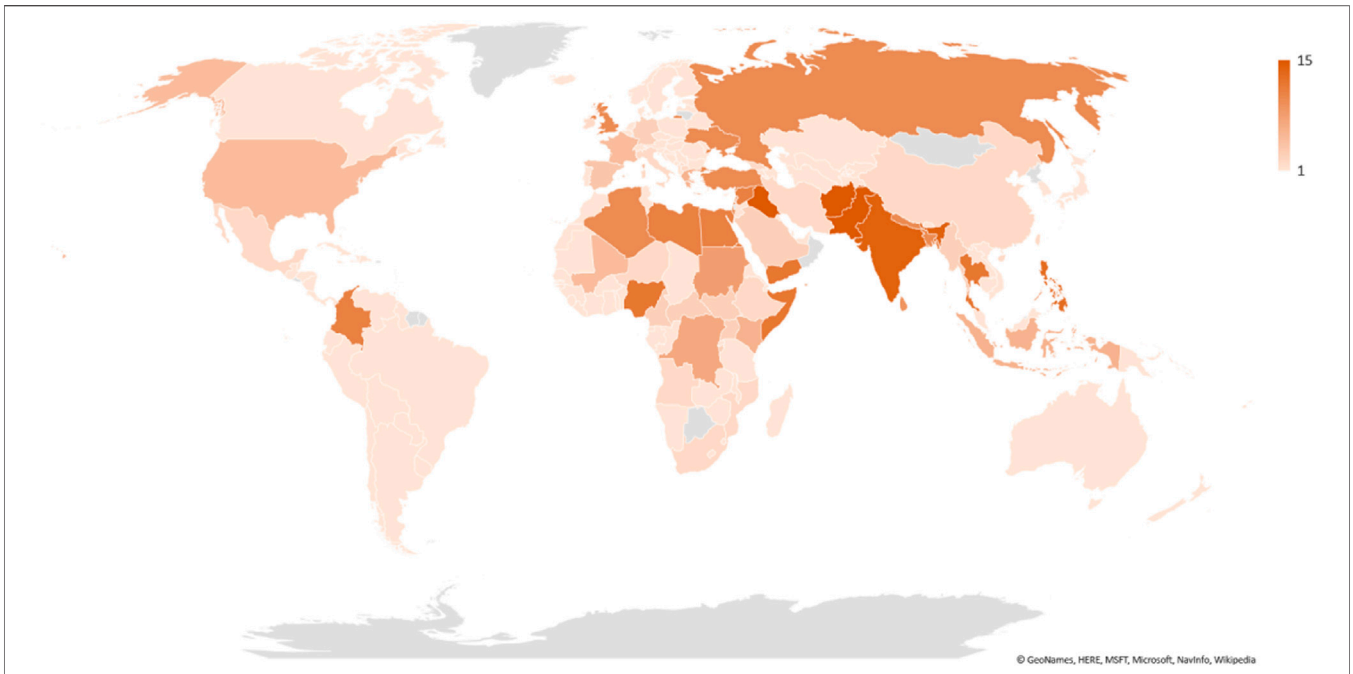


FIGURE 1 | The number of countries suffering from terrorist attacks from terrorist organizations: In this map, there are 15 levels (from 1 to 15). If the number of attacks is fewer than 100, the level is 1; if the number of terrorism attacks is more than 100 and fewer than 200, the level is 1. Thus, the corresponding level will be increased by 1 if the number of terrorism attacks is increased by 100.

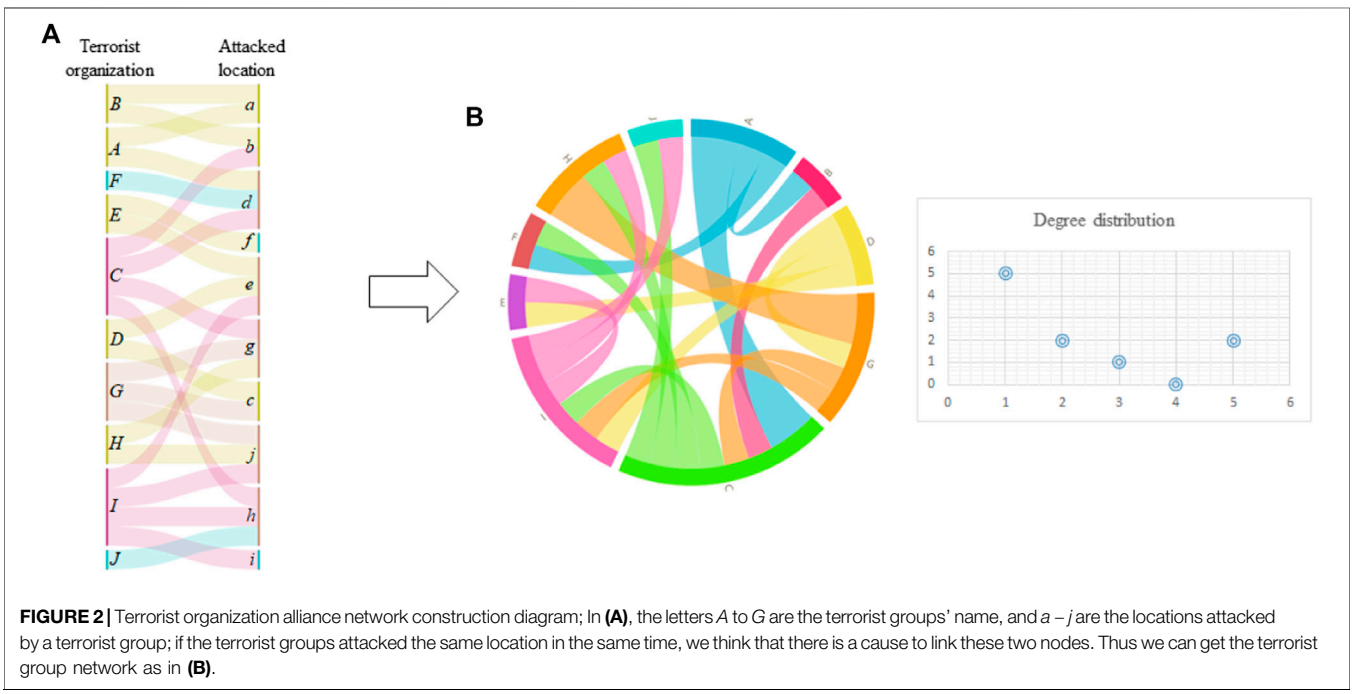


FIGURE 2 | Terrorist organization alliance network construction diagram; In **(A)**, the letters *A* to *G* are the terrorist groups' name, and *a* – *j* are the locations attacked by a terrorist group; if the terrorist groups attacked the same location in the same time, we think that there is a cause to link these two nodes. Thus we can get the terrorist group network as in **(B)**.

where g_{jk} denotes the number of the shortest paths between nodes j and k, and $g_{jk}(i)$ means the number of the shortest paths between nodes j and k that go through node i.

3.1.3 Closeness Centrality

The closeness centrality of terrorist organization alliance network's node i, being denoted as $CC(i)$, is defined as

TABLE 1 | The statistical features of terrorist organization alliance network.

Name	Terrorist organization alliance network
Node	567
Edge	663
Average degree	2.339
Network diameter	15
Weighted average degree	6.688

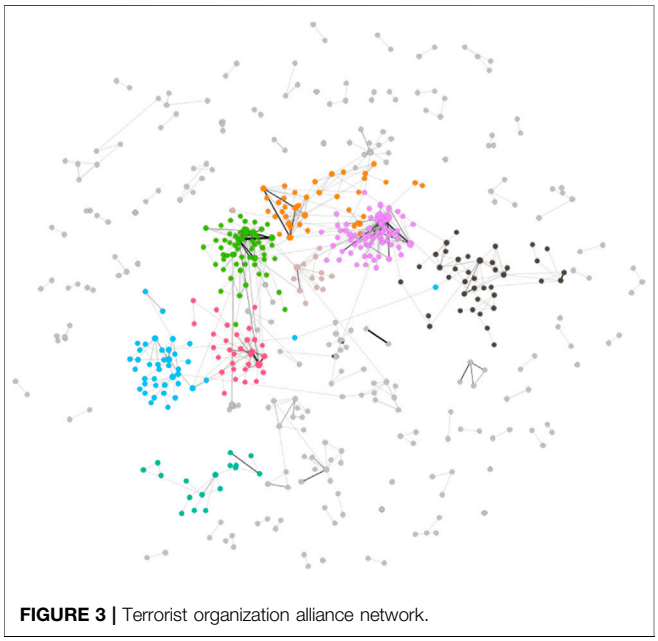


FIGURE 3 | Terrorist organization alliance network.

$$C_C(i) = \left[\sum_j^N d_{ij}\right]^{-1}, \tag{3}$$

where d_{ij} denotes the distance between node i and node j.

3.1.4 Eigenvector Centrality

Let $\mathbf{A}$ be an $n \times n$ similarity matrix. The eigenvector centrality x_i of terrorist organization alliance network's node i is the ith entry in the normalized eigenvector, which belongs to the largest eigenvalue of $\mathbf{A}$. In the matrix $\mathbf{A}$, λ is the largest eigenvalue and n is the number of vertices.

$$Ax = \lambda x, \quad x_i = u\sum_{j=1}^{n} a_{ij}x_j, \quad i = 1, 2, \ldots, n, \tag{4}$$

with proportionality factor $u = \frac{1}{\lambda}$ so that x_i is proportional to the sum of similarity scores of all nodes connected to it.

3.1.5 PageRank

The PageRank is a eigenvector centrality which is used to rank the websites, PageRank is one of these key nodes fingding ways for the key nodes. Mathematically, the PR value of terrorist organization alliance network's node v_i at t step is

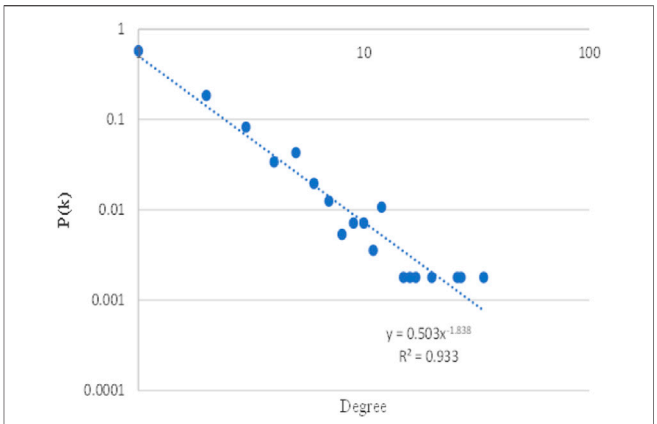


FIGURE 4 | The degree of TOA Network: Plot the number of the degree on the vertical *Y*-axis to logarithmic against weight degree on the horizontal *x*-axis to logarithmic.

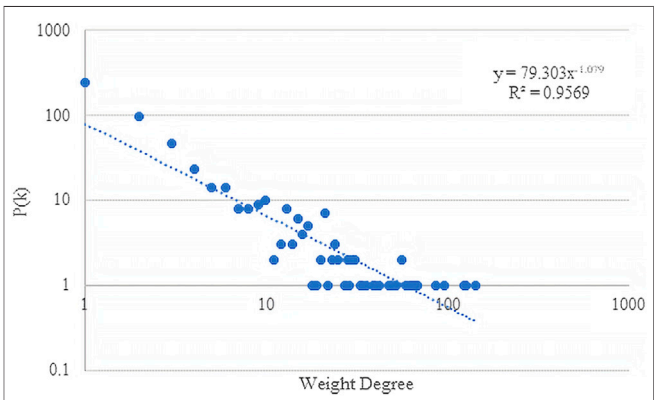


FIGURE 5 | The weight degree distribution of the TOA Network: Plot the number of the weight degree on the vertical *Y*-axis to logarithmic against weight degree on the horizontal *x*-axis to logarithmic.

$$PR_i(t) = \sum_{j=1}^{n} a_{ji}\frac{PR_j(t-1)}{k_j^{out}}, \tag{5}$$

where n is the total number of nodes in the network, and k_j^{out} is the out-degree of node v_j. The above iteration will stop if the PR values of all nodes reach the steady state.

3.2 Information Entropy

In information theory, entropy is defined as measuring the level of uncertainty. The order of the data determines the degree of entropy, i.e., according to the definition of information entropy, we know that the information entropy of this group of data will be greater when a group of data contains more information. Therefore, the larger the entropy of the data, the greater its weight.

Whether people can get high-quality decision-making information is determined by the quality of data. Therefore, it is particularly important to find high-quality data in multiple

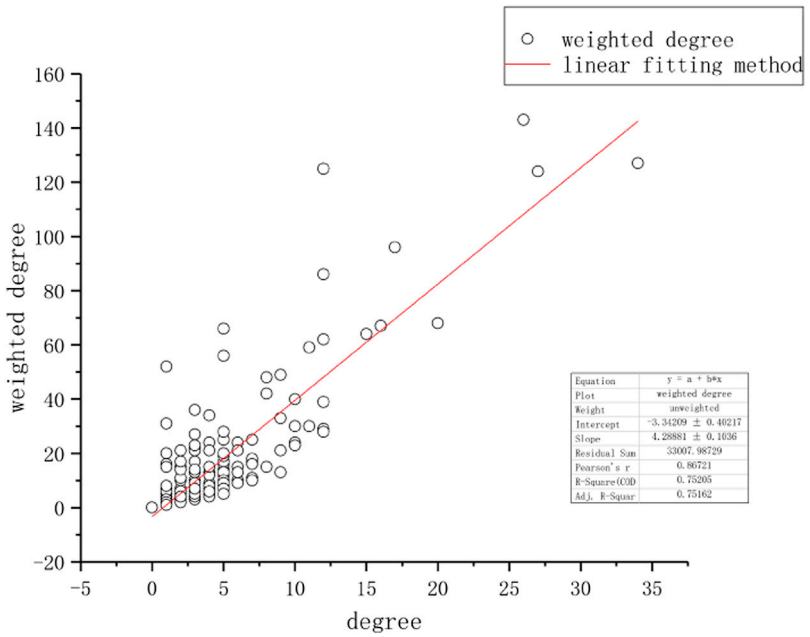


FIGURE 6 | The relationship between the degree and the weight degree of TOA Network: The linear relation between the degree and the weighted degree in TOA network is shown in this figure. After applying a linear fitting method, an linear equation, $y = 4.28881x - 3.34209$, is plotted. Note that the Pearsons *r*, *R*-Square, *Adj*. *R*-Square are all greater than 0.5, meaning that this fitting is quite appropriate.

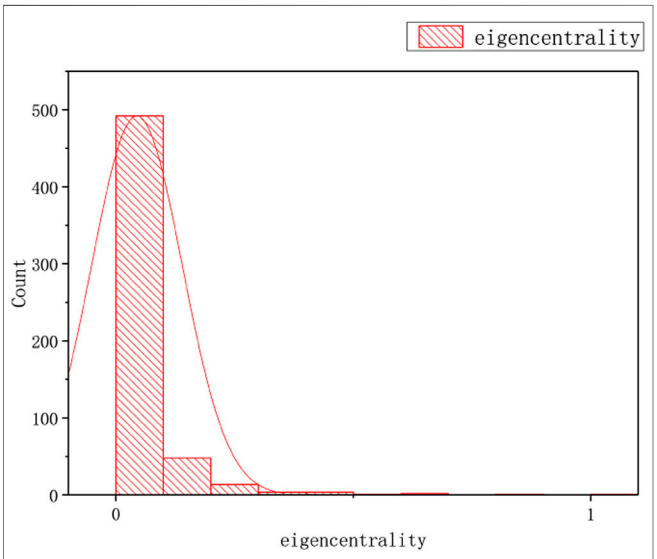


FIGURE 7 | The distribution of eigencentrality and RangeRank of the TOA Network: Plot the number of the number of eigencentrality on the vertical *Y*-axis against eigencentrality on the horizontal *x*-axis.

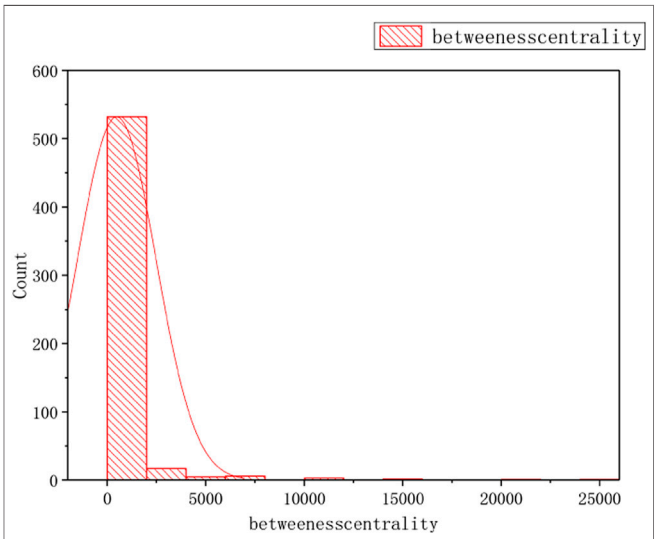


FIGURE 8 | The distribution of betweeness centrality of the TOA Network: Plot the betweeness centrality on the vertical *Y*-axis against betweeness centrality on the horizontal *x*-axis.

data. Entropy is utilized to measure the data's order, which can reflect the importance of data. Thus, based on the information theory, we can find the important factors, which are the high-weight data. In the multi-index decision-making problem, the greater the variability of the index is, the smaller the information entropy is, thus we can more information from this index. Therefore, the weight of this index is bigger than other indexes [20–22].

Definition 1. Among the problems of evaluating n objects and M evaluation indicators, the entropy of the i–th evaluation indicator is

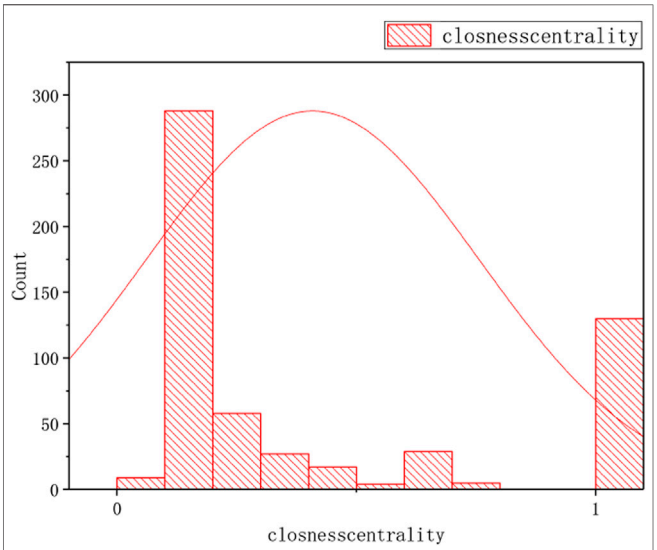


FIGURE 9 | The distribution of closness centrality of the TOA Network: Plot the number of the number of closness centrality on the vertical *Y*-axis against closness centrality on the horizontal *x*-axis.

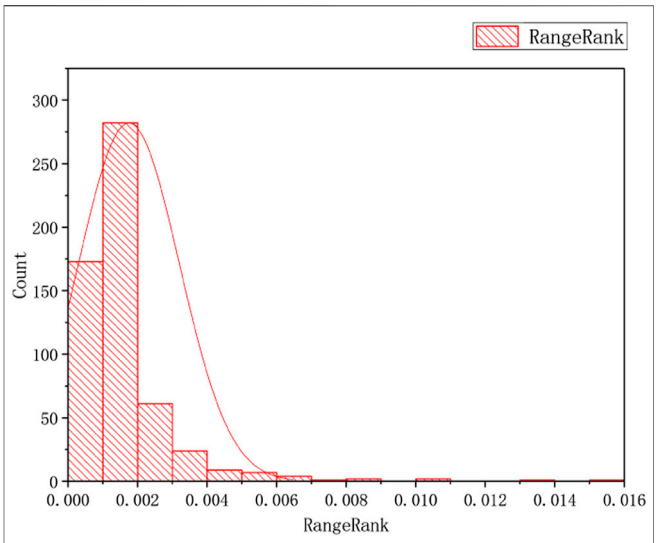


FIGURE 10 | The distribution of RangeRank of the TOA Network: Plot the number of the number of RangeRank on the vertical *Y*-axis against RangeRank on the horizontal *x*-axis.

$$H_i = -K\sum_{j=1}^{n} f_{ij}\ln f_{ij}, i = 1, 2, \ldots, n, \tag{6}$$

where $K = (\ln n)^{-1}, f_{ij} = \frac{r_{ij}}{\sum_{j=1}^{n} r_{ij}}$, *assume that,* $f_{ij} = 0$, $f_{ij}\ \ln f_{ij} = 0$.

Because $0 \le f_{ij} \le 1$, we can get $0 \le -\sum_{j=1}^{n} f_{ij}\ln f_{ij} \le 1$ and $0 \le H_i \le 1$.

Definition 2 In the problem of (m, n) evaluation, the entropy weight of the first evaluation index is defined as

$$w_i = \frac{1 - H_i}{m - \sum_{i=1}^{m} H_i} \tag{7}$$

From the above definition and the properties of the entropy function, the following properties of the entropy weight can be obtained:

Remark 1.

1) When the values of each evaluated object on index i are identical, the maximum value of entropy is 1 and the weight of entropy 0. This also means that the indicator does not provide any useful information to decision makers, and the indicator can be considered to be canceled.
2) When the values of each evaluated object on index i are quite different, the entropy value is small, and the entropy weight is large, it shows that the index provides useful information for decision makers. At the same time, it is pointed out that in this problem, there are obvious differences among the objects in this index, which should be investigated emphatically.
3) The bigger the index's entropy is, the smaller its entropy weight is. The less important the index is, the more satisfied it is.

$$0 < w_i < 1, \sum_{i=1}^{m} = 1, \tag{8}$$

4) Entropy weight, as a weight, has special significance. It is not the actual importance coefficient of an index in decision-making or evaluation, but the relative intensity coefficient of each index in the sense of competition when the value of various evaluation indexes is determined after the set of evaluated objects is given.
5) Considering from the information point of view, it represents the extent to which the index provides useful information in this problem.
6) The Size of the Entropy Weight is Directly Related to the Targets Being Evaluated

Algorithm 1 Calculating entropy weight.

(1) *Case* 1 the Large Value of the indicator, the Better indicator

$$r_{ij} \leftarrow \frac{x_{ij} - x_i^{\min}}{x_i^{\max} - x_i^{\min}} + 1$$

Case 2 the less value of the indicator, the better indicator

$$r_{ij} \leftarrow \frac{x_i^{\max} - x_i}{x_i^{\max} - x_i^{\min}} + 1$$

(2) *For* $i \leftarrow 1$: n *for* $j \leftarrow 1$: m

$$f_{ij} \leftarrow \frac{r_{ij}}{\sum_{i=1}^{m} r_{ij}},$$

end end

(3) *For* $i \leftarrow 1$: n

TABLE 2 | The top-20 ranked Terrorist Organizations base on degree centrality (DC), betweenness centrality (BC), eigenvector centrality (EC), and PageRank (PR) in TOA network.

Rank	DC	BC	EC	RageRank
1	Islamic state of Iraq and the levant (ISIL)	Al-qaida	Islamic state of Iraq and the levant (ISIL)	Islamic state of Iraq and the levant (ISIL)
2	Tehrik-i-Taliban Pakistan (TTP)	Islamic state of Iraq and the levant (ISIL)	Al-Nusrah front	Tehrik-i-Taliban Pakistan (TTP)
3	Al-Nusrah front	Lashkar-e-Jhangvi	Ahrar al-Sham	Lashkar-e-Jhangvi
4	Lashkar-e-Jhangvi	Hamas (Islamic resistance movement)	Free syrian army	Al-Nusrah front
5	Lashkar-e-Taiba (LeT)	United liberation front of Assam (ULFA)	Jaysh al-Islam (Syria)	Lashkar-e-Taiba (LeT)
6	Free syrian army	Ansar al-sharia (Libya)	Tehrik-i-Taliban Pakistan (TTP)	Kurdistan workers' party (PKK)
7	Ahrar al-Sham	Harkatul Jihad-e-Islami	Shamiya front	Al-qaida in the arabian peninsula (AQAP)
8	Al-qaida	Jemaah Islamiya (JI)	Al-Sham legion	United liberation front of Assam (ULFA)
9	Hamas (Islamic resistance movement)	Al-qaida in the Islamic maghreb (AQIM)	Southern front	Al-qaida
10	Al-qaida in the Islamic maghreb (AQIM)	Jaish-e-Mohammad (JeM)	Jaish al-mujahideen (Syria)	Free syrian army
11	Kurdistan workers' party (PKK)	Kurdistan workers' party (PKK)	Hay'at Tahrir al-sham	Garo national liberation army
12	Taliban	Taliban	Lashkar-e-Jhangvi	Taliban
13	Khorasan chapter of the Islamic state	Tehrik-i-Taliban Pakistan (TTP)	Khorasan chapter of the Islamic state	Al-qaida in the Islamic maghreb (AQIM)
14	United liberation front of Assam (ULFA)	Al-Nusrah front	Al-Nasir army (Syria)	Al-Shabaab
15	Hizbul mujahideen (HM)	Lashkar-e-Taiba (LeT)	Taliban	Khorasan chapter of the Islamic state
16	Popular front for the liberation of Palestine (PFLP)	Popular front for the liberation of Palestine (PFLP)	Lashkar-e-Taiba (LeT)	Ahrar al-Sham
17	Bangsamoro Islamic freedom movement (BIFM)	Bangsamoro Islamic freedom movement (BIFM)	Jund al-Aqsa	Hizbul mujahideen (HM)
18	Abu sayyaf group (ASG)	Abdullah Azzam brigades	Abu sayyaf group (ASG)	Hamas (Islamic resistance movement)
19	Al-qaida in the arabian peninsula (AQAP)	Kurdistan freedom hawks (TAK)	Bangsamoro Islamic freedom movement (BIFM)	Bangsamoro Islamic freedom movement (BIFM)
20	Garo national liberation army	Abu sayyaf group (ASG)	Nur-al-Din al-Zinki movement	Abu sayyaf group (ASG)

TABLE 3 | The top-20 ranked Terrorist Organizations.

	Closnesscentrality	Harmonicclosnesscentrality	Betweenesscentrality	Eigencentrality	Pageranks	Degreecentrality	Score	Rank
Islamic state of Iraq and the levant (ISIL)	0.018,499,169	0.02,617,742	0.426,525,361	0.399,236,595	0.289,086,571	0.367,830,974	0.356,905,192	1
Al-qaida	0.020,097,098	0.023,554,514	0.514,773,789	0.078,136,989	0.120,363,525	0.129,822,697	0.293,586,168	2
Lashkar-e-Jhangvi	0.018,754,365	0.023,681,403	0.300,544,024	0.13,940,783	0.196,538,999	0.216,371,161	0.21,420,798	3
Hamas (Islamic resistance movement)	0.019,549,456	0.022,610,798	0.286,553,015	0.080,022,983	0.100,945,885	0.129,822,697	0.180,368,367	4
Al-Nusrah front	0.017,091,169	0.02,288,531	0.106,081,816	0.337,380,474	0.191,149,312	0.28,128,251	0.17,081,277	5
Tehrik-i-Taliban Pakistan (TTP)	0.016,344,832	0.02,275,058	0.123,231,172	0.192,874,792	0.25,290,922	0.292,101,068	0.149,096,878	6
United liberation front of Assam (ULFA)	0.015,128,797	0.018,866,622	0.226,040,032	0.034,797,462	0.123,704,393	0.119,004,139	0.139,322,917	7
Ansar al-sharia (Libya)	0.016,693,467	0.020,895,569	0.210,544,543	0.067,099,295	0.068,940,004	0.075,729,906	0.132,724,445	8
Harkatul Jihad-e-Islami	0.016,834,956	0.020,520,254	0.209,595,612	0.056,055,613	0.068,792,341	0.075,729,906	0.129,633,555	9
Taliban	0.017,555,938	0.021,712,772	0.124,631,219	0.114,142,541	0.109,953,308	0.129,822,697	0.108,712,025	10
Al-qaida in the Islamic maghreb (AQIM)	0.014,944,893	0.018,399,174	0.152,349,996	0.041,950,184	0.109,067,332	0.129,822,697	0.104,964,168	11
Lashkar-e-Taiba (LeT)	0.016,020,379	0.021,086,619	0.094,572,412	0.113,598,381	0.164,053,214	0.183,915,487	0.101,417,827	12
Kurdistan workers' party (PKK)	0.018,302,509	0.021,748,358	0.125,795,344	0.066,760,743	0.155,636,443	0.129,822,697	0.100,306,514	13
Jaish-e-Mohammad (JeM)	0.017,810,194	0.020,816,329	0.137,317,124	0.061,754,316	0.074,606,559	0.086,548,464	0.09,670,911	14
Free syrian army	0.015,344,473	0.020,503,818	0.022,845,596	0.255,806,856	0.114,586,224	0.173,096,929	0.096,085,992	15
Jemaah Islamiya (JI)	0.018,646,057	0.020,398,491	0.15,284,233	0.043,668,898	0.047,621,208	0.05,409,279	0.095,710,347	16
Ahrar al-Sham	0.015,344,473	0.020,427,669	0.015,337,685	0.256,398,524	0.104,729,741	0.162,278,371	0.090,997,043	17
Bangsamoro Islamic freedom movement (BIFM)	0.018,367,541	0.021,592,216	0.089,187,671	0.103,211,842	0.092,178,415	0.108,185,581	0.085,742,084	18
Popular front for the liberation of Palestine (PFLP)	0.018,781,677	0.022,174,791	0.090,760,294	0.093,768,699	0.084,629,162	0.108,185,581	0.084,000,809	19
Khorasan chapter of the Islamic state	0.015,931,401	0.020,317,442	0.068,340,787	0.123,114,186	0.10,618,791	0.129,822,697	0.08,267,129	20

TABLE 4 | Statistical description of the results of traditional methods and our ways.

Group	Observation	Sum	Mean	Variance
Score	567	9.695	0.0179	8.461×10^{-4}
CC	567	231.671	0.409	0.121
HCC	567	250.555	0.442	0.115
BC	567	271,982	479.686	4.01×10^{6}
DC	567	0.008	1.458	3.735×10^{-10}
EC	567	25.121	0.044	0.009
PP	567	0.993	0.002	2.115×10^{-6}

$$H_i \leftarrow -K\sum_{j=1}^{n} f_{ij} \ln f_{ij}$$

end.

(4) *For* $i \leftarrow 1$: n

$$w_i \leftarrow \frac{1 - H_i}{m - \sum_{i=1}^{m} H_i}.$$

end.

4 APPLICATION

In this section, we used a network of Terrorist Organization Alliance Network to demonstrate that the proposed method does a lot better than other centrality approaches when the influential nodes in the network are not entirely determined by a high degree or good robustness. The Terrorist Organization Alliance Network is a network of Terrorist Organizations between 567 organizations with similarities. The Statistical characteristics of the TOA Network are shown in **Table 1**.

According to **Table 1**, we find that the number of nodes in the TOA network is 567, which means that in this network, there are 567 terrorist organizations with a joint attack on the same area. The average degree of this network is 2.399, which means that the number of other organizations joined by each organization is two. The weight average degree of this network is 6.68.

Figure 3 depicts the network topology of the TOA network. As a simple corollary of community funding [23] of our analysis, we found there is a community structure in this network. Thus there is evidence that terrorist organization make attacks with other terrorist organizations. Mining the terrorist organizations for details is very useful. Various community detection algorithms can be applied [24].

As shown in the **Figures 4**, **5**, **6**, the double-logarithmic relationship between cumulative node degree function $P(k)$ and degree k, node weight degree function $P(k)$, and weight degree k is described. First of all, we used the power law fitting for the degree distribution and weight distribution and found that all R2 are greater than 0.9, which means that the TOA network is a scale-free network. Thus the TOA network is a social network. And in the TOA network, we find that there are a large number of nodes with degree 1. According to analysis of the source data, most of the organizations carry out the attack once or the event is only between the two organizations creating terrorist attacks. However, such organizations are not rare. It is very likely that these organizations are temporarily organized to launch an attack and then disband or change their names.

As shown in the **Figures 7–8**, we find that the distribution of the eigencentrality of the TOA network is a power law.

The **Figure 9** to **Figure 10** display the distribution of CC, BC, EC, and RR, respectively. The distribution of the BC, EC, and RangeRank are the same; however, the distribution of CC is different from the three, which is very interesting.

The top-20 ranked groups by betweenness centrality (BC), degree centrality (DC), eigenvector centrality (EC), and PageRank (PR) in TOA network as shown in the **Table 2**.

Table 2 compares four identity nodes finding ways in the TOA network and gives the top 20 critical terrorist organizations in different ways. The terrorist organization ISIL always occupies the top one in the DC, EC, and RR ways, and in the BC, the ISLT is the second critical organization. Due to the closeness of the 130 Terrorist Organizations (centrality of 1), it is not possible to display these Terrorist Organizations. Comparison of the proposed method ranks ISIL, TTP, AI-Musrah Front, Lashkar-e-Jhangvi, LeT, BIFM, and ASG all as being in the top 20; others are not all in the top 20, as some are in three methods like the Taliban only note in the RangRank and PKK only in the DC and BC, some are in two methods like the Free Syrian Army in the DC and RangRank, some are in one method like the Garo National Liberation Army only in the DC.

In this part, we compare the traditional methods with the results obtain in the DC, BC, CC, EC, and RR. At the same time, from **Table 2** and **Table 3**, we can find the results of these traditional ways are different; thus, a global way to measure the key nodes is very necessary.

TABLE 5 | The *p*-value of these ways.

	CC	HC	BC	EC	Pageranks	DC	Ourway
CC	1	3.93E-01	9.70E-26	7.19E-13	8.31E-01	2.47E-04	2.41E-18
HC	3.93E-01	1	1.52E-29	1.36E-15	5.21E-01	6.72E-06	1.72E-22
BC	9.70E-26	1.52E-29	1	7.97E-04	1.14E-26	4.92E-12	5.46E-04
EC	7.19E-137	1.36E-15	7.97E-04	1	1.59E-13	3.73E-04	7.70E-01
Pageranks	8.31E-01	5.21E-01	1.14E-26	1.59E-13	1	1.07E-04	2.46E-19
DC	2.47E-04	6.72E-06	4.92E-12	3.73E-04	1.07E-04	1	1.19E-05
Ourway	2.41E-18	1.72E-22	5.46E-04	7.70E-01	2.46E-19	1.19E-05	1

TABLE 6 | T-test results of these ways.

	CC	HC	BC	EC	Pageranks	DC	Ourway
CC	0	0	1	1	0	1	1
HC	0	0	1	1	0	1	1
BC	1	1	0	1	1	1	1
EC	1	1	1	0	1	1	0
Pageranks	0	0	1	1	0	1	1
DC	1	1	1	1	1	0	1
Ourway	1	1	1	0	1	1	0

5 SIGNIFICANT ANALYSIS

In this section, we used the two-tail test to find the significant among these ways. The t-text is shown below.

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_{X_1}^2 + \sigma_{X_2}^2 - 2\gamma\sigma_{X_1}\sigma_{X_2}}{n-1}}} \tag{9}$$

Where $\bar{X}_1$ and $\bar{X}_2$ are the mean of the two samples, respectively; $\sigma_{X_1}^2$ and $\sigma_{X_2}^2$ are the variance of the two samples, respectively.

As shown in **Tables 3**, **4** and **Table 6**, $H = 0$ indicates that the zero hypothesis is not rejected under 5% confidence; $H = 1$ indicates that the zero hypothesis is rejected, that is, that there is discrimination.

6 CONCLUSION

At present, the research on terrorist attacks is mainly based on multi-agents, such as Refs. reference [25, 26]. These two articles analyze terrorist organizations through multi-agent simulation and study their change rules. In this paper, the complex network method is used to find the key terrorist organizations. The entropy method is used to measure the harm of terrorist attack organizations and find the key terrorist organizations. Compared with other models, it is relatively novel.

In this paper, we sort the terrorist organizations by using the calculation model of the key nodes in the complex network and find the key terrorist organizations. Through the previous calculation, we find that the terrorist attacks made by terrorist organizations are very harmful, and their concentration will obviously concentrate some weapon resources together, and attack at different locations at the same time, These researches seriously endanger the security of today's society. The purpose of this paper is to better prepare for the fight against terrorist organizations and the maintenance of world peace. And we observe some characteristics of the TOA network. Based on the traditional methods, a new method is given to find the key nodes as soon as critical organizations. We find that there are significant differences among these traditional methods; comparing these ways, we found that there is no significant difference among CC, HC, and Pageranks, and there is no significant difference between EC and our way. There are

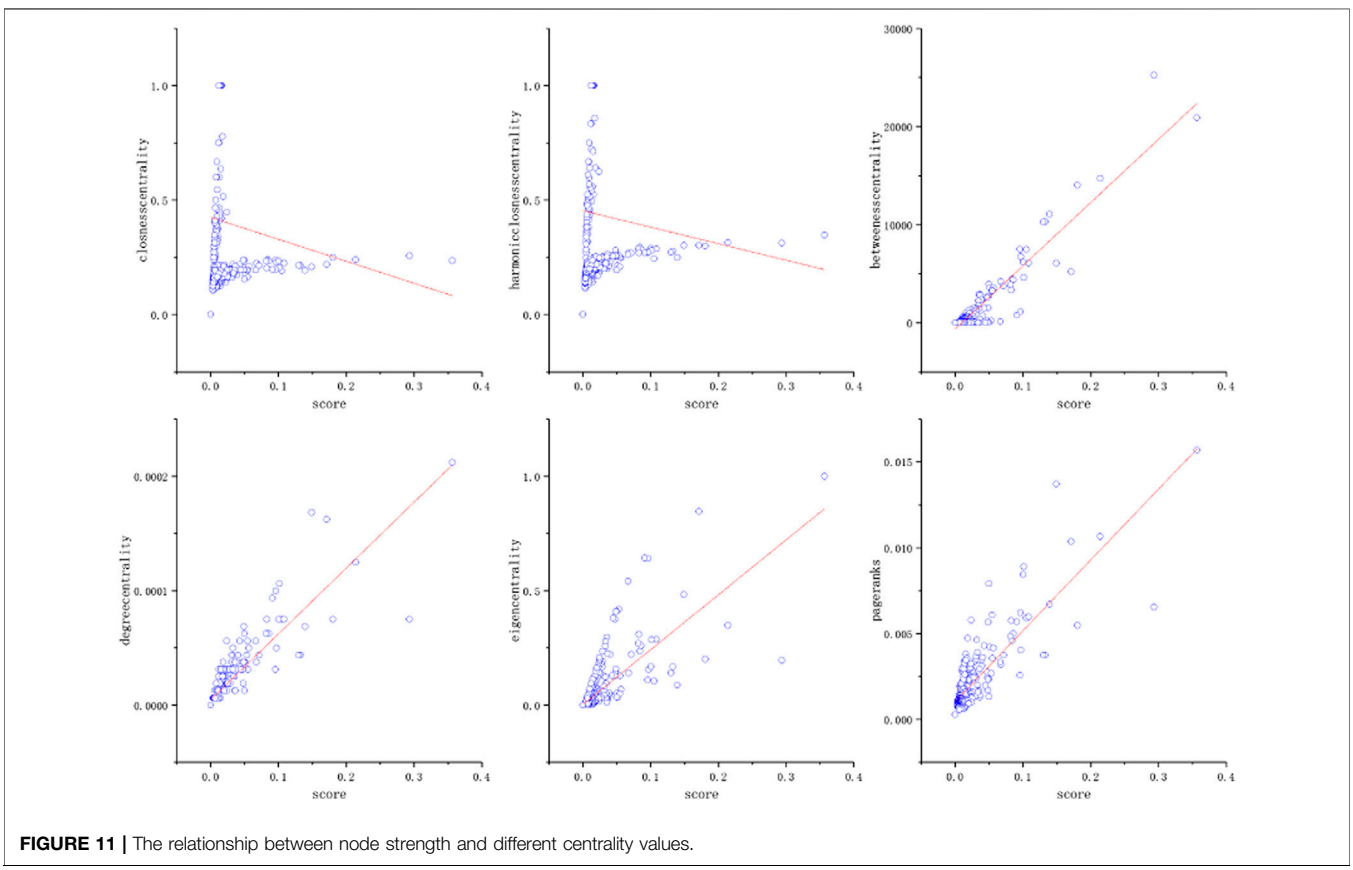


FIGURE 11 | The relationship between node strength and different centrality values.

significant differences between other indicators. Therefore, through t-tests and **Figure 11**, we found that there are differences between the integrated score and each centrality, which also shows that the information obtained by traditional methods is local information and cannot fully reflect the importance of nodes. Therefore, the weighted method can better integrate all the information, and we can get the more accurate and important nodes.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: https://www.start.umd.edu/gtd/.

AUTHOR CONTRIBUTIONS

JH: Visualization, Software, Computation, Drawing and Writing CC: Writing-Reviewing and Editing LX: Conceptualization, PW: Investigation, Visualization, Software HL: Methodology, Validation.

FUNDING

This work is supported by the National Natural Science Foundation of China (Nos. 71871233, 71701049, and 717871159), the Fujian Science and Technology Economic Integration Service Platform, Fundamental Research Funds for the Central Universities of China (Nos.2020XD-A01-1).

REFERENCES

1. Balcells L, and Torrats-Espinosa G, Using a Natural experiment to Estimate the Electoral Consequences of Terrorist Attacks, *Proc Natl Acad Sci USA* 115 (2018) 10624–9. doi:10.1073/pnas.1800302115
2. Mitts T. Terrorism and the Rise of Right-Wing Content in Israeli Books. *Int Org* (2019) 73:203–24. doi:10.1017/s0020818318000383
3. Xue A, Wang W, and Zhang M. "Terrorist Organization Behavior Prediction Algorithm Based on Context Subspace [C]" in International Conference on Advanced Data Mining and Applications.. Berlin, Heidelberg: Springer, 7121 (2011) 332–45. doi:10.1007/978-3-642-25856-5_25
4. Nurudeen M, Zou B, and Zhu C, A Neuro-Fuzzy Crime Prediction Model Based on Video Analysis[J], *Chin J Electro*, 27 (2018) 968–75. doi:10.1049/cje.2018.02.019
5. Li Z, Sun D, Li B, Li Z, and Li A, Terrorist Group Behavior Prediction by Wavelet Transform-Based Pattern Recognition, *Discrete Dyn Nat Soc* 2018, (2018) 1–16. doi:10.1155/2018/5676712
6. Carley K, Lee J-S, and Krackhardt D. Destabilizing Networks[J]. *Connections* (2002) 24:79–92.
7. Li G, Hu J, Song Y, Yang Y, and Li H-J, Analysis of the Terrorist Organization Alliance Network Based on Complex Network Theory, *IEEE Access* 7 (2019) 103854–62. doi:10.1109/access.2019.2929798
8. Fang L, Fang H, Tian Y, Yang T, and Zhao J. The alliance Relationship Analysis of International Terrorist Organizations with Link Prediction. *Physica A: Stat Mech its Appl* (2017) 482:573–84. doi:10.1016/j.physa.2017.04.068
9. Hakim MA, and Mujahidah DR, Social Context, Interpersonal Network, and Identity Dynamics: A Social Psychological Case Study of Terrorist Recidivism, *Asian J Soc Psychol* 23 (2020) 3–14.doi:10.1111/ajsp.12349
10. Li S, Zhao D, Wu X, Tian Z, Li A, and Wang Z. Functional Immunization of Networks Based on Message Passing. *Appl Math Comput* (2020) 366:124728. doi:10.1016/j.amc.2019.124728
11. Zhu P, Han J, Guo Y, and Lombardi F, Reliability and Criticality Analysis of Communication Networks by Stochastic Computation, *IEEE Netw* 30 (2016) 70–6. doi:10.1109/mnet.2016.1500221nm
12. West DB. *Introduction to Graph Theory[M]*. Upper Saddle Rive: Prentice hall (2001) 2.
13. Gao C, Fan Y, Jiang S, Deng Y, and Liu J, Dynamic Robustness Analysis of a Two-Layer Rail Transit Network Model, *IEEE Trans Intell Transport Syst* (2021) 1–16. doi:10.1109/tits.2021.3058185
14. Washio T, and Motoda H, State of the Art of Graph-Based Data Mining. *SIGKDD Explor Newsl* 5 (2003) 59–68. doi:10.1145/959242.959249
15. Gao C, Su Z, Liu J, and Kurths J, Even central Users Do Not Always Drive Information Diffusion, *Commun ACM* 62 (2019) 61–7. doi:10.1145/3224203
16. Zhu P, Guo H, Zhang H, Han Y, and Chu C, The Role of Punishment in the Spatial Public Goods Game[J], *Nonlinear Dyn* 102 (2020) 1–10. doi:10.1007/s11071-020-05965-0
17. Zhu P, Hou X, Guo Y, Xu J, and Liu J. Investigating the Effects of Updating Rules on Cooperation by Incorporating Interactive Diversity[J]. *The Eur Phys J B* (2020) 94:58. doi:10.1140/epjb/s10051-021-00059-1
18. Pappi FU, and Scott J. Social Network Analysis: A Handbook. *Contemp Sociol* (1993) 22:128. doi:10.2307/2075047
19. Du Y, Gao C, Hu Y, Mahadevan S, and Deng Y. A New Method of Identifying Influential Nodes in Complex Networks Based on TOPSIS. *Physica A* (2014) 399:57C69. doi:10.1016/j.physa.2013.12.031
20. Bialynicki-Birula I, and Mycielski J, Uncertainty Relations for Information Entropy in Wave Mechanics[J], *Commun Math Phys* 44 (1975) 129.
21. Campbell J. *Grammatical Man: Information, Entropy, Language, and life[M]*. New York: Simon & Schuster (1982).
22. Devroye L, Gyrfi L, and Lugosi G. *A probabilistic Theory of Pattern Recognition*. Berlin, Heidelberg: Springer Science and Business Media. (1996) 31.
23. Li H-J, Bu Z, Wang Z, and Cao J, Dynamical Clustering in Electronic Commerce Systems via Optimization and Leadership Expansion, *IEEE Trans Ind Inf*, 16 (2020) 5327–34. doi:10.1109/tii.2019.2960835
24. Zhu P, Wang X, Zhi Q, Ma J, and Guo Y. Analysis of Epidemic Spreading Process in Multi-Communities. *Chaos, Solitons Fractals* (2018) 109:231–7. doi:10.1016/j.chaos.2018.02.007
25. Lu P, Yang H, Li H, Li M, and Zhang Z, Swarm Intelligence, Social Force and Multi-Agent Modeling of Heroic Altruism Behaviors under Collective Risks[J], *Knowledge-Based Syst* 214 (2021) 106–725. doi:10.1016/j.knosys.2020.106725
26. Moya I, Chica M, Sáez-Lozano JL, and Cordón Ó, An Agent-Based Model for Understanding the Influence of the 11-M Terrorist Attacks on the 2004 Spanish Elections, *Knowledge-Based Syst* 123 (2017) 200–16. doi:10.1016/j.knosys.2017.02.015

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

frontiers in Physics

ORIGINAL RESEARCH
published: 20 August 2021
doi: 10.3389/fphy.2021.739202

Information Cascades Prediction With Graph Attention

*Zhihao Chen [1,2,3], Jingjing Wei [4], Shaobin Liang [1,2,3], Tiecheng Cai [1,2,3] and Xiangwen Liao [1,2,3,5]**

[1]School of Computer and Data Science, Fuzhou University, Fuzhou, China, [2]Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing, Fuzhou University, Fuzhou, China, [3]Digital Fujian Institute of Financial Big Data, Fuzhou University, Fuzhou, China, [4]College of Electronics and Information Science, Fujian Jiangxia University, Fuzhou, China, [5]Peng Cheng Laboratory, Shenzhen, China

The cascades prediction aims to predict the possible information diffusion path in the future based on cascades of the social network. Recently, the existing researches based on deep learning have achieved remarkable results, which indicates the great potential to support cascade prediction task. However, most prior arts only considered either cascade features or user relationship network to predict cascade, which leads to the performance limitation because of the lack of unified modeling for the potential relationship between them. To that end, in this paper, we propose a recurrent neural network model with graph attention mechanism, which constructs a seq2seq framework to learn the spatial-temporal cascade features. Specifically, for user spatial feature, we learn potential relationship among users based on social network through graph attention network. Then, for temporal feature, a recurrent neural network is built to learn their structural context in several different time intervals based on timestamp with a time-decay attention. Finally, we predict the next user with the latest cascade representation which obtained by above method. Experimental results on two real-world datasets show that our model achieves better performance than the baselines on the both evaluation metrics of HITS and mean average precision.

Keywords: social network, information diffusion, cascade prediction, deep learning, graph attention

OPEN ACCESS

Edited by:
Daihai He,
Hong Kong Polytechnic University, SAR China

Reviewed by:
Kam-Fai Wong,
The Chinese University of Hong Kong, China
Liang Y. Ang,
Dalian University of Technology, China

****Correspondence:***
Xiangwen Liao
liaoxw@fzu.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *10 July 2021*
Accepted: *05 August 2021*
Published: *20 August 2021*

Citation:
Chen Z, Wei J, Liang S, Cai T and Liao X (2021) Information Cascades Prediction With Graph Attention. Front. Phys. 9:739202. doi: 10.3389/fphy.2021.739202

1 INTRODUCTION

Social media online platforms, such as Twitter, Sina Weibo, WeChat, and so on, have greatly promoted the rapid spread of information on the Internet, which leads to an increasingly important impact on daily life of people. Cascade [1] consists of a series of users' behaviors on the social network like share, comment, like and so on, which is regarded as a temporal sequence as shown in **Figure 1**. Who participated in the diffusion we call them infected users. Cascade is usually considered as the basis of information diffusion on social networks. Modeling and predicting cascades is conducive to understanding and quantifying user influence on social network. Cascade prediction aims to predict the process of information diffusion in the future based on observed cascades, which is of great significance to decision-making on social networks such as viral marketing [2] and support for Internet of Things [3].

Traditionally, the existing works on cascade prediction mainly learn the cascade feature from the diffusion path or user relationship graph, which can be summarized into the following two categories: 1) Analysis methods based on traditional topological structure and characteristics. In the early stage, these methods were usually based on the network topology [4] and the network propagation mechanism [5, 6, 7] to model the cascading diffusion process; In order to further

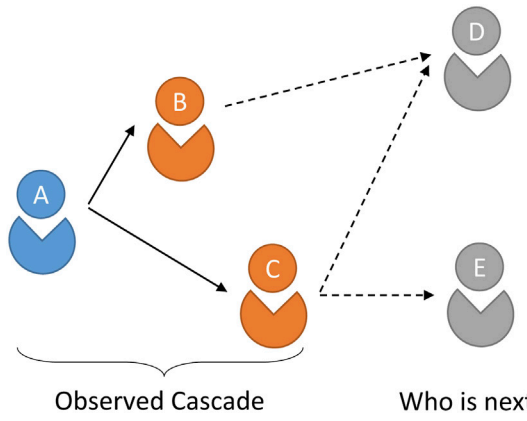


FIGURE 1 | A toy example of cascade prediction, which contains several users who participated in information diffusion. A → B → C is a observed cascade in social network, we will predict who is the next infected user, D or E?

improve the predictive performance of the model, a series of feature-driven methods [8] such as learning user influence and susceptibility [9] have been proposed. 2) Method based on deep learning. With the successful application of end-to-end learning models such as DeepCas [10], various neural network models applied to cascade prediction have further improves the model performance. It has gradually become the main method on cascade diffusion prediction task. In recent works, Topo-LSTM [11] and DeepInf [12] modeled the network topology and predicted propagation through user-level representation learning. On the other hand, some researchers tried to use neural networks to learn the temporal feature in the cascade, and thus DeepHawkes [13], RNN-based CRPP method [14] and some other models were proposed.

However, traditional methods based on propagation mechanism [5] and features [9] often rely on manual definition through a large number of studies and observations. What's more, the complex manual features [8] often limit the generalization and robustness. As for deep learning methods, existing works either study from the relationship of users from network topology [11] or cascade temporal feature [13] unilaterally. While in realistic social network, information diffusion is affected by spatial and temporal feature together. It is of great significance to study the potential relationship between the two aspects in information diffusion.

To solve this problems, in this paper, we propose an end-to-end neural network framework DLIC (Deep Leaning Information Cascades) that combines social network topology and temporal cascade information. The framework first learns the feature representations of each user through a graph attention based on the topological of social network. And then a recurrent neural network is built for learning cascade representation in different time intervals, and a time-decay attention mechanism is introduced for assigning different weights to them. The final user representation which learns previous cascade feature would be fed into decoder layer to predict the next infected user.

Extensive experiments on two public cascade diffusion datasets, namely Twitter [15] and Douban [16], validate the performance of our model compared with several baseline methods. The results indicate the improvements on HITS@100 by 2.2 and 2% on the twitter and douban datasets, respectively. Besides, our model performs best in all other metrics on both real datasets.

The paper is organized as follows: **Section 2** introduce the relative works in cascade prediction task, while **Section 3** describes our DLIC model. **Section 4** shows our experimental results and discusses the effect of different features. Finally, **Section 5** gives a conclusion of our main findings and future works.

2 BACKGROUND

The goal of cascade prediction is to model the diffusion regularity in social network, which aims to effectively describe the propagation mechanism of information and predict the future diffusion path. The research on cascade prediction can be divided into tow categories, namely the methods based on *traditional machine learning* or *deep learning techniques*.

In terms of cascade feature learning, the most of works based on traditional machine learning have proposed many methods to learn the diffusion probability among different users from the observed cascade information. [8] modeled cascade information through a marked Hawkes self-exciting point process and predicted with content virality, memory decay and user influence. [17] learned the embedded feature representations of users on social networks in a latent space through independent cascade model. [18] proposed IEDP model based on information-dependent embedding, which mapped users to a latent embedding space in observed time sequence of the cascade diffusion process, and the prediction is made according to the distance of embedding representation. [19] proposed an opinion leader mining model EIC based on the extended independent cascade model, which integrated network structure characteristics, individual attributes and behavior characteristics together. [20] designed a route decision model by a data-driven method. [21] constructed interaction rules based on multi-dimensional features such as user influence, sentiment and age to simulate the process of information diffusion in social networks. [22] used the survival analysis model to learn the susceptibility and influence of users, which were used to calculate the diffusion probability among users. [23] proposed a feature extraction method from user behavior under urban big data. [24] argued that the spread of rumors is composed of multiple factors and proposed a multi-featured spread model. [25] computed the epidemic risk of COVID-19 by combining the number of infected persons and the way they pass through the station.

In recent years, with the rise of representation learning in deep learning methods, more and more deep learning models such as LSTM, RNN, GCN, etc. have also been used in the work of cascade prediction. The DeepHawkes model proposed by [13] used end-to-end deep learning to simulate the explainable factors

of the Hawkes process and modeled cascade information. [26] proposed an attention-based RNN to capture the cross-dependence in the cascade and a coverage strategy to overcome the misallocation of attention caused by the memoryless of the traditional attention mechanism. [10] also proposed an end-to-end model to learn the cascade graph, which automatically learned the representation of a single cascade from the global network structure without manual features. [11] introduced a new data model named diffusion topologies and proposed a novel topological recurrent neural network Topo-LSTM. DeepInf proposed by [12] took the local area network among users as input, and learned their potential influence in social network through graph convolutional networks. [27] proposed a sequential information diffusion neural network with structure attention that considers the process of information diffusion and the structural characteristics of the user graph through a recurrent neural network. [28] also proposed an attention network to solve the diffusion prediction problem, which can effectively explore the implicit diffusion dependence among information cascade users. [14] proposed competing recurrent point process on RNN network, which models both the diffusion process and the competition process. [29] proposed a multi-scale diffusion prediction model based on reinforcement learning, which integrates the macroscopic information into the RNN-based microscopic spread model for predicting infected users. [30] proposed to perform multi-task joint learning framework to understand user relationships and predict cascades with graph attention networks and recurrent neural networks. [31] estimated traffic time from trajectory of taxi in different fine-grained time intervals based on deep learning. [32] designed a RNN model with a multi-relational structure, which not only captures the traditional time dependence, but also captures the explicit multi-relational topological dependence through a hierarchical attention mechanism.

In particular, the spatial-temporal feature learning methods that using the Graph Network and RNNs achieve remarkable results. They also belong to deep learning methods. [33] and [34] aimed to predict objective trajectories. They constructed graph based on spatial coordinate and learned the subsequent positional information with RNNs. Moreover, soft attention and self attention are used to enhance representation learning, separately. [35] proposed to a social recommendation via a dynamic graph method. They encoded the long-short term preferences for users in a session based on RNN. And then they learn the dynamic graph features for user and his friends through graph attention, which are used for recommendation. On the basis of above-mentioned work, we proposed to learn the user spatial features with graph attention and then encode cascade temporal features with RNN.

In summary, the methods based on deep learning which avoid the defects of feature engineering have gradually become the major technique in cascade prediction task, but most of previous research only focused on the representation of cascade. The lack of unified modeling about user structure and temporal feature is still a key problem to be solved.

3 METHODS

In this section, we start with formalization of the cascade diffusion prediction problem. Then we introduce the framework of our model, which learns the structural context among users through graph attention and then integrates the temporal feature into cascade representation by time decay effect. Finally, we present the overall algorithm and details.

3.1 Problem Formalization

Cascade is a behavior of information adoption by people in a social network. To formalize our problem, we first introduce some terminologies. A user who shares information in social network is called infected user. Given users set $U = \{u_1, u_2, \ldots, u_N\}$, cascades set $C = \{c_1, c_2, \ldots, c_M\}$, where N and M are the number of users and cascades respectively. A cascade $c_i = \{(u_1, t_1), (u_2, t_2), \ldots, (u_{|c_i|}, t_{|c_i|}), t_1 \leq t_2 \leq \ldots \leq t_{|c_i|}\}$ is a sequence of infected user and timestamp in a diffusion process, where $|c_i|$ is the number of infected user, t_i is the infected timestamp of u_i. The relationship among users can be represented by $G = \{U, E\}$, where $E = [e_{ij}] \in R^{|U|\times|U|}$ is the adjacency matrix of social graph. $e_{ij} = 1$ implies that there is an edge between user u_i and u_j in the social graph.

Cascade prediction can be divided into macro level and micro level. Macroscopic diffusion prediction aims to predict the final cascade scale. The purpose of this paper is microscopic diffusion prediction, whose aim is to predict who is the next infected user u_{i+1} based on social graph G and cascades set C before time t_i.

3.2 Overview of Technical Framework

In order to illustrate how to capture the potential spatial-temporal information in cascade. We introduce the proposed DLIC model. As shown in **Figure 2**, the framework of DLIC model takes the social network and cascade as input and outputs the next infected user one by one.

The main part of the DLIC model consists of four components: 1) **User embedding layer**: learning user relationship based on social graph to obtain their representation through graph attention, which reflects the different influence of users. 2) **Cascade encoding layer**: feeding the embedding representation according to the order of observed cascade to encode cascade path through recurrent neural network. 3) **Time-decay attention**: the cascade representation would be further extracted through assigning different attention weights based on timestamp slice. 4) **Decoding and Output layer**: The last hidden state of encoding layer is the representation of this cascade. It would be took as input into decoding layer to predict the next infected user and output them one by one. Next, we give a detailed introduction to these components.

3.3 User Embedding

Social network refers to the relationship graph among users. The behaviors such as follow, like, reply and forward forms the topology of social network. This structure affects and promotes the information diffusion. Therefore, it makes sense that learn the feature of users in the social

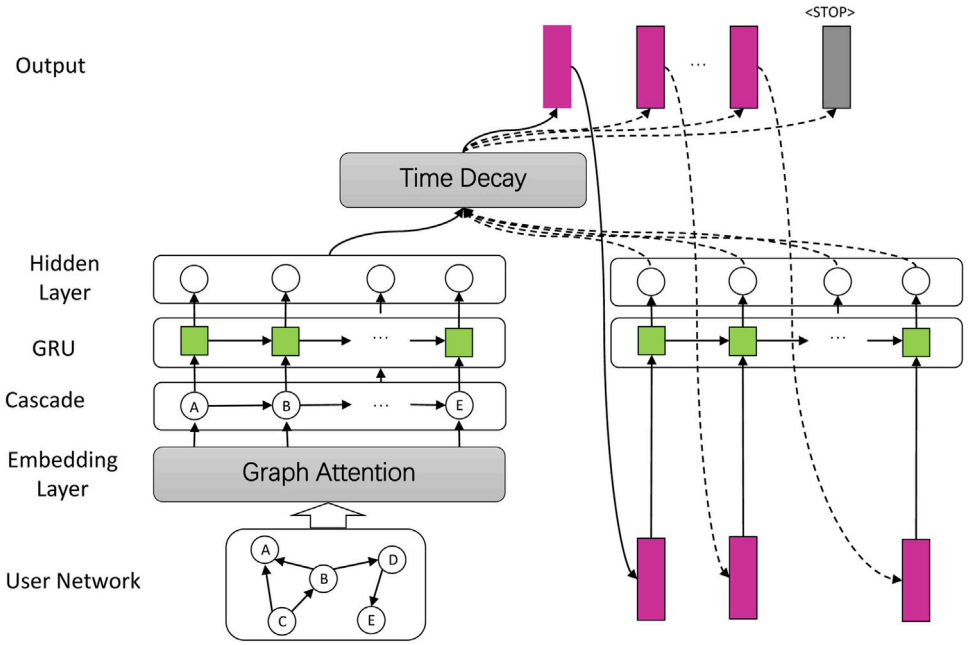


FIGURE 2 | Illustration of proposed model DLIC. Given user network and a series of cascades. Our model learns their feature separately and predicts the following infected users.

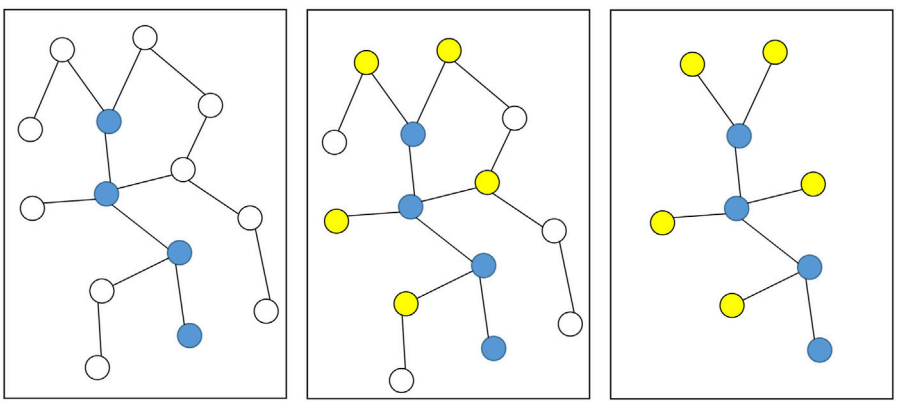


FIGURE 3 | An example of users sampling method. Blue nodes is observed users. Yellow nodes is what we select.

network. However, The user graph in the social network is usually very huge and complicated. If we learn the features from all users directly, it will not only take up a lot of hardware resources and time, but also may cause model performance degradation due to some noise nodes such as paid posters. So we propose a user sampling method. As **Figure 3** shows, for the K observed users in cascade, we randomly selected several their neighbor nodes shown as yellow nodes. The nodes with higher degree would be selected more easier. And the others would be discarded show as white. We can obtain a subgraph for each cascade and finally integrate them as the new user graph G.

For the obtained subgraph through above methods, we feed the adjacency matrix into a multi-layer graph attention network [36] with multi-heads to learn the user representations. Specifically, we assume vector sets $h = \{h_1, h_2, \ldots, h_N\}$, $h_i \in R^F$ as features of all users, where N is the number of users, F is the number of feature dimension. And then we apply a linear transformation in h as the **Eq. 1** shows:

$$v_i = Wh_i \tag{1}$$

Where $W \in R^{F' \times F}$ is an independent trainable weight matrix. For a pair of neighbor nodes i and j, i.e., $e_{ij} = 1$, we learn the attention weight z_{ij} between them. Firstly, for each neighbor j of user i, we apply a linear transformation again after a concatenation operation for their feature vector v_i and v_j to obtain the attention coefficients $c_{ij} = a\ (Wv_i \| Wv_j)$, where $\|$ is the concatenation operation, $a \in R^{(F' \times F')}$ is a trainable matrix. c_{ij}

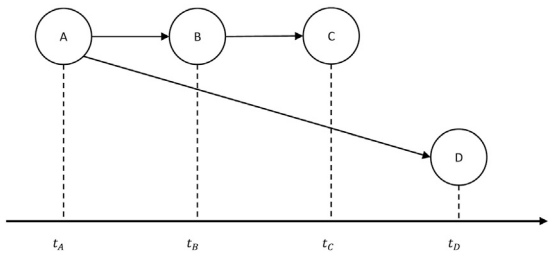


FIGURE 4 | An illustrative example of structural context in cascade.

represents the importance of node j relative to node i. And then it is activated with LeakyRelu function. Finally, we obtain their neighbor attention weight for each node by softmax function. The process as **Eq. 2** shows:

$$z_{ij} = \frac{exp(LeakyRelu(c_{ij}))}{\sum_{k\in\mathcal{N}(i)} exp(LeakyRelu(c_{ik}))} \tag{2}$$

Where $\mathcal{N}(i)$ is neighbor set of user i. Then user feature representation will be updated according to above attention weight of neighbors. Specifically, we obtain the new hidden representation h_i through a weighted sum operation based on above attention coefficients as **Eq. 3** shows:

$$h_i = \sigma\left(\sum_{j\in\mathcal{N}(i)} z_{ij} v_i\right) \tag{3}$$

Where σ is a nonlinear activation function, i.e., $RELU(\cdot)$. Finally, we adopt the multi-head attention to stabilize the process of user feature learning. Each head attention executes the transformation of **Eq. 3** independently and then concatenate them to obtain feature presentation, which contains different attention of user neighborhood. The result h_i^{concat} would be regarded as the input on next GAT layer and it is computed by **Eq. 4**:

$$h_i^{concat} = \|_{e=1}^{E} h_i^e \tag{4}$$

Where $\|$ is concatenation operation, E is the number of heads, h_e^i is the ouput of each head attention. In final layer of GAT, the sensitivity of splicing operation is reduced. Therefore, the output of user embedding presentation would be calculated by average pooling operation with **Eq. 5** to extract the feature among all attention heads and this operation can also save the memory.

$$h_i^{user} = \frac{1}{E}\sum_{e=1}^{E} h_i^e \tag{5}$$

3.4 Cascade Encoding Layer

A user who participated in cascade diffusion is not only affected by the latest infected user, but also influenced by previous users. As shown in **Figure 4**, we construct a cascade path A → B → C → D which ordered by the timestamps of infected users. We can see that B is affected by A and C is affected by B which looks like a chain according to the relationship among them. Though C is the latest infected user, A may have greater influence so that D still receives message from A and become the next infected user. It means each user in cascade may affect users in the whole diffusion process from start to finish. However, a cascade does not record the message forwarding source of users. The long distance dependence of cascade feature is a problem that needs to be solved.

RNN has shown its effectiveness in many fields, which provides a theoretical support for the learning of cascade sequences. For a given cascade sequence $c = \{(u_1, t_1), (u_2, u_2), \ldots, (u_{|c_i|}, t_{|c_i|})\}$ which is composed of user u_i and timestamp t_i together, we encode them, respectively, to learn for the later prediction. We adopt Gated Recurrent Unit (GRU) to learn the cascade sequential information based on users in our model. According to the order of cascade, the embedded presentation h_i^{user} of ith user would be feed into GRU cell to obtain the hidden state $h_i^s = GRU(h_i^{s-1}, h_i^{user})$ one by one. Where h_i^{s-1} is the hidden state of previous user, $s = 1, 2, \ldots, |c_i|$ is the step of recurrent neural network. GRU is mainly composed of reset gate and update gate. They are calculate as follow:

The reset gate r_i is calculated as **Eq. 6** shows:

$$r_i = \sigma(W_r h_i^{user} + U_r h_{i-1} + b_r) \tag{6}$$

Where $\sigma(\cdot)$ is sigmoid activation function, $W_r \in R^{H\times F}$, $U_r \in R^{H\times H}$ and $b_r \in R^H$ are independent trainable parameters.

The update gate v_i is calculated as **Eq. 7** shows:

$$v_i = \sigma(W_v h_i^{user} + U_v h_{i-1} + b_v) \tag{7}$$

Similarly, where $W_v \in R^{H\times F}$, $U_v \in R^{H\times H}$ and $b_v \in R^H$ are also trainable parameters.

Then GRU uses reset gate r_i to remember the current hidden state $\hat{h}_i$ as **Eq. 8** shows:

$$\hat{h}_i = tanh(W_h h_i^{user} + U_h(r_t \odot h_{i-1}) + b_h) \tag{8}$$

Where $\odot$ represents hadamard product, $W_h \in R^{H\times F}$, $U_h \in R^{H\times H}$ and $b_h \in R^H$.

Finally, update gate z_i activates the actual hidden state as **Eq. 9** shows:

$$h_i^s = v_i \odot h_{i-1} + (1 - v_i) \odot \hat{h}_i \tag{9}$$

3.5 Time Decay Attention

In social network, the influence of previous message usually decays with time passing because of timeliness. Users are usually more sensitive to the latest messages. The information that is closer to the user's infection time usually has a greater impact on users. However, the time function of traditional methods based on artificial definition generally can not describe this effect exactly and is hard to decide which one should be used.

In order to learn the influence of time on cascades, the following time decay attention mechanism is employed to learn the weight coefficient of current user to the previously infected users. Firstly, we divide the maximum observed time $t_{|c_i|}$ into k equal-sized intervals $\{t_0 = 0, t_1, t_1, t_2, \ldots, [t_{|c_i|-1}, t_{|c_i|}\}$. The mapping function from continuous time to each interval is showed as **Eq. 10**:

$$f(T - t_i) = k, t_{k-1} \le t_i < t_k \tag{10}$$

Where t_i is the infected time of user i in cascade c. We define a parameter $\lambda_{f(T-t)}$ for each interval as the time decay weight. We can get the final hidden state h'_i as the presentation of cascade c which is assembled by a weighted average pooling mechanism as showed in **Eq. 11**:

$$h'_i = \frac{\sum_{j=1}^{i} \lambda_{f(T-t)} h_j^s}{i} \tag{11}$$

3.6 Decoding and Output Layer

In order to predict the subsequent infected user, we feed the presentation $h'_{|c_i|}$learned by encoding layer into GRU cell in decoding layer. The output of GRU cell is the predicted infected user and it would be feed into next GRU cell as input to continue predicting. To identify when to stop predicting, we append a tag < EOS > in the end of cascade when training. The decoding layer will continuously output the predicted infected users until the output is < EOS >. It means the cascade stop propagating. The calculation process of infection probability for each user is showed as **Eq. 12**:

$$p_i = softmax(W_p h'_i + b_p) \tag{12}$$

Where $p_i \in R^{|U|}$ is the infection probability of user i in next propagation, W_p, b_p is the trainable weight matrix and bias, respectively. The training objective function to maximize the log-likelihood of all cascades is defined as **Eq. 13**:

$$L(\Theta) = \sum_{c=1}^{|C|} \sum_{i=1}^{|c|-1} log\, p_i^c[u_{i+1}^c] \tag{13}$$

Where $p_i^c[j]$ is the infection probability of the user i to the user j in cascade c. Θ is all the trainable parameters in training model. The whole calculation process of our model is as shown in **Algorithm 1**.

4 EXPERIMENTS

In this section, we compare the prediction performance of the proposed DLIC model with baselines and present the empirical evaluations to demonstrate the effectiveness of our model. Moreover, we perform detailed analysis to understand the role of each component in DLIC.

TABLE 1 | Statistics of datasets.

Datasets	Users	Links	Cascades	Avg. Length
Twitter	12,627	309,631	3,442	32.60
Douban	23,123	348,280	10,602	27.14

4.1 Datasets

In this paper, we verify the performance of our model on two public datasets. The datasets are split into training, validation and test set for 80, 10, and 10%, respectively. **Table 1** shows the statistics of datasets.

Twitter [15] dataset records retweets URL among users on Twitter during October 2010. The cascade consists of all the users who retweeted are sorted according to the time. There are 3,442 cascades which contains 12,627 users in this dataset.

Users on **Douban** [16] can comment on books they have read. Users' comments on a book at a certain time can be regarded as infected. The diffusion process of a book is regarded as a cascade. There are 10,602 cascades which contains 23,123 users in this dataset.

4.2 Evaluation Metric

The purpose of cascade prediction is to predict the next infected user based on given observed users. In order to simplify the task and make it easy to evaluate, we regard it as a retrieval task that detect k infected users in the remaining users. Therefore, we first rank the uninfected nodes according to the predicted infection probability, and then evaluate the Top-k infected users according to k = 10, 50, 100, respectively. The evaluation metrics are mean average precision (MAP) and HITS.

MAP@k: Mean average precision for a set of cascade predictions is the mean of the average precision scores for each cascade. We assume there are M infected users in top-k users so we can obtain a set of recall value $R = (1/M, 2/M, \ldots, M/M)$. Then for each $r \in R$, we can calculate the maximum precision $max_{r'>r} P(r')$ to obtain average precision AP. Finally, mean

Algorithm 1 | The Algorithm description of DLIC

Input: cascades set $C = \{c_1, c_2, \ldots, c_M\}$ and social network $G = \{U, E\}$, where $c_i = \{(u_1, t_1), (u_2, t_2), \ldots, (u_{|c_i|}, t_{|c_i|}) | t_1 \le t_2 \le \cdots \le t_{|c_i|}\}$
Output: User infection probability $P = p_1, p_2, \ldots, p_{|U|}$

Initial user representation $h = \{h_1, h_2, \ldots, h_{|U|}\}, h_i \in R^F$ for each user $u_i \in U$.
Update user representation with graph attention network, as shown in Eq. 1 - Eq. 5.
while $i <$ cascade lenth $|c_i|$ **do**
    Feed the corresponding user u_i into GRU cell according to c_i, as shown in Eq. 6 - Eq. 9
end while
Calculate time decay attention to update the cascade representation h'_i ,as shown in Eq 10, 11
Calculate the User infection probability $P = p_1, p_2, \ldots, p_{|U|}$, as shown in 12
return P

TABLE 2 | The setting of hyperparameters.

Hyperparameter	Value
User embedding	64d
Learning rate	1e-03
Dropout	0.1
Epoch	20
Heads of attention	8

average precision is calculated by the average of AP in cascades set C that we predicted. The formula is showed as **Eq. 14**:

$$AP = \frac{\sum_{r \in C} max_{r' > r} P(r')}{M} \qquad MAP = \frac{\sum_{c \in C} AP(c)}{|C|} \tag{14}$$

HITS@k: The rate of the top-k ranked nodes containing the next infected node. The formula is showed as **Eq. 15**:

$$HITS = \frac{\sum_{c \in C} p(c)}{|C|} \tag{15}$$

Where $p(\cdot)$ is an indicator function. If there is actual infected user in prediction result of cascade c, then $p = 1$. Otherwise, $p = 0$.

4.3 Baselines

In order to evaluate the performance of the DLIC model, the following baselines are applied to the same dataset to compare with the proposed model.

Topo-LSTM [11] is a model based on LSTM, which extracts directed acyclic graph from social graph and integrates its features into hidden state, which is used to predict the next node and its network structure.

SNIDSA [19] calculates the pairwise similarity of all user pairs, captures the structural dependency among users, and designs a gating mechanism to merge temporal and structural information into RNN.

FOREST [29] uses RNN to encode microscopic cascade information, which is used to learn the structural characteristics of the cascade. Also, it improved the performance through a reinforcement learning framework from macroscopic level to predict the infected nodes.

4.4 Experimental Settings

The DLIC model proposed in this paper adopts the seq2seq as the framework and graph attention network with 8 heads is used to optimize the target task. During the experiment, we set the number of observed users $K = 10$, then randomly sampled 20 neighbors of each node and used half-precision fp16 for training, and the objective function is optimized through the Adam algorithm. The specific hyperparameters setting of the experiment are shown in **Table 2**.

4.5 Experimental Results

In order to verify the effectiveness of our proposed model, we compared it with the state-of-the-art cascade prediction methods on two datasets, trying to evaluate the effect of predicting the future infected users with the metric of HITS and MAP. The result is the average of five experiments as shown on **Table 3**.

The experimental results show that the DLIC model proposed in this paper has improved all metrics on the two datasets. The results indicate improvements on MAP@k and HITS@k by more than 1 and 1.6% separately on Twitter, more than 0.4 and 0.9% separately on Douban, which proved that DLIC performs the best over other SOTR baselines in cascading prediction task.

The overall superiority of DLIC over the baselines mainly comes from two facts: 1) In the aspect of user network structure, thanks to the improvements of encoding of structural context, we achieve a better performance in the user embedding representation. The previous works only considered the influence of neighboring user nodes, while DLIC learns global user features through graph attention network. 2) In the aspect of cascading features, the improvements mostly come from the latent influence of user activation time. The previous works only regarded timestamps as a sequence of users activation order or simple learning parameters, while DLIC learns the weights of different time periods by introducing time decay attention mechanism.

TABLE 3 | Experimental results of our proposed model on Twitter and Douban. Results are evaluated by MAP@k and HITS@k in different categories of model on the cascade prediction task. For both metrics, scores are the higher the better.

Datasets		Twitter			Douban	
MAP@k	**MAP@10**	**MAP@50**	**MAP@100**	**MAP@10**	**MAP@50**	**MAP@100**
TOPO-LSTM	0.046	0.047	0.048	0.051	0.053	0.054
SNIDSA	0.149	0.152	0.153	0.065	0.069	0.071
FOREST	0.161	0.167	0.168	0.077	0.081	0.081
DLIC	**0.171**	**0.178**	**0.179**	**0.081**	**0.086**	**0.087**
HITS@k	**HITS@10**	**HITS@50**	**HITS@100**	**HITS@10**	**HITS@50**	**HITS@100**
TOPO-LSTM	0.005	0.017	0.024	0.098	0.153	0.189
SNIDSA	0.231	0.351	0.449	0.126	0.223	0.278
FOREST	0.246	0.384	0.472	0.134	0.231	0.283
DLIC	**0.262**	**0.410**	**0.494**	**0.143**	**0.244**	**0.303**

The bold values are the best results among all models.

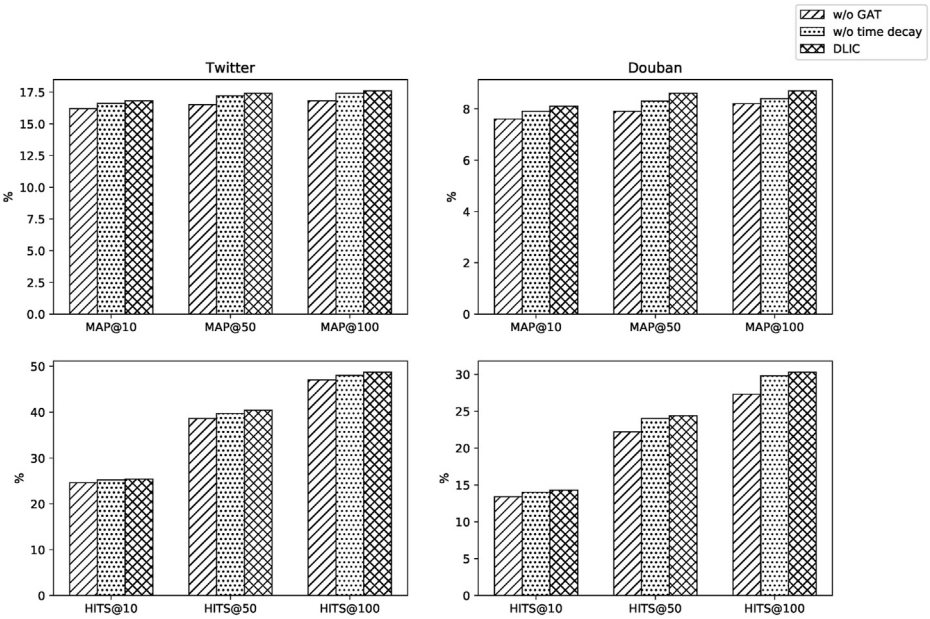


FIGURE 5 | The experimental results of ablation study, which show the influence of different components, w/o means removing the corresponding component.

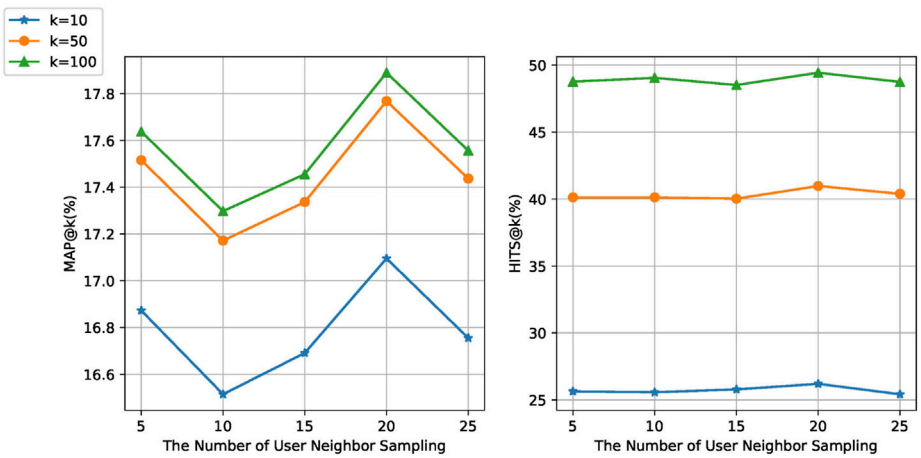


FIGURE 6 | The influence of number of user neighbor sampling on the metrics of MAP@k and HITS@k.

Overall, the proposed DLIC model which combines users relationship and cascade feature could achieve better performance. It proves that research on unified modeling is a effective way to predict cascade diffusion. We will consider it for our future work.

4.6 Ablation Study

Our model uses a graph attention on the embedding layer to learn the user topology structure and takes the time decay attention to learn temporal feature of cascades. In order to explore the influence of each component in our proposed model, we remove them separately for experiments. For this purpose, we present two simplified versions of DLIC, denoted as w/o GAT and w/o time decay.

The results of ablation experiments on two datasets verify the effectiveness of components mentioned above. When we remove the corresponding component, all metrics have decreased on both datasets, which show they are effective and reflect the different impacts of them in our model. As we can see on **Figure 5**, results on MAP@50 and HITS@50 decrease by 0.9 and 1.8%, respectively, on Twitter, and 0.7 and 2.2% respectively on Douban when we remove GAT. Results on MAP@50 and HITS@50 decrease by 0.2 and 0.7% respectively on Twitter, and 0.3 and 0.6% respectively on Douban when we remove time decay attention.

In summary, it can be seen that GAT has a great impact on our model, because the user topology structure is learned so that more context information is integrated to user embedding representation. The time-decay attention alone has little effect on the model when we remove time-decay. However, it can improve model performance when combined with the graph attention network.

4.7 Analysis of User Neighbor Sampling

To effectively utilize the user relationship from social network, we construct subgraph by sampling several user neighbors to learn the user representation with graph attention. For this purpose, we select different number of user neighbor sampling for experiment.

As we can see in **Figure 6**, with the increase of the number of user neighbor sampling, the performance of our model decreases first, then increases to the top, finally continues to decline on the metric of MAP@k. The result on metric of HITS@k also increases first and then decrease. Our proposed model achieves the best performance on both metric when we sample 20 neighbors. It shows that user representation with relationship structure is conductive to the cascade prediction. The main reason is the position of user in social network can reflect the influence to a certain extent. However, too many neighbors who is lack of influence may also introduce noise, which leads to the decline of prediction performance. Based on the above analysis, finally we sample 20 user neighbors for experiment.

5 CONCLUSION

In this paper, we proposed a cascade prediction method based on graph attention recurrent neural network for cascade prediction task. The main creative point is that our model can learn the spatial-temporal feature at the same time based on GAT and time-decay attention, respectively. Experiments on two public real-word datasets verify the effectiveness of our model in cascade prediction task and analyse the performance of different components. In the future, we plan to explore more attention mechanism to further mine the structural information between cascades.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

All authors conceived the paper. ZC drafted the initial manuscript. ZC and SL developed the software, run the analyses. TC participated in discussion of theories. JW and XL supervised the study and edited manuscript. All authors reviewed the paper.

FUNDING

This research was Supported by National Natural Science Foundation of China (No.61976054), the science and technology guiding project (2019H0040) of Fujian Province of the people's Republic of China.

REFERENCES

1. Guille A, Hacid H, Favre C, and Zighed DA. Information Diffusion in Online Social Networks: A Survey. *ACM Sigmod Rec* (2013) 42:17–28. doi:10.1145/2503792.2503797
2. Barabási A-L, and Albert R. Emergence of Scaling in Random Networks. *science* (1999) 286:509–12. doi:10.1126/science.286.5439.509
3. Qiu J, Tian Z, Du C, Zuo Q, Su S, and Fang B. A Survey on Access Control in the Age of Internet of Things. *IEEE Internet Things J* (2020) 7:4682–96. doi:10.1109/JIOT.2020.2969326
4. Wang Xiaoming YJ, and Wang L. Research on Microblog Information Diffusion Network Structural Properties. *J Chin Inf Process* (2014) 28:55.
5. Zhao L, Wang J, Chen Y, Wang Q, Cheng J, and Cui H. Sihr Rumor Spreading Model in Social Networks. *Physica A: Stat Mech its Appl* (2012) 391:2444–53. doi:10.1016/j.physa.2011.12.008
6. Xia C, Wang L, Sun S, and Wang J. An Sir Model With Infection Delay and Propagation Vector in Complex Networks. *Nonlinear Dyn* (2012) 69:927–34. doi:10.1007/s11071-011-0313-y
7. Zhao L, Cui H, Qiu X, Wang X, and Wang J. Sir Rumor Spreading Model in the New Media Age. *Physica A: Stat Mech its Appl* (2013) 392:995–1003. doi:10.1016/j.physa.2012.09.030
8. Mishra S, Rizoiu M, and Xie L. Feature Driven and point Process Approaches for Popularity Prediction. In: Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016; October 24-28, 2016; Indianapolis, IN, USA. (ACM) (2016). p. 1069–78. doi:10.1145/2983323.2983812
9. Aral S, and Walker D. Identifying Influential and Susceptible Members of Social Networks. *Science* (2012) 337:337–41. doi:10.1126/science.1215842
10. Li C, Ma J, Guo X, and Mei Q. Deepcas: An End-To-End Predictor of Information Cascades. In: Proceedings of the 26th International Conference on World Wide Web, WWW 2017; April 3-7, 2017; Perth, Australia. (ACM) (2017). p. 577–86. doi:10.1145/3038912.3052643
11. Wang J, Zheng VW, Liu Z, and Chang KC. Topological Recurrent Neural Network for Diffusion Prediction. In: 2017 IEEE International Conference on Data Mining, ICDM 2017; November 18-21, 2017; New Orleans, LA, USA. IEEE Computer Society (2017). p. 475–84. doi:10.1109/ICDM.2017.57
12. Qiu J, Tang J, Ma H, Dong Y, Wang K, and Tang J. Deepinf: Social Influence Prediction With Deep Learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018; August 19-23, 2018; London, UK. (ACM) (2018). p. 2110–9. doi:10.1145/3219819.3220077
13. Cao Q, Shen H, Cen K, Ouyang W, and Cheng X. Deephawkes: Bridging the Gap Between Prediction and Understanding of Information Cascades. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. (ACM) (2017). p. 1149–58. doi:10.1145/3132847.3132973
14. Saha A, Samanta B, Ganguly N, and De A. CRPP: Competing Recurrent Point Process for Modeling Visibility Dynamics in Information Diffusion. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018; October 22-26, 2018; Torino, Italy. (ACM) (2018). p. 537–46. doi:10.1145/3269206.3271726
15. Hodas NO, and Lerman K. The Simple Rules of Social Contagion. *Scientific Rep* (2014) 4:4343. doi:10.1038/srep04343

16. Zhong E, Fan W, Wang J, Xiao L, and Li Y. Comsoc: Adaptive Transfer of User Behaviors over Composite Social Network. In: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12; August 12-16, 2012; Beijing, China. (ACM) (2012). p. 696–704. doi:10.1145/2339530.2339641
17. Bourigault S, Lamprier S, and Gallinari P. Representation Learning for Information Diffusion Through Social Networks. In: Proceedings of the Ninth ACM international conference on Web Search and Data Mining. (ACM) (2016). p. 573–82. doi:10.1145/2835776.2835817
18. Gao S, Pang H, Gallinari P, Guo J, and Kato N. A Novel Embedding Method for Information Diffusion Prediction in Social Network Big Data. *IEEE Trans Ind Inform* (2017) 13:2097–105. doi:10.1109/TII.2017.2684160
19. Mi Z, Hui Z, Chunming Y, Bo L, and Xujian Z. Microblog Opinion Leader Mining Based on a Multi-Feature Information Diffusion Model. *J Chin Inf Process* (2018) 32:129–38.
20. Tian Z, Su S, Shi W, Du X, Guizani M, and Yu X. A Data-Driven Method for Future Internet Route Decision Modeling. *Future Gener Comput Syst* (2019) 95:212–20. doi:10.1016/j.future.2018.12.054
21. Liu Xiaoyang HD, and Tang T. Mathematical Modeling and Public Opinion Evolution Analysis of Information Diffusion With the User Attributes. *J Chin Inf Process* (2019) 33:115.
22. Liu S, Shen H, Zheng H, Cheng X, and Liao X. Ct Lis: Learning Influences and Susceptibilities Through Temporal Behaviors. *ACM Trans Knowledge Discov Data (Tkdd)* (2019) 13:1–21. doi:10.1145/3363570
23. Tian Z, Luo C, Lu H, Su S, Sun Y, and Zhang M. User and Entity Behavior Analysis Under Urban Big Data. *Trans Data Sci* (2020) 1:16. doi:10.1145/3374749
24. Guo J, Chen T, and Wu W. A Multi-Feature Diffusion Model: Rumor Blocking in Social Networks. *IEEE/ACM Trans Networking* (2021) 29:1. 386–397. doi:10.1109/TNET.2020.3032893
25. Gu Z, Wang L, Chen X, Tang Y, Wang X, Du X, et al. Epidemic Risk Assessment by a Novel Communication Station Based Method. *IEEE Trans Netw Sci Eng* (2021):1. doi:10.1109/TNSE.2021.3058762
26. Wang Y, Shen H, Liu S, Gao J, and Cheng X. Cascade Dynamics Modeling With Attention-Based Recurrent Neural Network. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017; August 19-25, 2017; Melbourne, Australia. (ijcai.org) (2017). p. 2985–91. doi:10.24963/ijcai.2017/416
27. Wang Z, Chen C, and Li W. A Sequential Neural Information Diffusion Model With Structure Attention. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018; October 22-26, 2018; Torino, Italy. ACM) (2018). p. 1795–8. doi:10.1145/3269206.3269275
28. Wang Z, Chen C, and Li W. Attention Network for Information Diffusion Prediction. In: Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018; April 23-27, 2018; Lyon , France. (ACM) (2018). p. 65–6. doi:10.1145/3184558.3186931
29. Yang C, Tang J, Sun M, Cui G, and Liu Z. Multi-scale Information Diffusion Prediction with Reinforced Recurrent Networks. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019; August 10-16, 2019; Macao, China. (ijcai.org) (2019). p. 4033–9. doi:10.24963/ijcai.2019/560
30. Chen X, Zhang K, Zhou F, Trajcevski G, Zhong T, and Zhang F. Information Cascades Modeling via Deep Multi-Task Learning. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. (ACM) (2019). p. 885–8. doi:10.1145/3331184.3331288
31. Qiu J, Du L, Zhang D, Su S, and Tian Z. Nei-tte: Intelligent Traffic Time Estimation Based on Fine-Grained Time Derivation of Road Segments for Smart City. *IEEE Trans Ind Inform* (2020) 16:2659–66. doi:10.1109/tii.2019.2943906
32. Cai H, Nguyen TT, Li Y, Zheng VW, Chen B, and Cong G. Modeling Marked Temporal point Process Using Multi-Relation Structure Rnn. *Cogn Comput* (2020) 12:499–512. doi:10.1007/s12559-019-09690-8
33. Vemula A, Muelling K, and Oh J. Social Attention: Modeling Attention in Human Crowds. In: 2018 IEEE International Conference on Robotics and Automation, ICRA 2018; May 21-25, 2018; Brisbane, Australia. IEEE (2018). p. 1–7. doi:10.1109/ICRA.2018.8460504
34. Ye L, Wang Z, Chen X, Wang J, Wu K, and Lu K. GSAN: Graph Self-Attention Network for Interaction Measurement in Autonomous Driving. In: 17th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2020; December 10-13, 2020; Delhi, India. IEEE (2020). p. 274–82. doi:10.1109/mass50613.2020.00042
35. Song W, Xiao Z, Wang Y, Charlin L, Zhang M, and Tang J. Session-based Social Recommendation via Dynamic Graph Attention Networks. In: JS Culpepper, A Moffat, PN Bennett, and K Lerman, editors. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC; February 11-15, 2019; Australia. (ACM) (2019). p. 555–63. doi:10.1145/3289600.3290989
36. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, and Bengio Y. Graph Attention Networks. *CoRR* (2017).

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Physics

ORIGINAL RESEARCH
published: 24 September 2021
doi: 10.3389/fphy.2021.711221

Multilevel Attention Residual Neural Network for Multimodal Online Social Network Rumor Detection

*Zhuang Wang and Jie Sui**

School of Engineering Science, University of Chinese Academy of Sciences, Beijing, China

In recent years, with the rapid rise of social networks, such as Weibo and Twitter, multimodal social network rumors have also spread. Unlike traditional unimodal rumor detection, the main difficulty of multimodal rumor detection is in avoiding the generation of noise information while using the complementarity of different modal features. In this article, we propose a multimodal online social network rumor detection model based on the multilevel attention residual neural network (MARN). First, the features of text and image are extracted by Bert and ResNet-18, respectively, and the cross-attention residual mechanism is used to enhance the representation of images with a text vector. Second, the enhanced image vector and text vector are concatenated and fused by the self-attention residual mechanism. Finally, the fused image–text vectors are classified into two categories. Among them, the attention mechanism can effectively enhance the image representation and further improve the fusion effect between the image and the text, while the residual mechanism retains the unique attributes of each original modal feature while using different modal features. To assess the performance of the MARN model, we conduct experiments on the Weibo dataset, and the results show that the MARN model outperforms the state-of-the-art models in terms of accuracy and F1 value.

Keywords: online social networks, rumor detection, neural networks, multimodal fusion, attention residual network

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Keke Huang,
Central South University, China
Chengyi Xia,
Tianjin University of Technology, China
Yan Wang,
Macquarie University, Australia

****Correspondence:***
Jie Sui
suijie@ucas.ac.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *18 May 2021*
Accepted: *03 August 2021*
Published: *24 September 2021*

Citation:
Wang Z and Sui J (2021) Multilevel Attention Residual Neural Network for Multimodal Online Social Network Rumor Detection. Front. Phys. 9:711221. doi: 10.3389/fphy.2021.711221

INTRODUCTION

Since the beginning of the 21st century, with the rapid development of the Internet technology and the gradual popularization of computers and other network terminal equipment, the dissemination speed of all kinds of news has been a qualitative leap, which has changed the inherent living habits of human beings to a certain extent. Especially after 2004, with the advent of the Web2.0 era [1], online social media represented by Facebook, Twitter, and Sina Weibo have developed rapidly, which not only have a great impact on the traditional news industry but also facilitate people's access to news.

Compared with the traditional news industry, social media have a lower release threshold, a faster spread, and a wider range of influence. These network rumors reduce the quality of people's access to information and seriously endanger the security of the whole society and even at the national level. In particular, rumors about some major public emergencies can easily cause panic and social unrest. Take the COVID-19 transmission incident in early 2020 as an example, from "COVID-19 is the evolutionary version of SARS" to "double coptis can prevent coronavirus infection;" rumors about the event emerge endlessly, which greatly hinder the overall prevention and control of the epidemic and causes adverse social effects.

Recently, many social media have been allowing users to add corresponding images or videos while publishing texts. News with images is more confusing and disseminating, and its forwarding frequency is 11 times more than that of pure text news [2]. However, most of the existing rumor detection models only focus on the propagation path or text of the news but ignore the images related to the event. At present, only a few works focus on the image in the news, but generally these multimodal rumor detection models only simply concatenate image features and text features for classification. In fact, the semantic features of each mode are heterogeneous in the feature space, which may lead to the following two problems:

1) The fusion of multimodal features is insufficient.
2) The noise information generated by the fusion is large, which affects the final classification results.

To solve these problems, we propose a multimodal social network rumor detection model based on the multilevel attention residual neural network. Among them, the multilevel attention mechanism selectively fuses the image–text features from the semantic level. Compared with the traditional fusion method of image–text features, the proposed method greatly improves the joint representation performance between different modal features. The residual structure retains the unique attributes of different modal features on the basis of the image–text joint representation, which effectively alleviates the noise information caused by different modal fusions. The contribution of this article can be summarized as the following three points:

1) This article proposes a multimodal social network rumor detection model based on the multilevel attention residual neural network.
2) The multi-layer attention mechanism improves the feature fusion effect between multiple modalities, and the residual structure effectively alleviates the adverse effects of the noise information generated during the fusion.
3) The experimental results on the real Weibo dataset show that the accuracy and F1 value of the MARN model are higher than those of the current mainstream multimodal rumor detection models.

RELATED WORK

Concept and Development of Rumor

The spread of rumors is a social phenomenon that develops with the development of the human society. It is often used as a weapon by hostile parties to fight. It has long been a hot topic of research. The systematic research on rumors began with Alport and Postman's [3] *The Psychology of Rumors*, which defines rumors as statements of information on specific or current topics that tend to spread from person to person, usually by oral means, without any evidence to prove their authenticity.

Compared with traditional rumors, network rumors have some different characteristics, such as faster spread and wider impact, which also brings great challenges to the detection of network rumors. The most original measures to prevent and control network rumors are by basically using a combination of user reports and manual verification for rumor detection and tracking, which not only consumes a large amount of human resources but also has a strong time lag. It is often difficult to predict and eliminate the rumors in the early stage of spread. To solve these problems, in recent years, a large number of scholars have used machine learning or neural network learning methods to detect rumors on Weibo and Twitter news and have achieved a series of results.

Research Status at Home and Abroad

From the data sources of the model, rumor detection can be roughly divided into two categories: propagation-based and content-based rumor detection. The former is based on the principle of the network structure [4–7] and uses the propagation path of the posts to classify them [8]. The latter is to use the post or its additional modal information for classification. The rumor detection referred to in this article is all content-based. It can generally be divided into three types: traditional machine learning–based methods, unimodal feature–based neural network methods, and multimodal feature–based neural network methods.

Rumor Detection Based on Traditional Machine Learning

Castillo et al. [9], who first introduced the machine learning method to the field of network rumor detection, used a variety of traditional machine learning methods to detect the reliability of the datasets collected on Twitter and achieved some results. The first one to automatically detect rumors on Sina Weibo was the method suggested by Yang et al. [10], which uses the SVM (Support Vector Machine, SVM) classifier to test and classify the datasets collected from Sina Weibo's official rumor development platform and proposes new detection features for the differences between Chinese and English language characteristics, pioneering the rumor detection in Chinese social networking platforms. On the basis of the above two studies, many experts and scholars [11, 12] have added text features, user features, and propagation features for rumor detection, which all improve the performance of rumor detection to a certain extent. Rumor detection based on traditional machine learning pioneers automated rumor detection and has a profound impact on the technological development of this field. However, such methods also have some drawbacks, such as the selection of indicators depends heavily on the experimenter's experience and the accuracy of the model's classification needs to be improved. This is also an important problem to be solved based on the neural network model.

Rumor Detection of the Neural Network Based on a Unimodal Feature

With the continuous progress of neural network technology in the field of natural language processing [13], more and more scholars have applied it to the field of rumor detection [14]. Ma et al. [15] applied the RNN (Recurrent Neural Network) model to

network rumor detection for the first time, which greatly improves the efficiency of rumor detection compared with traditional machine learning methods. Liu et al. [16] proposed an improved CNN (convolutional neural network) model for microblog rumor detection. The model is simple and easy to implement. Chen et al. [17] combined the attention mechanism with the RNN model for rumor detection, which solved the problem of excessive redundancy of text features and weak remote information connection to some extent. In 2019, Chen et al. [18] proposed an attention residual neural network combined with the CNN network for social network rumor detection, which is the first model to combine an attention model with a residual network for social network rumor detection. Experiments on two Twitter datasets show that the attention residual network can capture long-term dependencies and achieve high classification accuracy and F1 value regardless of the choice of policy. However, these traditional neural network models only focus on the text feature of rumors, ignoring the accompanying images and social characteristics, which limits the detection performance of the model and needs to be improved to adapt to the rapid development of the network era.

Rumor Detection of the Neural Network Based on Multimodal Features

Similar to sentiment classification [19, 20], social network rumor detection tasks have also entered the multimodal era in recent years. In 2017, Jin et al. [21] first introduced image features into fake news detection and created a corresponding multimodal microblog rumor dataset. The model first extracts event-related image semantic features through a pre-trained VGG19 (Visual Geometry Group, VGG) model and uses an attention mechanism to extract key features in the text and social context and then multiplies them element by element with image semantic features to adjust the weight of the visual semantic features. Experiments show that this method can detect many fake news cases which are difficult to distinguish under a unimodal feature. Wang [22] proposed an event-based antagonism network based on the work of the former. The multimodal feature extractor in this network is forced to learn the invariant representation of events to deceive the discriminator. In this way, it eliminates the strong dependency on specific events in the collected datasets and gains better generalization capabilities for unknown events. Dhruv et al. [23] then constrained the fused multimodal vectors through an automatic encoder to better learn the joint representation. Liu et al. [24] made full use of the text information contained in the image and improved the detection performance of the model by extracting hidden texts from the image.

Summarizing the previous research, it can be found that these multimodal rumor detection models using image and text features have become a major trend in the field of rumor detection. Compared with the traditional pure text rumor detection models, the multimodal rumor detection models can effectively make use of the feature differences between different modes to complement each other and improve the performance of rumor detection. However, due to the huge semantic gap and redundant information among the modal features, the existing models still have the problem of insufficient feature fusion among the modes and huge noise information when fusing, which is also an important problem to be solved by our model.

PROBLEM STATEMENT

In essence, the detection of rumors in social media is a two-category problem. That is, the experimenter divides the input content into rumors or non-rumors through a specific model. If the input content is a series of information such as the post itself and its related comments, forwarding, etc., it is called event-level rumor detection; if the input content is just the post, it is called post-level rumor detection. For example, user U1 posts a post, user U2 comments on the post, and user U3 retweets the post. Event-level rumor detection uses all of this relevant information as the basis for rumor detection, while post-level rumor detection uses only the posts posted by user U1. Our model belongs to post-level rumor detection, with the aim of identifying rumors in their early stages to avoid greater social harm.

We define a post $X = \{T, P\}$ as a tuple representing two different content patterns. $T = \{w_1, w_2, \ldots, w_n\}$ represents the text content contained in the post, where n is the number of words (w). $P = \{p_1, p_2, \ldots, p_m\}$ represents the image content attached to the post, where m is the number of images (p). The true tag of a post is $y = \{0, 1\}$ when $y = 0$ it means that the content of the post is true and when $y = 1$ the post is a rumor. Formally, rumor detection on the post-level aims to learn a projection $F(X) \rightarrow \{0, 1\}$

MODEL

In this section, cross-attention and self-attention mechanisms are used to enhance the fusion of image and text representation, and the residual mechanism is used to alleviate the adverse effects of the noise information generated during the fusion. We first describe the general framework of the model and then describe in detail the principle and operation of each component that makes up the model.

Building Model Framework

We propose a multimodal online social network rumor detection model based on the multilevel attention residual mechanism. Its overall framework is shown in **Figure 1** and consists of the following four parts:

1) Image–text embedding: The pre-trained models ResNet-18 and Bert are used to extract the original features of images and texts and transform them into the corresponding vectors R_P and R_T.
2) Cross-attention residual module: The text vector R_T is used to enhance the representation of the image vector R_P by cross-attention, and then the residual mechanism is used to add the vector R_P to the enhanced picture vector to get the vector R'_P
3) Self-attention residual module: The concatenated image–text vector R_C is fused by the self-attention process and sliced, and

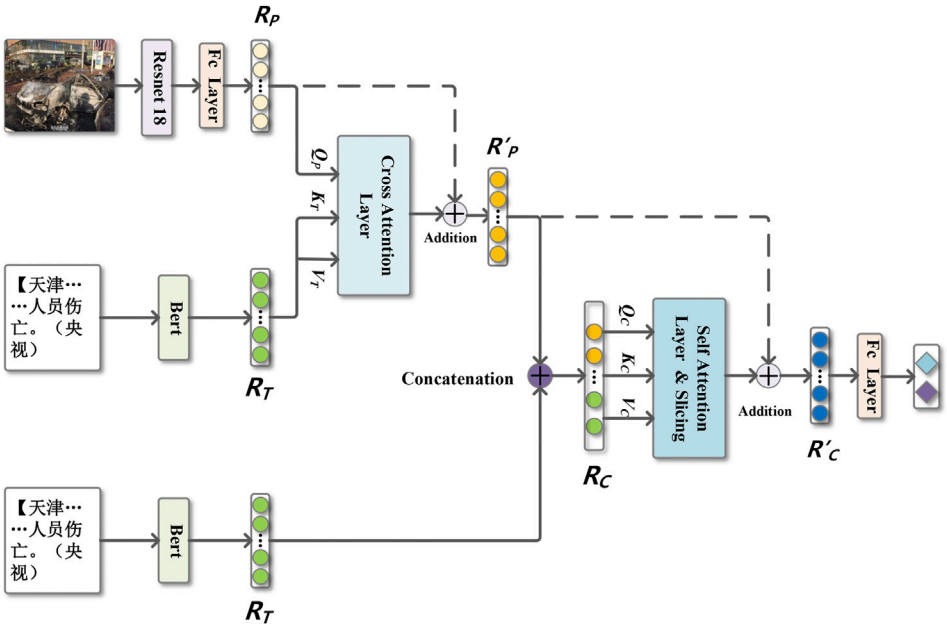


FIGURE 1 | Network structure of the MARN model. ***Instruction***: Dotted lines represent residual connections.

then the vector R_P' is added to the sliced vector to get the vector R_C' by using the residual mechanism.

4) Rumor classifier: It consists of a fully connected layer that binds the vector R_C' to get the final result.

Defining Image–Text Embedding

Using the Bert Text Extractor

Our model uses the pre-trained model Bert [25] as a feature extraction method, which improves the performance significantly compared with the traditional language model. The main reason is that Bert proposed a new pre-trained target and masked language model, which randomly masks 15% of the words in each sentence and uses the context to encode them in both directions, enriching the contextual feature of each word. In addition, Bert also pre-trains whether the two sentences are continuous. Specifically, Bert selects some sentences for A and B during the pre-training process, where statement B has a 50% probability of being the next sentence in statement A and a 50% probability of being randomly selected in the corpus. The goal is for Bert to learn the relevance of the two sentences and to better accommodate downstream tasks that require an understanding of the relationship between the upper and lower sentences.

The overall structure of the Bert model is shown in **Figure 2** [25], which is mainly composed of three parts: embedding layer, coding layer, and output layer. The embedding layer consists of three parts: token embedding, sentence embedding, and position embedding, which represent the word vector of the word, which sentence it belongs to, and where it is in the sentence. The coding layer is composed of the encode parts of a multilayer transformer. Through the multi-head attention mechanism and residual module, Bert can better enhance the extraction of deep semantic features of the text. There are two forms of the output layer where one is the vector *encode_out*, which represents the features of the whole statement. The other is the vector *pooled* representing the information of the first position [CLS]. In this model, the first output form is used. Each sentence can get a text vector $R_T \in \mathbb{R}^{\text{pad}\times d_T}$ after the Bert model, where represents the number of words embedded in each sentence and d_T represents the embedding dimension of each word.

Using ResNet-18 Image Extractor

This section uses a deep residual network ResNet-18 model based on transfer learning to extract image features. Compared with a traditional VGG model, the ResNet-18 model has smaller parameters, faster training speed, and higher accuracy. The ResNet model was originally proposed by Kaiming He et al. [26] and is widely used in image processing and computer vision. The main idea is to use multilevel residual modules to connect, which effectively alleviates the disappearance of back propagation gradient and model performance degradation caused by too many layers in traditional deep convolutional neural network models.

Each residual unit consists of a residual learning branch and an identical mapping branch, the structure of which is shown in **Figure 3** [26]. Here, i is the input, $G(i)$ is the result of the residual learning branch, ReLU is the activation function, and the output of the residual unit can be expressed as $H(i) = G(i) + i$. When the residual learning branch does not work, it can be expressed as $H(i) = i$. The two 1*1 layers in the residual branch function to reduce and increase the dimension of $G(i)$ to ensure that the dimension of $G(i)$ is consistent with that of i for subsequent operations.

We extract the last layer of the feature vector D in the ResNet-18 model by first stretching it through the flatten layer and then extracting the image vector R_P through a fully connected layer,

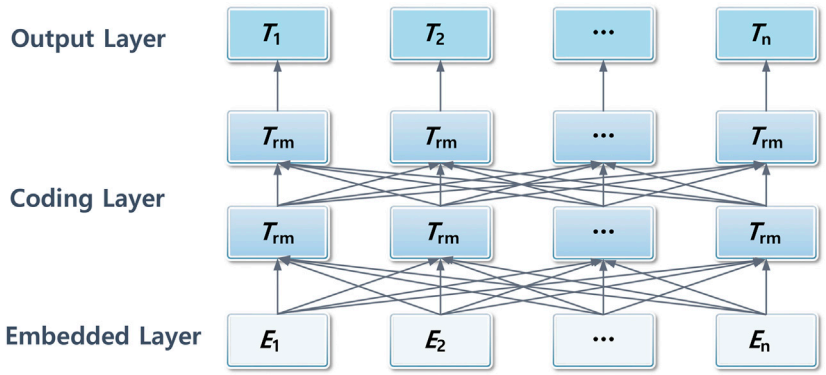


FIGURE 2 | Structure of the Bert model.

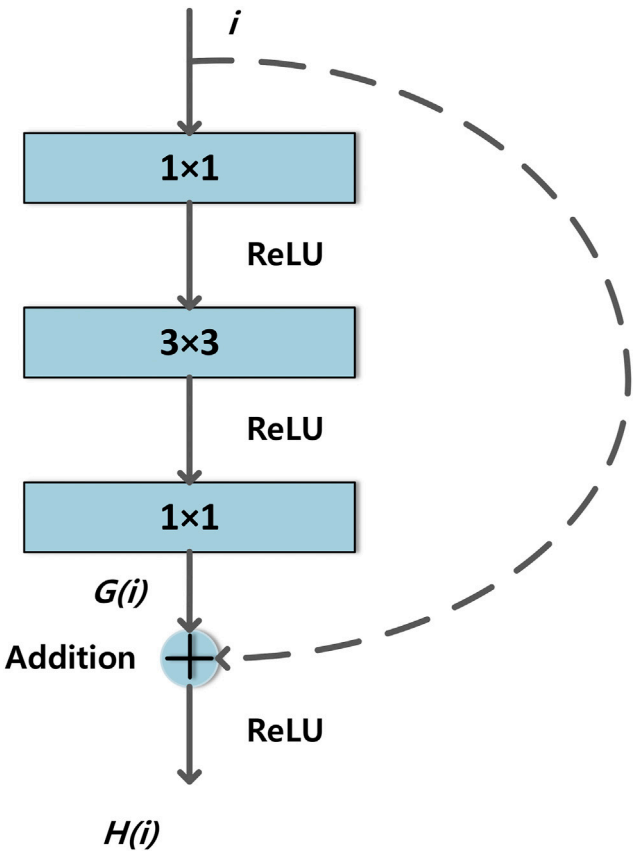


FIGURE 3 | Structure of the ResNet-18 unit. ***Instruction***: Dotted lines represent residual connections.

$$R_P = \mathrm{ReLU}\left(\mathrm{Flatten}\left(D\right) \times W_D + b_D\right) \tag{1}$$

Where, $R_P \in \mathbb{R}^{m \times d_P}$, $D \in \mathbb{R}^{m \times d_D \times 1 \times 1}$, $W_D \in \mathbb{R}^{d_D \times d_P}$, W_D and b_D are the weight matrix and bias term of this fully connected layer, respectively, and the dimension of b_D is the same as that of R_P. ReLU is the activation function, and the function of the flatten layer is to stretch the multidimensional vector into one dimension. m is the number of images attached to each post, and the value taken is m = 1. d_D is the dimension of the last output vector of the ResNet-18 model, d_P is the output dimension after image extraction, and $d_P = d_T$.

Building A Cross-Attention Residual Module

Just like human vision, the attention mechanism [27] automatically gives greater weight to the more noteworthy parts. There are two reasons why the MARN model can enhance the fusion between images and texts. On the one hand, the powerful pre-training models can express the semantic level of common words or item shapes; on the other hand, the weight distribution of the attention mechanism itself can be continuously studied to obtain better results.

We use text vectors to enhance image vectors by the cross-attention mechanism.

First, define Q_P, K_T, and V_T as follows:

$$\begin{aligned} Q_P &= R_P \times W_{QP} \\ K_T &= R_T \times W_{KT} \\ V_T &= R_T \times W_{VT} \end{aligned} \tag{2}$$

Where $Q_P \in \mathbb{R}^{m \times d_K}$, $K_T \in \mathbb{R}^{pad \times d_K}$, $V_T \in \mathbb{R}^{pad \times d_V}$, $W_{QP} \in \mathbb{R}^{d_P \times d_K}$, $W_{KT} \in \mathbb{R}^{d_T \times d_K}$, $W_{VT} \in \mathbb{R}^{d_T \times d_V}$. d_K and d_V are the second dimensions of matrix W_{QP} (or W_{KT}) and W_{VT}, note that $d_K = d_V = d_P = d_T$.

Then, compute the enhanced image vector ATT_P:

$$ATT_P = \mathrm{Softmax}\left(\frac{Q_P \times K_T^{\top}}{\sqrt{d_K}}\right) \times V_T \tag{3}$$

Where $ATT_P \in \mathbb{R}^{m \times d_V}$, $K_T^{\top}$ is the transposition of vector K_T and Softmax is the normalization function.

Finally, in order to ensure that the performance of the image feature after attention enhancement is not inferior to that of the original vector R_P, the residual mechanism is used to fuse R_P with the enhanced image feature ATT_P to get the vector and accumulate the results of each fusion of m images. If the fusion effect between the image and text is not ideal, the model will automatically adjust the value of ATT_P through back propagation until it is very small so that R_P' is almost equal to R_P, ensuring that the effect after fusion will not deteriorate,

TABLE 1 | Statistics of the dataset.

	Rumor	Non-rumor	All
Training set	3,561	3,584	7,145
Testing set	1,187	1,195	2,382
All	4,748	4,779	9,527

$$R_P' = \sum_{\mathrm{m}} (ATT_P + R_P) \tag{4}$$

Where $R_P' \in \mathbb{R}^{1\times \mathrm{d}_V}$.

Compared with single-head attention, multi-head attention can learn the weight relationship between each element from different angles and then concatenate to get the final vector representation. Under normal circumstances, its performance is better than single-head attention. Our model uses multi-head attention for fusion, and the specific related parameters are shown in **Table 2**. Since its principle is the same as single-head attention, it will not be repeated here.

Building a Self-Attention Residual Module

After obtaining the image vector R_P' with enhanced text features, we will proceed to fuse the image–text features.

First, we concatenate the image vector R_P' with the text vector R_T to get the initial fusion vector R_C,

$$R_C = \mathrm{Concat}\left[R_P',\ R_T\right] \tag{5}$$

where $R_C \in \mathbb{R}^{(1+\mathrm{pad})\times \mathrm{d}_T}$ and Concat is the concatenation function.

Then, similar to the cross-attention residual module, define Q_C, K_C, and V_C,

$$\begin{aligned} Q_C &= R_C \times W_{QC} \\ K_C &= R_C \times W_{KC} \\ V_C &= R_C \times W_{VC} \end{aligned} \tag{6}$$

Where $Q_C \in \mathbb{R}^{(1+\mathrm{pad})\times \mathrm{d}_K}$, $K_C \in \mathbb{R}^{(1+\mathrm{pad})\times \mathrm{d}_K}$, $V_C \in \mathbb{R}^{(1+\mathrm{pad})\times \mathrm{d}_V}$, $W_{QC} \in \mathbb{R}^{\mathrm{d}_T\times \mathrm{d}_K}$, $W_{KC} \in \mathbb{R}^{\mathrm{d}_T\times \mathrm{d}_K}$, $W_{VC} \in \mathbb{R}^{\mathrm{d}_T\times \mathrm{d}_V}$.

Then, the self-attention mechanism is used to calculate the weight of the integrated vector R_C to obtain the enhanced integrated vector ATT_C,

$$ATT_C = \mathrm{Softmax}\left(\frac{Q_C \times K_C^{\top}}{\sqrt{\mathrm{d}_K}}\right) \times V_C \tag{7}$$

Where $ATT_C \in \mathbb{R}^{(1+\mathrm{pad})\times \mathrm{d}_V}$, $K_C^{\top}$ is the transposition of vector K_C.

Finally, use the residual mechanism to connect the vector ATT_C with the vector R_P'. It is worth noting that the dimension of ATT_C is not the same as that of vector R_P', so the vector ATT_C needs to be sliced before residual joining,

$$R_C' = \mathrm{Slice}\left(ATT_C\right) + R_C' \in \mathbb{R}^{\mathrm{d}_V} \tag{8}$$

Where $R_C' \in \mathbb{R}^{\mathrm{d}_V}$, Slice is the slicing function.

Defining Classifier and Loss Function

The vector R_C' is sent to the fully connected layer to obtain the prediction probability, $\hat{y}$

$$\hat{y} = \mathrm{Softmax}\left(R_C' \times W_C + b_C\right) \tag{9}$$

where $\hat{y}$ is the probability predicted by the model, W_C is the weight matrix of the fully connected layer, b_C is the bias term, $W_C \in \mathbb{R}^{\mathrm{d}_V\times 2}$ and the dimension of b_C is the same as $\hat{y}$.

We use cross-entropy as the loss function of this model, and the formula is as follows:

$$\mathrm{L}(\theta) = -\frac{1}{\mathrm{z}}\sum\nolimits_{X}\left[\mathrm{y}\ln\hat{y} + (1-\mathrm{y})\ln(1-\hat{y})\right] \tag{10}$$

where θ represents all parameters of the model, z is the total number of training samples, and y is the real label of samples.

We use the Adam optimizer to carry out back propagation, so as to obtain the best model parameters, and test the actual performance of the model on the testing dataset.

EXPERIMENTS

In this section, we first introduce the dataset and various hyper-parameters used in our experiment, then briefly introduce the baseline model we used, and finally analyze the results of the comparison experiment and the ablation experiment.

Dataset

In order to fairly compare the detection performance of this model and the baseline models, this article uses the Weibo dataset, which is commonly used in the field of multimodal rumor detection to carry out the experiment. This dataset was first published by Jin et al. [21], and it contains roughly the same number of rumor posts and non-rumor posts. Among them, rumor posts came from the official rumor debunking system of Weibo from May 2012 to January 2016 and non-rumor posts came from news verified by the authoritative Chinese news agency Xinhua News Agency. At the same time, in order to ensure the availability of the dataset, Jin et al. [21] deleted duplicated images and very small or very long images in the original image set. This article adopts the same method as paper [21], setting the ratio of the training set to test set to 4:1. The details of the dataset are shown in **Table 1**.

Hyperparameters

The experiment of this model is based on Python3.7, using the PyTorch deep learning framework, computing on GPU, and

TABLE 2 | Hyperparameters.

Hyperparameter	Value
Word embedding dimension (d_T)	768
Sentence length (pad)	64
Attention heads	96
Learning rate	0.001
Batch size	216
Dropout	0.2
Epochs	50

TABLE 3 | Results of different baseline models on a dataset.

Model	Accuracy	F1
Textual	0.8077	0.8074
Visual	0.6969	0.6954
att-RNN	0.7720	0.7685
MSRD	0.7940	0.7790
EANN	0.8270	0.8290
MVAE	0.8240	0.8230
MARN	**0.8581**	**0.8580**

Bold font represents the largest number in this column.

using cross-entropy loss function and Adam optimizer for back propagation optimization. At the same time, in order to prevent overfitting, a dropout layer is added after each fully connected layer to randomly delete some parameters when training the model. To save the training time and GPU memory space, we fixed the internal parameters of Bert and ResNet-18 models and did not participate in the back propagation training of the models. Other hyperparameters are shown in **Table 2**.

Baseline Models

To compare the performance of each model fairly, the following models are tested based on the above dataset, and the partition ratio of the training dataset and testing dataset is the same.

1) Textual Model

The textual model only uses the text features in the samples for experiments and directly transfers the text features into the Bert model for training, followed by two fully connected layers for classification.

2) Visual Model

The visual model only uses the image features in the sample for experiments and uses 0 to fill the sample with missing image features, that is, a pure black image is used to replace the image features in the sample. The image is encoded and input into the ResNet-18 model, followed by a dimension of 32 fully connected layers, and finally input into the classifier to get the sample classification results. In order to enhance the generalization ability of the model and reduce the training time, the ResNet-18 network adopts the method of migration learning, selects the model parameters that have been trained on the large dataset Image 1000, and does not participate in the back propagation. It only fine tunes the back wiring layer.

TABLE 4 | Results of different ablation models on a dataset.

Model	Accuracy	F1
MARN-CA-SA	0.8359	0.8357
MARN-CA	0.8472	0.8469
MARN-SA	0.8489	0.8486
MARN-residual	0.8484	0.8484
MARN	**0.8581**	**0.8580**

Bold font represents the largest number in this column.

3) Att-RNN

This model [21] uses the attention mechanism to fuse the text, image, and social features and then input them into the classifier for judgment. In order to make a fair comparison, we adopt the model after deleting the social characteristics, and the other parameters are consistent with those in the literature.

4) MSRD

In this model [24], First, the text in the image is extracted, and then it is connected with the text content in the sample. Finally, the image and the connected text are fused and classified at the feature level.

5) EANN

EANN [22] uses VGG19 and the Text-CNN (text-convolutional neural network) to extract the image and text features and uses the event discriminator to take the concatenated vector of constraints and finally input the concatenated vector to the classifier for classification.

6) MVAE

This model [23] uses the VAE (variational autoencoder) module to constrain the vector after multimodal feature fusion and then classify the feature vector.

7) MARN

The whole model is proposed in this article.

Comparison and Analysis of Baseline Models

We use common indicators such as F1 value and accuracy to evaluate each model. The results of each model are shown in **Table 3**.

Table 3 shows that the MARN model achieves 0.8581 and 0.8580 of the most important performance indicators F1 value and accuracy, respectively, which are higher than the mainstream multimodal rumor detection model and fully demonstrate the advanced performance of the MARN model. On the one hand, the multilevel attention mechanism selectively fuses the text and image features, making full use of the feature complementary function between each mode. On the other hand, the residual mechanism keeps the unique attributes of each mode while using the fused features, which ensures that the final result will not be worse than before.

In addition, it can be seen from **Table 3** that the accuracy and F1 value of the visual model are lower than those of the textual model. After all, in the current social network, text is still the most important source of information for people and images only play a minor role. Moreover, the performance of the textual model is better than that of the att-RNN model and MSRD model because we used Bert instead of the traditional LSTM (Long Short-Term Memory) as a text extractor. The EANN model and MVAE model use an event discriminator and a VAE, respectively, to constrain the concatenated vectors, which makes their accuracy higher. However, these two models only connect the vectors of different modes in series, so it is difficult to fuse the information of

different modes at the semantic level, which is what the MARN model focuses on.

Comparison and Analysis of Ablation Models

To further analyze the influence of each module on the overall model results, we deleted each module and carried out experiments.

MARN-CA-SA: The cross-attention residual module and self-attention residual module are deleted in this model. It can be understood as directly concatenating the vectors R_V and R_T into the classifier for classification.
MARN-CA: This model removes the cross-attention residual module. The R_V and R_T vectors are concatenated and then classified by self-attention residual fusion.
MARN-SA: This model removes the self-attention residual module. The vector R'_V and R_T are concatenated and classified.
MARN-Residual: This model removes the residual connection in the cross-attention residual and self-attention residual modules (the dotted line in **Figure 1**), and all other aspects remain unchanged.

The classification results of each ablation model are shown in **Table 4**.

From **Table 4**, it can be seen that when the cross-attention residual module and self-attention residual module are deleted at the same time, the accuracy of the model is at least 0.8359, but it is still higher than that of the textual model and all baseline models. There are two reasons for this result. One is that we used the Bert model with better text extraction, and the other is that the overall results are improved by the image features.

When the model removes the cross-attention residual module, the accuracy of the model is slightly lower than that of the overall model, which shows that it is effective to use the text feature to strengthen the image by using the cross-attention residual mechanism. The enhanced image can give greater weight to the key areas related to the text so that the features can be selected to deal with. Similarly, when the self-attention residual module is removed, the model results are lower than that of the overall model. This is because the cross-attention residual module only enhances the image and does not involve the text mode. The self-attention residual module enhances the attention of the text feature and image feature at the same time and further improves the classification performance of the model.

It is worth noting that when we remove the residual connection between the two modules, the model performance is also reduced, which is easier to understand. The function of the residual mechanism is to prevent the overall model performance from being worse than the original model. However, it can be seen from **Table 4** that the role of the residual mechanism does not stop there. It can make full use of the complementarity of multimodal information while keeping the unique attributes of each mode as much as possible, avoiding the adverse effects of noise information.

CONCLUSION

In this article, we propose the MARN model to solve the problem of insufficient feature fusion between modes and serious information redundancy after fusion. The model uses the multilevel attention residual module to fuse text and image features selectively. On the basis of making full use of each mode feature, the noise information generated during mode fusion is minimized, to a certain extent, resulting in the above two problems being solved. The experimental results show that the performance of the MARN model is better than the related baseline models and ablation models in terms of accuracy and F1 value. To the best of our knowledge, there is no research on video rumors. We are going to collect short video rumors and explore them, so as to expand the application scope of rumor detection.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. These data can be found here: https://github.com/wangyajun-ops/Weibo-dataset.

AUTHOR CONTRIBUTIONS

WZ: responsible for article conception, model building, code implementation, and writing; SJ: corresponding author, responsible for guidance and revision work.

FUNDING

National Key Research and Development Program of China, No.2017YFB0803001; National Natural Science Foundation of China, No.61572459.

REFERENCES

1. O'reilly T. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *MPRA Paper* (2007) 97(7):253–9.
2. Jin Z, Cao J, Zhang Y, Zhou J, and Tian Q. Novel Visual and Statistical Image Features for Microblogs News Verification. *IEEE Trans Multimedia* (2017) 19(3):598–608. doi:10.1109/TMM.2016.2617078
3. Alport GW, and Postman L. *The Psychology of Rumor*. Henry Holt: public opinion quarterly (1947). p. 45p.
4. Li S, Zhao D, Wu X, Tian Z, Li A, and Wang Z. Functional Immunization of Networks Based on Message Passing. *Appl Maths Comput* (2020) 366: 124728–8. doi:10.1016/j.amc.2019.124728
5. Huang K, Li S, Dai P, Wang Z, and Yu Z. SDARE: A Stacked Denoising Autoencoder Method for Game Dynamics Network Structure Reconstruction. *Neural Networks* (2020) 126:143–52. doi:10.1016/j.neunet.2020.03.008
6. Li S, Jiang L, Wu X, Han W, Zhao D, and Wang Z. A Weighted Network Community Detection Algorithm Based on Deep Learning. *Appl Maths Comput* (2021) 401(2021):126012–9. doi:10.1016/j.amc.2021.126012

7. Huang K, Wang Z, and Jusup M. Incorporating Latent Constraints to Enhance Inference of Network Structure. *IEEE Trans Netw Sci Eng* (2020) 7(1):466–75. doi:10.1109/TNSE.2018.2870687
8. Zhang P, Ran H, Jia C, Li X, and Han X. A Lightweight Propagation Path Aggregating Network with Neural Topic Model for Rumor Detection. *Neurocomputing* (2021) 458(2021):468–77. doi:10.1016/J.NEUCOM.2021.06.062
9. Castillo C, Mendoza M, and Poblete B. Information Credibility on Twitter. In: Proceedings of the 20th International Conference on World Wide Web; 2011 March 28 - April 1; Hyderabad, India (2011) p. 675–84. doi:10.1145/1963405.1963500
10. Yang F, Liu Y, Yu X, and Yang M. Automatic Detection of Rumor on Sina Weibo. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics; 2012 August 12; Beijing, China. New York, NY: MDS (2008) doi:10.1145/2350190.2350203
11. Mohammad SM, Sobhani P, and Kiritchenko S. Stance and Sentiment in Tweets. *ACM Trans Internet Technol* (2017) 17(3):1–23. doi:10.1145/3003433
12. Zhao Z, Resnick P, and Mei Q. Enquiring Minds. In: Proceedings of the 24th International Conference on World Wide Web; 2015 May 18-May 22; Florence, Italy (2015) doi:10.1145/2736277.2741637
13. Zhang X, Chen F, and Huang R. A Combination of RNN and CNN for Attention-Based Relation Classification. *Proced Comp Sci* (2018) 131:911–7. doi:10.1016/j.procs.2018.04.221
14. Liu Y, Jin X, Shen H, Bao P, and Cheng X. A Survey on Rumor Identification over Social Media. *Chin J Comp* (2018) 41(07):1536–58. doi:10.11897/SP.J.1016.2018.01536
15. Ma J, Gao W, Mitra P, Kwon S, Jansen B, Wong K, et al. Detecting Rumors from Microblogs with Recurrent Neural Networks. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence; 2016 July 9 - July 15; San Francisco, USA (2016). p. 3818–24.
16. Liu Z, Wei Z, and Zhang R. Rumor Detection Based on Convolutional Neural Network. *J Comp Appl* (2017) 37(11):3053–6. doi:10.11772/j.issn.1001-9081.2017.11.3053
17. Chen T, Li X, Yin H, and Zhang J. Call Attention to Rumors: Deep Attention Based Recurrent Neural Networks for Early Rumor Detection. In: Proceedings of the Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference; 2018 June 3- June 6; Melbourne, Australia (2018) p. 40–52. doi:10.1007/978-3-030-04503-6_4
18. Chen Y, Sui J, Hu L, and Gong W. Attention-Residual Network with CNN for Rumor Detection. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management; 2019 November 3 - November 7; Beijing, China (2019) doi:10.1145/3357384.3357950
19. Poria S, Cambria E, Howard N, Huang G-B, and Hussain A. Fusing Audio, Visual and Textual Clues for Sentiment Analysis from Multimodal Content. *Neurocomputing* (2016) 174:50–9. doi:10.1016/j.neucom.2015.01.095
20. Huang F, Zhang X, Zhao Z, Xu J, and Li Z. Image-text Sentiment Analysis via Deep Multimodal Attentive Fusion. *Knowledge-Based Syst* (2019) 167:26–37. doi:10.1016/j.knosys.2019.01.019
21. Jin Z, Cao J, Guo H, Zhang Y, and Luo J. Multimodal Fusion with Recurrent Neural Networks for Rumor Detection on Microblogs. In: Proceedings of the 25th ACM international conference on Multimedia; 2017 October 14 - October 19; California, USA (2017) p. 795–803. doi:10.1145/3123266.3123454
22. Wang Y, Ma F, Jin Z, Yuan Y, Xun G, Jha K, Su L, and Gao J. Eann. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018 July 19 - July 23; London, England (2018) doi:10.1145/3219819.3219903
23. Dhruv K, JaiPal S, Manish G, and Varma V. MVAE: Multiimodal Variational Autoencode for Fake News Detection. In: Proceedings of the 19th World Wide Web Conference; 2019 May 10 - May 14; San Francisco, USA (2019). doi:10.1145/3308558.3313552
24. Liu J, Feng K, Jeff Z, Juan D, and Wang L. MSRD: Multi-Modal Web Rumor Detection Method. *J Comp Res Dev* (2020) 57(11):2328–36. doi:10.7544/issn1000-1239.2020.20200413
25. Devlin J, Chang M, Lee K, and Toutanova K. *Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding* (2018). p. 04805. arXiv:1810.
26. He K, Zhang X, Ren S, and Sun J. Deep Residual Learning for Image Recognition. In: Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition; 2016 June 27 - June30; USA. Piscataway: NV (2016) p. 770–8. doi:10.1109/CVPR.2016.90
27. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, et al. Attention Is All You Need. *Adv Neural Inf Process Syst* (2017) 12:5999–6009.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

frontiers in Physics

ORIGINAL RESEARCH
published: 28 September 2021
doi: 10.3389/fphy.2021.763081

COVID-19 Rumor Detection on Social Networks Based on Content Information and User Response

*Jianliang Yang and Yuchen Pan**

School of Information Resource Management, Renmin University of China, Beijing, China

The outbreak of COVID-19 has caused a huge shock for human society. As people experience the attack of the COVID-19 virus, they also are experiencing an information epidemic at the same time. Rumors about COVID-19 have caused severe panic and anxiety. Misinformation has even undermined epidemic prevention to some extent and exacerbated the epidemic. Social networks have allowed COVID-19 rumors to spread unchecked. Removing rumors could protect people's health by reducing people's anxiety and wrong behavior caused by the misinformation. Therefore, it is necessary to research COVID-19 rumor detection on social networks. Due to the development of deep learning, existing studies have proposed rumor detection methods from different perspectives. However, not all of these approaches could address COVID-19 rumor detection. COVID-19 rumors are more severe and profoundly influenced, and there are stricter time constraints on COVID-19 rumor detection. Therefore, this study proposed and verified the rumor detection method based on the content and user responses in limited time CR-LSTM-BE. The experimental results show that the performance of our approach is significantly improved compared with the existing baseline methods. User response information can effectively enhance COVID-19 rumor detection.

Keywords: rumor detection, COVID-19, social networks, social physics, user responses

OPEN ACCESS

Edited by:
Chengyi Xia,
Tianjin University of Technology, China

Reviewed by:
Zhan Bu,
Nanjing University of Finance and Economics, China
Yuan Bian,
University of Chinese Academy of Sciences, China

****Correspondence:***
Yuchen Pan
panyuchen@ruc.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *23 August 2021*
Accepted: *15 September 2021*
Published: *28 September 2021*

Citation:
Yang J and Pan Y (2021) COVID-19 Rumor Detection on Social Networks Based on Content Information and User Response. Front. Phys. 9:763081. doi: 10.3389/fphy.2021.763081

INTRODUCTION

Nowadays, the social network has become an indispensable tool in people's daily life. People carry out activities such as social communication, obtaining information, and expressing opinions on social network platforms. In the above activities, securing information and expressing opinions are particularly frequent on social networks. However, most of the content on social networks is user-generated content (UGC), and the veracity of UGC is challenging to be guaranteed. The net structure of a social network is convenient for the viral dissemination of information, which makes it easy to generate rumors in a social network, and rumors are easier to spread on a large scale. Rumors in social networks are particularly rampant when public incidents occur. During the COVID-19 epidemic outbreak in 2020, a large number of rumors spread widely on social platforms such as Twitter and Weibo, which aggravated people's fear and anxiety about the epidemic, and made people experience an "information epidemic" in the virtual space [1]. Rumor governance on social networks is essential and necessary work.

For social network users, removing rumors on social networks could effectively reduce people's anxiety and stress during COVID-19 and help people reduce wrong behavior (such as refusing vaccines) caused by misinformation, thus protecting their health. For social network platforms,

removing rumors could reduce the spread of false information and improve the platforms' environment and user experience. For public health departments, removing rumors could reduce the cost of responding to the epidemic by allowing truthful and correct policies and guidelines to be disseminated effectively. The effective detection of rumors is the key to rumor governance. If false rumors or fake news on social networks can be detected sooner, relevant measures (e.g., rumor refutation and timely disclosure of information) will be taken more timely.

For the detection of rumors, existing studies proposed methods from various perspectives. Most methods for rumor detection are based on rumor content information, rumor source, and propagation path. Rumor detection methods based on content information focuses on language style, emotional polarity, text and picture content features [1]. Rumor detection methods based on rumor source focuses on web address (e.g., the source URLs of rumors), website credit, and webpage metadata [2]. Rumor detection methods based on propagation focus on the propagation structural features during rumor propagation, such as the retweeting and commenting behavior by social platform users [3]. With the development of artificial intelligence, deep learning methods make a significant contribution to various tasks. Some studies had adopted artificial intelligence based methods in rumor detection and achieved decent performance [4]. With the advent of language models based on transfer learning like BERT [5] and GPT3 [6], the analysis ability of deep learning models for natural language is further improved, which indicates us to utilize the language models based on transfer learning on rumor detection.

Time constraints are an essential factor that needs to be taken into consideration. The timelier we detect the fake news on a social network, the less harm it will cause. Public health emergencies like COVID-19 epidemic-related information are radically concerned and could profoundly affect psychology and behavior. There is a stricter time constraint on COVID-19 rumor detection. With the time constraints, methods based on propagation path are not applicative. It takes time to form the propagation path of a rumor. This indicates that we pay more attention to the content of rumors and user comments, and retweets. Because the users of a social network can comment and retweet on a rumor, known as user responses, the user responses usually contain information on the rumor's veracity. However, most of the existing studies did not take the content of user responses. The responses from users can be considered as discussions or arguments around the rumor. By extracting user response features, we may be able to implement rumor detection better. Facing rumor detection on COVID-19 on social networks, this study proposes a novel deep learning method based on rumor content and user responses. Our method has the following contributions:

1. Our method incorporates user response sequence into the rumor detection system. On the one hand, the information contained in user responses is fully utilized; on the other hand, the sequence of user responses also contains a part of the features of the rumor propagation path.
2. Time limit is added in our study. Only user responses within 24 h of rumor release are used as model input for detection.
3. Our method is based on the language model with transfer learning to obtain content features. Moreover, to capture richer information about COVID-19 in the social context, we use post-training mechanism to post train BERT on the corpus of COVID-19 related posts on Twitter and Weibo.

The structure of this paper is as follows: *Related Work* introduces the research progress on this topic, especially the progress in methods development. *Methods* introduces the problem statement of COVID-19 rumor detection and the methods proposed in this study. *Experiments* introduces the experimental dataset, baselines, evaluation methods, experiment settings, and experimental results. In *Discussion*, the experimental results are deeply analyzed and discussed. *Conclusion* summarizes the research findings of our work and points out some future directions.

RELATED WORK

With the development of intelligent devices and mobile internet, human beings are experiencing an era of information explosion. At present, countless information is flooded in our lives. However, not all of this information is true, and even in the outbreak of a major public health crisis such as the COVID-19 epidemic, much of the information we have obtained is false rumors. Generally speaking, a rumor refers to a statement whose value can be true, false, or uncertain. Rumor is also called fake news [7]. Rumor detection means to determine whether a statement or a Twitter post is a rumor or non-rumor. The task of determining whether a statement or a Twitter post is a rumor or non-rumor is also called rumor verification [8]. According to recent studies, rumor detection refers to the veracity value of a rumor. Therefore, rumor detection is equivalent to rumor verification [9].

Since information is easier to spread on social networks, rumor detection on social networks is more complex than general fake news detection. For detecting fake news, text features, source URL, and source website credit can be considered [2]. The source of information is more complex on the social network, and information spreading is much faster and wider. Rumor detection on social media is critical. Existing studies show that rumor detection on social networks is often based on text content features, user features, rumor propagation path features. Among them, the text content features and rumor propagation path features are significant for rumor detection.

For rumor detection methods based on text content features, writing style and topic features are an essential basis for determining whether rumors are true or not [10]. In addition to the text content, postag, sentiment, and specific hashtags such as "#COVID19" and "#Vaccine" are also important content features [11]. Chua et al. summarized six features, including comprehensiveness, sentence, time orientation, quantitative details, writing style, and topic [12]. With the development of deep learning and artificial intelligence, deep learning models

such as CNN have been used to extract the text features of rumors and combined with word embedding generation algorithms such as Word2vec, GloVe. Deep learning models can automatically extract the features related to rumors detection through representation learning and have achieved decent performance in the rumor detection task. Using CNN to extract the features of rumor content has a good effect on limited data and early detection of rumors [13]. CNN is also applied to feature extraction of text content in multimodal fake news detection [4].

Rumor propagation path is another common and essential feature of rumor detection. Real stories or news often have a single prominent spike, while rumors often have multiple prominent spikes in the process of spreading. Rumors spread farther, faster, and more widely on social networks than real stories or news [14]. Focusing on the rumor recognition path, Kochkina et al. proposed the branch-LSTM algorithm, which uses LSTM to transform propagation path into a sequence, combines text features and propagation path features and conducts rumor verification through a multi-task mechanism [8]. Liu et al. regarded the rumor propagation path as a sequence and utilized RNN to extract propagation path information [15]. Kwon et al. combined text features, user network features, and temporal propagation paths to determine rumors [16]. Bian et al. transformed rumor detection into a graph classification problem and constructed the Bi-GCN from Top-Down and Bottom-Up two directions to extract the propagation features on social networks [3].

Because rumor detection needs a high-quality dataset as support, few studies are focusing on COVID-19 rumor detection. Glazkova et al. proposed the CT-BERT model, paying attention to the content features, and fine-tuned the BERT model based on other news and Twitter posts related to COVID-19 [17]. For the datasets, Yang et al. [18] and Patwa et al. [19] provided rumor datasets on COVID-19, which are mainly based on social network platforms such as Twitter, Facebook, and Weibo, and news websites such as PolitiFact.

Compared to routine rumor detection, COVID-19 rumor detection has a strict time constraint, especially during the outbreak stage of the epidemic. Once the rumor detection is not timely enough, the negative impact brought by rumor propagation is enormous. The damage caused by COVID-19 rumors can increase rapidly over time and have an even more significant and broader impact than other rumors. Therefore, early rumor detection on COVID-19 needs to be considered, and early detection and action should be taken. Most of the existing studies focus on the features of rumor content and propagation path but pay insufficient attention to user responses and rumor detection within a limited time. User responses to a rumor often include stance and sentiment toward the rumor. Particularly for false rumors, user responses are often more controversial [20].

In the existing studies, some suggested that user response can better assist systems in detecting rumors [9, 20]. However, more studies use user response to determine user stance and regard user stance classification as a separate task. User stance refers to users' attitudes toward rumors. Similar to sentiment polarity classification, user stance is generally a value of [−1,1], where one indicates full support for the rumor to be true, 0 indicates neutrality, and −1 indicates no support for the rumor to be true at all [21]. There are studies on implementing rumor verification and user stance simultaneously through a multi-task mechanism [8]. However, there are very few studies that directly use user responses to enhance rumor detection. Given the shortcomings of existing studies, this study proposes a rumor detection method based on rumor content and user response sequence in a limited time and uses the language model based on transfer learning to extract the features of rumor text.

METHODS

This section introduced the method based on rumor content and the user response sequence proposed in our study. *Problem Statement* presents the problem statement of rumor detection. *Rumor Content Feature Extractor* introduces the feature extracting method for the COVID-19 rumor content. User *Response Feature Extractor* introduces the feature extracting method for the user response of the COVID-19 rumor content.

Problem Statement

The problem of rumor detection on COVID19 on social networks can be defined as: for a rumor detection dataset $R = \{r_1, r_2, \ldots, r_n\}$. r_i is the i-th rumor event, and n is the number of rumors in the rumor dataset. $r_i = \{x_i, s_1^i, \ldots s_j^i, \ldots, s_{m_i}^i\}$, where x_i is the source post of rumor event r_i, and s_j^i is the response to the post x_i from other users within a certain period of time. Specifically, user responses $s_{1\ldots m_i}^i$ to the post x_i can be defined as a sequence. For each rumor events r_i is associated with a ground-truth label $y_i \in \{\mathrm{F}, \mathrm{T}, \mathrm{U}\}$ corresponding to False Rumor, True Rumor and Unverified Rumor. Given a rumor dataset on COVID-19, the goal of rumor detection is to construct a classification system f, that for any r_i, its label y_i can be determined. In many studies, this definition is the same as rumor veracity classification task [9, 22].

Rumor Content Feature Extractor

In this study, we implemented a deep learning model based on content features and user responses for COVID-19 rumor detection in limited time. Therefore, content features are an important basis for rumor detection. We need to extract the features for the rumor content and map the rumor content to embedding in a vector space. In the common representation learning process, for a rumor text x_i, pre-training models such as Word2vec or GloVe are generally transform the words $\{w_{i,1} \ldots w_{i,n}\}$ composed of rumor text x_i into word embedding, and then deep learning models such as RNN and CNN are used to extract features related to rumor detection and form rumor content feature C. For example, the last step h_n of RNN or the vector C from CNN pooling layer is normally used to represent the content feature of the whole rumor text x_i.

Along with the development of natural language processing technology, language models based on transfer learning, such as ELMo [23], BERT [5], and XLNet [24], have achieved excellent performance in text feature extraction. Benefit from the transfer learning mechanism, language models like BERT significantly improved backend tasks, including text classification, machine

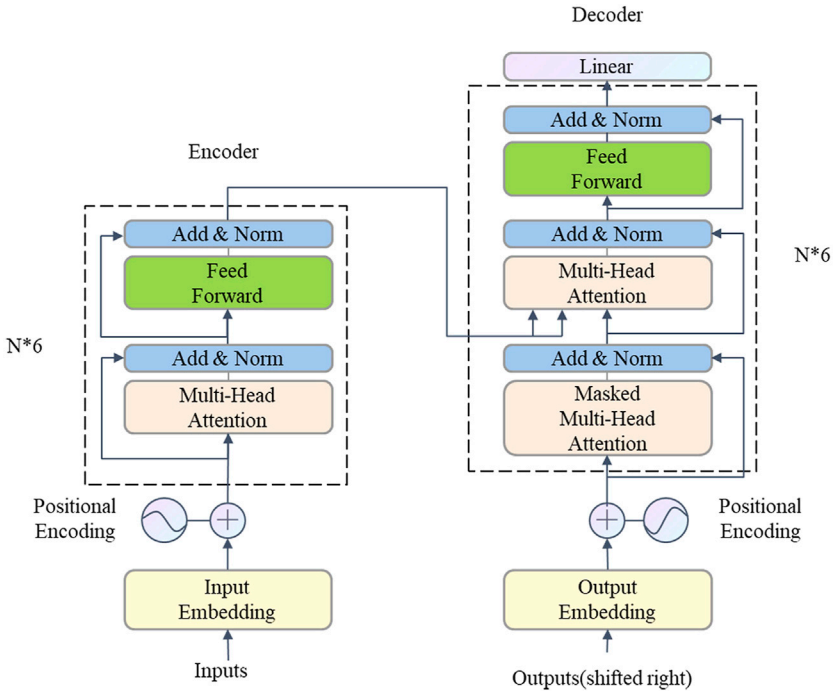


FIGURE 1 | The architecture of BERT.[1]

translation, named entity recognition, reading comprehension, and automatic question answering tasks. Since language models based on transfer training have better performance in natural language processing tasks, this study will use such models to extract the features of rumor content. Specifically, this study uses the post-trained BERT model to extract features from COVID-19 rumor post texts.

BERT is short for Bidirectional Encoder Representations from Transformers proposed by Jacob et al. (2018). Through the mechanism of the transformer network and the transfer learning mechanism, BERT contains vibrant text lexicon information and semantic information. BERT model has been trained on more than 110M corpus and can be directly loaded and used. It is pre-trained by MLM (Masked Language Model) and NSP (Next Sentence Prediction) task. The basic architecture of BERT is shown in **Figure 1**. Rumor text first goes through the BERT tokenizer and creates token embedding, segment embedding, and position embedding in the BERT model. Then the embedding of the text enters the encoder of BERT. The encoder is composed of multi-head attention layers and a feed-forward neural network. After six layers of encoding, the encoded text is embedded into the decoder, composed of a multi-head attention layer and feed-forward neural network. After six layers of decoding, the feature of rumor content is extracted. The multi-head attention mechanism is the critical process to extract text features. It can be formulated as:

$$Q_i = QW_i^Q, K_i = KW_i^k, V_i = VW_i^V$$

$$Head_i = Softmax\left(\frac{Q_i K_i^T}{\sqrt{d_K}}\right)V_i$$

$$MultiHead\,(Q,K,V) = Concat\,(Head_1, Head_2, \ldots, Head_n)W^O$$

where Q represents the input of the decoder in a step, the K and V represent the rumor text embedding. W_i^Q, W_i^k, and W_i^V are the weight parameters of Q, K, and V. d_K is the number of dimensions in K to scale the dot product of Q_i and K_i. $Head_i$ represents the output of the i-th attention head layer. W^O is the weight parameters for concatenated outputs. $MultiHead\,(Q,K,V)$ represents the final output of the multi-head attention layer.

Existing studies have shown that post-train on BERT by domain-specific corpus can significantly improve the performance on the natural language processing task in specific domains [25]. In combination with the COVID nine rumor detection task, post-training on BERT was carried out through a COVID 19 Twitter dataset [26] and a COVID19 Weibo dataset [27], respectively. Specifically, we use the MLM task to post-train BERT so that our BERT model contains more semantic and contextual information on COVID 19-related posts from social networks. This study uses BERT and Chinese BERT in the PyTorch version released by Huggingface[2] as our primary model.

[1]The figure is modified based on: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin "Attention is all you need." In Advances in neural information processing systems, pp. 5998–6008. 2017.

[2]https://github.com/huggingface/pytorch-transformers

An Example of a Rumor Post and Its User Responses:

Twitter Post:

"CDC is preparing for the 'likely' spread of coronavirus in the US, officials say https://t.co/cm9pRyVTcU Do we have anyone left in the CDC who knows what the fuck they are doing," Mon Feb 24, 2020.

User Responses:

– "Georgia Doctor Appointed Head Of The CDC: Health News Dr. Brenda Fitzgerald, who leads the Georgia Department of Public Health, has been appointed CDC director. She'll take over as the Trump administration seeks big cuts to the CDC's budget." Mon Feb 24 10:40:44 + 0000 2020, 0, 1, 1, 49:33.4.

– "@NikitaKitty @PerfumeFlogger We used to. This a travesty.", Mon Feb 24 10:47:35 + 0000 2020, 1, 0, 0, 49:33.4.

–"As we've reported, that would include a $186 million cut to programs at the CDC's center on HIV/AIDS, hepatitis and other sexually transmitted diseases.", Mon Feb 24 10:43:18 + 0000 2020, 1, 2, 2, 49:33.4.

– "The CDC's chronic disease prevention programs, such as those for diabetes, heart disease, stroke and obesity, would be cut by $222 million. What will she do stave the fucking virus?", Mon Feb 24 10:43:18 + 0000 2020, 1, 4, 4, 49:33.4.

After the pre-training of 20 epochs on the COVID-19 dataset, we post-trained the original BERT and original Chinese BERT to the COVID-19 Social Network BERT (CSN-BERT) model.

User Response Feature Extractor

Users on social networks would reply or forward a Twitter, whether it is a true rumor or a false rumor. These responses and retweets contain users' views. Some of these views are to the rumor, and others are to other users' responses or retweets. The user responses and retweets can be considered as discussions or arguments around the rumor. An example of a Twitter post's user responses is shown below. Typically, the responses and retweets can be seen as a tree structure. Rumors and their responses and retweets are called conversational threads. Many studies focused on the tree structure consisting of user responses and retweets and determine rumor veracity based on its structure known as propagation path. However, they do not pay much attention to the content of user responses. Because COVID-19 rumors are more likely to cause panic, there are stricter time constraints for discovering these rumors. In limited time, the structure of responses and retweets, the propagation path, may not be comprehensive enough to determine the veracity of rumors. This indicates that we need to dig into the user responses for essential features on rumor detection.

In this study, we focus on the opinions expressed from user responses. We think of user responses as a sequence, $R_i = \{s_1^i, \ldots s_j^i, \ldots, s_{m_i}^i\}$. The sequence is arranged by response time. To be sure, the original rumor post is not recorded in the sequence. This responses sequence is constructed with time limits. We start with the time of the first responses or retweets and only record responses within 24 h. For the response sequence R_i, we need to extract features from the user response sequence R_i for rumor detection. In order to extract features from the user response sequence, we proposed COVID-19 Response-LSTM (CR-LSTM) to learn about the user response sequences. We implemented the post-trained BERT model (CSN-BERT) mentioned in *Rumor Content Feature Extractor* and a textCNN extractor to learn the sentence embedding of each user response. To be specific, BERT's [CLS] vector is used to represent the feature of user responses. The structure of the entire model is shown in **Figure 2**.

For a user response, its sentence embedding firstly generated through the CSN-BERT. Then, the sentence embedding enters a bidirectional LSTM layer in the order of release time. Each hidden layer in the LSTM layer corresponds to a response, denoted as $\vec{h}_t^i$. After encoding by two LSTM layers, the vector is weighted by an attention layer. We use the multi-head self-attention mechanism to find the responses that have more influence on the results. This process can be represented as:

$$\vec{h}_t^i = \overrightarrow{LSTM}_i\left(\vec{h}_{t-1}^i, e_t^i\right)$$

$$\overleftarrow{h}_t^i = \overleftarrow{LSTM}_i\left(\overleftarrow{h}_{t+1}^i, e_t^i\right)$$

$$L_i = Concat_1^n\left(Softmax\left(\left[\frac{\vec{h}_t^i, \overleftarrow{h}_t^i\right]e_t^i}{\sqrt{d_K}}\right)e_t^i\right)W^O$$

where $\overrightarrow{LSTM}_i$ indicates the encoding operation in the forward direction, and $\overleftarrow{LSTM}_i$ in the backward direction. $\vec{h}_t^i$ represents the forward hidden state of the t-th embedding in R_i, also corresponding to word s_j^i in R_i, which is calculated by its previous hidden state $\vec{h}_{t-1}^i$ and current post sentence embedding e_t^i. $\overleftarrow{h}_t^i$ represents the backward hidden state of the t-th embedding in R_i. The hidden state of the t-th embedding is obtained by concatenating $\vec{h}_t^i$ and $\overleftarrow{h}_t^i$, denoted by $h_t^i = [\vec{h}_t^i, \overleftarrow{h}_t^i]$. L_i is the final embedding of the CR-LSTM. $Concat_1^n\left(Softmax\left(\left[\frac{\vec{h}_t^i, \overleftarrow{h}_t^i\right]e_t^i}{\sqrt{d_K}}\right)e_t^i\right)W^O$ indicates the multi-head self-attention.

The Full View

Combining the rumor content feature extractor and the user response feature extractor, we can extract the integrated rumor feature. For a rumor $r_i = \{x_i, s_1^i, \ldots s_j^i, \ldots, s_{m_i}^i\}$ in a rumor dataset $R = \{r_1, r_2, \ldots, r_n\}$, the rumor content feature extractor (CSN-BERT) can extract the rumor content feature C_i from x_i. The user response feature extractor (CR-LSTM) can extract the user response feature L_i from $\{s_1^i, \ldots s_j^i, \ldots, s_{m_i}^i\}$.

We concatenate the user response feature L_i extracted by CR-LSTM with the rumor content feature C_i extracted by CSN-BERT into the integrated rumor feature. The rumor detection feature then goes through a fully-connected layer dimension, activated by Relu function, and at last output the probability distribution of

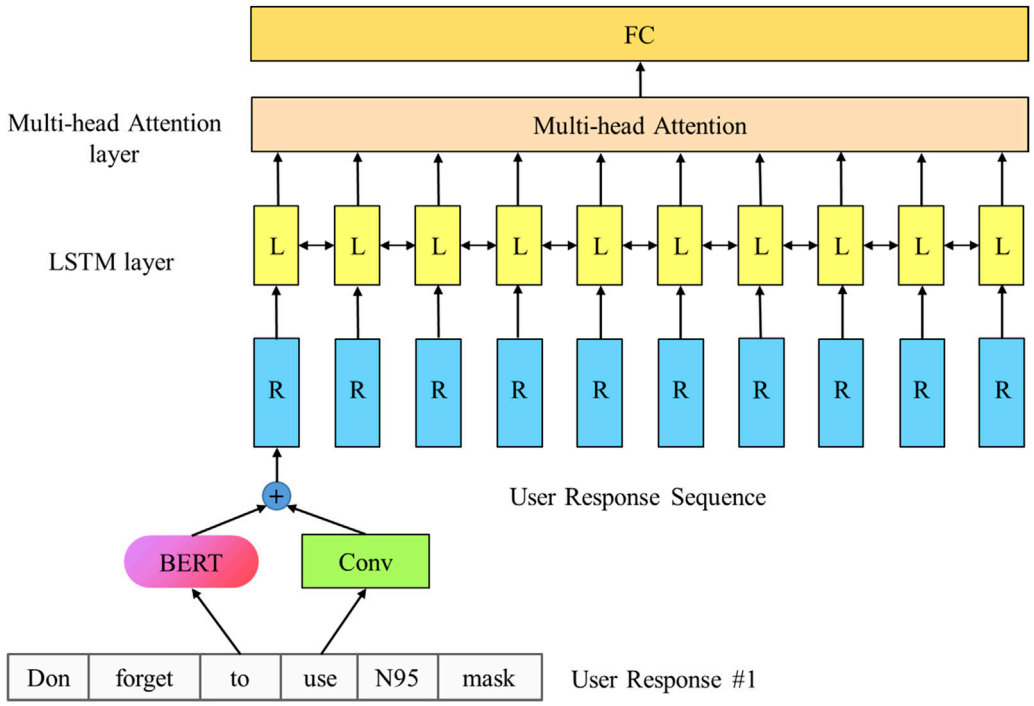


FIGURE 2 | The architecture of CR-LSTM

rumor detection by a Softmax function. The total model is called CR-LSTM-BE (COVID-19 Response LSTM with BERT Embedding). The full view of our model is shown in **Figure 3**.

EXPERIMENTS

In this section, we introduced the experimental preparation and the experimental results. *Dataset* introduces the two datasets for the COVID-19 rumor detection task used in our experiments. *Evaluation Metrics* introduces the evaluation metrics with the computing methods. *Experiment Settings* introduces the experiment settings, especially the hyperparameters selected in the experiments. *Results* presents the experimental results and compares and analyzes the results with baseline methods.

Dataset

To confirm the performance of the CR-LSTM-BE model proposed by us on the COVID-19 rumor detection task. Since there are not many datasets for COVID-19 rumors and considering the data requirements, this study conducted experiments on two datasets. The datasets selected to conduct experiments are the COVID-19 rumor dataset and the CHECKED dataset. The experimental results and related indicators tested the performance of the CR-LSTM-BE model.

The COVID-19 rumor dataset is provided by Cheng et al. [28] and consists of rumors from two types of sources. One is news from various news sites, and the other is from Twitter. There are 4,129 news and 2,705 Twitter posts in this dataset. This study focuses on COVID-19 rumor detection on the social network, so only the Twitter post part of the dataset is selected as the experimental data. The Twitter part of the dataset contains rumor Twitter post id (Hashed), Twitter post content, rumor label (True, False or Unverified), number of likes, number of retweets, number of comments, user responses over a while, user response time and stance of user response. This study mainly used the Twitter post content in the dataset and the user responses of each Twitter post within 24 h to conduct experiments.

The CHECKED data set was provided by Yang et al. [18], and the data came from the Chinese Weibo social network. This dataset contained 2,104 tweets. The dataset contains the rumor microblog's post id (hashed), microblog's post id content, rumor label (True or False), user id (hashed), the time the microblog was posted, number of likes, number of retweets, number of comments, user responses over some time, user retweet over some time, user response time, and user retweet time. This study mainly used the contents of the rumor microblog and the responses and retweets of each microblog within 24 h to conduct experiments. Statistics of the relevant data are shown in **Table 1**. We randomly split the two datasets into the training set, validation set, and testing set with the proportion of 70, 10, and 10%, respectively.

Due to the uncontrollable quality of user response data, we performed resample on the data while preprocessing the user response data. Specifically, we removed user responses that are very concise (less than three words), contain more emoji (over 80%), and have only one hyperlink without other information.

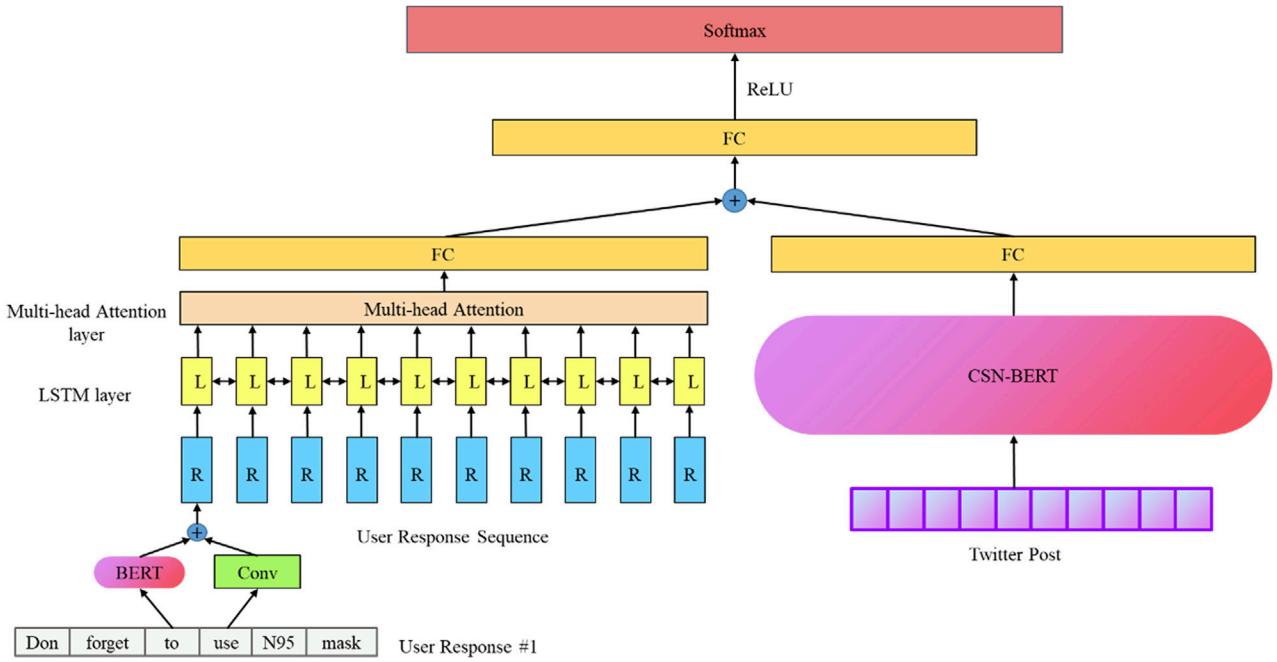


FIGURE 3 | The full view of CR-LSTM-BE.

Evaluation Metrics

The evaluation metric followed most of the existing studies, which regards rumor detection as a classification task. We used the Macro F1, precision score, recall score, and accuracy to evaluate the performance of our model. Macro F1 is used because the labels of rumor posts are imbalanced, which means the distribution is skewed. Marco F1 allows us to evaluate the classifier from a more comprehensive perspective. The precision and recall score in our evaluation is also macro. The definitions of precision, recall, Marco F1, and accuracy are shown below:

$$Precision_c = \frac{TP_c}{TP_c + FP_c}$$

$$Recall_c = \frac{TP_c}{TP_c + FN_c}$$

$$\mathrm{F1}_c = \frac{2*(Recall_c * Precision_c)}{Recall_c + Precision_c}$$

$$\text{Marco F1} = \sum_{c=1}^{n} \mathrm{F1}_c / n$$

$$\text{Accuracy} = \frac{Correct\ Predictions}{all\ samples}$$

where c is the label of a rumor, which could be True, False, or Unverified. TP_c stands for the true positives of rumor label c, which means that the actual label of this rumor is c, and the predicted label is also c. FP_c stands for the false positives, which means that the actual label of this rumor is not c, but the predicted one is c. FN_c stands for false negatives, which means that the actual label c, but the predicted label is not c. Macro F1 was used to integrate all $\mathrm{F1}_c$.

Experiment Settings

In our experiments, we fine-tuned the CSN-BERT on rumor veracity classification task. To prevent overfitting, we disabled backpropagation of CSN-BERT while training the CR-LSTM-BE model. We implemented our model by Pytorch, and the bias was initialized to 0. We used the dropout mechanism to prevent the model from quickly overfitting, the dropout rate was set to 0.5. Random Search method [29] was used to find the optimum hyperparameters. For post training the BERT model and fine-tuning the CSN-BERT, AdamW optimizer [30] was applied with an initial learning rate 1e-5 for model updating, and a mini-size batch of 16 was set. Early stopping is used, and the patience was set to five epochs. In the CR-LSTM-BE model, the optimum number of RNN layers is one and the optimum hidden size is 512. the one optimum number of attention head is 8, and the optimum attention size is 512. We used the Word2vec [31] embedding to initialize word embedding vectors in the textCNN part of the CR-LSTM-BE model, the word embedding vectors were pretrained on English corpus provided by Google. The dimension of word embedding vector was set to 300. For training the CR-LSTM-BE model, Adam optimizer [32] was applied with an initial learning rate 1e-3 for model updating, and a mini-size batch of 16 was set. Early stopping is used, and the patience was set to 15 epochs. All the experiments were done on a GeForce TITAN X.

RESULTS

The datasets adopted in this study do not provide detailed rumor detection results based on different methods. The COVID-19 rumor dataset provides rumor detection results for all data,

TABLE 1 | Statistics of the datasets.

	COVID-19-rumor dataset	CHECKED dataset
Sentence per tweets	1.39	4.76
Words per sentence	11.4	25.96
Words per tweets	15.87	123.67
Total words	42,939	260,197
Total tweets	2,705	2,104
Total responses	34,963	2997063

including news and Twitter data. However, only the Twitter dataset was used in this study. The CHECKED dataset includes benchmark results of FastText, TextCNN, TextRNN, Att-TextRNN, and Transformer methods, but the test only gives Macro F1 score, which lacks more specific indicators such as accuracy and F1 scores on different labels. In order to compare and analyze the performance of our model. We set up several baseline methods based on rumor content features. Referring to related studies and the CHECKED dataset, baseline methods in this study include SVM classifier with word bags, textCNN with word2vec embedding, TextRNN with word2vec embedding, AttnRNN with word2vec embedding, Transformer with word2vec embedding, and BERT-base. We used the Word2Vec embedding pretrained on the English corpus published by Google and the Word2Vec embedding pretrained on the Chinese corpus published by Sogou.

We repeatedly conducted experiments with each method ten times in our study. With the results of the ten experiments, the median of Macro F1 in each group was selected as the experimental results for comparison. We conducted the t-test to confirm if the proposed model performed significantly differently from the baseline methods. The results of the t-test show a significant improvement (p-value<0.05) between CSN-BERT and the baseline methods, CR-LSTM-BE and the baseline methods, and CR-LSTM-BE and CSN-BERT. The experimental results of this study in the COVID-19 rumor dataset are shown in **Table 2**. According to the experimental results, the best-performed method in the baselines is the BERT-base, of which the precision, recall, Marco F1, and accuracy score achieved 55.22, 55.53, 55.34, and 55.42, respectively. In our methods, the post-trained CSN-BERT model showed significant improvement on the data set. Its precision, recall, Marco F1, and accuracy score achieved 58.47, 58.64, 58.55, and 58.87, respectively. Compared to the best-performed baseline, the CSN-BERT showed a 5.8% improvement on Macro F1. The CR-LSTM-BE method based on rumor content feature and user responses proposed in this study has achieved the best performance in the COVID-19 rumor dataset. The precision, recall, Marco F1, and accuracy score of the CR-LSTM-BE achieved 63.15, 64.39, 63.64, and 63.42, respectively. Compared to the best-performed baseline, the CR-LSTM-BE improves 15.0% on Macro F1. Compared to the post-trained CSR-BERT method, this is an 8.7% improvement on Macro F1.

TABLE 2 | Performance on the COVID-19 rumor twitter dataset.

Methods	Precision	Recall	Macro F1	Accuracy
WB-SVM	43.20	43.50	43.33	43.35
textCNN	52.87	52.82	52.80	53.45
textRNN	51.18	51.83	51.45	51.35
attnRNN	51.79	53.04	52.23	51.97
Transformer	52.85	52.63	52.72	52.22
BERT-base	55.22	55.53	55.34	55.42
CSN-BERT	58.47	58.64	58.55	58.87
CR-LSTM-BE	63.15	64.39	63.64	63.42

TABLE 3 | Performance on the CHECKED dataset.

Methods	Precision	Recall	Macro F1	Accuracy
WB-SVM	62.35	69.21	62.33	70.09
textCNN	81.99	89.08	84.76	89.87
textRNN	71.57	81.00	73.77	80.54
attnRNN	81.98	91.44	85.32	89.87
Transformer	84.84	92.36	87.82	91.93
BERT-base	95.74	98.16	96.89	98.10
CSN-BERT	97.13	99.32	98.18	98.89
CR-LSTM-BE	100.00	100.00	100.00	100.00

The experimental results of this study on the CHECKED dataset are shown in **Table 3**. According to the experimental results, the best-performed method in the baselines is the BERT-base, of which the precision, recall, Marco F1, and accuracy score achieved 95.74, 98.16, 96.89, and 98.10, respectively. In our methods, the precision, recall, Marco F1, and accuracy score of the post-trained CSN-BERT model achieved 97.13, 99.32,98.18, and 98.89, respectively. Compared to the best-performed baseline, the CSN-BERT slightly improved Macro F1 (1.3%). The CR-LSTM-BE method based on rumor content feature and user responses proposed in this study has achieved the best performance in the CHECKED dataset. The precision, Recall, Marco F1, and accuracy score of the CR-LSTM-BE all achieved 100. Compared to the best-performed baseline, the CR-LSTM-BE improves 3.2% on Macro F1. Compared to the post-trained CSN-BERT method, this is a 1.9% improvement on Macro F1.

DISCUSSION

In this section, we discussed the performance and the characters of our proposed models. *Improvement Analysis* analyzes the improvements of CSN-BERT and CR-LSTM-BE compared with the baseline methods. Number of Responses Analysis analyzes the effect of the number of responses to rumor detection.

Improvement Analysis

Among the methods experimented in this study, CSN-BERT has a particular improvement than the baseline methods according to the experimental results, which indicates CSN-BERT has a better performance in the feature representation of rumor content by post-training COVID-19 twitter dataset than the original BERT (BERT-base). Compared with general deep learning models (such as textCNN and LSTM), it is not surprising that the BERT model, which is based on transfer learning, performs better in the

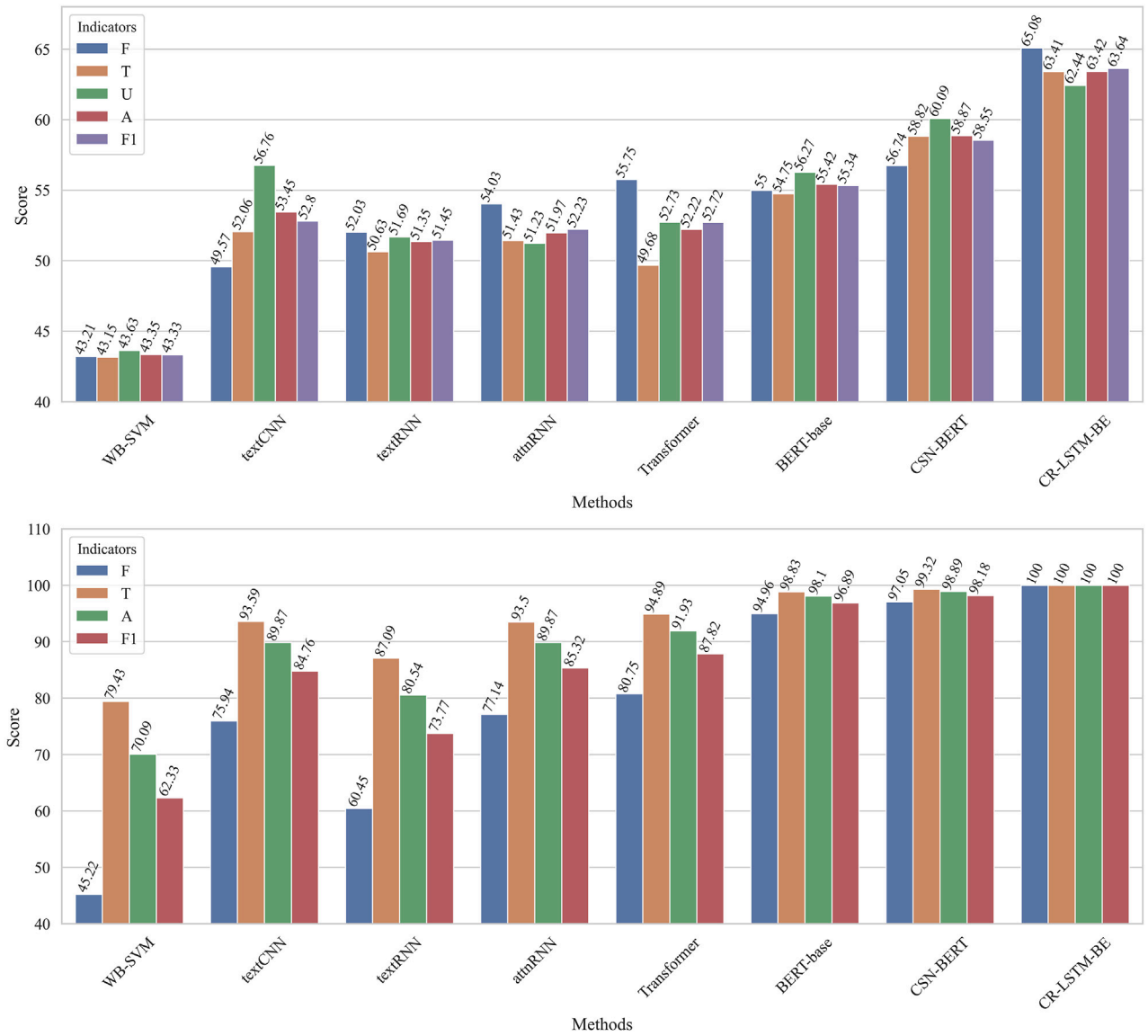


FIGURE 4 | The performance of methods on different rumor labels (F: False, T: True, U: Unverified, A: Accuracy, F1: Macro F1).

problem of rumor detection because the model is based on transfer training has more contextual semantic information—continuing with the idea of allowing the model to acquire more contextual semantic information, CSN-BERT allowing the BERT model to learn more information on COVID-19 discussed by users in the social network in advance. Compared with the original BERT, BERT after post-training is more suitable for COVID-19 rumor detection.

The CR-LSTM-BE proposed in this study adds user responses information into the deep learning model and encodes user responses through the LSTM network with multi-head attention. Use responses contains much information to the original twitter post [33]. In our hypothesis, adding user responses into the model can provide richer information standing for user feedback for the learning process and enable the model to determine the veracity of rumors based on user feedback. The experimental results show that CR-LSTM-BE achieves the best results on both datasets. The experimental results confirmed our hypothesis. In **Figure 4**, we compare the F1 scores of all methods on the various rumor labels (F: False, T: True, U: Unverified). The legend "A" in **Figure 4** is the accuracy, and legend "F1" is the Macro F1. It can be seen that the F1 score on each rumor label of CR-LSTM-BE is better than other methods. In addition, this method can still get a more balanced classification result from unbalanced training data.

Number of Responses Analysis

Interactions on social networks could help better represent user profiles. More user responses can be seen as connections on social network and will provide richer information to describing an

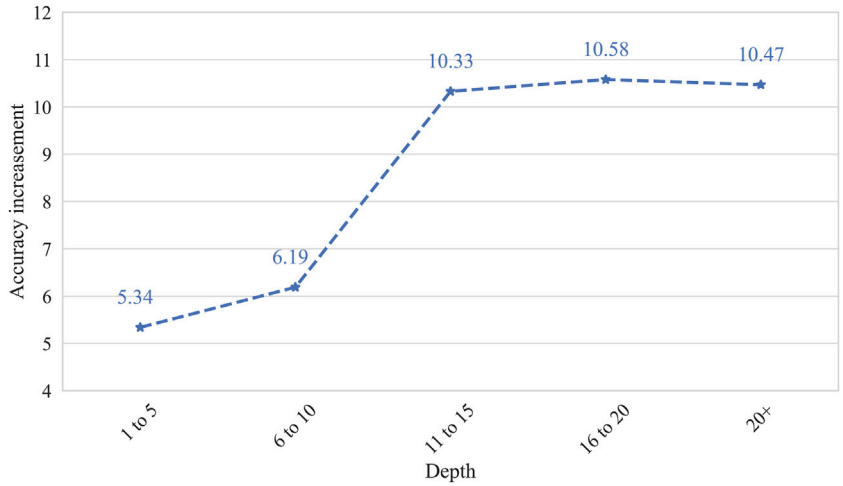


FIGURE 5 | The accuracy increasement of twitter post with various number of responses.

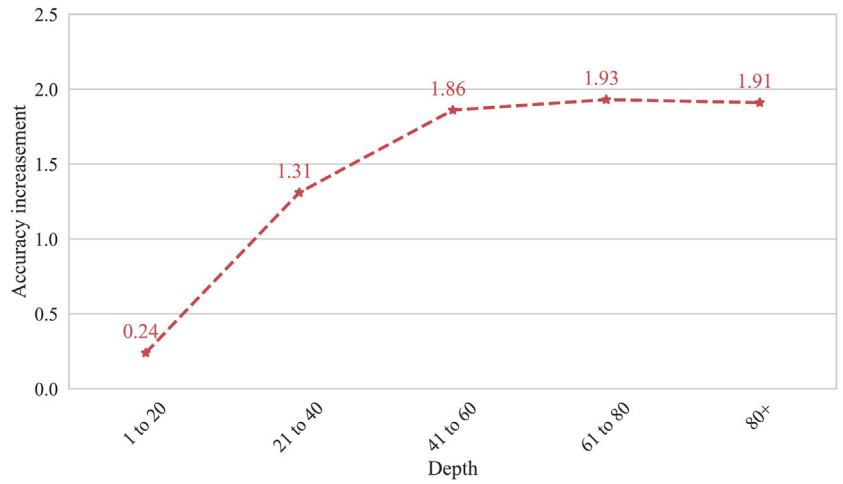


FIGURE 6 | The accuracy increasement of microblog post with various number of responses.

event from a more abundant perspective [34–39]. To further understand the effect of user responses on rumor detection, we compared the accuracy of a different group of Twitter and microblog posts with various responses within 24 h. **Figure 5** shows the rumor detection accuracy improvements of a different group of Twitter and microblog posts with various responses tested on CR-LSTM-BE and CSN-BERT. While the number of user responses is 0, CR-LSTM-BE will degenerate into CSN-BERT, and the accuracy will not be improved. As shown in **Figure 5**, while the number of user responses is 1–5, the accuracy of rumor detection increased by 5.34%. While the number of user responses is 6–10, the accuracy of rumor detection increased by 6.19%. While the number of user responses is more than 11, the accuracy improvement of rumor detection is stabilized at about 10%. This indicates that we should consider including more than 11 user responses for COVID-19 rumor detection on Twitter. For Weibo, due to a large number of retweets and responses, we use another category scheme in the division of the number of user responses. As can be seen from **Figure 6**, the curve of accuracy promotion is similar to that of Twitter (**Figure 5**). While the number of user responses is more than 41, the improvement of rumor detection accuracy tends to be stable. This suggests that we should consider including more than 41 user responses for COVID-19 rumor detection on Weibo.

CONCLUSION

In this study, we proposed rumor detection methods based on the features of rumor content and user responses because of the rapid propagation and prominent domain characteristics of COVID-19 rumor detection on social networks. In order to better capture

and extract rumor content features, we combined the language model based on transfer learning with a post-training mechanism to construct CSN-BERT based on COVID-19 user posts on social networks. In order to make better use of the information in user responses, we further proposed CR-LSTM-BE, which incorporated the information of user responses into the learning process through LSTM. The experimental results show that the post-trained CSN-BERT model can better extract the content features of COVID-19 rumors on social networks than other deep learning models. The CR-LSTM-BE model that integrates user responses achieves the best performance on both datasets. In addition, we found that more user responses can help the CR-LSTM-BE model to achieve better results. On the Twitter network, more than 11 user responses can help to achieve the best performance. On the Weibo network, more than 41 user responses can help to achieve the best performance.

This study focuses on exploring the enhancement of user responses information on rumor detection. Limited by the experimental data, this study did not consider the structural features of user responses and retweets, known as propagation path. Future research will focus on the structural features of user response and retweets and implementing deep learning methods to implement rumor detection better. One direction is to utilize the GCN or hierarchical attention model to incorporate and extract structural and user response features simultaneously.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: DATASET1: https://github.com/MickeysClubhouse/COVID-19-rumor-dataset DATASET2: https://github.com/cyang03/CHECKED

AUTHOR CONTRIBUTIONS

JY and YP conceived and designed the study. JY and YP conducted the experiments. YP reviewed and edited the manuscript. All authors read and approved the final manuscript.

REFERENCES

1. Cinelli M, Quattrociocchi W, Galeazzi A, Valensise CM, Brugnoli E, Schmidt AL, et al. The COVID-19 social media infodemic. *Sci Rep* (2020) 10(1): 16598–10. doi:10.1038/s41598-020-73510-5
2. Mazzeo V, Rapisarda A, and Giuffrida G. Detection of Fake News on COVID-19 on Web Search Engines. *Front Phys* (2021) 9:685730. doi:10.3389/fphy.2021.685730
3. Bian T, Xiao X, Xu T, Zhao P, Huang W, Rong Y, et al. Rumor detection on social media with bi-directional graph convolutional networks. *Aaai* (2020) 34(01):549–56. doi:10.1609/aaai.v34i01.5393
4. Wang Y, Ma F, Jin Z, Yuan Y, Xun G, Jha K, et al. Eann: Event adversarial neural networks for multi-modal fake news detection. In: *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*. Stroudsburg: ACL Press (2018). p. 849–57.
5. Devlin J, Chang M, Kenton L, and Kristina T, Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of NAACL-HTL*. Stroudsburg: ACL Press (2019). p. 4171–86.
6. Brown TB., Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. "Language models are few-shot learners." arXiv [Preprint].14165 (2020). Available at https://arxiv.org/abs/2005.14165 (Accessed September 20, 2021).
7. Qazvinian V, Rosengren E, Radev D, and Mei Q. Rumor has it: Identifying misinformation in microblogs. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg: ACL (2011). p. 1589–99.
8. Kochkina E, Liakata M, and Zubiaga A. "All-in-one: Multi-task learning for rumour verification." arXiv preprint arXiv:1806.03713 (2018).
9. Cao J, Guo J, Li X, Jin Z, Guo H, and Li J. "Automatic rumor detection on microblogs: A survey." arXiv [Preprint] (2018). Available at https://arxiv.org/abs/1807.03505 (Accessed September 20, 2021).
10. Ma J, Gao W, and Wong K. Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning. *Proc 55th Annu Meet Assoc Comput Linguistics* (2017) Vol. 1:708–17.Long Papers)
11. Castillo C, Mendoza M, and Poblete B. Information credibility on twitter. *Proc 20th Int Conf World wide web* (2011) 675–84. doi:10.1145/1963405.1963500
12. Chua AYK, and Banerjee S. Linguistic predictors of rumor veracity on the internet. *Proc Int MultiConference Eng Comp Scientists* (2016) 1:387–91.
13. Yu F, Liu Q, Wu S, Wang L, and Tan T. A Convolutional Approach for Misinformation Identification. *IJCAI* (2017) 3901–7. doi:10.24963/ijcai.2017/545
14. Vosoughi S, Mohsenvand MN, and Roy D. Rumor Gauge. *ACM Trans Knowl Discov Data* (2017) 11:1–36. doi:10.1145/3070644
15. Liu Y, and FangWu YB. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In: *32nd AAAI Conference on Artificial Intelligence*. California: AAAI press (2018). p. 354–61.
16. Kwon S, Cha M, and Jung K. Rumor Detection over Varying Time Windows. *PloS one* (2017) 12:e0168344–1. doi:10.1371/journal.pone.0168344
17. Glazkova A, Glazkov M, and Trifonov T. g2tmn at Constraint@AAAI2021: Exploiting CT-BERT and Ensembling Learning for COVID-19 Fake News Detection. *Commun Comput Info Sci* (2021) 116–27. doi:10.1007/978-3-030-73696-5_12
18. Yang C, Zhou X, and Zafarani R. CHECKED: Chinese COVID-19 fake news dataset. *Soc Netw Anal Min* (2021) 11:58–8. doi:10.1007/s13278-021-00766-8
19. Patwa P, Sharma S, Pykl S, Guptha V, Kumari G, Akhtar MS, et al. Fighting an infodemic: COVID-19 fake news dataset. In: *International Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation*. Cham: Springer (2021) p. 21–9. doi:10.1007/978-3-030-73696-5_3
20. Li Q, Zhang Q, Luo S, and Liu Y. Rumor Detection on Social Media: Datasets, Methods and Opportunities. In: *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship*. Snyder: Disinformation, and Propaganda (2019) p. 66–75. doi:10.18653/v1/d19-5008
21. Zubiaga A, Kochkina E, Liakata M, Procter R, Lukasik M, Bontcheva K, et al. Discourse-aware rumour stance classification in social media using sequential classifiers. *Inf Process Manag* (2018) 54(2):273–90. doi:10.1016/j.ipm.2017.11.009
22. Zhou X, and Zafarani R. "Fake news: A survey of research, detection methods, and opportunities." arXiv [Preprint] (2018). Available at https://arxiv.org/abs/1812.00315v2 (Accessed September 20, 2021).
23. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, et al. Deep contextualized word representations. *Proc NAACL-HLT* (2018) 2227–37. doi:10.18653/v1/n18-1202
24. Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, and Quoc V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Adv Neural Inf Process Syst* (2019) 32:5753–63.
25. Lee J, Yoon W, Kim S, Kim D, Kim S, So CH, et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* (2020) 36(4):1234–40. doi:10.1093/bioinformatics/btz682
26. Lamsal R. Design and analysis of a large-scale COVID-19 tweets dataset. *Appl Intell* (2021) 51(5):2790–804. doi:10.1007/s10489-020-02029-z

27. Leng Y, Zhai Y, Sun S, Wu Y, Selzer J, Strover S, et al. Misinformation during the COVID-19 outbreak in China: Cultural, social and political entanglements. *IEEE Trans Big Data* (2021) 7(1):69–80. doi:10.1109/tbdata.2021.3055758
28. Cheng M, Wang S, Yan X, Yang T, Wang W, Huang Z, et al. A COVID-19 Rumor Dataset. *Front Psychol* (2021) 12(2021):644801. doi:10.3389/fpsyg.2021.644801
29. Bergstra J, and Bengio Y. Random Search for Hyper-Parameter Optimization. *J Machine Learn Res* (2012) 13:281–305.
30. Loshchilov I, and Hutter F. *Fixing weight decay regularization in adam*. arXiv [Preprint] (2012). Available at https://arxiv.org/abs/1711.05101 (Accessed September 20, 2021).
31. Mikolov T, Chen K, Corrado G, and Dean J. *Efficient estimation of word representations in vector space* (2013). arXiv preprint arXiv:1301.3781.
32. Kingma DP, and Jimmy B. *Adam: A method for stochastic optimization* (2014). arXiv preprint arXiv:1412.6980.
33. Tuzón P, Fernández-Gracia J, and Eguíluz VM. From Continuous to Discontinuous Transitions in Social Diffusion. *Front Phys* (2018) 6:21. doi:10.3389/fphy.2018.00021
34. Omodei E, De Domenico M, and Arenas A. Characterizing interactions in online social networks during exceptional events. *Front Phys* (2015) 3:59. doi:10.3389/fphy.2015.00059
35. Bellingeri M, Bevacqua D, Scotognella F, Alfieri R, Nguyen Q, Montepietra D, et al. Link and Node Removal in Real Social Networks: A Review. *Front Phys* (2020) 8:228. doi:10.3389/fphy.2020.00228
36. Lou J, Xu Z, Zuo D, Zhang Z, and Ye L. Audio Information Camouflage Detection for Social Networks. *Front Phys* (2021) 9:715465. doi:10.3389/fphy.2021.715465
37. Bu Z, Li H, Zhang C, Cao J, Li A, and Shi Y. Graph K-means based on leader identification, dynamic game, and opinion dynamics. *IEEE Trans Knowledge Data Eng* (2019) 32(7):1348–61.
38. Wang Z, Xia C, Chen Z, and Chen G. Epidemic Propagation with Positive and Negative Preventive Information in Multiplex Networks. *IEEE Trans Cybern* (2020) 51 (3):1454–62. doi:10.1109/TCYB.2019.2960605
39. Wang Z, and Xia C. Co-evolution Spreading of Multiple Information and Epidemics on Two-layered Networks Under the Influence of Mass Media. *Nonlinear Dyn* (2020) 102:3039–52. doi:10.1007/s11071-020-06021-7

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

frontiers in Physics

ORIGINAL RESEARCH
published: 25 October 2021
doi: 10.3389/fphy.2021.768093

A Local Search Algorithm for the Influence Maximization Problem

Enqiang Zhu[1], *Lidong Yang*[1] *and Yuguang Xu*[2]*

[1]*Institute of Computing Science and Technology, Guangzhou University, Guangzhou, China,* [2]*Medical Artificial Intelligence Research Institute, Binzhou Medical University (Yantai Campus), Yantai, China*

How to select a set of top k nodes (called seeds) in a social network, through which the spread of influence under some certain diffusion models can achieve the maximum, is a major issue considered in the social network analysis. This problem is known as the *Influence Maximization Problem* (IMP). Due to its **NP**-hard nature, designing a "good" algorithm for the IMP is a very challengeable work. In this paper, we propose an efficient local search algorithm called DomIM to solve the IMP, which involves two main ideas. The first one is an approach to constructing an initial solution based on a dominating set, while the second is a degree based greedy strategy in the local search phase. DomIM is evaluated on three real world networks, under three widely-used diffusion models, including independent cascade (IC) model, weighted cascade (WC) model, and linear threshold (LT) model. Experimental results show that DomIM is competitive and efficient, and under all of these diffusion models it can obtain the best performance (in terms of solution quality) on the networks we consider.

Keywords: social network, influence maximization, dominating set, local search, heuristic

OPEN ACCESS

Edited by:
Chengyi Xia,
Tianjin University of Technology, China

Reviewed by:
Changjun Zhou,
Zhejiang Normal University, China
Ye Liu,
University of Illinois at Chicago, United States

***Correspondence:**
Yuguang Xu
xuyuggmw@bzmc.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *31 August 2021*
Accepted: *23 September 2021*
Published: *25 October 2021*

Citation:
Zhu E, Yang L and Xu Y (2021) A Local Search Algorithm for the Influence Maximization Problem. Front. Phys. 9:768093. doi: 10.3389/fphy.2021.768093

1 INTRODUCTION

A social network is an interconnected structure which consists of a set of socially relevant nodes (e.g., individuals, groups, organizations, or related systems) connected with one or more relations, such as shared ideas, social contacts, financial stock exchanges, and affinities [1,2]. Needless to say, exploring valuable information related to nodes and revealing relations between them are very meaningful and significant. For this, many topics have been introduced to analyze social networks, from a different perspective; please refer to [3,4] for an overview of social network analysis.

One of the most studied problems in the social network analysis is the influence maximization problem (IMP), whose task is to select a set of k nodes from a given social network, called *seed set*, through which the number of influenced nodes under some certain diffusion model can achieve the maximum. Due to its potential applications in practice, the IMP has attracted wide-spread attention. Especially, in today's era, with the rapid development in the communication field, the size of social networks is becoming increasingly large. As a consequence, information exchange among users throughout social networks has become an indispensable part in our daily life, and meanwhile a large number of users may be influenced by such information diffusion. So, there is a growing body of literature analyzing the influence and information propagation in social networks [5–8].

The well-known application of the IMP is viral marketing, which aims to exploit the network value of customers, i.e., the potential influence of a customer who may recursively influence his neighbors (e.g., family members, colleagues, friends, friend's colleagues, friend's friends, and so on) to buy a product through the "word-of-mouth" propagation [9]. Clearly, a small number of highly influential customers can be specified as potential customers to market to, so that the expected profit

can be maximized. Besides viral marketing, there are also many other applications, e.g., analyzing human behavior [10], target advertisement [11], rumor blocking [12], social recommendation [13], etc. Practically, the spread of influence can occur with the aid of some operational models. Three widely-used diffusion models are *independent cascade (IC) model*, *weighted cascade (WC) model*, and *linear threshold (LT) model*, where the WC model is a special case of the IC model [14]; see **Section 2** for an detailed discussion of these models.

1.1 Related Works

The IMP in social networks was first studied in 2001 by [9], who regarded it as an algorithm problem. Since then, it has been studied extensively, especially after the work by [14] who proved that the IMP is **NP**-hard by defining it as a combinatorial optimization problem. Nevertheless, it is still challengeable to solve the IMP, due to the following two difficulties: the first one is how to accurately measure the influence of a given seed set, which has been shown to be ♯**P**-hard; the second is how to select a seed set with the maximum influence [15,16]. We make an overview of related works on the IMP from the following two aspects: greedy based approaches and heuristic approaches. For more detailed categories on this problem, please see the survey papers [17–19].

1.1.1 Greedy Based Approach

It is widely believed that the initial work using greedy based idea to solve the IMP is attributed to [14], who proposed a simple hill-climbing greedy algorithm to solve the IMP under the IC model and the LT model. Despite the algorithm can get a guarantee that obtains the optimal solution with a high probability (about 63%), it is very time-consuming, because it has to search for the whole network (every node) and implement tens of thousands Monte-Carlo simulations. To optimize the efficiency of the simple greedy algorithm, [20] proposed an improved greedy algorithm with an approximation ratio of $\frac{1}{2}(1-\frac{1}{e})$, called CELF, which selects influential nodes leveraging the submodular property. They showed that CELF can achieve up to 700 times faster than the simple greedy algorithm. Whereas, CELF has a poor performance in large network since it has to compute the marginal influence spread of each alternative node repeatedly [21]. [22] designed new schemes to optimize the greedy algorithm under the IC model, by which they generated a faster greedy algorithm based on CELF. [23] developed an improved version of CELF, called CELF++, and showed that it is 35–55% faster than CELF. [24] proposed a deprecation based greedy algorithm for the IMP, called DGS. This algorithm first orders the nodes of a social network by applying three heuristic influence functions, and then selects the most influential nodes from a list of pre-ordered vertices. Although DGS takes less time than CELF, it is still time-consuming in large networks. In [25], Heidari et al. proposed a fast greedy algorithm SMG to solve the IMP. By reducing calculations in counting the traversing nodes and Monte-Carlo graph construction, SMG improves the efficiency of greedy algorithms. To deal with the time-consuming drawback of greedy algorithms, [26] proposed a CascadeDiscount algorithm for solving the IMP. The algorithm uses PageRank to measure the initial influence of nodes, measures node's marginal gain of influence spread by considering the influence loss on their neighbors, and then selects the most influential nodes based on a greedy strategy. [27] proposed a community-based framework for the IMP, which was further improved in [28] by designing an objective function to evaluate the influence spread and then generating an efficient greedy algorithm to find the influential nodes.

A simple greedy algorithm can yield nearly optimal solutions, but it is often time-consuming, which limits its application on large-scale networks. As a useful technique to deal with **NP**-hard problems, heuristic approaches have been widely used in a variety of problems, such as partition coloring problem [29], network immunization [30], dominating set problem [31], etc. Also, heuristic algorithms for the IMP are proposed sequentially.

1.1.2 Heuristic Approach

To solve the low efficiency of simple greedy algorithms, [22] in 2009 proposed a degree discount heuristics to improve influence spread, by considering the degree discount of a candidate node caused by its seed neighbors. However, compared with greedy algorithms, the algorithm has a poor accuracy, though it reduces the running time. Later on, [32] introduced a heuristic approach called MIP to measure node's influence from other nodes, by which a heuristic algorithm called PMIA was developed to solve the IMP on large-scale social networks. The drawback of PMIA is that it has to design different thresholds for different networks and there is no uniform method to set the thresholds, which may affect the accuracy of the algorithm. Since then, a large body of heuristic algorithms for the IMP are developed. In 2011, [33] designed a simulated annealing based algorithm for the IMP under the IC model, which integrates two heuristic approaches to optimize the convergence process and a method to speed up the selection of the most influential nodes. [34] proposed a scalable influence approximation algorithm IPA for the IMP under the IC model, which uses an independent influence path to estimate the influence of nodes. For the purpose of bridging the theory and practice in influence maximization, [35] proposed an algorithm called TIM. They showed that TIM runs in $O((k+\ell)(n+m)\log\frac{n}{\varepsilon^2})$ time and guarantees an approximation ratio of $1-\frac{1}{e}-\varepsilon$ (with at least $1-n^{-\ell}$ probability). By utilizing the genetic approach and the strength greedy algorithm, [36] proposed an efficient algorithm for solving the IMP in social networks. Based on evolutionary methods, [37] introduced a simple genetic algorithm for the IMP. In [38], Kim proposed a Random Walk and Rank Merge based algorithm, which uses a random walk method to speed up the algorithm. [39] analyzed the reason why the greedy approaches have low efficiency and proposed a degree-descending search strategy, based on which they designed an evolutionary algorithm. By eliminating the time-consuming simulations in a greedy algorithm, the efficiency of the algorithm is improved significantly. Recently, [16] proposed a discrete shuffled frog-leaping algorithm for the IMP, which selects influential nodes based on network topology characteristic. In [21], Qin et al. introduced a discount-degree descending technology and lazy-forward technology to identify a set of candidate nodes, based on which they designed a two-stage selection algorithm for the IMP in social networks. [6] proposed a path-based approach, which

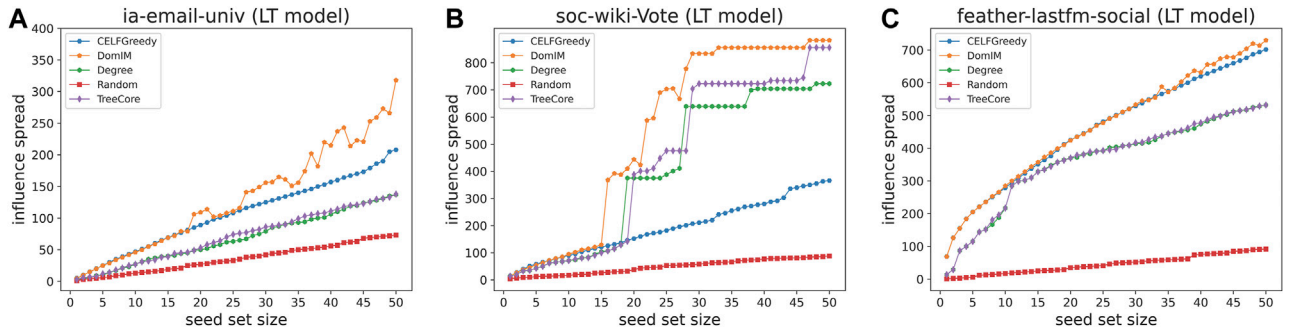


FIGURE 1 | Results of influence spread under LT model.

uses the degree and the independent influence path to estimate the influence spread and uses a heuristic method to reduce the computation volume.

The heuristic algorithms usually have better running time and scalability. But, they cannot provide any performance guarantee.

1.2 Contribution

In this paper, we propose an efficient local search algorithm named DomIM to solve the IMP in social networks. Our contributions mainly include the following three aspects.

(1) We propose a mechanism to construct a high quality initial solution based on dominating set, and an approach to building candidate set.
(2) A degree based greedy strategy is introduced in the local search.
(3) DomIM is evaluated on three real world graphs, under IC model, WC model, and LT model. Compared with four heuristic algorithms, DomIM is competitive and efficient, and obtains the best performance on these graphs.

The remainder of the paper is organized as follows. **Section 2** introduces basic definitions, including the influence maximization problem and three diffusion models. **Section 3** gives a brief overview of dominating set problem and a heuristic algorithm for finding minimum dominating set that we will quote. **Section 4** describes our DomIM algorithm. **Section 5** presents experimental results and **Section 6** concludes this paper with future work.

2 PRELIMINARIES

To study the IMP, we often abstract a social network as a graph, where the *vertex set* represents the set of nodes in the social network and *edge set* represents the social ties among nodes. From now on, we use the term "graphs" to replace "social networks", and follow the standard terminologies in graph theory.

All graphs considered in this paper are simple undirected graphs. Let $G = (V, E)$ be a graph with vertex set V and edge set E. We use a 2-length string uv to denote an edge connecting two vertices u and v. The two vertices u, v are called *endpoints* of edge uv. An edge is said to be *incident with* its two endpoints, and the two endpoints of an edge are said to be *adjacent to* each other. A vertex is called a *neighbor* of another vertex, if they are adjacent in G. Given a vertex $v \in V$, the *neighborhood* of v in G, denoted by $N_G(v)$, is the set of neighbors of v, and let $N_G[v] = N_G(v) \cup \{v\}$. The number of neighbors of v in G (or equally the number of edges incident with v), denoted by $d_G(v)$, is called the *degree* of v in G. For a set $S \subseteq V$, we use $G[S]$ to denote the subgraph of G induced by S, i.e., the resulting graph obtained from G by deleting all vertices in $V \setminus S$ and their incident edges.

2.1 Influence Maximization Problem

Given a graph $G = (V, E)$ and a positive number k, the task of the IMP is to find a set S of k vertices (called seed set) such that the influence spread by S [denoted by $\sigma(S)$], i.e., the number of influenced vertices triggered by S, reaches maximum under a given diffusion model. This problem was formulated as an optimization problem by [14], which is shown as follows.

$$S^* = \arg \max_{S \subseteq V, |S|=k} \sigma(S) \tag{1}$$

In **Equation 1**, the maximum is taken over all seed sets S and S^* is the best one that can maximize the spread of influence.

Now, we describe three widely-adopted diffusion models that we will use for the IMP.

2.2 Independent Cascade Model

As the simplest model of dynamic cascade models, the IC model was first investigated by [40]. In this model, the influence spread, starting with a set of active vertices (seed set), follows a randomized rule: an active vertex can activate its inactive neighbors only when it first becomes active. Specifically, let u be a vertex activated at step t. Then, for each inactive neighbor $v \in N_G(u)$, there is a single change for v to be activated by u with probability $p_{u,v}$ (a parameter independent of all previous attempts to active v). If u succeeds, then v will become active at step $t + 1$; otherwise, v is still inactive. Note that whether or not v is activated successfully, it cannot be further activated by u at subsequent steps. If at step t, an inactive vertex u has more than one newly

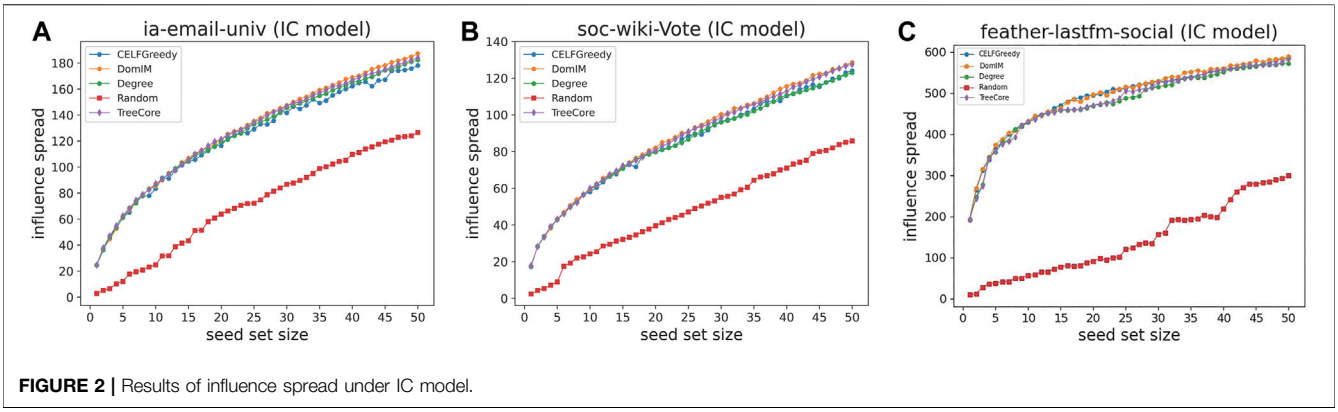


FIGURE 2 | Results of influence spread under IC model.

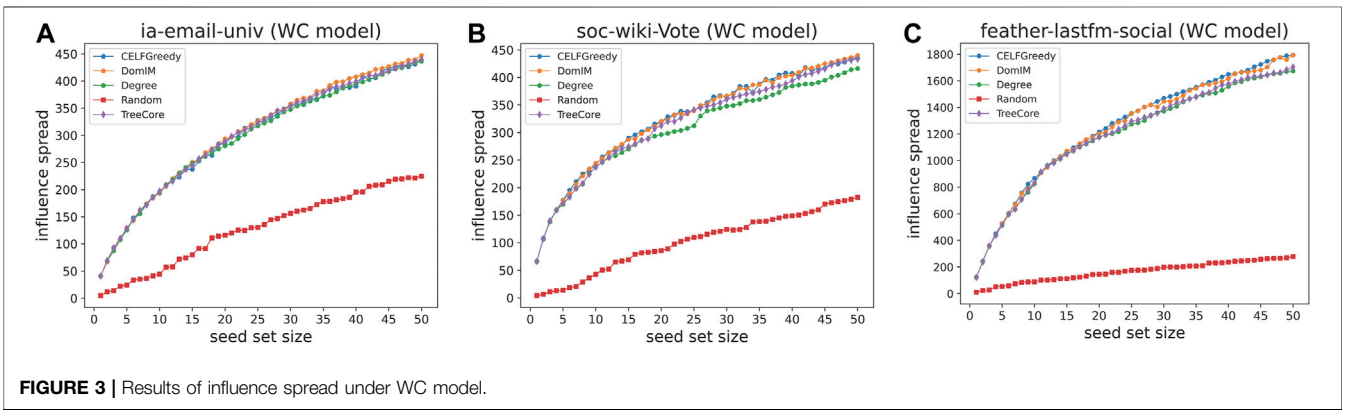


FIGURE 3 | Results of influence spread under WC model.

activated neighbors, then they can activate u one by one in any order. In this way, the diffusion process stops when no more possible vertices will be activated.

2.3 Weighted Cascade Model

The WC model is a special IC model [14], in which a newly activated vertex u activates its inactive neighbor v with a probability related to the degree of v, i.e., $p_{u,v} = \frac{1}{d_G(v)}$. It is clear to see that a high-degree vertex may be activated by each of its activated neighbors with low probability. In a certain sense, this simulates the actual interpersonal relationships. Consider the case that if a person has only one friend, then suggestions from his unique friend will play a very important role in his decisions. In contrast, if a person has many friends, then suggestions from one of its friends may be less important to his decisions.

2.4 Linear Threshold Model

The LT model is different from the IC model and the WC model, which estimates the spread process by using vertex-specific thresholds [14]. In this model, an inactive vertex v is influenced by each of its active neighbor u with a weight $b_{v,u}$, under the limitation of $\sum_u b_{v,u} \leq 1$, where u is taken over all active neighbors of v. Indeed, this limitation has its own significance, since the probability that u can be activated is at most one.

The dynamic process can be described as follows. We preassign randomly a threshold $\theta_v \in [0, 1]$ to each vertex v. Then, start with a seed set as an initial set of active vertices; in sept t ($\geq$2), each active vertex in step $t-1$ ($t \geq 2$) is still active and an inactive vertex v is activated successfully if the total weight of its active neighbors is at least θ_v, i.e.,

$$\sum_{u \in N_G(v) \ is \ active \ at \ step \ t} b_{v,u} \geq \theta_v.$$

It is intuitive that the thresholds of vertices represent the distinct potential tendencies of vertices to become active. Due to the lack of knowledge, we assign the same threshold to all vertices in the experiment.

3 DOMINATING SET

A *dominating set* of a given graph $G = (V, E)$ is a subset S of vertices such that $V \setminus S \subseteq N_G(S)$, where $N_G(S) = \{v | v$ has a neighbor in $S\}$. The minimum dominating set problem (MDS) aims to find a dominating set with the minimum cardinality. The MDS is a classic **NP**-hard problem, which has been widely studied in both theoretical and application aspects [41,42], especially for designing efficient approximation algorithms [43,44]. Given that vertices in a dominating set may have some important properties,

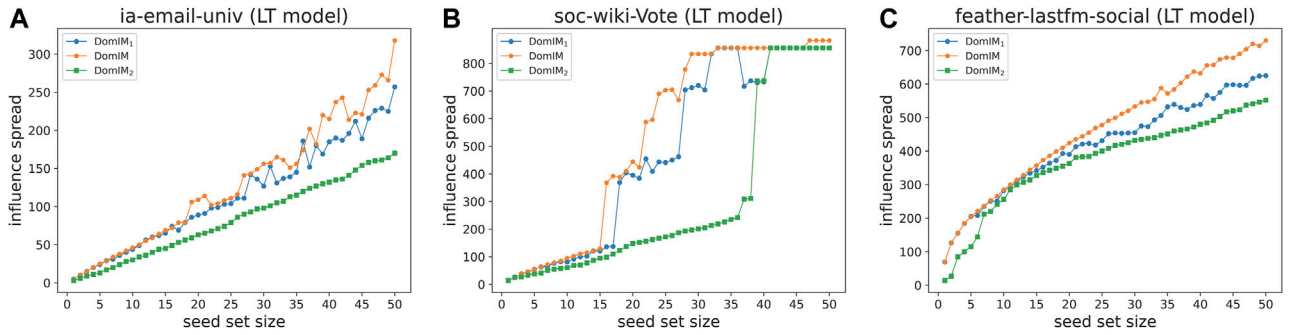


FIGURE 4 | Comparison results between DomIM and its alternatives under LC model.

we have reason to believe that vertices from a (minimum) dominating set can have high influence. So, we can construct an initial solution based on a dominating set of a given social network.

In our algorithm DomIM which will be presented in the subsequent section, an algorithm finding a minimum dominating set will be used. We quote such an algorithm called ScBppw proposed by [31]. Here we present the local search framework of ScBppw for the reader's convenience.

Algorithm 1 | ScBppw [31].

Input: An undirected graph $G = (V, E)$, the *cutoff* time
Output: A dominating set of G

1 **begin**
2 $D \leftarrow$ InitDS(G);
3 **while** *elapsed_time* $<$ *cutoff* **do**
4 **if** D *covers all vertices* **then**
5 $D^* \leftarrow D$;
6 remove a vertex from D with the minimum *loss*, breaking ties randomly;
7 $lastStepImproved \leftarrow false$;
8 **else**
9 ExchangeVertices(G, D)
10 **return** S^*;

Notice that **Algorithm 1** integrates two sub-procedures, InitDS and ExchangeVertices, where InitDS is a simple greedy strategy to generate an initial solution and ExchangeVertices is an exchanging procedure based on a proposed tabu strategy. For more information about this algorithm, please refer to paper [31].

4 THE DOMIM ALGORITHM

We develop a local search algorithm for IMP named DomIM (**Algorithm 2**), which is based on dominating set and a degree-related rule for selecting vertices.

In the beginning, the algorithm finds a minimum dominating set, by which an initial solution will be constructed. Considering that the MDS problem is **NP**-hard, we adopt a fast heuristic algorithm (Algorithm 1) to approximatively find a minimum dominating set of the input graph (line 2). A key concept of our algorithm is the uncorrelated degree.

Definition 1. Let S be a subset of vertices in a graph $G = (V, E)$, and $v \in V$ be an arbitrary vertex. The *S-uncorrelated neighborhood of v*, denoted by $N_{\bar{S}}(v)$, is defined as the set of vertices in $V \setminus S$ that are adjacent to v in G, and the *S-uncorrelated degree of v* is the cardinality of $N_{\bar{S}}(v)$, denoted by $d_{\bar{S}}(v)$, i.e., $d_{\bar{S}}(v) = |N_{\bar{S}}(v)|$.

For a fixed set S, the vertices with the maximum S-uncorrelated degree may have higher influence in some sense, since it can influence more vertices directly when vertices in S are not considered. This observation is used to construct an initial solution and design an approach to improving solutions by exchanging vertices.

The algorithm utilizes a greedy strategy (related to the uncorrelated degree) based on dominating set to construct an initial solution. Notice that when the dominating set D contains less than k vertices, the algorithm selects k–$|D|$ vertices from $V \setminus D$ (according to the S-uncorrelated degree from large to small) to generate an initial solution by adding them into D (lines 3–5); otherwise, the algorithm chooses top k vertices from D in terms of the uncorrelated degrees (as large as possible) as an initial solution (line 7).

After the construction of an initial solution S, the algorithm computes the minimum S-correlated degree of vertices in S, according to which a candidate set T is constructed for the local search phase (Line 9). Note that T may be not large enough to improve the current solution S by exchanging vertices repeatedly between T and S; if this happens, i.e., $|T| < \alpha|V|$, the algorithm selects $\lfloor \alpha|V| \rfloor - |T|$ vertices from $V \setminus (S \cup T)$ with S-uncorrelated degree as high as possible and adds them to T, where α is a real number in the interval (0,1) related to the value of $|V|$ and the type of the diffusion model (lines 10–12).

Subsequently, a loop (lines 13–24) is executed until a given termination condition is reached. DomIM returns the best found seed set S^* (line 25). In each iteration of the loop, a local search process is executed to exchange vertices between the current solution S and the candidate set T for improving the current solution (starting with the initial solution). Specifically, the algorithm chooses a vertex $u \in S$ with the minimum S-uncorrelated degree (randomly select one when there is more than one vertices with the minimum value). Note that it is possible that u is not be selected for the first time; if so, u is reselected randomly (lines 15–16). Then, remove u from S, and

Algorithm 2 | DomIM

Input: An undirected graph $G = (V, E)$, a positive inter k
Output: A seed set S^* of size k

```
1  begin
2      D ← a dominating set obtained by ScBppw (Algorithm 1);
3      if |D| < k then
4          D' ← select top k − |D| vertices from V \ D accordign to the D-uncorrelated degree from large to small;
5          S ← D ∪ D';
6      else
7          S ← a set of k vertices in D with D-uncorrelated degree as large as possible;
8      S* ← S;
9      t ← min{d_S(v)|v ∈ S} and T ← {v|v ∈ V \ S and d_S(v) > t};
10     if |T| < ⌊α|V|⌋ then
11         T' ← a set of ⌊α|V|⌋ − |T| vertices in V \ (S ∪ T) with S-uncorrelated degree as high as possible; // α is a real number in the interval (0,1),
           which is dependent on the value of |V| and the type of the diffusion model
12         T ← T ∪ T';
13     while elapsed_time < cutoff do
14         u ← a vertex in S with the minimum S-uncorrelated degree, breaking ties randomly;
15         if u is chosen more then one time then
16             u ← a random vertex in S;
17         Remove u from S;
18         v ← a random vertex in T, and add v into S;
19         if ComputingInfluence(S) > ComputingInfluence(S*) then
20             S* ← S;
21             Add u into T;
22         else
23             Remove v from S;
24             Add u into S;
25     return S*;
```

select a vertex *v* from *T* randomly (for the diversity) and add it to *S* (lines 17–18). If the exchanging can produce more influence, then it is viewed as a valid process, and update $S^\star$ by *S* and *T* by $T \cup \{u\}$ (for the diversity of solutions) (lines 19–21); otherwise, *S* is back to the previous state (lines 22–24).

5 EXPERIMENTS

We evaluate DomIM on three real world (undirected) networks under the three diffusion models mentioned in **Section 2**, i.e., the LT model, the IC model, and the WC model. The data come from two databases: Network Repository[1] and SNAP (Stanford Large Network Dataset Collection)[2].

ia-email-univ (IEU) [45]: This network is from Network Repository, which is an email communication network at the University Rovira i Virgili in Tarragona in the south of Catalonia in Spain. There are in total 1,133 vertices and 5,451 edges. Each vertex represents a user and an edge connecting two users indicates that one sent at least one email to another.

soc-wiki-Vote (SWV) [45]: This network is also from Network Repository, which involves all the Wikipedia voting data from the inception of Wikipedia till January 2008. This graph contains 889 vertices and 2,914 edges, where vertices represent Wikipedia users and a direct edge from vertex *i* to vertex *j* represents that user *i* vote on user *j*. In our experiment, we consider only the underlying undirected graph of this graph.

feather-lastfm-social (FLS) [46]: This is a social network of LastFM users which was collected from the public API in March 2020. This graph is from SNAP, consisting of 7,624 vertices and 27,806 edges, where vertices represent LastFM users from Asian countries and edges are mutual follower relationships between them.

5.1 Experiment Setup

DomIM is implemented in C++ and complied by g++ 8.2.0. All experiments are run on a computer with Intel i7-8565U 1.80 GHz with 16 GB RAM under Windows 10.

We compare the overall performances of DomIM with four heuristic algorithms, including Degree [14], Random [14], CELFGreedy [20], and TreeCore [47]. Degree is a simple

[1]http://networkrepository.com
[2]https://snap.stanford.edu/data/feather-lastfm-social.html

algorithm that selects high-degree vertices. Random chooses vertices randomly. CELFGreedy is a greedy algorithm with lazy-forward optimization, in which for each candidate seed set, it executes 10,000 simulations to obtain an accurate estimation of influence spread. Therefore, CELFGreedy is time-consuming. TreeCore is an approach based on a network connectivity parameter called tree coritivity.

For each instance, all algorithms are executed 3 times with seed set size from 1 to 50, from which we select the best solutions for each situation. The time limit of each run is at most 90 s, which is dependent on the size of networks.

5.2 Results on Real World Social Networks

Experimental results under the three diffusion models are shown in three groups of figures (**Figures 1–3**), where each group contains three figures corresponding to the results on the three networks we consider, respectively [1) for the IEU network, 2) for the SWV network, and 3) for the FLS network]. In each figure, the *x*-axis represents the size of seed set (denoted by seed set size which is from 1 to 50) and the *y*-axis represents the number of all vertices that are activated at the end of the diffusion process (denoted by influence spread). Each figure depicts the results obtained by five different algorithms (represented by distinct colors), where CELFGreedy, Degree, Random, and TreeCore are the four approaches mentioned above, and DomIM is our algorithm.

5.2.1 Results Under the LT Model

Under the LT model, all vertices are assigned to the same threshold 0.5 in the experiment, and we assume that an inactive vertex v is influenced by each of its neighbor u with the same weight $b_{v,u} = \frac{1}{d_G(v)}$. In **Figure 1** (a), α is set to 0.1 and *cutoff* is 10 s; in **Figure 1** (b), α is set to 0.05 and *cutoff* is 15 s; and in **Figure 1** (c), α is set to 0.06 and *cutoff* is 50 s.

In each figure, the trend of influence spread is on the rise as the seed set size increases, although some exceptions may occur due to the random selection in the local search phase. Of all these approaches, Random did worst on these instances. And the reason is simple because Random does not consider any network properties and does not use any strategy to improve the solution. We use Random here just for the sake of comparison. As a whole, our algorithm DomIM preforms the best in terms of solution quality, but Degree and TreeCore are worse on the IEU instance and CELF is worse on the SWV instance. In particular, for the IEU and SWV instances, DomIM is essentially better than the other algorithms. For the FLS instance, CELFGreedy performs slightly worse than DomIM, but Degree and TreeCore are worse.

5.2.2 Results Under IC Model

Under the IC model, for every two adjacent vertices u and v such that u is an active vertex and v is an inactive vertex, the probability $p_{u,v}$ that v is activated by u is set to the same value 0.05 (this is based on the consideration that the networks we use are sparse). In **Figure 2** (a), α is set to 0.01 and *cutoff* is 10 s; in **Figure 1** (b), α is set to 0.01 and *cutoff* is 20 s; and in **Figure 1** (c), α is set to 0.0015 and *cutoff* is 20 s.

As shown in **Figure 2**, all of these algorithms (except for Random) can obtain a better influence spread, and they have a very similar performance on all the three instances. Our algorithm DomIM slightly outperforms Degree, TreeCore, and CELFGreedy (especially when the size of seed set increases), and TreeCore and CELFGreedy perform very closely to DomIM. Note that the result obtained by the simple approach Degree is also not bad. The reason is because the diffusion probability is not so large, which limits the propagation depth of an active vertex. So, vertices with high-degree may influence much more neighbors. This shows that selecting high-degree vertices as seed set is possible to produce a good influence spread for this case.

5.2.3 Results Under WCM

For the WC model, in **Figure 3** (a), α is set to 0.01 and *cutoff* is 20 s; in **Figure 1** (b), α is set to 0.01 and *cutoff* is 20 s; and in **Figure 1** (c), α is set to 0.0065 and *cutoff* is 90 s.

As shown in **Figure 3**, all algorithms (except for Random) can achieve a similar influence spread. For the IEU instance, DomIM has the best performance under almost all cases (in terms of the seed set size); For the SWV and FLS instances, DomIM and CELFGreedy are better than other algorithms, and they have a similar performance. However, DomIM is efficient, while CELFGreedy is inefficient which will take a long time to obtain a better solution.

5.3 Analysis of Underlying Strategies

We also study the effectiveness of the key strategies of our algorithm. We modify DomIM to obtain two alternative approaches, denoted by DomIM_1 and DomIM_2, where DomIM_1 uses standard degree to replace the uncorrelated degree and DomIM_2 removes the local search procedure on the basis of DomIM.

The comparison experiment of DomIM and its alternatives on the three real-world instances is implemented under the LT model, and the results are presented in **Figure 4**, from which we see that DomIM is better than DomIM_1 and DomIM_2. This implies that these two strategies play an important role in our algorithm DomIM.

6 CONCLUSION

We proposed a local search algorithm DomIM for the IMP. Compared with four distinct types of algorithms, DomIM is efficient and robust, and obtains the best performance for all graphs and all diffusion models we use. However, for the purpose of obtaining an improved solution in the local search phase, our algorithm has to compute the influence of a newly constructed seed set in each iteration. This may slightly effect the efficiency of DomIM. We would like to consider this issue in our future work.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

EZ performed the measurements, analyzed the results, and wrote the paper. LY: performed the experiments and analyzed the results. YX: analyzed the results and refined our manuscript.

FUNDING

This work was supported in part by National Natural Science Foundation of China under Grant 61872101; in part by Natural Science Foundation of Guangdong Province of China under Grant 2021A1515011940.

REFERENCES

1. Scott J, and Carrington PJ. *The SAGE Handbook of Social Network Analysis*. 1st ed. London: Sage Publications Ltd (2011).
2. Can U, and Alatas B. A New Direction in Social Network Analysis: Online Social Network Analysis Problems and Applications. *Physica A* (2019) 355: 122372. doi:10.1016/j.physa.2019.122372
3. Tabassum S, Pereira FSF, Fernandes S, and Gama J. Social Network Analysis: An Overview. *WIREs Data Mining Knowledge Discov* (2018) 8:e1256. doi:10.1002/widm.1256
4. Han W, Tian Z, Huang Z, Li S, and jia Y. Topic Representation Model Based on Microblogging Behavior Analysis. *World Wide Web* (2020) 23:11–2. doi:10.1007/s11280-020-00822-x
5. Tong G, Wu W, Tang S, and Du D-Z. Adaptive Influence Maximization in Dynamic Social Networks. *IEEE/ACM Trans Networing* (2017) 25:112–25. doi:10.1109/tnet.2016.2563397
6. Kianian S, and Rostamnia M. An Efficient Path-Based Approach for Influence Maximization in Social Networks. *Expert Syst Appl* (2021) 167:114168. doi:10.1016/j.eswa.2020.114168
7. Li S, Jiang L, Wu X, Han W, Zhao D, and Wang Z. A Weighted Network Community Detection Algorithm Based on Deep Learning. *Appl Maths Comput* (2021) 401:126012. doi:10.1016/j.amc.2021.126012
8. Centola D. The Spread of Behavior in an Online Social Network experiment. *Science* (2010) 329:1194–7. doi:10.1126/science.1185231
9. Domingos P, and Richardson M. Mining the Network Value of Customers. In: Proceedings of the Seventh ACM SIGKDD international Conference on Knowledge Discovery and Data Mining; 2001 August 26–29; San Francisco, CA (2001) p. 57–66. doi:10.1145/502512.502525
10. Bond RM, Fariss CJ, Jones JJ, Kramer ADI, Marlow C, Settle JE, et al. A 61-Million-Person experiment in Social Influence and Political Mobilization. *Nature* (2012) 489:295–8. doi:10.1038/nature11421
11. Li Y, Zhang D, and Tan K-L. Real-time Targeted Influence Maximization for Online Advertisements. *Proc VLDB Endowment* (2015) 8:1070–81. doi:10.14778/2794367.2794376
12. Fan L, Lu Z, Wu W, Thuraisingham B, Ma H, and Bi Y. Least Cost Rumor Blocking in Social Networks. In: Proceeding of the 2013 IEEE 33rd International Conference on Distributed Computing Systems; 2013 July 8–11; Philadelphia, PA. IEEE (2015). p. 540–9.
13. Ye M, Liu X, and Lee W-C. Exploring Social Influence for Recommendation-A Generative Model Approach. In: 35th International ACM SIGIR conference on research and development in information retrieval; 2012 August 12–16; Portland, OR. New York: ACM Press (2012). p. 671–80.
14. Kempe D, Kleinberg J, and Tardos É. Maximizing the Spread of Influence through a Social Network. In: Proceedings of the 9th ACM SIGKDD international Conference on Knowledge Discovery Data Mining; 2003 August 24–27; Washington, DC. Washington: ACM Press (2003) p. 137–46. doi:10.1145/956750.956769
15. Lee JR, and Chung CW. A Query Approach for Influence Maximization on Specific Users in Social Networks. *IEEE Trans Knowledge Data Eng* (2015) 27: 340–53. doi:10.1109/tkde.2014.2330833
16. Tang J, Zhang R, Wang P, Zhao Z, Fan L, and Liu X. A Discrete Shuffled Frog-Leaping Algorithm to Identify Influential Nodes for Influence Maximization in Social Networks. *Knowledge-Based Syst* (2020) 187:104833. doi:10.1016/j.knosys.2019.07.004
17. Banerjee S, Jenamani M, and Pratihar DK. A Survey on Influence Maximization in a Social Network. *Knowl Inf Syst* (2020) 62:3417–55. doi:10.1007/s10115-020-01461-4
18. Li Y, Fan J, Wang Y, and Tan K-L. Influence Maximization on Social Graphs: A Survey. *IEEE Trans Knowledge Data Eng* (2018) 30:1852–72. doi:10.1109/tkde.2018.2807843
19. Peng S, Zhou Y, Cao L, Yu S, Niu J, and Jia W. Influence Analysis in Social Networks: a Survey. *J Netw Comput Appl* (2018) 106:17–32. doi:10.1016/j.jnca.2018.01.005
20. Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen J, and Glance N. Cost-effective Outbreak Detection in Networks. In: Proceedings of the 13th ACM SIGKDD international Conference on Knowledge Discovery Data Mining; 2007 August 12–15; San Jose, CA. San Jose: ACM Press (2007) p. 420–9. doi:10.1145/1281192.1281239
21. Qiu L, Gu C, Zhang S, Tian X, and Zhang M. Tsim: A Two-Stage Selection Algorithm for Influence Maximization in Social Networks. *IEEE Access* (2020) 8:12084–95. doi:10.1109/access.2019.2963100
22. Chen W, Wang Y, and Yang S. Efficient Influence Maximization in Social Networks. In: Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery Data Mining; 2009 June 28–July 1; Paris, France. Pairs: ACM Press (2009) p. 199–208. doi:10.1145/1557019.1557047
23. Goyal A, Lu W, and Lakshmanan L. Celf++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011; March 28-April 1, 2011; Hyderabad, India (2011). Companion Volume.
24. Kundu S, and Pal S. Deprecation Based Greedy Strategy for Target Set Selection in Large Scale Social Networks. *Inf Sci* (2015) 316:107–22. doi:10.1016/j.ins.2015.04.024
25. Heidari M, Asadpour M, and Faili H. Smg: Fast Scalable Greedy Algorithm for Influence Maximization in Social Networks. *Physica A* (2015) 420:124–33. doi:10.1016/j.physa.2014.10.088
26. Lu F, Zhang W, Shao L, Jiang X, Xu P, and Jin H. Scalable Influence Maximization under Independent cascade Model. *J Netw Comput Appl* (2016) 86:15–23. doi:10.1016/j.jnca.2016.10.020
27. Shang J, Zhou S, Li X, Liu L, and Wu H. Cofim: A Community-Based Framework for Influence Maximization on Large-Scale Networks. *Knowledge-Based Syst* (2016) 117:88–100. doi:10.1016/j.knosys.2016.09.029
28. Wu H, Shang J, Zhou S, Zhong J, Yong F, and Qiang B. Impc: Influence Maximization Based on Multi-Neighbor Potential in Community Networks. *Physica A*. (2018) 512:1085–103. doi:10.1016/j.physa.2018.08.045
29. Zhu E, Jiang F, Liu C, and Xu J (2020). Partition Independent Set and Reduction-Based Approach for Partition Coloring Problem. *IEEE Trans Cybernetics*. 1–10. doi:10.1109/TCYB.2020.3025819
30. Li S, Zhao D, Wu X, Tian Z, Li A, and Wang Z. Functional Immunization of Networks Based on Message Passing. *Appl Maths Comput* (2020) 366:124728. doi:10.1016/j.amc.2019.124728
31. Fan Y, Lai Y, Li C, Li N, Zongjie M, et al. Efficient Local Search for Minimum Dominating Sets in Large Graphs. In: 24th International Conference on Database Systems for Advanced Applications; 2019 April 22–25; Chiang Mai, Thailand (2019) p. 211–28. doi:10.1007/978-3-030-18579-4_13
32. Chen W, Wang C, and Wang Y. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In: Proceedings of the 16th ACM SIGKDD international Conference on Knowledge Discovery Data Mining; 2010 July 25–28; Washington, DC. Washington: ACM Press (2010) p. 1029–38. doi:10.1145/1835804.1835934
33. Jiang Q, Song G, Cong G, Wang Y, Si W, and Xie K. Simulated Annealing Based Influence Maximization in Social Networks. In: Proceedings of the

Twenty-Fifth AAAI Conference on Artificial Intelligence; 2011 August 7–11; San Francisco, CA. San Francisco: AAAI Press (2011). p. 127–32.
34. Kim J, Kim S-K, and Yu H. Scalable and Parallelizable Processing of Influence Maximization for Large-Scale Social Networks? In: Data Engineering (ICDE), 2013 IEEE 29th International Conference on; 2013 April 8–12; Brisbane, QLD (2013). p. 266–77.
35. Tang Y, Xiao X, and Shi Y. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. In: SIGMOD'14: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data; 2014 June 22–27; Snowbird, UT. New York; ACM Press (2014). p. 75–86.
36. Tsai C-W, Yang Y-C, and Chiang M-C. A Genetic Newgreedy Algorithm for Influence Maximization in Social Network. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics; 2015 October 9–12; Hong Kong, China. IEEE Press (2015) p. 2549–54. doi:10.1109/smc.2015.446
37. Bucur D, and Lacca G. Influence Maximization in Social Networks with Genetic Algorithms. In: 19th European Conference on the Applications of Evolutionary Computation; 2016 March 30–April 1; Porto, Portugal. Springer (2016). p. 379–92. doi:10.1007/978-3-319-31204-0_25
38. Kim S, Kim D, Oh J, Hwang J-H, Han W-S, Chen W, et al. Scalable and Parallelizable Influence Maximization with Random Walk Ranking and Rank Merge Pruning. *Inf Sci* (2017) 415:171–89. doi:10.1016/j.ins.2017.06.018
39. Cui L, Hu H, Yu S, Yan Q, Ming Z, Wen Z, et al. Ddse: A Novel Evolutionary Algorithm Based on Degree-Descending Search Strategy for Influence Maximization in Social Networks. *J Netw Comput Appl* (2018) 103:119–30. doi:10.1016/j.jnca.2017.12.003
40. Goldenberg J, Libai B, and Muller E. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-Of-Mouth. *Marketing Lett* (2001) 12: 211–23. doi:10.1023/a:1011122126881
41. Liu C. A Note on Domination Number in Maximal Outerplanar Graphs. *Discrete Appl Maths* (2021) 293:90–4. doi:10.1016/j.dam.2021.01.021
42. Wuchty S. Controllability in Protein Interaction Networks. *Pnas* (2014) 111: 7156–60. doi:10.1073/pnas.1311231111
43. Wang Y, Cai S, Chen J, and Yin M. A Fast Local Search Algorithm for Minimum Weight Dominating Set Problem on Massive Graphs. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18); 2018 July 13–19; Stockholm, Sweden (2018) p. 1514–22. doi:10.24963/ijcai.2018/210
44. Cai S, Hou W, Wang Y, Luo C, and Lin Q. Two-goal Local Search and Inference Rules for Minimum Dominating Set. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20); 2021 January 7–15; Yokohama, Japan (2020) p. 1467–73. doi:10.24963/ijcai.2020/204
45. Rossi RA, and Ahmed NK. The Network Data Repository with Interactive Graph Analytics and Visualization. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence; 2015 January 25–30; Austin, TX (2015).
46. Rozemberczki B, and Sarkar R. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM' 20) (ACM); 2020 October 19–23 (2020) p. 1325–34. doi:10.1145/3340531.3411866
47. Zhu E, Wu Y, Xu Y, and Niu Y. Tree-coritivity-based Influence Maximization in Social Networks. *Acta Electronica Sinica* (2019) 47:161–8. doi:10.3969/j.issn.0372-2112.2019.01.021

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

frontiers in Physics

ORIGINAL RESEARCH
published: 25 October 2021
doi: 10.3389/fphy.2021.733224

Assessing the Structural Vulnerability of Online Social Networks in Empirical Data

Dayong Zhang[1,2], *Changyong Guo*[3]*, *Zhaoxin Zhang*[3]* and *Gang Long*[3,4]

[1]State Key Laboratory of Communication Content Cognition, People's Daily Online, Beijing, China, [2]Department of New Media and Arts, Harbin Institute of Technology, Harbin, China, [3]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, [4]China Information Technology Security Evaluation Center, Beijing, China

Assessing the structural vulnerability of online social networks has been one of the most engaging topics recently, which is quite essential and beneficial to holding the network connectivity and facilitating information flow, but most of the existing vulnerability assessment measures and the corresponding solutions fail to accurately reveal the global damage done to the network. In order to accurately measure the vulnerability of networks, an invulnerability index based on the concept of improved tenacity is proposed in the present study. Compared with existing measurements, the new method does not measure a single property performance, such as giant component size or the number of components after destruction, but pays special attention to the potential equilibrium between the removal cost and the removal effect. Extensive experiments on real-world social networks demonstrate the accuracy and effectiveness of the proposed method. Moreover, compared with results of attacks based on the different centrality indices, we found an individual node's prominence in a network is inherently related to the structural properties of network. In high centralized networks, the nodes with higher eigenvector are more important than the others in maintaining stability and connectivity. But in low centralized networks, the nodes with higher betweenness are more powerful than the others. In addition, the experimental results indicate that low centralized networks can tolerate high intentional attacks and has a better adaptability to attacks than high centralized networks.

Keywords: online social network, vulnerability, structural property, centrality index, vulnerability assessment

OPEN ACCESS

Edited by:
Chengyi Xia,
Tianjin University of Technology, China

Reviewed by:
Michele Bellingeri,
University of Parma, Italy
Zhen Wang,
Hangzhou Dianzi University, China

***Correspondence:**
Changyong Guo
guocy@hit.edu.cn
Zhaoxin Zhang
heart@hit.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *30 June 2021*
Accepted: *13 September 2021*
Published: *25 October 2021*

Citation:
Zhang D, Guo C, Zhang Z and Long G (2021) Assessing the Structural Vulnerability of Online Social Networks in Empirical Data. Front. Phys. 9:733224. doi: 10.3389/fphy.2021.733224

INTRODUCTION

With the revolution of the WWW technology, Web 2.0 characterized by social collaborative technologies is emerging and fast-growing. People are increasingly inclined to cultivate their virtual social relations and virtual life on the existing prevalent online social networks [1], such as Facebook, Blogger, Wiki, and Digg. These online social networks can provide favorable platforms for people to exchange opinions or information with one another [2]. Specifically, online social networks are creating ties for us with a very wide range of people, which not only are bonded in relationships with acquaintances, as well as maintain close relationships with friends, schoolmates, and family members, but also are embodied in some new relationships in an online virtual world.

In order to defend some potential disruptions and facilitate information flow, assessing the vulnerability of online social networks has been one of the most engaging topics [3, 4]. The concept of

vulnerability is generally used to find and characterize a lack of robustness and resilience of a complex system [5]. The vulnerability of a network structure was analyzed first by Albert et al. [6] and was regarded as a previously overlooked "Achilles' heel." Initially, vulnerability assessment was focused on some simple and generic models such as the Erdös–Rényi (ER) random model and the Barabási–Albert (BA) scale free model [6, 7]. Over the years, some scholars have found that the inherent preferential attachment mechanism and the structural properties of network may be responsible for the vulnerability of network [8–10]. Especially, a series of numerical simulations were introduced to study tolerance to random removals and intentional attacks in complex networks [11–13]. Most experimental studies have shown that the Barabási–Albert (BA) network and other similar heterogeneous networks are very robust to random removals but are very fragile against intentional attacks based on the degree or betweenness [6, 14]. For homogeneous networks such as regular networks and random networks, the effect of random removals is equivalent to that of intentional attacks [6], while for small-world networks, long-range link attacks can cause their collapse directly [13]. Some achievements have been made in the research of some typical network models, but how the dynamical processes, such as resilience to damage or tolerance to attacks, are influenced by the specific topological structure of a network remains unknown.

In recent years, there has been much effort directed at developing methods for vulnerability assessment [15–17]. The main results are largely based on two aspects, including critical node identification and removal effect evaluation. The former reflects the nodal prominent position in maintaining the network connectivity or facilitating information flow, while the latter refers to how to quantify the effect caused by the removal of a finite number of nodes. Indeed, the identification of critical individuals is an influence maximization problem [18], which aims to select a minimal node set to generate a maximal outcome in a given network. The quantification methods can be roughly classified into three categories: centrality-based algorithms, random-walk algorithms, and greedy-based algorithms. Structural connectivity has become the primary test criterion for vulnerability assessment [19]. In most instances, these evaluation metrics such as the characteristic path length [6] and the network efficiency [20, 21] are relatively straightforward and can more clearly characterize the changes in the connectivity of the target network before and after some nodes are removed. However, they only provide a useful topological snapshot for connected networks and are not suitable to assess the network vulnerability in terms of disconnectivity [15]. In addition, the existing measurements are difficult to reach equilibrium between the removal cost and the removal effect.

The primary purpose of this article is to fill this gap by exploring a new method to effectively quantify the vulnerability of the network structure. The new method focuses on how to identify the importance, or status, of a node in the network, and on further use of available resources to efficiently disrupt network operation, which comprehensively takes account of the cost with which one can disrupt a network and the attack effect. The contributions of this study can be summarized as the following:

1) An invulnerability index based on the concept of improved tenacity is proposed to measure the adaptability to attacks.
2) Low centralized networks can have a better adaptability to attacks than high centralized networks.
3) The experimental results verify the outperformance of the proposed method.

The rest of the article is organized as the following: In *Methods*, in order to assess the vulnerability of networks more properly, we present an invulnerability index based on the concept of improved tenacity to examine network adaptability to attacks. Generally, a network with a higher invulnerability index performs better under intentional attacks. In *Network Data,* we will examine the static properties of real online social networks empirically, in order to summarize the generalized differences in the topological structure of various online social networks. Especially, in *Discussion*, we will display the threshold behavior of the aforementioned networks on experimental observation, and further compare the efficiency of node removal with different centrality indices to find the vulnerability of online social networks.

METHODS

The Attack Strategies

Inspired by the well-known percolation theory in statistical physics, the robustness and resilience of a network is usually defined as the network structural degradation caused by the removal of some critical nodes [19]. Tolerance to random removals and intentional attacks is understood as the ability of the network to maintain operations and connectivity under the loss of some nodes or links [8]. In order to ensure the efficiency of attacks, it is necessary to identify the most vulnerable nodes in a network. Indeed, the identification of critical nodes is an influence maximization problem [18]. The quantification algorithms can be roughly classified into three categories: centrality-based algorithms, random-walk algorithms, and greedy-based algorithms.

Centrality-based algorithms perform a fundamental quantification by considering geodesics between nodes to evaluate nodal importance. Up to now, many centrality-based algorithms have been proposed, such as degree centrality [22, 23], betweenness centrality [22, 24], closeness centrality [22, 25], eigenvector centrality [26], and other improved centrality-based algorithms [27–29]. The random-walk algorithms include the well-known PageRank [30] and other improved algorithms [31]. Random-walk algorithms work only well for directed networks. Greedy-based algorithms formulate the influence measurement as a discrete optimization problem, and their elementary strategies are to select the spreaders that contribute the largest incremental influence one by one, according to a specified influence cascade model [32]. In terms of algorithm construction, although greedy-based algorithms can

achieve excellent results, they also have very high computational complexity and are not suitable for large-scale social networks.

Previous studies have shown that the adaptability of networks behaves differently from various attack strategies [20, 33]. Thus, in this article, we will study tolerance to various attacks in real online social networks and further find a minimized set of nodes triggering the collapse of network. We will consider four straightforward and efficient centrality indices as attack strategies to identify the importance of nodes.

1) Degree centrality: The algorithm measures a node's influence according to the number of edges attached to it, which reflects the ability of a node to connect directly with other nodes.
2) Betweenness centrality: The algorithm measures a node's influence through the ratio of the shortest path over the nodes to the number of all paths, which considers the global structure information of a given graph.
3) Closeness centrality: The basic idea behind the closeness centrality is that a node is central if it is "close" to many other nodes [34]. Thus, the closeness centrality score of node i is defined as the reciprocal of the sum of geodesic distances to all other nodes.
4) Eigenvector centrality: The algorithm is based on the principle that a node should be viewed as important if it is linked to other nodes which are important themselves. Thus, the eigenvector centrality of node i is defined as the proportional to the sum of the eigenvector centralities of the nodes it is connected to [28].

Because the removal of nodes under intentional attacks changes the balance of structure and leads to a global redistribution over all the networks, we recalculate the degree centrality, the betweenness centrality, the closeness centrality, and the eigenvector centrality, every time a small fraction of nodes is removed.

Evaluation Metrics

Numerous empirical results on real networks have revealed that the heterogeneous topology structure may be fit for most real networks [35–37], where degree distribution significantly deviates from a Poisson and low degree nodes are far more abundant than the nodes with high degrees. Due to the inhomogeneity of general networks, removing some critical nodes will decrease the network connectivity and lead to the loss of the global information-carrying ability of the network [6, 20]. Generally, when assessing the vulnerability of a network under intentional attacks, three performance criteria should be concerned in the framework of graph theory [38]:

1) The number of components that are being removed.
2) The number of disconnected subgroups after intentional attacks.
3) The size of the largest remaining group within which mutual communication can still occur.

Most of the online social networks can be abstracted as a non-complete connected graph $G = (V, E)$, where individual members and personal relationships can be defined by a set of nodes V and a set of edges E, respectively. In general, a good social network should have short distance between nodes, average distance, and high connectivity. In fact, there has been much effort directed at developing methods for evaluating network adaptability to attacks. In most instances, the characteristic path length and the network efficiency as evaluation metrics can only provide a useful topological snapshot for connected networks [39]. But for disconnected networks, the geodesic distance between any two nodes belonging to two disconnected subgroups is identically zero or infinity, which will directly affect the accuracy of evaluation results. In graph theory, some helpful indicators of evaluating network vulnerability have been proposed. These metrics relates to network topology and attributes, such as toughness [40], integrity [41], tenacity [42], and scattering number [43]. The detailed description of each indicator is shown in **Table 1**.

As one of the basic concepts of graph theory, connectivity plays a vital role in network performance and is fundamental to vulnerability measures. The concept of connectivity $K(G)$ of G is defined as

$$K(G) = \min\{|S|: S \subset V\}, \tag{1}$$

where $|S|$ is a cutset of $V(G)$. In **Eq. 1**, the connectivity$K(G)$ asks for the minimum number of nodes whose removal renders the graph Gdisconnected. As one of the graph theoretical concepts, connectivity deals with the criterion (1).

Toughness and integrity are two other graph concepts used in the vulnerability assessment. The notion of the toughness $T(G)$ of$G$, originally introduced by Chvátal [40], is defined as follows:

$$T(G) = \min\left\{\frac{|S|}{w(G-S)}: S \subset V, w(G-S) \geq 2\right\}, \tag{2}$$

where $w(G-S)$ stands for the number of components of $G-S$. Unlike the connectivity $K(G)$, the toughness $T(G)$ incorporates the relationship between the size of the cutset and the number of components after destruction and takes into account of the criteria (1) and (2). But the toughness $T(G)$ is still insufficient to measure the network vulnerability. Considering the graphs G_1 and G_2 (see **Figure 1**), two graphs have the same connectivity and toughness, where $K(G_1) = K(G_2) = 1$ and $T(G_1) = T(G_2) = \frac{1}{3}$, but they are really different in the vulnerability of graphs. For instance, after the minimum cutset $\{u_1\}$ is removed, we find that the G_1has been divided into three small components, while the vast majority of nodes of G_2 have been retained in the largest connected component $\{u_2, u_6, u_5\}$, within which mutual communication among the remaining nodes can still occur. It implies that the connectivity $K(G)$ and toughness $T(G)$ cannot accurately reveal the global damage done to the network.

The notion of integrity introduced as another vulnerability parameter of graphs [41] focuses on the criteria (1) and (3). For a non-complete connected graph G, its integrity $I(G)$ is defined as follows:

$$I(G) = \min\{|S| + m(G-S): S \subset V\}, \tag{3}$$

where $m(G-S)$ denotes the giant component size after destruction.

TABLE 1 | Evaluation metrics based on graph theory.

Name	Acronym	Meaning	References
Toughness	T(G)	Focusing on the relationship between the removal cost and the number of components after destruction	Chvátal [40]
Integrity	I(G)	Focusing on the relationship between the removal cost and the largest connected component after destruction	Barefoot et al. [41]
Tenacity	R(G)	Focusing on the relationship among the removal cost, the number of components, and the largest connected component after destruction	Cozzens et al. [42]
Scattering number	S(G)	Focusing on the relationship between the removal cost and the number of components after destruction	Hendry [43]

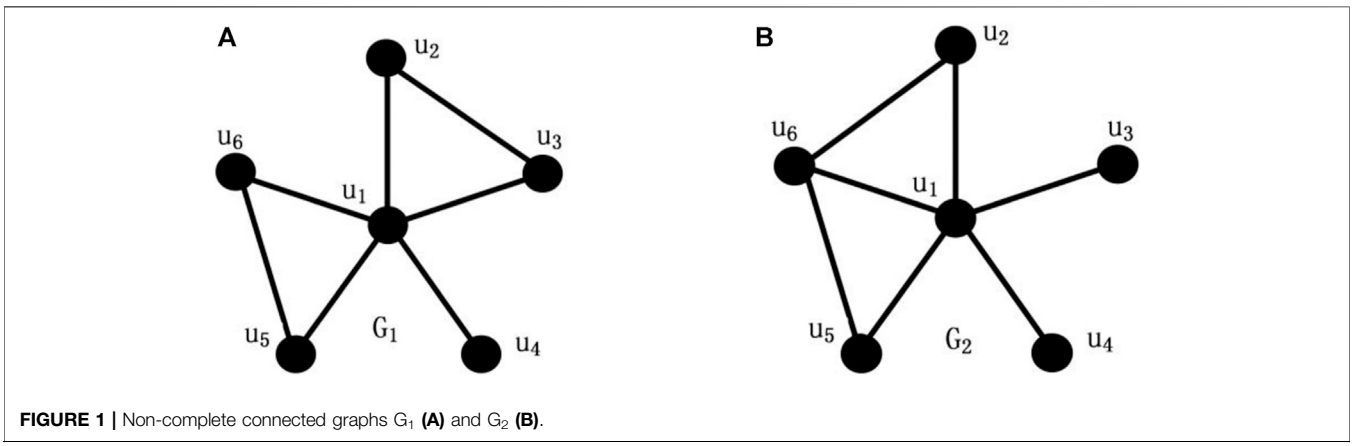


FIGURE 1 | Non-complete connected graphs G_1 **(A)** and G_2 **(B)**.

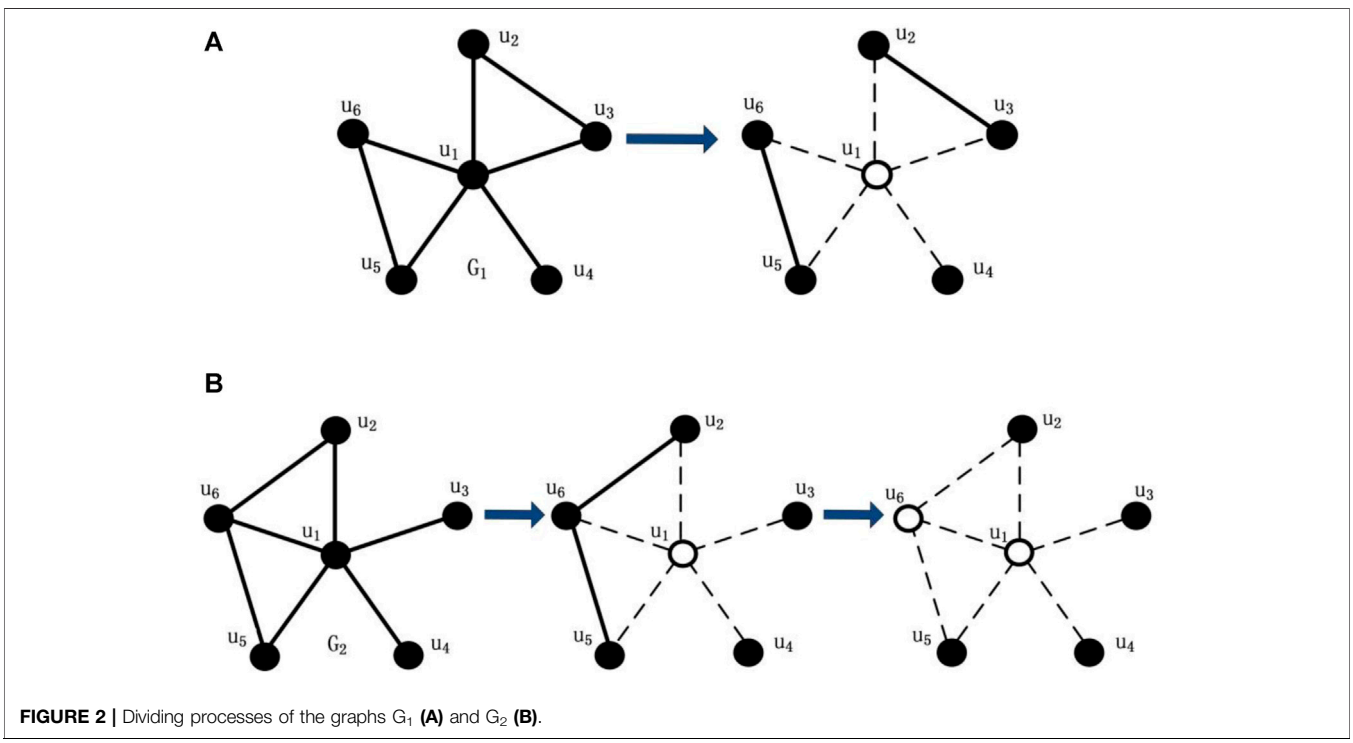


FIGURE 2 | Dividing processes of the graphs G_1 **(A)** and G_2 **(B)**.

Obviously, the disruption is more successful if the disconnected network contains more components and is much more successful if, in addition, the components are small. Unfortunately, connectivity and toughness give the minimum cost to disrupt a network but fail to indicate accurately what remains after the disruption. Although Barefoot's integrity has taken the size of the largest remaining component after destruction into account, it cannot indicate the extent of the damage.

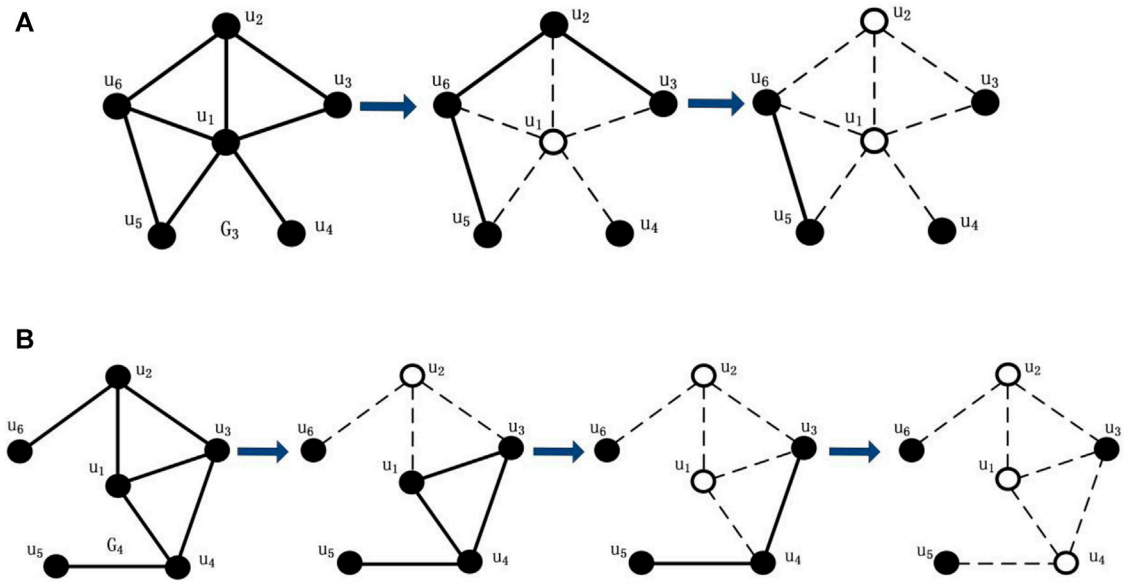


FIGURE 3 | Dividing processes of the graphs G_3 **(A)** and G_4 **(B)**.

The notion of tenacity was originally proposed in Ref. [42], where they introduced the mix-tenacity to measure the vulnerability of Harary graphs. The precise definition of tenacity is defined as follows:

$$R(G) = \min\left\{\frac{|S| + m(G-S)}{w(G-S)}: S \subset V, w(G-S) \geq 2\right\} \quad (4)$$

The tenacity $R(G)$ of graphs directly integrates all three criteria, such as the cost of network breakage, the number of components, and the giant component size, and is considered to be a reasonable measure for the vulnerability of graphs. As shown in **Figures 2A,B**, it is easy to know that $R(G_1) = \min\left\{\frac{1+2}{3}\right\} = 1$ and $R(G_2) = \min\left\{\frac{2+1}{4}\right\} = \frac{3}{4}$, $R(G_1) > R(G_2)$, which indicates the adaptability to attacks for $G_1$is better than G_2.

In general, if the network remains more disconnected subgroups and smaller connected component size after destruction, the disruption is more successful. As shown in **Figures 3A,B**, G_3 and G_4 all have the same number of nodes and edges. After $\{u_1, u_2\}$ and $\{u_2, u_1, u_4\}$ are removed respectively, the minimum toughness and the minimum tenacity can be obtained, which are $T(G_3) = T(G_4) = \frac{1}{2}$ and $R(G_3) = R(G_4) = \frac{4}{3}$. The cutsets $\{u_1, u_2\}$ and $\{u_2, u_1, u_4\}$ are the minimum removal costs of the graph G_3 and the graph G_4, respectively. But we find that there are differences both in the attack efficacy in the graph G_3 and graph G_4, where for the graph G_3, the minimum removal cost is 2 and the giant component size also is 2; while for the graph G_4, the minimum removal cost is 3 and the giant component size is 1. As discussed earlier, the tenacity $R(G)$is still an imperfect criterion to assess network vulnerability.

In Ref. [43], Hendry used the concept of scattering number to measure the vulnerability of extremal non-Hamiltonian graphs and found that it was more efficient for measuring the degree of global destruction. The scattering number $S(G)$ of $G$is defined as follows:

$$S(G) = \max\{w(G-S) - |S|: S \subset V, w(G-S) \geq 2\}. \quad (5)$$

So we think that it is necessary to add the criterion to reveal the global damage done to networks under attacks, and keep its priority in assessing the vulnerability of networks. Therefore, we propose an improved tenacity based on the concepts of scattering number and tenacity, which is named $R_{sca}(G)$ and defined as follows:

$$R_{sca}(G) = \min\left\{\frac{m(G-S)}{|w(G-S) - |S||}: S \subset V, w(G-S) \geq 2\right\}. \quad (6)$$

As shown in **Figures 3A,B**, $R_{sca}(G_3) = \min\left\{\frac{1+2}{3}\right\} = 1$ and $R_{sca}(G_4) = \min\left\{\frac{2+1}{4}\right\} = \frac{3}{4}$, $R_{sca}(G_3) > R_{sca}(G_4)$, which indicates the anti-interference capability of $G_3$is better than G_4.

Compared with existing evaluation metrics [6, 20, 21, 39–42], our notion of improved tenacity is not to measure a single property performance such as giant component size or the number of components after destruction, which is insufficient to evaluate the network vulnerability by only considering whether or not it is a disconnected network, and how fragmental the network becomes, but to pay special attention to the potential equilibrium between the giant component size and the number of components under intentional attacks. As shown in definition (6), the number of components$w(G-S)$and the size of the largest connected component$m(G-S)$are re-calculated after each iteration, so the whole computational complexity of the proposed method is$O(n^2)$. The result demonstrates that the proposed algorithm has relatively less computational burden in evaluating the

TABLE 2 | The basic topological properties of the six online social networks.

Network	*N*	*E*	<*k*>	*L*	*D*	*C*
Lilac	3,414	10,353	6.065	4.055	10	0.042
OClinks	1,893	13,835	14.617	3.055	8	0.138
Wiki-Vote	7,115	103,689	14.573	3.341	7	0.141
Twitter	3,656	154,824	84.696	2.506	6	0.279
RenRen	1,130	14,332	25.366	3.241	8	0.263
Facebook	4,039	88,234	43.691	3.693	8	0.606

Note: N, number of nodes; E, number of edges; <k>, average degree; L, characteristic path length; D, diameter; and C, clustering coefficient.

vulnerability of networks and can be applicable to large-scale networks.

NETWORK DATA

Data Description

Because online social networks serve as much social function as other kinds of social interaction, including e-mail exchanging, text messaging, instant messaging, digital video sharing, and so on, each edge meaning a connection in online social networks is complex and changeable, whereas each node meaning an individual of online social networks is constant. To measure the vulnerability of networks more properly, our method will pay more attention to the importance of nodes, rather than edges. Our data set is composed of six undirected and unweighted networks, that is, networks that have a binary nature, where the edges between nodes are either present or not, and each edge has no directional character. **Table 2** indicates that the six real social network include the following: OClinks is a representative online community network, where users are from the University of California, Irvine, which is from Panzarasa et al.'s [44]; Twitter, Wiki-Vote, and Facebook can be downloaded from the Stanford network dataset (http://snap.stanford.edu/data/index.html); and Lilac and RenRen are collected from the online social networks by us.

In **Table 1**, we show the main topological properties of a series of online social networks, belonging to two different types: the online community service based on group-centered service and the social network service based on individual-centered service. The first three networks, Lilac, OClinks, and Wiki-vote, are three examples of online community services, where the users have a common interest and purpose and can exchange information or seek help. In these networks, the nodes are the registered users, and the links represent a relationship between two users existing message exchange. In the latter part of **Table 1**, we show three examples of social network services. Twitter is a social news website, where users may mention other people or follow other people to make his/her posts, so the links imply communication between users existing mention or comment. RenRen is a real-name social networking internet platform in China, where users can connect and communicate with each other or enjoy a wide range of other features and services, so the links reflect different kinds of social relationships between the users. Facebook is an ego network consisting of friends lists from Facebook.

Structure of Networks

The most basic topological characterization of networks can be obtained in terms of the degree distribution $P(k)$, defined as the probability that a node is chosen uniformly at random has degree k or, equivalently, as the fraction of nodes in the graph having degree k [19]. In recent years, scientists approached the study of real networks from the available databases and found most of the real networks having inhomogeneous structure, where the connections within nodes of the highest degree are rather sparse, and a large number of nodes just have a few connections. Moreover, most of real networks exhibit power law–shaped degree distribution$P(k) \sim k^{-\gamma}$, with exponents varying in the range $2 < \gamma < 3$[35, 36], and a little of them follow shifted power law distribution, stretched exponential distribution, or more complicated distributions [37].

The empirical results demonstrate that the degree distributions $P(k)$of aforementioned networks are subjected to two types: the segmented power law distribution (Lilac, OClinks, and Wiki-Vote) and the stretched exponential distribution (Twitter, RenRen, and Facebook). The insets in **Figures 4A–C** show the degree distributions $P(k)$ of Lilac, OClinks, and Wiki-Vote are heavy-tailed, and approximately follow the power law distribution, where $\gamma = 1.72$, $\gamma = 0.99$, and $\gamma = 1.31$, respectively. However, when the size of data set is small, it may happen that the data have a rather strong intrinsic noise due to the finiteness of the sampling. In order to avoid the statistical fluctuations, one better possibility is to measure the cumulative degree distributions$P(x \geq k)$. The cumulative degree distributions $P(x \geq k)$of Lilac, OClinks, and Wiki-Vote show two different scaling regions: a slow region and a rapid decaying region, and are well-approximated by the segmented power law distribution. The crossover takes place between k = 10 and k = 100, and the cumulative degree distributions $P(x \geq k)$ of Lilac network can be defined by the following equation:

$$P(x \geq k) \sim k^{-(r-1)} \begin{cases} r-1 = 0.89, & \text{if } k \leq 16 \\ r-1 = 2.11, & \text{if } k > 16 \end{cases}.$$

The similar trend is shown in **Figure 4B**, where the probability $P(x \geq k)$ of the OClinks network can be fitted by the following equation:

$$P(x \geq k) \sim k^{-(r-1)} \begin{cases} r-1 = 0.51, & \text{if } k \leq 20 \\ r-1 = 2.02, & \text{if } k > 20 \end{cases}.$$

As shown in **Figure 4C**, the probability $P(x \geq k)$ of the Wiki-Vote network can be fitted by the following equation:

$$P(x \geq k) \sim k^{-(r-1)} \begin{cases} r-1 = 0.45, & \text{if } k \leq 63 \\ r-1 = 2.26, & \text{if } k > 63 \end{cases}.$$

Otherwise, the insets in **Figures 5A–C** show the degree distributions of Twitter, RenRen, and Facebook, where $R^2 = 0.815$,$R^2 = 0.811$, and $R^2 = 0.809$, respectively, and the

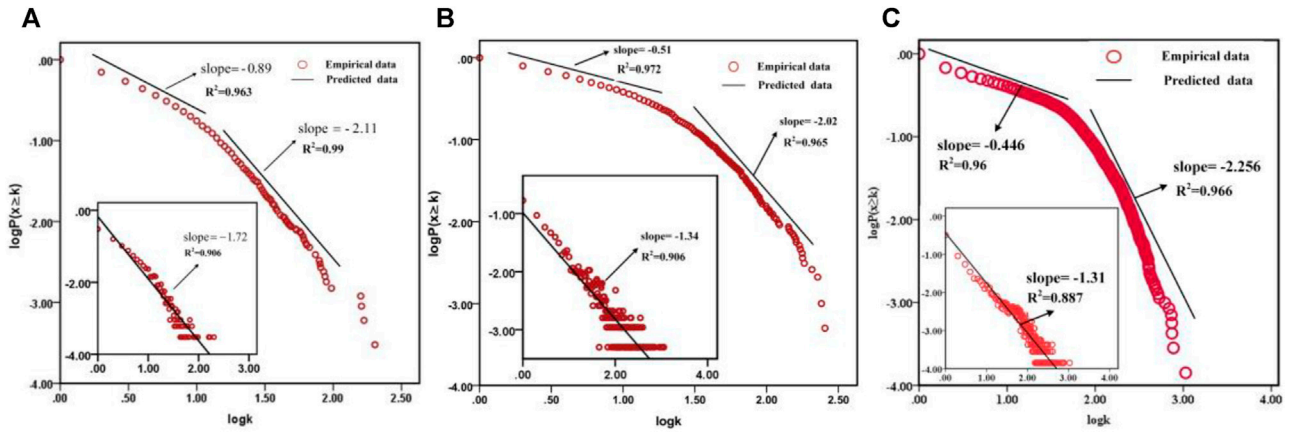


FIGURE 4 | Cumulative degree distribution $P(x \geq k)$in Lilac network **(A)**, OClinks network **(B),** and Wiki-Vote **(C)**. The insets show the degree distributions $P(k)$of Lilac, OClinks, and Wiki-Vote are heavy-tailed.

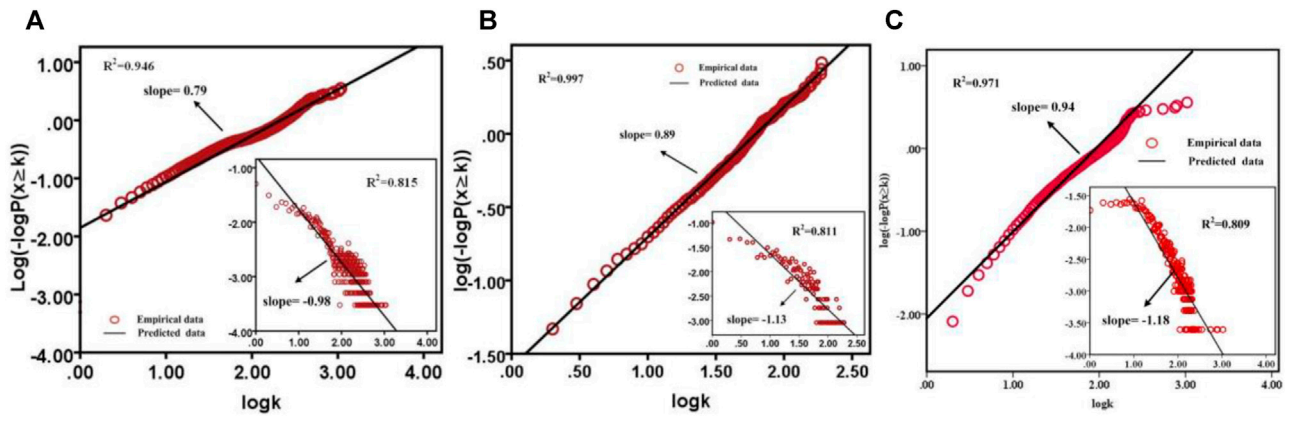


FIGURE 5 | Stretched exponential distributions of Twitter **(A)**, RenRen **(B),** and Facebook **(C)** are clear. The insets exhibit the degree distribution $P(k)$of Twitter network, RenRen, and Facebook network.

curves of the degree distributions are not well-approximated by a power law distribution. **Figures 5A–C** indicate that the degree distributions of Twitter, RenRen, and Facebook obey the stretched exponential distribution$P(x \geq k) = e^{-(\frac{k}{k_0})^c}$, where the graph of $\log P(x \geq k)$ versus $\frac{k}{k_0}$ is characteristically stretched, and a stretching exponent c takes a value between 0 and 1. The stretching exponent c can be obtained from$P(x \geq k)$, as we add the slope of $\log P(x \geq k)$ in a log–log plot. As shown in **Figure 5**, the distribution functions of Twitter, RenRen, and Facebook can be defined by$P(x \geq k) = e^{-(\frac{k}{k_0})^{0.79}}$, $P(x \geq k) = e^{-(\frac{k}{k_0})^{0.89}}$, and $P(x \geq k) = e^{-(\frac{k}{k_0})^{0.94}}$, respectively.

The stretched exponential distribution is obtained by inserting a fractional power law distribution into the exponential distribution: as$c = 1$, the usual exponential function is recovered; as$c \to 0$, the distribution follows the power law distribution. In general, the power law distribution is characterized by a slower than exponentially decaying probability tail. In contrast with Twitter, RenRen, and Facebook, where$c = 0.79$, $c = 0.89$, and $c = 0.94$ approaching$c = 1$, some extremum nodes in Lilac, OClinks, or Wiki-Vote can occur with a more non-negligible probability.

In this section, we have examined the static properties of a variety of online social networks empirically and found that two types of networks, the online community service and the social network service, have completely different structural properties. As shown in **Figures 4A–C**, Lilac network, OClinks network, and Wiki-Vote network exhibit a highly inhomogeneous degree distribution, where the simultaneous presence of a few nodes tending to link many other nodes, and a large number of poorly connected nodes. But in **Figures 5A–C**, the curves of degree distributions have witnessed that the nodes in Twitter, RenRen, and Facebook networks are more evenly distributed than those in Lilac, OClinks, and Wiki-Vote networks, where extreme value still runs at a relatively low level. As it can be noticed, Twitter, RenRen, and Facebook as social network services are mainly based on an individual-centered online platform for organizing

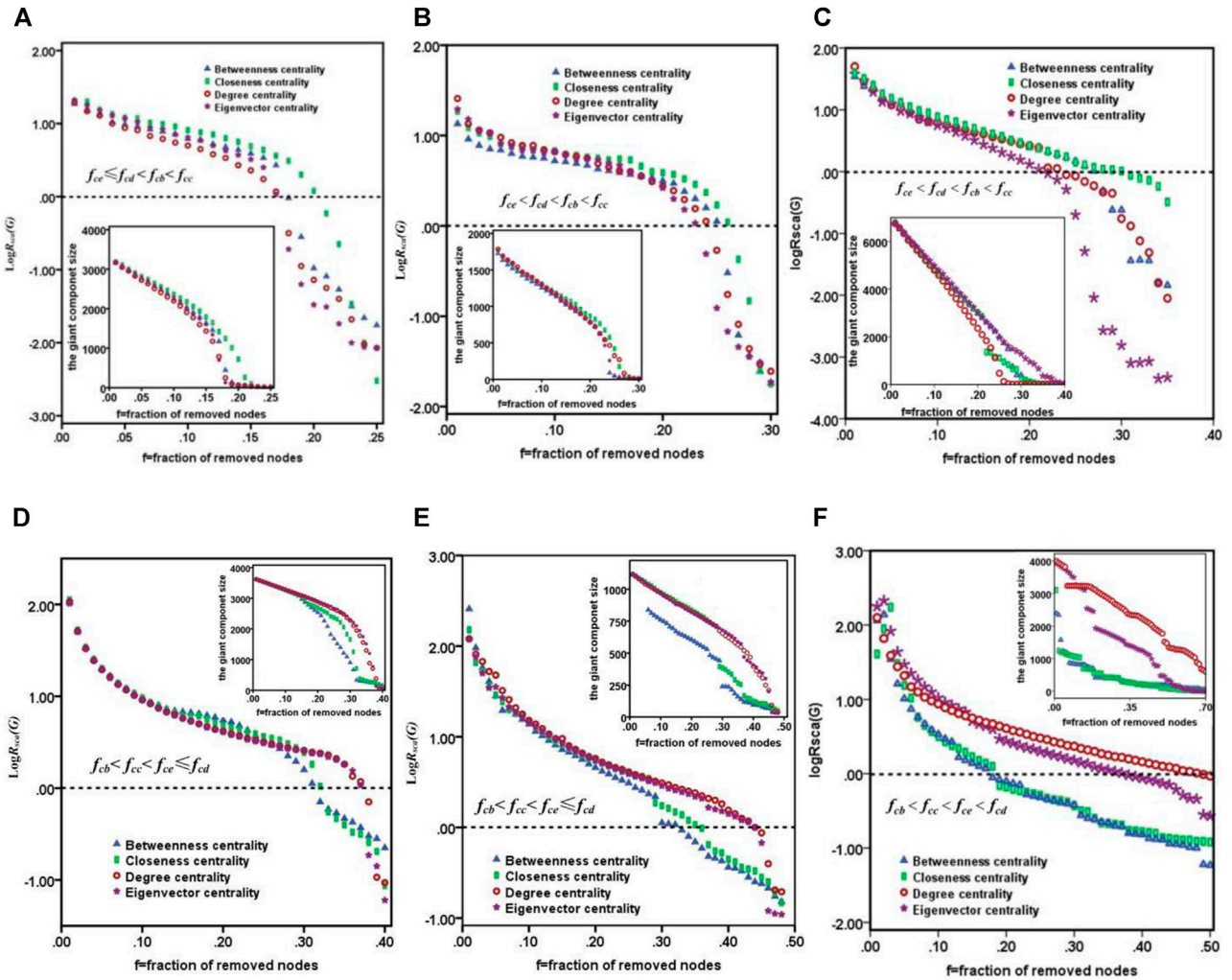


FIGURE 6 | From **(A–F)**, indicating the performance of improved tenacity $R_{sca}(G)$ in six online social networks (from Lilac to Facebook) when a fraction f of the nodes is removed. ○, preferential removal of the most connected nodes; △, preferential removal of the nodes with maximum betweenness; *, preferential removal of the nodes with maximum eigenvector; and □, preferential removal of the nodes with minimum closeness. Degree, betweenness, eigenvector, and closeness are recomputed each time the nodes are removed. Inset refers to the performance of the giant component size.

feature and technical feature themselves and have lower centralization than the Lilac, OClinks, and Wiki-Vote network based on group-centered service.

DISCUSSION

When the vast majority of real networks, especially online social networks, are fragmented into relatively tiny isolated components, these networks will lose transmission capacities between individual components, indicating the collapse of network is approaching. So, we only find an optimal threshold that can trigger the collapse of the network. In general, the problem can be analytically treated by using percolation theory, where one defines a critical probability f_c below which the network percolates, and a set of critical exponents can characterize the phase transition. In Ref. [6], Albert et al. have studied how the properties of some networks with given order and size change when a fraction f of the nodes are removed, where the average characteristic path length as an order parameter displays for both errors and attacks a threshold-like behavior. In this section, we first study the changes in the improved tenacity $R_{sca}(G)$ when a small fraction f of the nodes is removed gradually, and use the new criterion characterizing the phase transition to obtain the critical probability f_c. Then, we compare four attack strategies, that is, degree centrality, betweenness centrality, closeness centrality, and eigenvector centrality (the algorithms defined as in *The Attack Strategies*), to estimate which solution is the most effective and also to determine the most important nodes for online social networks.

As shown by the curved lines in **Figure 6**, the performance of improved tenacity $R_{sca}(G)$ in the aforementioned networks displays a threshold-like behavior: first, $R_{sca}(G)$ drops with the fraction of removed nodes f increasing, indicating the ability of the network to

TABLE 3 | Tenacity level of selected online social networks

Network	N	f_c	S′	m(G − S)	w(G − S)	R_{sca}(G)
Lilac	3414	0.17	580	641	1179	1.07
OClinks	1893	0.23	435	465	870	1.07
Wiki-Vote	7115	0.21	1483	1864	3258	1.05
Twitter	3656	0.31	1133	667	537	1.12
RenRen	1130	0.32	362	233	144	1.07
Facebook	4039	0.17	686	684	108	1.18

maintain its connectivity properties is sensitive to intentional attacks, and the size of fragments is increasing significantly. Especially, the insets of **Figure 6** show the giant component size $m(G-S)$ rapidly decreases with f increasing. Second, we find that the curve of improved tenacity $R_{sca}(G)$ abruptly decays at the critical value f_c, the same trend as the giant component size $m(G-S)$(see the inset of **Figure 6**, where the slope of curve is sharply downward), implying f_c is precisely the threshold triggering the collapse of network. In addition, it is worth noticing that the critical value f_c can be obtained fast by using the definition (6). Aslog $R_{sca}(G) \approx 0$, from Lilac network to Facebook network, the critical value $f_c = \{0.17, 0.23, 0.21, 0.31, 0.32, 017\}$. Although the largest component of the remaining nodes still is larger, where the giant component size$m(G-S)$ runs from 233 to 1864 (shown in **Table 3**), accounting for 16.93–26.2% of the total, intentional attacks have led to the breakdown of the overall connectivity.

The centrality concept seeks to quantify an individual node's prominence within a network by summarizing structural relations among the nodes. A node's prominence reflects its greater visibility to the other network nodes. In online social networks, central nodes are likely to be more influential and have greater access to information and can communicate their opinions to others more efficiently. Further research indicates that the various roles of the same node based on different centrality indices show the striking difference in maintaining network connectivity. Looking at the changes in the critical fraction f_c, from **Figures 6A–F**, $f_{cd} = \{0.17, 0.24, 0.23, 0.37, 0.44, 0.48\}$ relating to the degree-based attacks, $f_{cb} = \{0.18, 0.25, 0.27, 0.31, 0.32, 0.17\}$ relating to the betweenness-based attacks, $f_{cc} = \{0.2, 0.26, 0.3, 0.32, 0.35, 0.18\}$ related to the closeness-based attacks, while $f_{ce} = \{0.17, 0.23, 0.21, 0.37, 0.43, 0.37\}$ related to the eigenvector-based attacks. Obviously, in the cases of Lilac, OClinks, and Wiki-Vote, the attacks based on the eigenvector centrality actually have a better performance than the attacks based on other indices, rooting in the critical fraction $f_{ce} < f_{cd} < f_{cb} < f_{cc}$. Although in Lilac $f_{ce} = f_{cd}$, the giant component size$m(G-S)$under the attacks based on the eigenvector centrality is significantly smaller than the result based on the degree centrality. The conclusion described before indicates the role of the nodes with high eigenvector is the most important than the others in maintaining connectivity of the network. In the diffusion of information, especially in online social networks, a user with high eigenvector centrality has connections to many other users that are themselves highly connected and central within the network, thus multiplying his or her capabilities in maintaining communication of network. But in the cases of Twitter, RenRen, and Facebook, the attacks based on the betweenness centrality f_{cb} cause a greater amount of damage than the others, where $f_{cb} < f_{cc} < f_{ce} \leq f_{cd}$. The main reason for the differentiation from various networks may closely relate with their organizing features and technical features, which are characterized by their topological structures.

Another important conclusion that can be drawn from the results presented is that the performance of improved tenacity $R_{sca}(G)$ in Twitter, RenRen, or Facebook is better than that in Lilac, OClinks, or Wiki-Vote, which implies the former has higher anti-interference capability than the latter. In general, a low centralized network can improve network resilience by reorganizing network to increase local control and the execution of a service. Analogously, for the lack of obvious centralization and a strict inhomogeneous topology structure, social network services, like Twitter, RenRen, or Facebook, can tolerate higher intentional attacks based on some critical individuals than high centralized networks. In addition, as shown in **Table 3**, although the critical probability f_c of Twitter or RenRen is much larger than the threshold f_c of Lilac, OClinks, or Wiki-Vote, the corresponding proportion of the giant component size of the former is alsways lower than that of the latter. Therefore, it is difficult to judge which network has higher adaptability facing intentional attacks from the attack cost or the giant component size scale. Compared with the single criterion, such as removal cost, the giant component size, and the number of components, our proposed method can comprehensively consider the attack effect and attack cost.

CONCLUSION

In this article, we synthetically take account of the cost with which one can disrupt a network and the effect of, and a new evaluation method based on the concepts of scattering number and tenacity. Compared with existing evaluation metrics, our method focuses on the potential equilibrium between the attack effect and the attack cost. For this purpose, we first examined empirically the static properties of six online social networks, including three online community services and three social network services, that is, Lilac, OClinks, Wiki-Vote, Twitter, RenRen, and Facebook, and found that there are wide differences in the topological structure of networks.

Second, we studied the changes in the improved tenacity $R_{sca}(G)$ when a small fraction f of the nodes is removed gradually and found the curve of improved tenacity displays a threshold-like behavior, when the minimum of tenacity approaches zero. Then, we compared the four solutions of intentional attacks based on the different indices, that is, degree centrality, betweenness centrality, closeness centrality, and eigenvector centrality, and found that an individual node's prominence in a network is inherently related to structural properties: the role of the nodes with higher eigenvector is more important than the others in maintaining stability and connectivity of high centralized networks, such as Lilac, OClinks, and Wiki-Vote, but the nodes with higher betweenness are more powerful than the others in low centralized networks, such as Twitter, RenRen, and Facebook. Moreover, the empirical study revealed that low

centralized networks can tolerate high intentional attacks and have higher anti-interference capabilities than high centralized networks.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**; further inquiries can be directed to the corresponding authors.

AUTHOR CONTRIBUTIONS

Conceptualization, DZ and CG; methodology, DZ and ZZ; validation, DZ and GL; formal analysis, DZ, ZZ, and CG; resources, DZ and GL; data curation, DZ; original draft preparation, DZ; revise and editing, DZ, ZZ, and CG; project administration, CG; funding acquisition, DZ and ZZ. All authors have read and agreed to the published version of the manuscript.

ACKNOWLEDGMENTS

This work was partly supported by State Key Laboratory of Communication Content Cognition, People's Daily Online (No. A12001) and Natural Science Foundation of Heilongjiang Province of China (No. LC2018031).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fphy.2021.733224/full#supplementary-material

REFERENCES

1. Leimeister J. M., Sidiras P., and Krcmar H. Exploring Success Factors of Virtual Communities: the Perspectives of Members and Operators. *J Organizational Comput Electron commerce* (2006) 16:279–300. doi:10.1207/s15327744joce1603&4_7
2. Liu Z., and Hu B. Epidemic Spreading in Community Networks. *Europhys Lett* (2005) 72:315–21. doi:10.1209/epl/i2004-10550-5
3. Deng X. L, Ding H., Chen Y., Chen C., and Lv TJ. Novel Node Centrality-Based Efficient Empirical Robustness Assessment for Directed Network. *Complexity* (2020) 2020. 1-14. doi:10.1155/2020/8715619
4. Zeng Y., and Xiao R. A Networked Approach to Dynamic Analysis of Social System Vulnerability. *J Intell Fuzzy Syst* (2015) 28:189–97. doi:10.3233/IFS-141289
5. Kuhnle A., Nguyen N. P., Dinh T. N., and Thai M. T. Vulnerability of Clustering under Node Failure in Complex Networks. *Soc Netw Anal Min* (2017) 7:8. doi:10.1007/s13278-017-0426-5
6. Albert R., Jeong H., and Barabási A.-L. Error and Attack Tolerance of Complex Networks. *Nature* (2000) 406:378–82. doi:10.1038/35019019
7. Holmgren A. J. Using Graph Models to Analyze the Vulnerability of Electric Power Networks. *Risk Anal* (2006) 26:955–69. doi:10.1111/j.1539-6924.2006.00791.x
8. Callaway D. S., Newman M. E. J., Strogatz S. H., and Watts D. J. Network Robustness and Fragility: Percolation on Random Graphs. *Phys Rev Lett* (2000) 85:5468–71. doi:10.1515/9781400841356.510
9. Buldyrev S. V., Parshani R., Paul G., Stanley H. E., and Havlin S. Catastrophic Cascade of Failures in Interdependent Networks. *Nature* (2010) 464:1025–8. doi:10.1038/nature08932
10. Gao J., Buldyrev S. V., Havlin S., and Stanley H. E. Robustness of A Network of Networks. *Phys Rev Lett* (2011) 107:195701. doi:10.1103/PhysRevLett.107.195701
11. Cohen R., Ben-Avraham D., and Havlin S. Percolation Critical Exponents in Scale-free Networks. *Phys Rev E* (2002) 66:36113. doi:10.1103/PhysRevE.66.036113
12. Vázquez A., and Moreno Y. Resilience to Damage of Graphs with Degree Correlations. *Phys Rev E* (2003) 67:015101. doi:10.1103/PhysRevE.67.015101
13. Jalili M. Error and Attack Tolerance of Small-Worldness in Complex Networks. *J Informetrics* (2011) 5:422–30. doi:10.1016/j.joi.2011.03.002
14. Wu J., Tan S-Y., Liu Z., Tan Y-J., and Lu X. Enhancing Structural Robustness of Scale-free Networks by Information Disturbance. *Sci Rep* (2017) 7:7559. doi:10.1038/s41598-017-07878-2
15. Grubesic T. H., Matisziw T. C., Murray A. T., and Snediker D. Comparative Approaches for Assessing Network Vulnerability. *Int Reg Sci Rev* (2008) 31: 88–112. doi:10.1177/0160017607308679
16. Estrada E., and Hatano N. A Vibrational Approach to Node Centrality and Vulnerability in Complex Networks. *Physica A: Stat Mech its Appl* (2010) 389: 3648–60. doi:10.1016/j.physa.2010.03.030
17. Chen B. Y., Lam W. H. K., Sumalee A., Li Q., and Li Z.-C. Vulnerability Analysis for Large-Scale and Congested Road Networks with Demand Uncertainty. *Transportation Res A* (2012) 46:501–16. doi:10.1016/j.tra.2011.11.018
18. Morone F., and Makse H. A. Influence Maximization in Complex Networks through Optimal Percolation. *Nature* (2015) 524:65–8. doi:10.1038/nature14604
19. Boccaletti S., Latora V., Moreno Y., Chavez M., and Hwang D. Complex Networks: Structure and Dynamics. *Phys Rep* (2006) 424:175–308. doi:10.1016/j.physrep.2005.10.009
20. Latora V., and Marchiori M. Efficient Behavior of Small-World Networks. *Phys Rev Lett* (2001) 87:198701. doi:10.1103/PhysRevLett.87.198701
21. Crucitti P., Latora V., Marchiori M., and Rapisarda A. Error and Attack Tolerance of Complex Networks. *Physica A: Stat Mech its Appl* (2004) 340: 388–94. doi:10.1016/j.physa.2004.04.031
22. Freeman L. C. Centrality in Social Networks: Conceptual Clarification. *Social Networks* (1979) 1:215–39. doi:10.1016/0378-8733(78)90021-7
23. Uddin S., Hossain L., and Wigand R. T. New Direction in Degree Centrality Measure: Towards a Time-Variant Approach. *Int J Info Tech Dec Mak* (2014) 13:865–78. doi:10.1142/S0219622014500217
24. Newman M. E. J. A Measure of Betweenness Centrality Based on Random Walks. *Soc Networks* (2005) 27:39–54. doi:10.1016/j.socnet.2004.11.009
25. Wehmuth K., and Ziviani A. DACCER: Distributed Assessment of the Closeness Centrality Ranking in Complex Networks. *Computer Networks* (2013) 57:2536–48. doi:10.1016/j.comnet.2013.05.001
26. Li X. J., Liu Y .Z., Jiang Y. C., and Liu X. Identifying Social Influence in Complex Networks: A Novel Conductance Eigenvector Centrality Model. *Neurocomputing* (2015) 210:141–54. doi:10.1016/j.neucom.2015.11.123
27. Chen D., Lü L., Shang M.-S., Zhang Y.-C., and Zhou T. Identifying Influential Nodes in Complex Networks. *Physica A: Stat Mech its Appl* (2012) 391: 1777–87. doi:10.1016/j.physa.2011.09.017
28. Iyer S., Killingback T., Sundaram B., and Wang Z. Attack Robustness and Centrality of Complex Networks. *Plos One* (2013) 8:e59613. doi:10.1371/journal.pone.0059613
29. Nguyen Q., Pham H. D., Cassi D., and Bellingeri M. Conditional Attack Strategy for Real-World Complex Networks. *Physica A: Stat Mech its Appl* (2019) 530:121561. doi:10.1016/j.physa.2019.121561
30. Brin S., and Page L. The Anatomy of A Large-Scale Hypertextual Web Search Engine. *Comp Networks ISDN Syst* (1998) 30:107–17. doi:10.1016/S0169-7552(98)00110-X
31. Zengin Alp Z., and Gündüz Öğüdücü Ş. Identifying Topical Influencers on Twitter Based on User Behavior and Network Topology. *Knowledge-Based Syst* (2018) 141:211–21. doi:10.1016/j.knosys.2017.11.021
32. Erkol Ş., Castellano C., and Radicchi F. Systematic Comparison between Methods for the Detection of Influential Spreaders in Complex Networks. *Sci Rep* (2019) 9:15095. doi:10.1038/s41598-019-51209-6

33. Nie T., Guo Z., Zhao K., and Lu Z.-M. New Attack Strategies for Complex Networks. *Physica A: Stat Mech its Appl* (2015) 424:248–53. doi:10.1016/j.physa.2015.01.004
34. Liao H., Mariani M. S., Medo M., Zhang Y.-C., and Zhou M.-Y. Ranking in Evolving Complex Networks. *Phys Rep* (2017) 689:1–54. doi:10.1016/j.physrep.2017.05.001
35. Eom Y. H., Jeon C., Jeong H., and Kahng B. Evolution of Weighted Scale-free Networks in Empirical Data. *Phys Rev E Stat Nonlin Soft Matter Phys* (2008) 77:056105. doi:10.1103/PhysRevE.77.056105
36. Kujawski B., Hołyst J., and Rodgers G. J. Growing Trees in Internet News Groups and Forums. *Phys Rev E Stat Nonlin Soft Matter Phys* (2007) 76:036103. doi:10.1103/PhysRevE.76.036103
37. Slanina F. Dynamics of User Networks in On-Line Electronic Auctions. *Advs Complex Syst* (2014) 17. 1–14. doi:10.1142/S0219525914500027
38. Kirlangic A. The Rupture Degree and Gear Graphs. *Bull Malaysian Math Sci Soc* (2009) 32:31–6.
39. Bellingeri M., Bevacqua D., Scotognella F., Alfieri R., Nguyen Q., Montepietra D, et al. Link and Node Removal in Real Social Networks: A Review. *Front Phys* (2020) 8:228. doi:10.3389/fphy.2020.00228
40. Chvátal V. Tough Graphs and Hamiltonian Circuits. *Discrete Maths* (1973) 5:215–28. doi:10.1016/0012-365x(73)90138-6
41. Barefoot C. A., Entringer R., and Swart H. Vulnerability in Graphs—A Comparative Survey. *J Combin Math Combin Comput* (1987) 1:13–22.
42. Cozzens M. B., Moazzami D., and Stueckle S. Tenacity of Harary Graphs. *J Combin Math Combin Comput* (1994) 16:33–56.
43. Hendry G. R. T. Scattering Number and Extremal Non-hamiltonian Graphs. *Discrete Maths* (1988) 71:165–75. doi:10.1016/0012-365x(88)90069-6
44. Panzarasa P, OpsahlCarley T. K. M., and Carley K. M. Patterns and Dynamics of Users' Behavior and Interaction: Network Analysis of an Online Community. *J Am Soc Inf Sci* (2009) 60:911–32. doi:10.1002/asi.21015

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

frontiers in Physics

ORIGINAL RESEARCH
published: 08 November 2021
doi: 10.3389/fphy.2021.768336

Oracle Recognition of Oracle Network Based on Ant Colony Algorithm

Xianjin Shi[1,2] *and Xiajiong Shen*[1,3]*

[1]School of Computer and Information Engineering, Henan University, Kaifeng, China, [2]Henan Center for Educational Technology, Zhengzhou, China, [3]Henan Key Laboratory of Big Data Analysis and Processing, Henan University, Kaifeng, China

Recent studies have shown that compared with traditional social networks, networks in which users socialize through interest recommendation have obvious homogeneity characteristics. Recommending topics of interest to users has become one of the main objectives of recommendation systems in such social networks, and the widespread data sparsity in such social networks has become the main problem faced by such recommendation systems. Particularly, in the oracle interest network, this problem is more difficult to solve because there are very few people who read and understand the Oracle. To address this problem, we propose an ant colony algorithm based recognition algorithm that can greatly expand the data in the oracle interest network and thus improve the efficiency of oracle interest network recommendation in this paper. Using the one-to-one correspondence between characters and translation in Oracle rubbings, the Oracle recognition problem is transformed into character matching problem, which can skip manual feature engineering experts, so as to realize efficient Oracle recognition. First, the coordinates of each character in the oracle bones are extracted. Then, the matching degree value of each oracle character corresponding to the translation of the oracle rubbings is assigned according to the coordinates. Finally, the maximum matching degree value of each character is searched using the improved ant colony algorithm, and the search result is the Chinese character corresponding to the oracle rubbings. In this paper, through experimental simulation, it is proved that this method is very effective when applied to the field of oracle recognition, and the recognition rate can approach 100% in some special oracle rubbings.

Keywords: weak relationships, oracle social networks, interest-based recommendations, ant colony algorithms, heterogeneous networks

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Yu Wu,
Aerospace Information Research Institute (CAS), China
Xiaohua Hu,
Drexel University, United States

***Correspondence:**
Xiajiong Shen
shenxj@henu.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *31 August 2021*
Accepted: *04 October 2021*
Published: *08 November 2021*

Citation:
Shi X and Shen X (2021) Oracle Recognition of Oracle Network Based on Ant Colony Algorithm. Front. Phys. 9:768336. doi: 10.3389/fphy.2021.768336

INTRODUCTION

Recommendation systems, as an effective means of information filtering, can help users discover the content they are more interested in from a large amount of information [1–4]. Most traditional recommendation algorithms perform collaborative filtering recommendations with strong relational objects that are closely related to the target users and have good interpretability. However, strong relational recommendations often lead to single recommendations due to their own small circle and more content of invalid information generated than quality information content. In contrast to strong relationships, weak relationships have richer semantic information and stronger information influence. Therefore, recommendations based on weak relationships can get rid of the problems of single recommendation type and duplication of recommendation items brought by traditional recommendations based on strong relationships.

Social networks are usually regarded as homogeneous or heterogeneous structural networks [5, 6], where homogeneous social networks have a single type of nodes and relationships with strong asymmetry and small group size, which is not conducive to eliminating users' perception bias [7]. Oracal social network is a classical homogeneous networks. And because of the presence of a large number of oracle images in the oracle network, which are generally in the form of CAPTCHA, the task of recommendation in oracle social networks is a very difficult task. Therefore, the data sample for this interest-based social network recommendation is very small, especially for the oracle-based interest-based social network. Since reading oracle texts requires a lot of time and expertise, the data samples in oracle-based interest-based social networks often do not meet the quantitative requirements needed for network recommendation algorithms. and in oracle interest social networks, oracle pictures are generally presented in the form of CAPTCHA, which becomes more difficult to recognize oracle CAPTCHA compared to traditional oracle recognition.

However, there exist a large number of deciphered oracle rubbings, and these topographies corresponding to the translated text would largely advance the recommendation work in oracle-based interest-based social networks if they could be used for training of oracle CAPTCHA recognition. However, these deciphered topographies are not labeled with the correspondence between each character and the translated text, so they cannot be directly used for training.

Based on the above problems, this paper proposes an oracle rubbings character recognition method based on ant colony algorithm. The oracle characters in the bones and the characters of the interpreted text can be corresponded one by one, laying the foundation for the establishment of an oracle bone character library. The following is the contribution of this paper:

- Oracle bones are used as the identification unit and the use of manual annotation for generating oracle character sets for identification is skipped, thus manpower loss is reduced and work efficiency is improved.
- The euclidean distance-based ant colony algorithm is improved to character matching degree based ant colony algorithm, and the improved ant colony algorithm is applied to oracle bones character recognition to improve the recognition accuracy. The recognition rate reaches 100% in some rubbings with more standardized oracle character arrangement.

In *Related Works*, the principle of topology-based oracle character recognition is elaborated; in *Oracle Captcha Recognition*, how the traditional ant colony algorithm can be improved and applied to oracle character recognition is shown; in *Experimental Design and Analysis of Results*, the topology-based oracle character recognition proposed in this paper is simulated and modeled; in *Conclusion*, the whole paper is summarized and the direction of future work is described.

RELATED WORKS

Interest Network Recommendation Research

Network recommendation is to use the interaction data generated by the whole user group when browsing the Internet to filter the huge amount of information and provide the target users with the required information. Network recommendation not only saves its own operation cost, but also improves the user's loyalty and satisfaction. The first step in the construction of a web recommendation system is data input. The dramatic increase in the volume of Internet data and the number of users has greatly increased the cost of explicit scoring data, and the problem of data sparsity has become more serious, leading to a decrease in recommendation accuracy and user satisfaction.

User-project implicit interaction data for the purpose is an ideal solution to the problem [8, 9]. At present, the collection of user browsing behavior data is mainly divided into three ways, namely server-side, client-side and server-client. Firstly, the quantitative collected browsing behavior data is analyzed to get the implicit interest score of users to the project, and then the personalized recommendation list is obtained by using recommendation algorithms such as collaborative filtering according to users' different personalized needs and application scenarios. Finally, the results are pushed to the target users on the page in the form of ranked lists, images, links, etc.,

The popularity of artificial intelligence and big data has made applied research in the field of personalized recommendations even hotter. RecSys has placed increasing emphasis on modeling and analyzing user interaction behavior in recent years, and the 2017 Xing Social Network Job Recommendation Challenge generated job recommendations for job seekers by fusing user browsing behavior data with user and project attribute data [10]. A "Million User Playlist Dataset (MPD)" was provided by the Spotify online music platform in 2018, which is a dataset consisting of 1,000 sparse playlists, including title and meta and metadata, as well as playlist data from users' collections, to recommend 500 ordered waitlists to users the data set consists of 1,000 sparse playlists, including title and metadata, as well as playlist data from users' collections [11, 12]. In online global hotel search platform of [13], an anonymous dataset of nearly 16 million session interactions was provided, which involves over 700,000 users who visited the trivago website during the week of November 2018 for the purpose of providing online travel recommendation questions. A large data set of approximately 160 million public tweets provided by Twitter in 2020 public dataset containing retweet, like, comment, and retweet-plus-comment social browsing behavior data, user data, and Tweet data. The data set will contain data on social browsing behavior of retweets, likes, comments and retweets plus comments, user data and tweet data, and the data set will be protected by user privacy regulations for deleted tweets and user profiles. The data set of deleted tweets and user profiles is updated according to user privacy regulations, making it necessary for the challenger to must continuously retrain the data test set for

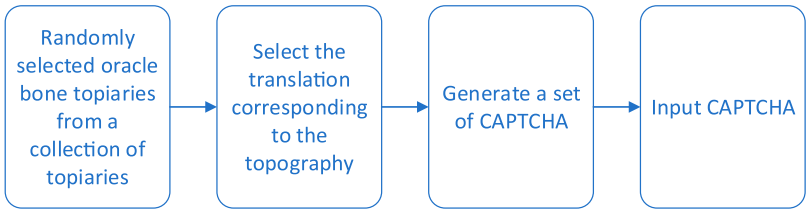


FIGURE 1 | Oracle-based CAPTCHA generation process.

predicting user browsing behavior Prediction of engagement [14]. Gao et al. solved the cold start problem caused by sparse user rating data by calculating the similarity between active users and a small number of expert users based on user collaborative filtering as the trust value [15]. Han et al. propose a collaborative filtering recommendation algorithm that incorporates item features and mobile user trust relationships to mitigate the effects of scoring data sparsity on collaborative filtering algorithms and improve the accuracy of mobile recommendations. Compared with traditional recommendation algorithms, the added "trust" mechanism effectively alleviates the data sparsity problem, but fundamentally still uses similarity as a criterion to measure the trust value, and does not change the subject of the recommendation, and the recommendation duplication problem is not solved. In contrast to strong relationships, weak relationships can be two-way or one-way, because they do not require strong interaction, so they are less costly to maintain. Relying on the power of "weak relationship" can expand the selection range of recommendations, increase the novelty of recommendations, reduce nagging content and make recommendations more clear [16].

Oracle-Based CAPTCHA Scheme

Due to the development of deep learning, both the text-based CAPTCHA scheme and the image-based CAPTCHA scheme mentioned above have different degrees of security risks. To improve the security line of CAPTCHA, researchers of text information systems have proposed an Oracle-based CAPTCHA scheme with the flow shown in **Figure 1**.

The captcha generation starts with a random selection of oracle rubbings, and a library of oracle rubbings is created in advance, each of which contains the corresponding translation, and the oracle rubbings contain annotation boxes for all oracle characters. **Figure 2** shows an example of saving to the oracle rubbings library.

As shown in **Figure 3**, the order of the oracle bone marker box starts from the top right and ends at the bottom left. Similarly, the order of the transliteration is also from the top right, "申(Shen)" is the first character in the topography, and "莫(Mo)" is the last character in the topography. The "□" represents the unrecognizable oracle bone character. Because of hand-carving, the shape of the same character in the Oracle rubbings is also very different, so the deep neural network still has a big defect in recognizing Oracle. This also provides a great security guarantee for this verification code solution. And for some text information systems with confidentiality requirements, using the captchas scheme based on Oracle can filter out some irrelevant personnel and improve the security of the system.

ORACLE CAPTCHA RECOGNITION

Using the correspondence between Oracle topology and interpretation text, this paper proposes an Oracle captcha recognition model based on ant colony algorithm. This model have both lower computing time and higher accuracy compared with the deep learning model.

Prerequisites and Conventions

Convention 1. Assume that all oracle bone inscriptions in oracle rubbings can be detected and annotated by applying oracle bone detection tools.

Constraint 2. Assume that the coordinates of the center point of the oracle markup box are the coordinates of the markup box. This is shown in **Figure 4**. The coordinates of this markup box are (123, 456).

FIGURE 2 | Example of storage in the oracle topography library.

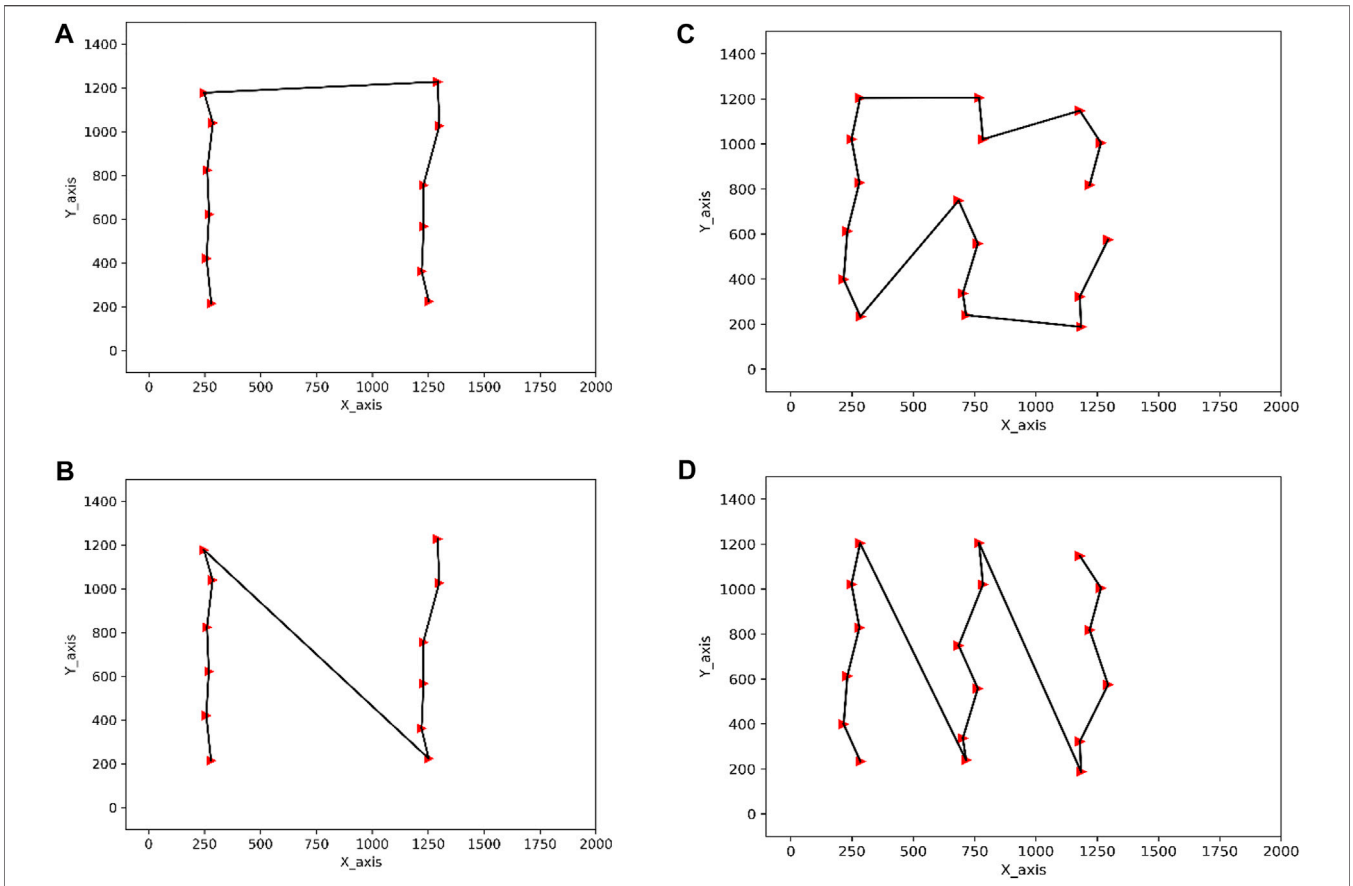


FIGURE 3 | (A) Basic ant colony algorithm—12 characters. **(B)** Improved ant colony algorithm—12 characters. **(C)** Basic Ant Colony Algorithm—18 characters. **(D)** Improved ant colony algorithm—18 characters.Coordinates of the oracle marker box.

Constraint 3. The distance of the markup box distance is the distance from the center of the markup box.

Constraint 4. The closer the distance between two oracle characters marker boxes in each line of an oracle rubbings, the higher the probability that these two oracle characters translations are adjacent to each other.

Identification Steps

The oracle bone writing is generally shown in **Figure 5**, and the writing is not uniform in any way. For researchers who do not know the oracle rubbings, they can only compare them one by one according to the interpretation. However, regardless of the rules of oracle bone writing, they all have a similar rule: the closer the distance between two oracle bone marker boxes, the higher the probability that the two oracle bone interpretations are adjacent to each other. Based on this law, this paper transforms the oracle bone captcha recognition problem into an optimization problem that can be solved using heuristic search algorithms.

The identification steps are as follows.

Step 1. Use the oracle bone detection algorithm according to convention 1 to detect all oracle bone characters in the oracle rubbings, extract the coordinates of all characters in the oracle rubbings and label them.

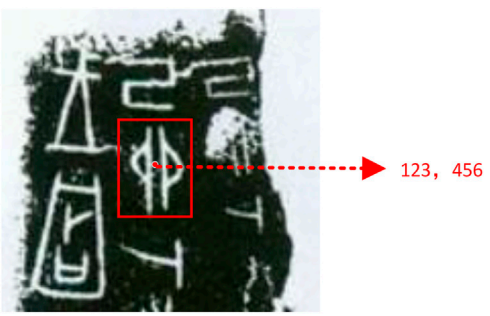


FIGURE 4 | Coordinates of the oracle marker box.

Step 2. Identify each oracle bone character in the topos according to the interpretation and the existing oracle bone character database, and the results of identification are arranged in descending order in this paper, and the top 3 are taken as the final results of identification. Here, the recognition results are named as matching degrees.

FIGURE 5 | Oracle rubbings.

Step 3. Calculate the distance between every two labeled boxes according to convention 3.

Step 4. Using the improved ant colony arithmetic, the oracle characters corresponding to the oracle rubbings translation are searched with the matching degree obtained in step 2 as the primary constraint and the distance obtained in step 3 as the secondary constraint, and the final search result is the recognition result of this paper.

Model

In this section, the oracle recognition problem is abstracted into a mathematical model for description. In this paper, all the labeled boxes on a piece of oracle rubbings are regarded as the vertices of the graph, and the coordinates of the center point of the labeled boxes are regarded as the coordinates of the vertices in the graph. Then they are defined as follows.

Definition 1. Figure $G = (V, W, D)$, the $V = \{v_1, v_2, v_3, \cdots, v_n\}$ denotes the graph G the set of vertices in the graph, and $W = \{w_1, w_2, w_3, \cdots, w_n\}$ denotes the set of matches between the vertices in graph G and the corresponding interpretations, and $E = \{d_{12}, d_{13}, d_{14}, \cdots, d_{(n-1)n}\}$ denotes the set of distances between the vertices in the graph, where d_{12} denotes the distance between the first vertex and the second vertex. In this paper, we call the graph G as the rubbings graph.

Definition 2. Assume the existence of paths $P = \{v_1, v_2, v_3, \cdots, v_n\}$, in which the order of the nodes is the same as the order of the oracle rubbings, then the path P is the Best Path.

According to definition 1 and definition 2, this paper transforms the Oracle recognition problem into finding the best path problem in graph G. The order of nodes in this path is the same as the order of the oracle interpretation. A formal language is used to describe it as follows.

It is known that a certain oracle rubbing has n oracle rubbings, the set $V = \{v_1, v_2, v_3, \cdots, v_n\}$ denotes the coordinates of each character in the oracle rubbings and $o(v_i, v_j) = c$, $1 \le i < j \le n$, $0 \le c \le 1$ indicates that the probability that the next Oracle character next to v_i on the Oracle rubbing is v_j is c. The sequence of a group of nodes is required to be $P = \{v'_1, v'_2, \cdots, v'_n\}$, and satisfies $max(\sum_{i=1}^{n} o(v'_i, v'_{i+1}))$, then P is the Best Path.

How to solve for the best path? Exhaustive enumeration is the simplest and the most accurate method. Suppose an oracle rubbings have n characters, then the size of the solution space is $S = (n-1)!/2$ and the solution with the highest matching degree is the best path solution. But when $n = 10$, we have $S \approx 1.8 \times 10^4$; when $n = 20$, we have $S \approx 6.0 \times 10^{16}$; when $n = 30$, we hve $S \approx 4.4 \times 10^{30}$. In the face of such a large solution space, it is impractical to use the exhaustive method.

In general, all problems that can be computed using polynomial-time algorithms and thus obtain results, we call P-problems [17–20]. All problems that can be solved in polynomial time to verify whether the solution of a problem is correct or not, we call NP problems, and it is important to note that the time used to solve the solution of the NP problem itself is unbounded. If there exists an NP problem where the time required to solve this NP problem itself is super-polynomial time, then we call this problem an NP-hard problem. Solving the optimal path problem is an NP-hard problem.

And a common method for solving NP-hard problems is to use heuristic search methods, of which the ant colony algorithm [21–23] is relatively common. In this paper, we propose a method for solving the optimal path based on the ant colony algorithm.

Ant Colony Algorithm

Ant colonies release a substance called pheromone in the process of foraging, and the more concentrated pheromone on a path means that more ants are walking on that path. Based on this principle, Maro Dorigo et al. proposed a bionic algorithm to simulate the foraging behavior of ants and named it the ant colony algorithm. The ant colony algorithm has two key steps: state transfer and pheromone update.

Let α denotes the information heuristic factor, which will affect the search range of the ant colony. With the increase of α, the search range of the ant colony will increase, but the randomness of the corresponding path selection will decrease. A decrease of α will make it easier to get a locally optimal solution. Let β denote the desired heuristic factor, and β affects the convergence speed of the algorithm. As the expected heuristic factor increases, the convergence speed of the algorithm will increase, but it will be easier to obtain the local optimal solution. Let τ_{ij} denote the path (i, j) the pheromone concentration of the path, d_{ij} denote the Euclidean distance of the path (i, j) and η_{ij} denote the heuristic function. Furthermore, let the set $J_k(i)$ denote the first city i that has not been visited by an ant. Then the path transfer formula of the ant colony algorithm is

$$P_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta}}{\sum_{\mu \in J_k(i)}[\tau(i,u)]^{\alpha}[\eta(i,u)]^{\beta}}, & j \in J_k(i) \\ 0, & , \ j \notin J_k(i) \end{cases} \quad (1)$$

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (2)$$

Let ρ denote the information volatility factor, then 1 − ρ denotes the residual factor. As ρ increases, it will cause some paths to be abandoned for search, which may contain valid paths and affect the search of optimal values. And as ρ decreases, it will cause repeated search and affect the convergence speed of the algorithm. Then the pheromone update formula at time $t + 1$ is

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (3)$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{M} \Delta\tau^{k}(t) \quad (4)$$

where $\Delta\tau_{ij}(t)$ denotes the pheromone concentration when the k-th ant circulates at time t. Let Q be the pheromone intensity, and L_k represents the total length of the path taken by the k-th ant, then

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L_k} \quad (5)$$

Find the best path based on ant colony algorithm

Based on the theory of the traditional ant colony algorithm, this paper designs an algorithm for finding the best path on Oracle rubbings according to the characteristics of Oracle recognition [24–26]. The transfer rules and pheromone update rules of the ant colony algorithm are mainly changed.

As mentioned above, the optimization goal of the best path is $max(\sum_{i=1}^{n} o(v'_i, v'_{i+1}))$, the same Convention 4 indicates that the closer the two Oracle characters are, the greater the probability that they are adjacent in the translation. However, when there is a line break in the Oracle rubbings, agreement 4 will not be met. Therefore, it is necessary to integrate when looking for the best path. Consider the distance relationship between the matching degree of the Oracle rubbings and the translation and the Oracle characters. According to (**Eq. 1**), this paper proposes the state transition formula for finding the best path as:

$$P_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta} o(i,j)}{\sum_{\mu \in J_k(i)}[\tau(i,u)]^{\alpha}[\eta(i,u)]^{\beta} o(i,u)} & j \in J_k(i) \\ 0, & , \ j \notin J_k(i) \end{cases} \quad (6)$$

$$\eta_{ij} = \frac{1}{d_{ij} o(i,j)} \quad (7)$$

where $o(i, j)$ denote Probability that the next node of the current node i is the j -th node. Then we have

$$o(i,j) = \frac{1}{2}(w_i + w_j) \quad (8)$$

where w_i denote the match between the oracle bone topos and the oracle bone translation of the first i character in the translation.

Since the pheromone concentration affects the paths chosen by the ant colony, the pheromone update rule needs to be modified as well. The goal of the modification is to increase the pheromone concentration on the paths with larger matches and decrease the pheromone concentration on the paths with smaller matches. The pheromone update rules are as follows.

$$\Delta\tau_{ij}^{k}(t) = \frac{Q \times o(i,j)^2}{L_k} \quad (9)$$

Implementation of improved ant colony algorithm

Input: Oracle rubbings $G = (V, W, D)$; Parameters α, β, ρ, m, maximum number of iterations $iter\_max$.

Output: The best path $P = (V', W', D')$.

Step 1: Initialize $iter\_max, \alpha, \beta, \rho, m$, and initialize W according to oracle identification algorithm.

Step 2: The roulette algorithm is used to select m nodes as the initial positions of the ants, and the initial positions are added to the taboo table.

Step 3: According to the state transfer formula, the probability of all possible arrival points is calculated, and the roulette algorithm is used to select the next node and add the next node to the forbidden table.

Step 4: The matches of the nodes on the path taken by all ants are summed and the total match W_{sum} is updated.

Step 5: Update the value of ρ to update the global pheromone.

Step 6: Determine if the maximum number of iterations is reached. If the maximum number of iterations is reached, the algorithm runs to the end and outputs the best path currently searched. If the maximum number of iterations is not reached, another iteration is added 1 and go to Step 3 to continue running the algorithm.

Based on the above analysis, this paper gives a pseudo-code example of the improved ant colony algorithm, as shown in **Algorithm 1**.

Line 1 indicates initialization α, β, ρ, m, $iter\_max$, w_{sum}, W, Q. Lines 2-4 indicate assigning a corresponding match value to each oracle character to be recognized. Line 5 indicates the use of the roulette wheel algorithm to select m The initial positions of the ants are selected using the roulette algorithm and the initial positions are added to the forbidden table. Line 6 indicates that the maximum iteration of the ant colony algorithm cannot be larger than iter_max and rows 7–9 indicate that the probability that all ants may reach each point is calculated according to the state transfer (**Eq. 1**) . Line 10 indicates that the roulette wheel algorithm is used to select the next node and add the next node to the forbidden table. Lines 11–16 indicate that the matches of the nodes on the path taken by all ants are summed up. Lines 17–24 indicate that the best path is selected W_{best}, and update the total matching degree W_{sum}. Line 25 indicates that the update of ρ according to (**Eqs 3, 4**). Line 26 indicates that the global pheromone is updated according to **Eq. 9**.

Algorithm 1 | The improved ant colony algorithm

Input: W, V

Output: W_{best}

Initialization $\alpha, \beta, \rho, m, iter\_max, w_{sum}, W, Q$

for $v_i \in V_i$, **do**

$v_i = w_i$

end for

use Roulette algorithm selects m vertices and **add** tabu list

for i $= 0,$ i $<$ iter_max; i $++$ **do**

  for $w_i \in W_i$ **do**

    recalculate w_i according to Eq. (1)

  End for

  use Roulette algorithm to select the next vertices and **add** a tabu list

  for $v_i \in V_i$, **do**

    $w_{temp} = 0$

  if the ants reach over v_i **then**

    $w_{temp} \mathrel{+}= w_i$

  end if

  end for

  if $w_{temp} > w_{sum}$ **then**

    $w_{sum} = w_{temp}$

  for $v_i \in V_i$, **do**

    if the ants reach over v_i **then**

w_i adds W_{best}

    end if

  end for

  end if

  update ρ according to Eq. (3) and Eq. (4)

  update Q according to Eq. (9)

end for

TABLE 1 | Parameter settings of ant colony algorithm.

Parameters	Basic ant colony algorithm	Improved ant colony algorithm
m Number of ant colonies	20	20
α	1	1
β	2	2
ρ	0.1	0.1
iter_max	100	100
Q Pheromone intensity	1	1
w	[0.9, 1) or (0, 0.5]	[0.9, 1) or (0, 0.5]

EXPERIMENTAL DESIGN AND ANALYSIS OF RESULTS

Experimental Environment and Data Set

The experimental setting for this paper is as follows.

Operating system	Windows 10
Python version	3.7
CPU	Intel i5-8300H
Memory	16GB
GPU	Quadro P4000
Video Memory	8GB
Deep Learning Framework	Tensorflow 2.1

Since there is no research on oracle CAPTCHA recognition and there is very little data about it, this paper uses simulation to conduct experiments. Firstly, the arrangement of oracle characters on the real oracle CAPTCHA is simulated on a 400* 240-pixel image and a series of coordinates are generated. This paper assumes that the recognition efficiency based on the individual characters of oracle is very high, so the corresponding matching degree is assigned as [0.9, 1), while the non-corresponding oracle match is assigned as (0,0.5]. The number of characters generated is divided into 2 categories: 12, 18.

Analysis of Experimental Results

The experimental parameters are set as shown in **Table 1**. The experimental results are shown in **Figure 3**.

From **Figure 3**, it can be seen that the improved ant colony algorithm, which is more in line with the recognition law of Oracle CAPTCHA, has a higher accuracy of recognition. **Table 2** gives a comparison of the convergence speed of the basic ant colony algorithm and the improved ant colony algorithm.

It can be seen from the above table that the improved ant colony algorithm has greatly reduced the number of iterations compared with the basic ant colony algorithm. However, due to the increase of the amount of calculation in each iteration, the total time of the improved ant colony algorithm increases compared with the basic ant colony algorithm, but the increase range is very limited.

TABLE 2 | Convergence speed of ant colony algorithm.

Parameters	Basic ant colony algorithm	Improved ant colony algorithm
Optimal number of iterations—12 characters	31	16
Optimal number of iterations—18 characters	52	29
System runtime—12 characters	1.21s	1.26s
System runtime—18 characters	2.31s	2.48s

CONCLUSION

The oracle based social network is a typical homogeneous network with strong asymmetry and small group size, and the recommendation task in oracle social network is a very difficult task because of the existence of a large number of oracle pictures in the oracle interest network, which generally exist in the form of CAPTCHA. Since oracle researchers have deciphered a very large number of oracle topographies, if we can use the deciphered oracle topographies to expand the data samples in oracle social networks, it will greatly improve the accuracy of recognizing oracle CAPTCHA in oracle social networks, and thus improve the accuracy of oracle network recommendation problem. Therefore, this paper proposes an oracle CAPTCHA recognition model based on ant colony algorithm. Firstly, the matching degree parameter is introduced based on the feature of higher recognition of single character rate of oracle topology. Secondly, the path selection rules and pheromone update rules of the ant colony algorithm are modified according to the matching degree. Finally, simulation experiments are conducted by simulating different classes of oracle verification codes. The experimental results show that the improved ant colony algorithm has better performance in the problem of identifying the best path, and it converges faster, with fewer iterations, and is more accurate than the traditional ant colony algorithm. This paper provides a new exploration direction for the recommendation problem in the oracle network, but there are few reference factors for the algorithm change at this stage, and there is still room for improvement in parameter optimization and other aspects.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

SHI and SHEN contributed to the conception or design of the work; All authors contributed to article revision, read, and approved the submitted version.

REFERENCES

1. Longe OB, Robert ABC, Onwudebelu U. *Checking Internet Masquerading Using Multiple CAPTCHA challenge-response Systems*. International Conference on Adaptive Science & Technology (2009). p. 244–9. doi:10.1109/ICASTECH.2009.5409718
2. Yan J, El Ahmad AS. Captcha Robustness: A Security Engineering Perspective. *Computer* (2011) 44(2):54–60. doi:10.1109/mc.2010.275
3. Saini B, Bala A. A Review of Bot protection Using CAPTCHA for Web Security. *IOSR J Comp Eng* (2018) 8(6):36–42. doi:10.9790/0661-0863642
4. Hernández C, Barrero D, Li S. *An oracle-based Attack on CAPTCHAs Protected against oracle Attacks*. arXiv preprint: arXiv:1702.03815 (2017).
5. Yan J, El Ahmad AS. Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms. In: Twenty-Third Annual Computer Security Applications Conference. Springer (2007). p. 279–91. doi:10.1109/ACSAC.2007.47
6. Yan J, El Ahmad AS. A Low-Cost Attack on a Microsoft Captcha. In: Proceedings of the 15th ACM Conference on Computer and Communications Security. Association for Computing Machinery (2008). p. 543–54. doi:10.1145/1455770.1455839
7. Lupkowski P, Urbanski M. SemCAPTCHA-user-friendly Alternative for OCR-Based CAPTCHA Systems. *Proc Int Multiconference Comp Sci Inf Tech* (2008) 3:325–9. doi:10.1109/IMCSIT.2008.4747260
8. Konstan JA, Miller BN, Maltz D, Herlocker JL, Gordon LR, Riedl J. GroupLens. *Commun ACM* (1997) 40(3):77–87. doi:10.1145/245108.245126
9. Nichols D. Implicit Rating and Filtering. In: Proc of the 5th DELOS Workshop on Filtering and Collaborative Filtering (1997). p. 10–2.
10. Peter K, Yashar D, Farshad B, Jens A, Gerard L, Philipp M. Proceedings of the Workshop on ACM Recommender Systems Challenge. In: RecSys Challenge '17: ACM Recommender Systems Challenge 2016 Workshop. Boston, United States: RecSys Community (2016).
11. Peter K, Yashar D, Farshad B, Jens A, Gerard L, Philipp M. Proceedings of the Workshop on ACM Recommender Systems Challenge. In: RecSys Challenge '17: ACM Recommender Systems Challenge 2017 Workshop. Como, Italy: RecSys Community (2017).
12. Peter K, Yashar D, Farshad B, Jens A, Gerard L, Philipp M. Proceedings of the Workshop on ACM Recommender Systems Challenge. In: RecSys Challenge '18: ACM Recommender Systems Challenge 2018 Workshop. Vancouver, Canada: RecSys Community (2018).
13. Peter K, Yashar D, Farshad B, Jens A, Gerard L, Philipp M. Proceedings of the Workshop on ACM Recommender Systems Challenge. In: RecSys Challenge '19: ACM Recommender Systems Challenge 2019 Workshop. Copenhagen, Denmark: RecSys Community (2019).
14. Peter K, Yashar D, Farshad B, Jens A, Gerard L, Philipp M. Proceedings of the Workshop on ACM Recommender Systems Challenge. In: RecSys Challenge '19: ACM Recommender Systems Challenge 2020 Workshop. [Online, Worldwide] RecSys Community (2020).
15. Gao F, Huang M, Zhang T. Collaborative Filtering Recommendation Algorithm Based on User Characteristics and Expert Opinions. *Comp Sci* (2017) 44(2):103–6. doi:10.11896/j.issn.1002-137X.2017.02.014
16. Han Z, Ghen Y, Liu W, Yuan B, Li M, Duan A. Research on Node Influence Analysis in Social Networks. *J Softw* (2017) 28(1):84–104. doi:10.13328/j.cnki.jos.005115
17. Tian Z, Gao X, Su S, Qiu J, Du X, Guizani M. Evaluating Reputation Management Schemes of Internet of Vehicles Based on Evolutionary Game Theory. *IEEE Trans Veh Technol* (2019) 68(6):5971–80. doi:10.1109/TVT.2019.2910217

18. Tian Z, Su S, Shi W, Du X, Guizani M, Yu X. A Data-Driven Method for Future Internet Route Decision Modeling. *Future Generation Comp Syst* (2019) 95: 212–20. doi:10.1016/j.future.2018.12.054
19. Gu Z, Wang L, Chen X, Tang Y, Wang X, Du X, et al. Epidemic Risk Assessment by A Novel Communication Station Based Method. *IEEE Trans Netw Sci Eng* (2021) 1:1. doi:10.1109/TNSE.2021.3058762
20. Gu Z, Li H, Khan S, Deng L, Du X, Guizani M, et al. IEPSBP: A Cost-Efficient Image Encryption Algorithm Based on Parallel Chaotic System for Green IoT. *IEEE Trans Green Commun Netw* (2021) 1:1. doi:10.1109/TGCN.2021.3095707
21. Gu Z, Hu W, Zhang C, Lu H, Yin L, Wang L. Gradient Shielding: Towards Understanding Vulnerability of Deep Neural Networks. *IEEE Trans Netw Sci Eng* (2021) 8(2):921–32. doi:10.1109/TNSE.2020.2996738
22. Zhang L, Huang Z, Liu W, Guo Z, Zhang Z. Weather Radar Echo Prediction Method Based on Convolution Neural Network and Long Short-Term Memory Networks for Sustainable E-Agriculture. *J Clean Prod* (2021) 298: 126776. doi:10.1016/j.jclepro.2021.126776
23. Zhang L, Xu C, Gao Y, Han Y, Du X, Tian Z. Improved Dota2 Lineup Recommendation Model Based on a Bidirectional LSTM. *Tinshhua Sci Technol* (2020) 25(6):712–20. doi:10.26599/TST.2019.9010065
24. Zhang L, Huo Y, Ge Q, Ma Y, Liu Q, Ouyang W. A Privacy Protection Scheme for IoT Big Data Based on Time and Frequency Limitation. *Wireless Commun Mobile Comput* (2021) 2021:1–10. doi:10.1155/2021/5545648
25. Han D, Chen J, Zhang L, Shen Y, Gao Y, Wang X. A Deletable and Modifiable Blockchain Scheme Based on Record Verification Trees and the Multisignature Mechanism. *CMES-Computer Model Eng Sci* (2021) 128(1):223–45. doi:10.32604/cmes.2021.016000
26. Lv L, Zheng C, Zhang L, Shan C, Tian Z, Du X, et al..Contract and Lyapunov Optimization-Based Load Scheduling and Energy Management for UAV Charging Stations. *IEEE Trans Green Commun Netw* (2021) 5(3):1381–94. doi:10.1109/TGCN.2021.3085561

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

frontiers in Physics

ORIGINAL RESEARCH
published: 12 November 2021
doi: 10.3389/fphy.2021.690748

Determining the Maximum States of the Ensemble Distribution of Boolean Networks

Xiaodong Cui[1]*, *Binghao Ren*[2] and *Zhenghan Li*[2]

[1]School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China, [2]School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, China

Inference of the gene regulation mechanism from gene expression patterns has become increasingly popular, in recent years, with the advent of microarray technology. Obtaining the states of genes and their regulatory relationships would greatly enable the scientists to investigate and understand the mechanisms of the diseases. However, it is still a big challenge to determine relationships from several thousands of genes. Here, we simplify the above complex gene state determination problem as an inference of the distribution of the ensemble Boolean networks (BNs). In order to investigate and calculate the distribution of the BNs' states, we first compute the probabilities of the different BNs' states and obtain the number of states Ω. Then, we find the maximum possible distribution of the number of the BNs' states and calculate the fluctuation of the distribution. Finally, two representative experiments are conducted, and the efficiency of the obtained results is verified. The proposed algorithm is conceptually concise and easily applicable to many other realistic models; furthermore, it is highly extensible for various situations.

Keywords: network, Gaussian distribution, state pattern, gene expression, Boolean

OPEN ACCESS

Edited by:
Chengyi Xia,
Tianjin University of Technology, China

Reviewed by:
Jinling Liang,
Southeast University, China
Zhipeng Zhang,
Tianjin University of Technology, China

***Correspondence:**
Xiaodong Cui
xdchoi@gmail.com

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *04 April 2021*
Accepted: *28 May 2021*
Published: *12 November 2021*

Citation:
Cui X, Ren B and Li Z (2021) Determining the Maximum States of the Ensemble Distribution of Boolean Networks. Front. Phys. 9:690748. doi: 10.3389/fphy.2021.690748

1 INTRODUCTION

Gene network is an important tool to study the biological system from the molecular level. Gene network is an interaction network formed by DNA, RNA, protein, and metabolic intermediates involved in gene regulation. Gene network research is expected to reveal the function and behavior of genome from a systematic perspective. It is helpful in explaining the life process in detail from the genomic level, so as to achieve the goal of systematically explaining cell activity, life activity, disease, and treatment. Therefore, gene network has attracted great attention in the study of biological growth, development, and diseases. The research results of gene network have important theoretical significance and application value.

Genetic regulatory network (GRN) has aroused lots of interests over the past years [1–3]. There exists a large proportion of genes regulating or interacting with the other genes through proteins, which can be modeled by the GRN. Various types of GRNs, such as Boolean networks (BNs) and extended probabilistic Boolean networks, stochastic Boolean networks, and multiple-valued networks [4–6], have been developed for different applications. For example, BNs were first proposed by Kauffpman [7, 8] to model the complex and nonlinear biological systems. Furthermore, various factors, such as gene perturbation, context-sensitive, and asynchronous, are also thoroughly investigated [9, 10].

However, the research results of BNs are relatively limited, due to the difficulties for solving logical dynamic systems with a systematic tool [11]. In the viewpoint of biology, considering there are a huge

number of genes expression states at the same time, this incurs the difficulties of inferring the states of the gene expression at a given time stamp.

Recently, Cheng et al. [11] proposed the semi-tensor product (STP) of matrices, which can only represent the logical equation as an algebraic equation, but also convert the dynamics of a BCN into a linear discrete-time control system. Based on such reformation, many interesting properties have been obtained for BCN [12–16]. The optimal control is an interesting topic in system control theory. Other than the STP technique, they developed statistical methods for solving problems in BNs. A Mayer-type optimal control problem for BCNs with multi-input and single-input has been well studied in Refs. [17] and [18], respectively. The states of biological networks and electronic networks are often influenced by instantaneous disturbances. In addition, they may still experience abrupt changes at certain points, because of the switching phenomenon and sudden noise, that is, impulsive effects. Impulsive dynamical networks have attracted the interests of many researchers for their various applications in information science, bioinformatics, and automated control systems.

There are many cells with the same function in an organ. However, it is hard to get the states of every single cell. Here in this study, we find that the states of a proportion of cells share one particular distribution. Thus, it is useful for biologists to conclude whether the illness is caused by the changes of the cell state distribution or not.

From a biological standpoint, inference of gene regulation mechanism from expression patterns is becoming increasingly important, along with the invent of DNA microarray technology. Thus, we need to get the ensemble distribution of the BNs and determine the states of genes, which is the key for further exploration of the expression profiles of thousands of genes. Specifically, in this study, we proposed an algorithm for inferring the distribution states of the BNs. First, we compute the probability of different BNs' states and get the value of Ω. Second, we find the maximum possible distribution of the number of BNs' states, as well as the function of this distribution. Finally, two representative experiments are conducted to verify the efficiency of the obtained results. Although the practical genetic networks are different from the BNs in this study, the theoretical and practical results can be extended easily to the real-world scenarios. Moreover, the proposed algorithm is highly extensible in various scenarios because of the computational simpleness.

2 THE FINITE NUMBER OF BOOLEAN NETWORKS

2.1 The States of Boolean Networks

This section provides a base knowledge for **Section 2.1**. Ω is the only hypothesis. In this section, we assume that the probability of each state is equivalent, which is used for the next efficiency.

First, we suppose that there are many Boolean networks in one group, and the probability of different BNs' states is P.

$$P(\Omega) = \frac{1}{\Omega}, \tag{1}$$

where Ω is the number of BNs.

We assume that ψ_j is however the jth state, M_j is the number of ψ_j in the BNs, and E_j is the weight of ψ_j. Evidently, the number of states is M, which is calculated as follows:

$$M = \sum_j M_j, \tag{2}$$

and the value of the cells $\mathcal{E}$ is gives as

$$\mathcal{E} = \sum_j M_j E_j. \tag{3}$$

Although we know the number of cells, it is difficult to determine, even if a distribution M_j is given, what the specific state of each cell is. For example, suppose there are three cells in state 1 and five cells in state 2, we do not know which three cells are in state 1 and which five cells are in state 2. So the theorem 1 is given as follows in order to solve this problem.

Theorem 1We know the number of BNs in the ensemble is M and the value of the ensemble $\mathcal{E}$. Given a distribution $\{M_j\}$, it is easy to determine the number of states Ω as

$$\Omega = \frac{M!}{\prod_j M_j}. \tag{4}$$

Proof: The system consists of M number of identical transforms, which have $M!$ permutations. Given the condition that the total number of states do not change, if there exists n transforms $M_1 \to M_2$, denoted as the state 1 switching to the state 2, the number of M_2 states will increase by n, while the number of M_1 states will decrease by n. Therefore, the state permutation number is $\prod (pi) M_j!$, and

$$\Omega = \frac{M!}{\prod_j \left(M_j!\right)}.$$

Two specific examples are given to illustrate Theorem 1, while there is an ensemble with 5 BNs. Thus, $M = 5$.

(1) We assume that there are three Boolean networks in state j_1 and two Boolean networks in state j_2, then Ω is

$$\Omega = \frac{5!}{3!2!} = 10. \tag{5}$$

(2) We assume that there are four Boolean networks in state j_1 and one Boolean network in state j_2, then Ω is

$$\Omega = \frac{5!}{4!1!} = 5. \tag{6}$$

We can easily find that even if the number of M is very large, the conclusion still holds.

2.2 The Maximum Probabilistic Distribution of Boolean Networks

In this section, we study and prove the maximum probabilistic distribution of the Boolean network. The maximum probabilistic

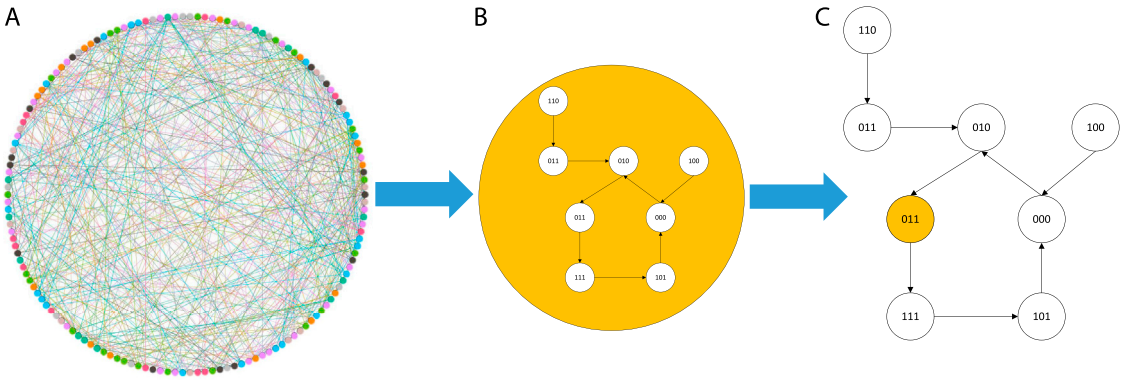


FIGURE 1 | State change in a system: **(A)** all nodes' state change, **(B)** one node of these nodes, and **(C)** the state change rule for one node, and there are eight states in these nodes.

TABLE 1 | Number of initial states of cells.

State of cell	**(0, 0)**	**(0, 1)**	**(1, 0)**	**(1, 1)**
Number	198	182	319	301

distribution is a Gaussian distribution, and then the cells' states distribution can be determined as shown in **Figure 1**.

Although given $\mathcal{E}$, M, and the distribution M_j, it is not easy to figure out the particular states where the BNs are. The best probability of the distribution needs further calculation. Given **Eq. 1**, we can find that the more states in the system, the larger probability the states are. The probability of each distribution of the ensemble networks is proportional to the number of the BN state Ω. Thus, when determining the maximum probability, the maximum Ω should be specified. Under the constrained conditions (2) and (3), we can use the differentiation to calculate the maximum value of the states. Two Lagrange multipliers α and ß will be utilized, and the condition of the peak can be written as follows:

$$\frac{\partial}{\partial M_j}\ln\Omega - \alpha\frac{\partial\sum_j M_j}{\partial M_j} - \beta\frac{\partial\sum_j M_j E_j}{\partial M_j} = 0. \tag{7}$$

To determine the probability, we need to assume that the number of M is relatively large. In contrast, the data of BNs do not need be large. When M goes to infinite, M_j also goes to infinite. For $M \gg 1$, we can use the Stirring's approximation to simulate $M!$

$$M! = \left(\frac{M}{e}\right)^M \sqrt{2\pi M}\left(1 + \frac{1}{12M} + \frac{1}{288M} + \cdots\right). \tag{8}$$

Using **Eq. 8** (the specific calculation process is shown in the Appendix), we can get the following equation:

$$\ln\Omega = M\ln M - M - \sum_j M_j \ln M_j + \sum_j M_j. \tag{9}$$

When we compute the partial derivative of ln Ω, there are two ways to solve this problem (**Eq. (8)**), that is, one is fixing the M, while the other does not fix the M. The difference between the two solutions is a constant. In order to boost the computation, the second way for solving **Eq. 9** is used.

$$\frac{\partial \ln\Omega}{\partial M_j} = -\ln M_j, \tag{10}$$

$$\frac{\partial M}{\partial M_j} = 1, \tag{11}$$

$$\frac{\partial \mathcal{E}}{\partial M_j} = E_j. \tag{12}$$

Substituting **Eqs 10–12** into **Eq. 7**, we can get the following equations:

$$-\ln M_j - \alpha - \beta E_j = 0, \tag{13}$$

$$\ln M_j = -\alpha - \beta E_j, \tag{14}$$

So there is

$$M_j = e^{-\alpha-\beta E_j}. \tag{15}$$

When given the number of BN M, represented as the scale, we can get the distribution M_j, given that the parameters α and ß should be specified in advance. To prove Theorem 2, two definitions are given as follows.

Definition II1 When P_j is the best probability distribution, the probability of system in the state j is

$$P_j \equiv \frac{M_j}{M} = \frac{e^{-\beta E_j}}{\sum_j e^{-\beta E_j}}. \tag{16}$$

Definition II 2 Partition function [19] is

$$Q \equiv \sum_j e^{-\beta E_j}, \tag{17}$$

where Q indicates the sum of the probability of all the states. The partition of **Eq. 17** plays an important role as a normalization

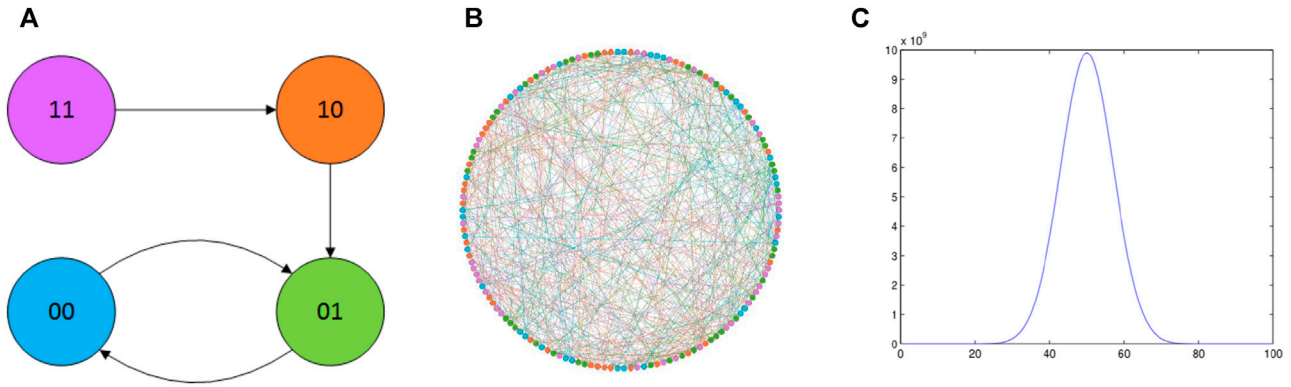


FIGURE 2 | Cells of the system **(**31**)**. The state change in the system (31), and the distribution of these states of combination. **(A)** Node's states change, and there are four states in these cells; **(B)** the relation of 100 cells, and these cells were randomly generated; **(C)** main result of the system (31): the distribution of these states of combination.

TABLE 2 | Cellular network statistical characteristics.

Node	Edge	Average degree	Clustering coefficient
1,000	2,781	2.45	0.04

constant.$E \equiv \frac{\mathcal{E}}{M}$ and $E = \frac{1}{Q}\sum E_j e^{-\beta E_j}$ is the definition of E for succinctly, and the latter one in terms of formula expression is good for clarity and following computation.After the computation of P_j, α can be eliminated, and β can be expressed by the mean value E:

$$E \equiv \frac{\mathcal{E}}{M}, \tag{18}$$

From **Eq. 3**, it can be rewritten as

$$\begin{aligned} E &= \frac{\sum_j M_j E_j}{\sum_j M_j} \\ &= \frac{\sum_j e^{-\beta E_j} E_j}{\sum_j e^{-\beta E_j}} \end{aligned}. \tag{19}$$

Replacing **Eq. 19** with **Eq. 17**, we can get

$$E = \frac{1}{Q}\sum_j E_j e^{-\beta E_j}. \tag{20}$$

From the result, we can get the information about that in a canonical ensemble. When E is given, M tends to infinite, P_j and β do not have any relationship with M.

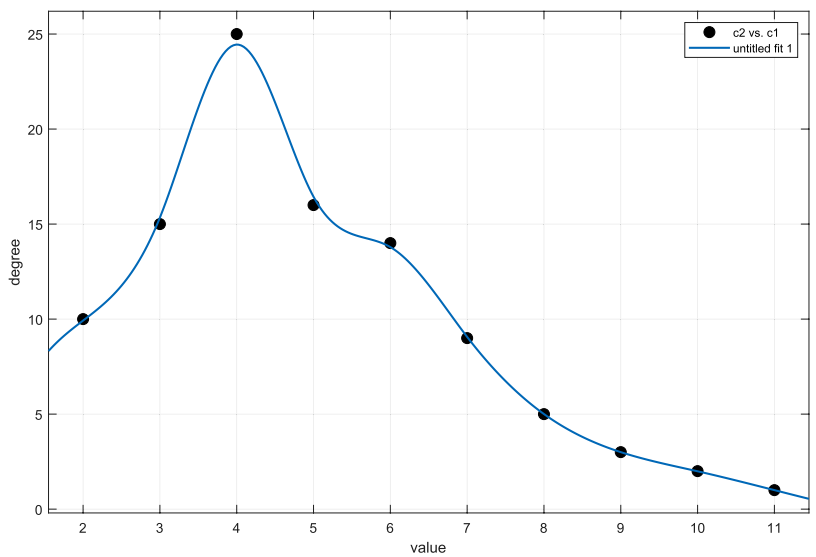


FIGURE 3 | Distribution of the cellular network.

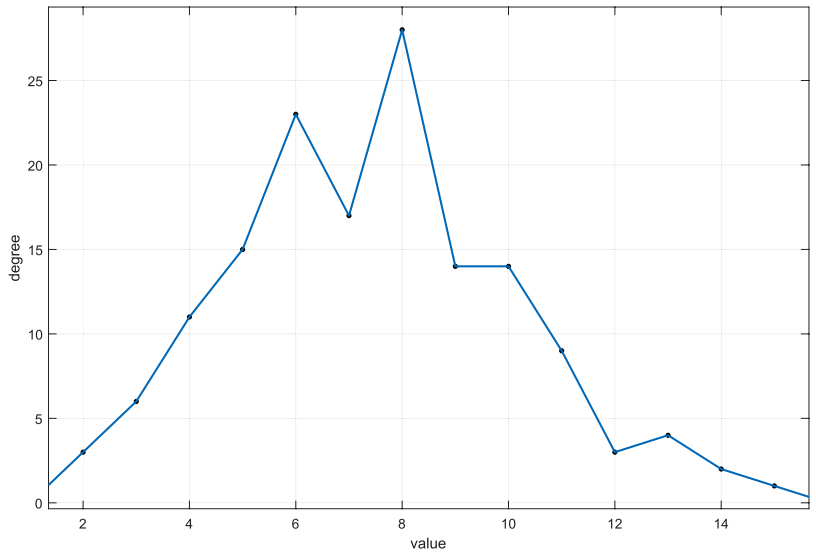


FIGURE 4 | Distribution of the cellular network.

TABLE 3 | Number of initial states of cells.

State of cell	**(0, 0, 0)**	**(0, 0, 1)**	**(0, 1, 0)**	**(0, 1, 1)**
Number	54	60	57	48
State of cell	(1, 0, 0)	(1, 0, 1)	(1, 1, 0)	(1, 1, 1)
Number	51	54	66	42

TABLE 4 | Cellular network statistical characteristics.

Node	**Edge**	**Average degree**	**Clustering coefficient**
450	1890	3.73	0.046

Theorem 2When H and E are given and M tends to infinite, the best of the distribution M is the true distribution. In other words, the fluctuation is equal to 0.

ProofWe need to talk about a function,

$$f \equiv f\left(M_j\right) \equiv \ln\,\Omega - \alpha\sum_j E_j - \beta\sum_j M_j E_j. \tag{21}$$

However,

$$\frac{\partial^2 f}{\partial M_j^2} = \frac{\partial^2 \ln\,\Omega}{\partial M_j^2} = -\frac{1}{M_j} < 0. \tag{22}$$

Since the second term and the third term of f are the linear functions, the second derivative of M_j equals to zero, which

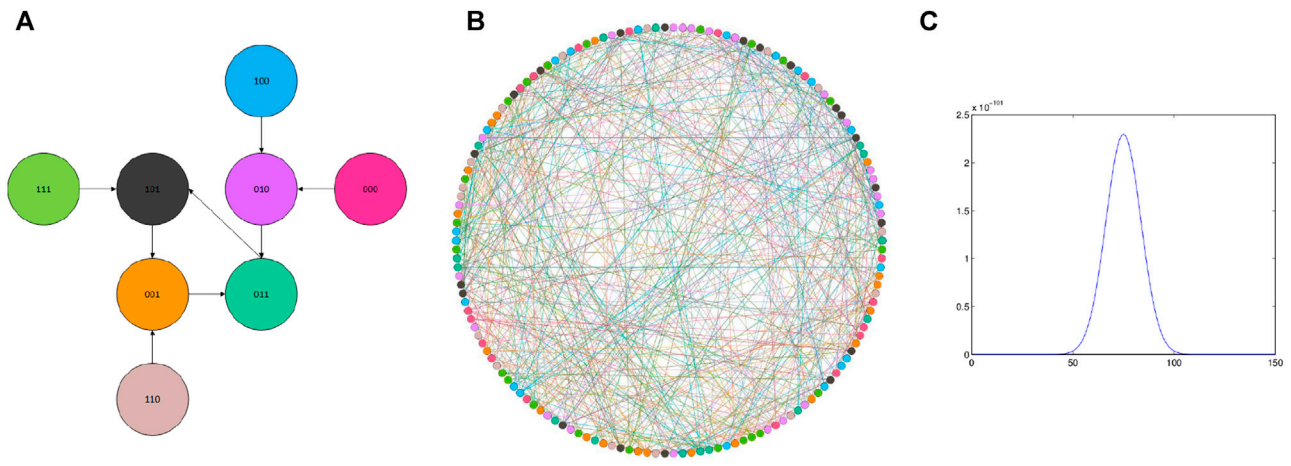


FIGURE 5 | Cells of the system **(**33**)**. State change in the system (33), and the distribution of these states of combination: **(A)** node's state change, and there are eight states in these cells; **(B)** relation of 150 cells, and these cells were randomly generated; **(C)** result of the system (33): the distribution of these states of combination.

means the peak is stable.Using the Taylor series which starts $f(M_j)$ at point $\overline{M_j}$, the equation can be obtained as follows:

$$\begin{aligned} f\left(M_j\right) &= f\left(\overline{M_j}\right) + \sum_j \frac{\partial f}{\partial M_j}\left(M - \overline{M_j}\right) + \sum_j \frac{1}{2!}\frac{\partial^2 f}{\partial M_j^2}\left(M - \overline{M_j}\right)^2 + \cdots, \\ &= f\left(\overline{M_j}\right) - \sum_j \frac{\partial f}{\partial M_j}\left(M - \overline{M_j}\right) + O\left(\frac{\Delta M}{M}\right).p. \end{aligned} \tag{23}$$

The peak of f is as follows:

$$\overline{f} = f\left(\overline{M_j}\right) = \ln \overline{\Omega} - \alpha M - \beta ME. \tag{24}$$

Substituting **Eqs 21** and **24** into **Eq. 23**, we can get

$$\ln \Omega = \ln \overline{\Omega} - \sum_j \frac{1}{2Mp_j}\left(M - \overline{M_j}\right)^2 + O\left(\frac{\Delta M}{M}\right). \tag{25}$$

Ignoring the term $O\left(\frac{\Delta M}{M}\right)$, we can get

$$\ln \frac{\Omega}{\overline{\Omega}} = -\sum_j \frac{1}{2Mp_j}\left(M - \overline{M_j}\right)^2. \tag{26}$$

So there is

$$\Omega = \overline{\Omega} e^{-\sum_j \frac{1}{2Mp_j}\left(M - \overline{M_j}\right)^2}. \tag{27}$$

Thus, we complete the proof of this theorem.

2.3 The Fluctuation of the Distribution

This section is aiming to prove that cells are impossible in the same states, when the number of cells goes to infinity.

It is easy to find that **Eq. 27** is a Gaussian distribution. Now, we need to prove the function **Eq. 7** is a δ function. We need to prove the fluctuation would be eliminated when $M \to \infty$. Here, Theorem three is provided as follows:

Theorem 3 When $M \to \infty$, the value of fluctuation tends to be 0, that is,

$$fluctuation \equiv \sqrt{\frac{\overline{M_j^2} - \overline{M_j}^2}{\overline{M_j}^2}} = \sqrt{\frac{MP_j}{\left(MP_j\right)^2}} \to 0. \tag{28}$$

Proof: There is a distribution that

Obviously, there is $P(x) \propto e^{-\frac{x^2}{2\Delta^2}}$.

$$\overline{x} = 0$$

and

$$\overline{x^2} = \frac{\int x^2 e^{-\frac{x^2}{2\Delta^2}} dx}{\int e^{-\frac{x^2}{2\Delta^2}} dx}. \tag{29}$$

Then **Eq. 29** can be rewritten as

$$\begin{aligned} \overline{x^2} &= \frac{\int x^2 e^{-\frac{x^2}{2\Delta^2}} dx}{\int e^{-\frac{x^2}{2\Delta^2}} dx} \\ &= -2\frac{\partial}{\partial\left(\frac{1}{\Delta^2}\right)} \ln\left[\int e^{-\frac{x^2}{2\Delta^2}} dx\right} \\ &= \Delta^3 \frac{\partial}{\partial\left(\frac{1}{\Delta}\right)} \ln \Delta\left[\int e^{-\frac{x^2}{2}} dx\right] \\ &= \Delta^3. \end{aligned} \tag{30}$$

Comparing **Eq. 27** with **Eq. 30**, we can get

$$\left(M_j - \overline{M_j}\right)^2 = \overline{M_j^2} - \overline{M_j}^2 = MP_j$$

and substituting it into **Eq. 28**, there is

$$fluctuation \equiv \sqrt{\frac{\overline{M_j^2} - \overline{M_j}^2}{\overline{M_j}^2}} = \sqrt{\frac{MP_j}{\left(MP_j\right)^2}} = \sqrt{\frac{1}{MP_j}} = 0,$$

where $M \to \infty$. Hence, the proof of the theorem is completed.Until now, the proof of Theorem three is finished. When H and E are fixed and $M \to \infty$, the distribution with the maximum probability is the true distribution.

3 EXPERIMENTS

In this section, we perform analysis of the cells' states distribution model, that is, **Eq 26**. We establish that two experiments are conducted in order to illustrate the distribution of the BNs' states, which can be used to verify our conclusions. Since there are no practical data for the state changes of the same type of cells, we can only simulate the transformation process of these cells through Boolean network, and then we also perform extensive analyses of the data of the state changes of these cells.

3.1 A Boolean Network with 100 Cells

In this example, we choose the state change function [17]. While the number of cells is 100, the number of the same Boolean is 1,000. And the Boolean network's state change rule is illustrated as follows:

$$\begin{cases} x_1(t+1) = x_1(t) \wedge x_2(t) \\ x_2(t+1) = \neg x_2(t) \end{cases}, \tag{31}$$

where (x_1, x_2) indicates the cell's state, while $x = 1$ or 0, and the function indicates the state change rule. Hence, in this example, there are four states in 100 cells, and the state change rule is shown in **Figure 2B**.

Assume that the number of four initial states in the cells is shown in **Table 1**.

From Theorem 1, we can obtain the k combinations.

$$k = \frac{1000!}{198!182!349!301!} = 8.584 \times 10^{200}.$$

We generate the particular network relationship between cells in a random manner, where each node represents a cell, and the edge indicates a connection between two cells. The probability of connecting the two cells is initialized as 0.05. The indicators of the association network between the cells are shown in the following table.

Through **Figure 3**; **Table 2**, we get the basic characteristics of this cellular network; there are 1,000 nodes, 2,781 edges, and so on. The visualization of the network is shown in **Figure 2B**. In this figure, different colors of the nodes are expressed as different states of the cells.

When the cell states change, they will be initialized with a random state, and the influence of states by other states is modeled as well. Assuming that the number of identical states between the connected cells is greater than 10, the other cells directly skip the changed state, and switch directly to the next state. Thus, the function Ω can be obtained as follows:

$$\Omega = \overline{\Omega} e^{-\sum_j \frac{1}{2M\overline{P_j}} \left(M - \overline{M_j}\right)^2}. \tag{32}$$

where Ω indicates the distribution, $P_i, i = 1, 2, 3, 4$ is the probability of the cells state, M is the number of cells, and $\overline{M_j}$ is the number of jth states.

The state change rule as shown in **Figure 2A** demonstrates the end state is $(0, 0)$, meaning the cells getting the state $(0, 0)$ twice. In addition, the state of the cells will be randomly assigned, in **Figure 2B**, and it is easy to find that when $x = 50$, the distribution reaches its mode, showing that when all the states of the cells are equal, the state in the collection of cells is the most prominent.

3.2 A Boolean Network with 150 Cells

In this example, we choose the state change function similar to the previous reported one [12]. Here, the number of cells is 500, meaning the number of the same Boolean is 500. Along with the Boolean network's state change, the mathematical rules can be formatted as

$$\begin{cases} x_1(t+1) = x_2(t) \wedge x_3(t) \\ x_2(t+1) = \neg x_1(t) \\ x_3(t+1) = x_2(t) \vee x_3(t) \end{cases}. \tag{33}$$

where (x_1, x_2) mean the cell's state, and $x = 1$ or 0, and the function is the state change rule, so in this example, there are eight states in 450 cells, and the state change rule is shown in **Figure 4B**.

Assume that the number of four initial states in the cells is as shown in **Table 3**.

Form Theorem 1, we can get there are about k combinations.

$$k = \frac{150!}{18!20!19!16!17!18!22!14!} = 8.6616 \times 10^{367}.$$

We generate a network relationship among cells in a random manner, where each node represents the cell, and the edge indicates that there is a connection between the two cells, and the probability of connecting the two cells is 0.05. The indicators of the association network between the cells are shown in the following **Table 4**.

Through **Figure 4**; **Table 3**, we get the basic characteristics of this cellular network; there are 450 nodes, 1890 edges, and so on. The visualization of the network is shown in **Figure 5B**. Here, different colors of the nodes are expressed as different states of the cells.

The state change rule as shown in **Figure 5A**, and the end state is $(0, 0, 1)$, meaning the cells get the state $(0, 0, 1)$ twice, and the state of the cells will be randomly assigned, in **Figure 5B**; it is easy to find that when $x \simeq 18$, the distribution reaches the peak. It means that when all the states of the cells are equal and the number of the eight states is approximately equal to 18, the collection of cells is the most prominent state.

From these two experiments, we verify that the distribution of these states is a Gaussian distribution, and these cells cannot be in the same state when the number of cells approaches to the infinity. Thus, the above theorems are right.

4 CONCLUSION

In this article, we study and calculate the distribution of the Boolean networks' states. First, we compute the probability of different BNs' states and get the value of Ω, then we find the maximum possible distribution of the number of BNs' states. Furthermore, we calculate the fluctuation of the distribution. Finally, two representative experiments are conducted to verify the efficiency of the obtained results. Although the real genetic networks are different from the BNs, the theoretical and practical results in this study may be extended for more realistic models. Since the proposed algorithm is conceptually concise and efficient, it is highly extensible for various situations.

5 DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material; further inquiries can be directed to the corresponding author.

6 AUTHOR CONTRIBUTIONS

XC drafted the idea. ZL did the derivation, while BR drafted the manuscript. All authors have read through the manuscript.

7 FUNDING

This work was supported in part by National Natural Science Foundation of China (Grant Nos. 62003273, 62073263), Natural Science Foundation of Shaanxi Province (Grant No. 2020JQ-217), Fundamental Research Funds for the Central Universities (Grant No. 3102019HHZY03002).

REFERENCES

1. Ideker T, Galitski T, and Hood L. A NEWAPPROACH TODECODINGLIFE: Systems Biology. *Annu Rev Genom Hum Genet* (2001) 2:343–72. doi:10.1146/annurev.genom.2.1.343
2. Kim J, Park S-M, and Cho K-H. Discovery of a Kernel for Controlling Biomolecular Regulatory Networks. *Sci Rep* (2013) 3:2223. doi:10.1038/srep02223
3. Zhang Z, Xia C, and Chen Z. On the Stabilization of Nondeterministic Finite Automata via Static Output Feedback. *Appl Math Comput* (2020) 365:124687. doi:10.1016/j.amc.2019.124687
4. Shmulevich I, Dougherty ER, Kim S, and Zhang W. Probabilistic Boolean Networks: a Rule-Based Uncertainty Model for Gene Regulatory Networks. *Bioinformatics* (2002) 18(2):261–74. doi:10.1093/bioinformatics/18.2.261
5. Liang J, and Han J. Stochastic Boolean Networks: An Efficient Approach to Modeling Gene Regulatory Networks*BMC Syst Biol* (2012) 6:113. doi:10.1186/1752-0509-6-113
6. Peican Zhu P, and Jie Han J. Stochastic Multiple-Valued Gene Networks. *IEEE Trans Biomed Circuits Syst* (2014) 8(1):42–53. doi:10.1109/tbcas.2013.2291398
7. Kauffman SA. Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets. *J Theor Biol* (1969) 22:437–67. doi:10.1016/0022-5193(69)90015-0
8. Kauffman SA. *The Origins of Order. Self-Organization and Selection in Evolution*. Oxford University Press (1993).
9. Zhu P, Liang J, and Han J. Gene Perturbation and Intervention in Context-Sensitive Stochastic Boolean Networks. *BMC Syst Biol* (2014) 8–60. doi:10.1186/1752-0509-8-60
10. Zhu P, and Han J. Asynchronous Stochastic Boolean Networks as Gene Network Models[J]. *J Compu. Biol.* (2014) 21(10):771–83. doi:10.1089/cmb.2014.0057
11. Cheng D. *Analysis and Control of Boolean Networks: A Semi-Tensor Product Approach[M]*. Berlin: Springer (2010).
12. Cheng D, and Qi H. A Linear Representation of Dynamics of Boolean Networks. *IEEE Trans Automat Contr* (2010) 55:2251–8. doi:10.1109/tac.2010.2043294
13. Li R, Yang M, and Chu T. State Feedback Stabilization for Boolean Control Networks. *IEEE Trans Automat Contr* (2013) 58:1853–7. doi:10.1109/tac.2013.2238092
14. Li B, Lu J, Liu Y, and Wu Z-G. The Outputs Robustness of Boolean Control Networks via Pinning Control. *IEEE Trans Control Netw Syst* (2020) 7(1):201–9. doi:10.1109/tcns.2019.2913543
15. Liu A, and Li H. On Feedback Invariant Subspace of Boolean Control Networks. *Sciece China Inf Sci* (2020) 63(12):229201. doi:10.1007/s11432-019-9869-6
16. Zhang Z, Xia C, Chen S, Yang T, and Chen Z. Reachability Analysis of Networked Finite State Machine with Communication Losses: A Switched Perspective. *IEEE J Select Areas Commun* (2020) 38(5):845–53. doi:10.1109/jsac.2020.2980920
17. Laschov D, and Margaliot M. *Observability of Boolean Networks: A Graph-Theoretic Approach*. Cambridge, U.K.: Cambridge Scientific Publishers, Cambridge (2013).
18. Laschov D, and Margaliot M. A Maximum Principle for Single-Input Boolean Control Networks. *IEEE Trans Automat Contr* (2011) 56:913–7. doi:10.1109/tac.2010.2101430
19. Baxter RJ. Partition Function of the Eight-Vertex Lattice Model. *Ann Phys* (2000) 281(1-2):187–222. doi:10.1006/aphy.2000.6010

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

APPENDIX

Logarithm **Eq. 4**, we can get

$$\begin{aligned} \ln \Omega &= \ln M! - \left(\ln \Pi_j M_j!\right) \\ &= \ln M! - \sum_j ln\, M_j \end{aligned} . \tag{A1}$$

Replacing **Eq. A2** with **Eq. 8**, we can get

$$\begin{aligned} \ln \Omega &= \ln\left(\tfrac{M}{e}\right)^M \sqrt{2\pi M}\left(1 + \frac{1}{12M} + \frac{1}{288M + \ldots}\right) \\ &-\sum_j \ln\left(\tfrac{M_j}{e}\right)^{M_j} \sqrt{2\pi M_j}\left(1 + \frac{1}{12M_j} + \frac{1}{288M_j + \ldots}\right) \\ &= M \ln M - M + ln\sqrt{2\pi M}\left(1 + \frac{1}{12M} + \frac{1}{288M + \ldots}\right) \\ &-\sum_j M_j \ln M_j - M_j + ln\sqrt{2\pi M_j}\left(1 + \frac{1}{12M_j} + \frac{1}{288M + \ldots}\right) \\ &= M \ln M - M - \sum_j M_j \ln M_j - \sum_j M_j \\ &+ln \frac{\sqrt{2\pi M_j}\left(1 + \frac{1}{12M_j} + \frac{1}{288M_j + \ldots}\right)}{\sum_j ln\sqrt{2\pi M_j}\left(1 + \frac{1}{12M_j} + \frac{1}{288M + \ldots}\right)} \\ &= M \ln M - M - \sum_j M_j \ln M_j - \sum_j M_j. \end{aligned} \tag{A2}$$

frontiers in Physics

ORIGINAL RESEARCH
published: 24 November 2021
doi: 10.3389/fphy.2021.768205

Identifying Influential SLD Authoritative Name Servers on the Internet

Haiyan Xu, Zhaoxin Zhang, Bing Han and Jianen Yan**

School of Cyberspace Science, Faculty of Computing, Harbin Institute of Technology, Harbin, China

DNS plays an important role on the Internet. The addressing of most applications depends on the proper operation of DNS. The root servers and the top-level domain servers are relied upon by many domains on the Internet, and their security affects the whole Internet. As a result, more attention has been paid to the security of servers at these two levels. However, the security of second-level domains and their servers also needs to be brought to the forefront. This paper focuses on showing the complex resolving dependencies and identifying influential name servers for second-level domains. We start by detecting domain name resolution paths and building up a name dependency graph. Then we construct domain name resolution networks of different numbers and sizes, which are connected by a certain number of domain name resolution graphs. On this basis, the network is analyzed from the perspective of complex network analysis, and a multi-indicators node importance evaluation method based on partial order is proposed to identify the influential name servers of the network. Once these name servers are not properly configured and fail or are compromised by DDoS attacks, it will cause resolution failure for a wide range of domain names.

Keywords: DNS, authoritative name servers, name dependency, complex networks, node importance, influential nodes, partial order

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
José-Fernán Martínez-Ortega,
Polytechnic University of Madrid, Spain
Shitala Prasad,
*A*STAR Graduate Academy (A*STAR), Singapore*

****Correspondence:***
Zhaoxin Zhang
heart@hit.edu.cn
Jianen Yan
yanjianen@hit.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *31 August 2021*
Accepted: *25 October 2021*
Published: *24 November 2021*

Citation:
Xu H, Zhang Z, Han B and Yan J (2021) Identifying Influential SLD Authoritative Name Servers on the Internet. Front. Phys. 9:768205. doi: 10.3389/fphy.2021.768205

INTRODUCTION

The domain name system (DNS) provides the service of address resolution for most kinds of Internet applications, which transforms the more easily remembered domain name into the actual identification of hosts on the Internet - IP address, and vice versa. DNS is a globally distributed system. To deal with the scale problem, it deploys a large number of domain name servers, which are organized in a hierarchical structure and distributed all over the world. In general, there are three types of DNS servers in the hierarchy: root DNS server, top-level domain (TLD) DNS server, and authoritative name server below the top-level domain, as shown in **Figure 1**.

There is another kind of local DNS server, which is equivalent to a proxy. It is responsible for receiving domain name resolution requests from clients and forwarding them to DNS servers in the hierarchy. The root and TLD servers do not store the mapping information of specific domain names, and this information is stored in the authoritative name servers below the top level. The local DNS server can only obtain the IP address of the corresponding authoritative name server through DNS communication with the root and TLD server. Therefore, the importance of the root and TLD server is self-evident, and its security is also vital. Although there are many distributed denial of service (DDoS) attacks against these two-tier servers, their security has been concerned by many researchers. At the same time, their management and configuration are in charge of professional

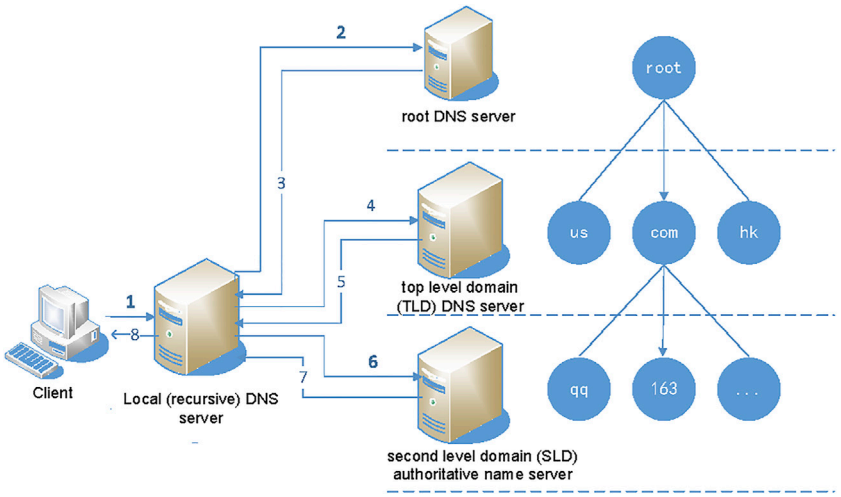


FIGURE 1 | DNS structure and domain name resolution flowchart.

organizations. However, some authoritative name servers at or below the second-level domain (SLD)are self-built, and some are entrusted to the DNS service provider or CDN service providers for trusteeship. There are great differences in operation, maintenance and security capability among service organizations, and the overall service capability is not high.

To ensure the availability of domain name resolution and the correctness of the resolution results, all DNS servers involved in the resolution need to work normally and stably. Even if the services of the root servers and the TLD servers are normal, the authoritative name servers below the TLD will bring risks to the domain name resolution. In recent years, some DDoS attacks have targeted DNS service providers' servers. For example, in October 2019, Amazon's DNS server suffered from a DDoS attack. The attacker sent a huge amount of junk network traffic to the target, resulting in the name resolution service being unable to access, which affected many websites and applications [1]. In October 2016, a DNS service provider encountered a large-scale DDoS attack, which made a large area of websites on the east coast of the United States inaccessible [2].

Moreover, according to the requirements of the DNS protocol specification [3, 4], most SLD administrators configure multiple name servers for domain zone to improve the performance of domain name resolution and distribute the servers in different regions to increase the reliability of domain name resolution. In addition, some large websites delegate resolution services directly to service providers or use services provided by CDNs. All these can make the resolution relationship of a domain very complicated, leading to associations between domains.

Therefore, we detected resolution paths of Alex [5] top one million domain names every month in 2020. According to the monthly survey, 86.18% of domain names involve more than two domains, and some domain name resolution relies on hundreds of name servers. When name resolution spans multiple domains, it will lead to name dependency and make the resolution process more complex. In addition, by analyzing these resolution dependencies, it is found that some SLD authoritative name servers (hereinafter referred to as name servers) provide resolution services for hundreds of domain names. Consequently, to identify influential SLD name servers, a domain name resolution network is built based on real data of a large number of domain name resolution paths data. The main contributions of this paper can be summarized as follows:

- The dependency relationships between domain names and SLD name servers are obtained by probing the resolving paths of domain names, and a name dependency graph model is proposed to express the resolution relationships of a domain name.
- The modeling of the domain name resolution network (DNRN) is proposed, which connects dependency graphs to construct a complex network. The DNRN represents the complex connection of many domain names on the resolution path, which is used to identify influential key domains and SLD name servers from the SLDs of DNS.
- A method for quantitative analysis of network nodes based on workload is proposed, which considers the actual process of workload transfer in the hierarchical resolution chain. The quantized weight of the node can be used as one of the indicators to describe the centrality of the node.
- A multi-indicators node importance evaluation method based on partial order is put forward combined with node workload weight and other classical networks centrality indicators, to identify influential servers from multiple perspectives.

The paper is organized as follows. *Related Work* gives an overview of the related work in this area. *Methods* introduces the methods of identifying influential SLD name servers on the Internet. *Experiments and Analysis* presents experiments and

analysis by using the method described in *Methods*. *Conclusion* concludes our analysis.

RELATED WORK

In the aspect of DNS security, there is some significant research. Yehuda Afek et al. [6] et al. observed that the number of packets involved in a typical name resolution process is much larger than theoretically expected, which is mainly due to the extra resolving of name server IP address. Based on this vulnerability, the corresponding attack (NXNSAttack) was constructed, which can be used to launch DDoS attacks against any victims. This paper also measured the popularity of domains with out-of-bailiwick name servers, showing that most of the top one million popular websites have out-of-bailiwick name servers. As for out-of-bailiwick name servers, V. Rama Subramanian et al. [7] first raised this issue and proposed the concept of name dependency, which showed that a typical name depends on an average of 46 servers, while some names depend on hundreds of servers. In addition, it found that 30% of domain names can be hijacked by two servers, both of which have well-known security vulnerabilities. Casey Deccio et al. [8–10] found that more than half of the queries for a domain name were affected by the namespace beyond the control of the domain name owner. Fujiwara et al. [11] measured the growth of DNS traffic, and the results showed that 60% of DNS traffic was generated by out-of-bailiwick name servers.

In the aspect of DNS vulnerability research, Kröhnke et al. [12] studied the impact on the resolution of some domain names when the routers, name servers, and resolvers in the AS failed or the interconnection between ASs failed, which could identify the bottleneck and single point of failure in the network. Abhishta et al. [13] showed the impact of DDoS attacks against DNS service providers and discovered that the number of domain names specifically using a single DNS service provider is decreasing, and the trend of using multiple providers to disperse risk is on the rise.

The above papers studied that DNS faced many resiliency and security issues. In our previous work [14], the impact of the resolution dependence on the DNS was studied by constructing a name resolution network based on large-scale data from a macro perspective. The similarity between these two articles is that they use the same name dependency graph and the name resolution network model. The difference between these two articles is that this paper pays more attention to the optimization of the centrality algorithm as well as the identification of influential name servers from multiple angles, while the previous article analyzed the DNS vulnerability from the perspective of structure.

The research of complex networks is dedicated to finding macro-statistical characteristics and discovering the relationship between structure and function. Many of these studies focus on the survivability of complex networks. How to mine the key nodes of the network to prevent the network from intentional attacks has attracted the attention of many researchers. Linyuan Lü et al. classified and summarized the method of vital nodes identification in complex networks, as well as pointed out that the criteria of important nodes are diverse [15]. Therefore, it is impossible to find a general index that can best quantify the importance of nodes in each case. Sun Peng et al. [16] propose a community-based k-shell decomposition algorithm adapted to a network with a hierarchically ordered structure. This algorithm is superior to other algorithms on networks with community structures. Dong Zhihao [17] proposes a joint nomination strategy that can discover important nodes without global knowledge. This strategy can effectively identify key nodes only using local information and can be implemented in the real world such as the sudden outbreak of covid-19. Shang Qiuyan [18] proposes an effective distance gravitation model based on information fusion and multi-level processing to identify influential nodes. This method can comprehensively consider the global information and local information of complex networks, and use the effective distance to fuse static and dynamic information.

In addition, research on complex networks based on neural networks are also the focus in recent years. Veličković, et al. [19] proposed a graph attention network by introducing attention mechanisms in the propagation process. The attention mechanism assigns different attention coefficients to different neighbor nodes of a node, so that more important nodes can be found. Shudong Li et al. [20] Proposed a community detection algorithm based on a deep sparse autoencoder. In this paper, the unsupervised deep learning method is used to construct a deep sparse encoder, which can obtain a feature matrix with a stronger ability to express network features, and the algorithm can identify the community structure more accurately.

METHODS

Domain Name Resolution Data Acquisition

A large number of DNS servers are deployed in different parts of the world and organized hierarchically. DNS uses a delegation-based architecture for domain name resolution, in which clients follow a set of rules to resolve domain names through multiple name servers, starting with the root, then the TLD, and then the SLD authoritative name server. Following the delegate, a DNS query requires performing additional name resolution to obtain the address of the intermediate name server, and the resolution of each additional name depends on the delegate chain. In summary, the resolution process that follows the chain of delegation induces complex dependencies between name servers. On the Internet, the resolution of many domain names depends on a large number of name servers, and an extremely complex dependency relationship is formed between these domain names and name servers, which can be represented by a complex network.

We perform a real probe of the resolution paths of one million hotspot domain names to obtain the name servers involved in the resolution path of a domain name and the relationship between them. Specifically, we simulate the actual DNS resolution process without considering caching by sending DNS query packets to DNS servers at all levels and getting the address of the server to be queried next from the response packets until the A record for the domain name is obtained. The domains and servers involved in

the path and the relationship between them are recorded throughout the process. The resolution path data of a domain name is recorded in XML format, as shown in **Figure 2**. It records all the domains and name servers involved in the resolution process, as well as all the dependencies of a domain name. Since we are mainly concerned with the resolution status below the TLD, the root and TLD servers from the path data are removed from the resolution path. In the probe, follow the following rules:

1) Stop detection when cycle dependency occurs. For example, the name server of the domain *A. com* is *ns.B.com*, and the name server of *B. com* is *ns.A.com*. According to the RFC [4], resolving domain *A. com* is needed to submit a DNS query to name server *ns.B.com*, then the DNS server software will first try to get the address of server *ns.B.com* by default. And in the process of resolving server *ns.B.com*, it is needed to query domain *B. com* and server *ns.A.com*, as shown in **Figure 3**. This will produce an endless loop. This is an incorrect way to configure a domain. This problem is discovered in some domains during the process of detecting the domain name resolution data.
2) Stop detection when the domain name servers are self-dependent. For example, the name server of *A. net* is *ns1.A.net*, which does not rely on other domains, so the detection process will stop.

Modeling the Domain Name Resolution Network

Name Dependency Graph Model Based on Resolution Path

To represent the dependencies on the domain name resolution path, a name dependency graph model is built to express the inner logic of domain name resolution. Based on the resolution path data recorded in the above XML file, the main domain name, parent domains, alias, and name servers are extracted as nodes, and the relation between nodes is extracted as edges to form a name dependency graph. A directed edge from a node u to a node v indicates that node v has a dependence on node u. According to the practical meaning of domain name resolution, the edges represent three different types of dependencies:

1) Parent domain dependency: The address mapping information of a subdomain is stored by the name servers of its parent domain, which is the basic specification of the DNS. For example, the domain name *www.abc.com* relies on the correct resolution of its upper level, namely the root domain, the TLD (*com*), and the SLD (*abc.com*). In this way, the domain node *www.abc.com* has a directed edge that points to its SLD node *abc. com* in its name dependency graph. Since this paper focuses on the resolution status below the TLD, the root and TLD servers are removed from the name dependency graph.
2) NS dependency: In the DNS specification [4], the address mapping information for domain names administrated by each domain zone is stored in their authoritative name servers (namely NS). Therefore, the dependency of a domain and its name servers is called NS dependency. Generally, a domain is configured with at least two NS servers, but it is also possible to configure multiple geographically dispersed NS servers, even those managed by other domains. The purpose is to improve resolution reliability, but it also brings the issue of resolution complexity. Nowadays, most popular sites rely on DNS or CDN service providers to support professional authoritative zone administration, which also leads to the phenomenon that some NS servers provide services for a large number of domain names. Once these NS servers fail or are attacked, it will affect resolution failures of a wide range of domain names. In this way, a directed edge from the domain node to its NS server node is formed in its name dependency graph.
3) Alias dependency: if an alias (namely Cname) is set for a domain name, the address of the Cname is needed to continue to resolve. Therefore, a domain name node and its alias can form an alias dependency. In this way, a directed edge from the domain name node to its alias node is formed in its name dependency graph.

```
<root>www.xinhuanet.com.
 <text parent_domain="xinhuanet.com." type="NS">ns3.news.cn.;ns1.cdns.cn.;ns1.news.cn.;ns2.cdns.cn.;ns3.cdns.cn.
  <text domain="news.cn." type="NS">
   <text parent_domain="news.cn." type="NS">ns3.news.cn.;ns1.cdns.cn.;ns1.news.cn.;ns2.cdns.cn.;ns3.cdns.cn.
    <text domain="cdns.cn." type="NS">
     <text parent_domain="cdns.cn." type="NS">ns1.cdns.cn.;ns2.cdns.cn.;ns3.cdns.cn.</text>
    </text>
   </text>
  </text>
  <text domain="cdns.cn." type="NS">ns1.cdns.cn.;ns2.cdns.cn.;ns3.cdns.cn.</text>
 </text>
 <text type="CNAME">www.xinhuanet.com.w.cdngslb.com.
  <text parent_domain="w.cdngslb.com." type="NS">ns2.vip.cdngslb.com.;ns3.vip.cdngslb.com.;ns1.vip.cdngslb.com.
   <text parent_domain="cdngslb.com." type="NS">ns1.cdngslb.com.;ns2.cdngslb.com.  </text>
  </text>
 </text>
</root>
```

FIGURE 2 | A XML format for resolution path data of a domain name.

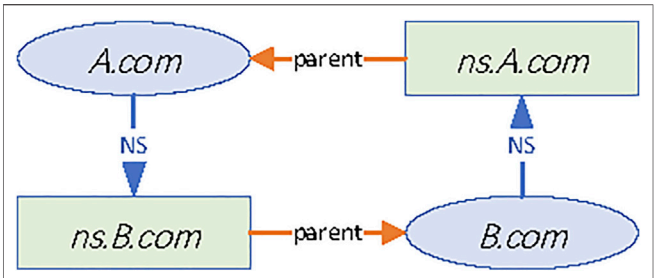


FIGURE 3 | Cycle dependency.

Figure 4 is the name dependency graph of *www.xinhuanet.com*. The ellipse box represents the domain node and the alias node, and the rectangular box represents the server node. The edges between the nodes are directed edges, expressing the above three dependencies. The labels on the edges indicate the type of dependencies.

Moreover, according to whether a NS server is a locally managed server, the NS dependency is further divided into Intra-domain and Inter-domain dependency.

1) Intra-domain dependency: If *v* is a NS server of domain *u*, and is administered by domain *u* itself, then the dependency between *u* and *v* is the Intra-domain dependency, such as the relationship between domain *cdngslb. com* and its server *ns1. cdngslb.com* in **Figure 4**. When the DNS resolver receives this type of resource record (RR), the server's IP address in the additional section of the DNS response packet is used to make the next query.
2) Inter-domain dependency: If *v* is a name server of domain *u*, and is administered by another domain, then the dependency between *u* and *v* is the Inter-domain dependency, such as the relationship between the domain *xinhuanet. com* and its server *ns1. cdns.*cn in **Figure 4**. In this case, there is no IP address of this server in the additional parts of the DNS response packet. So, the DNS resolvers will re-iterate to query the IP address of the out-of-bailiwick server. This leads to cross-domain resolution.

Because of the Inter-domain dependency, the resolution of a domain name involves more domains and more nameservers, which makes the administration and configuration of the SLD complicated.

Construction of Domain Name Resolution Network

Since some NS servers provide services for multiple domains, and the resolution of a domain name is associated with several different domains, a complex network is formed when connecting a certain number of name dependency graphs. This paper refers to this network as DNRN, which contains relationships between domains and name servers. **Figure 5** is an example of a domain name resolution network with 200 nodes, where the node types include domain names and aliases, domains, name servers.

Afterward, by using the statistical characteristics of complex networks, such as degree and aggregation coefficient, this paper analyzes the characteristics and structure of DNS networks and uses node importance analysis methods to identify the key name server nodes. These servers are highly dependent objects, which can affect the resolution of a large number of domain names. These are the focus of protection and should be paid attention to by domain zone administrators and service providers.

Multi-Indicators Node Importance Evaluation Based on Partial Order

In the research of complex networks, it is found that a small number of nodes play a key role [21, 22]. They have an irreplaceable role in network performance, and often determine the structure and function of the network. In the research on the identification of influential nodes of complex networks, the existing centrality algorithms of complex networks can be divided into the following categories according to their properties.

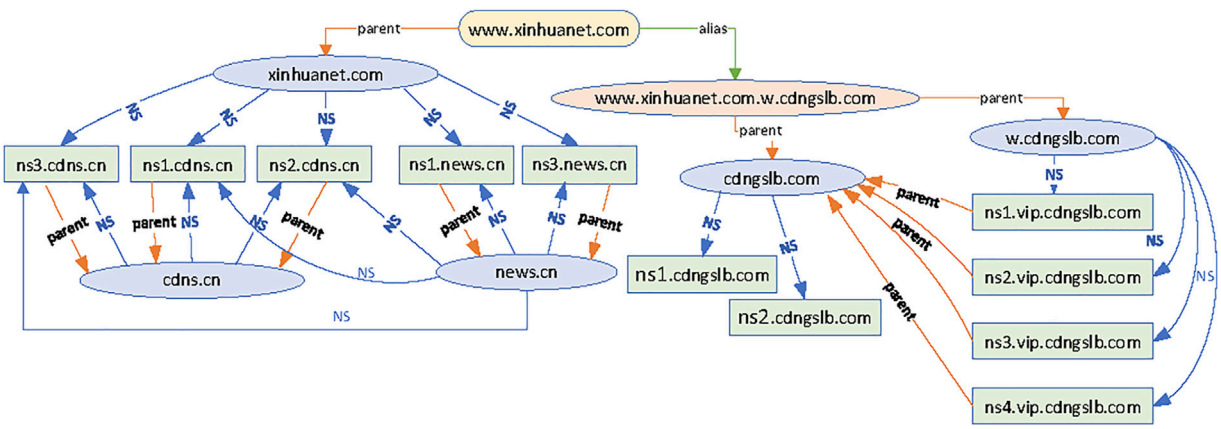


FIGURE 4 | An example of domain name dependency graph.

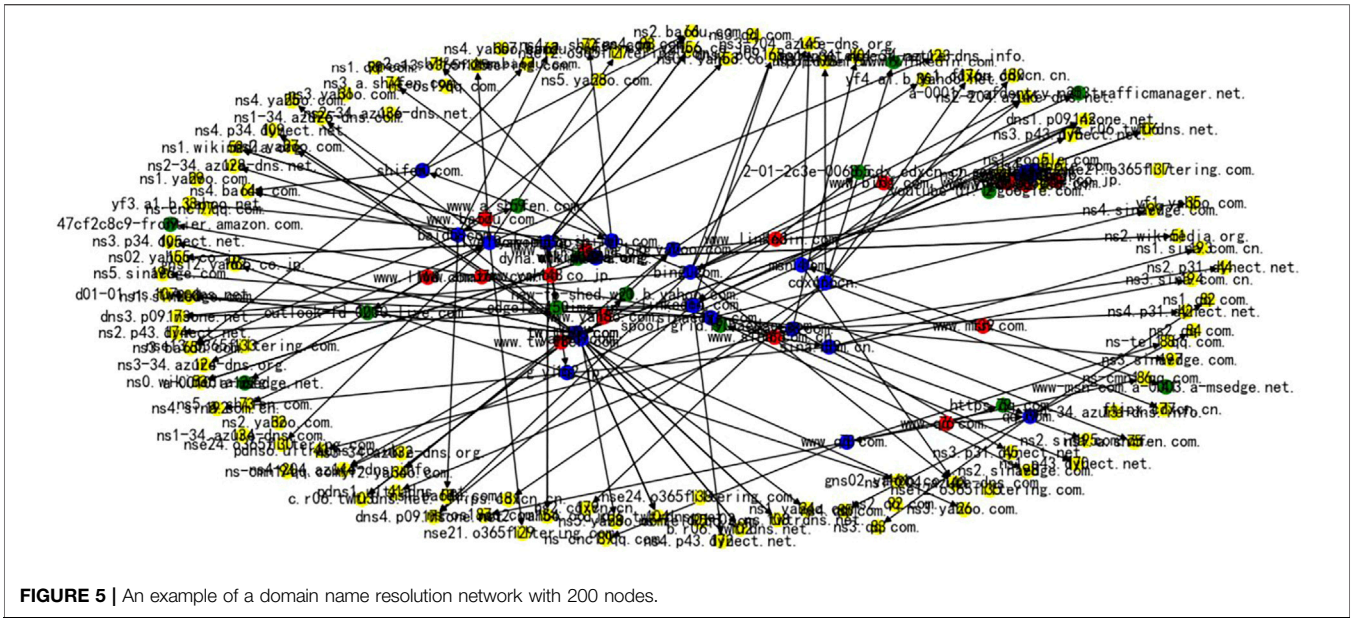


FIGURE 5 | An example of a domain name resolution network with 200 nodes.

Classical Centrality Algorithms for Complex Networks

Neighborhood-Based Method

This method is based on the number of neighboring nodes, e.g., degree centrality.

- In-degree centrality

The DNRN is a directed graph, so the node degree is divided into in-degree and out-degree. The in-degree of a node indicates the dependence of other nodes on it. For a network with N nodes, the maximum possible in-degree of each node is N-1, so the in-degree centrality of the node is obtained by normalizing N-1. For node i, its in-degree is k_i^{in}, then the in-degree centrality DC_i is:

$$DC_i = \frac{k_i^{in}}{N-1} \tag{1}$$

Degree centrality metrics are simple, intuitive, and have low computational complexity. The disadvantage of degree centrality metrics is that it only considers the local information of the node, and does not explore the surrounding environment of the node (e.g., the location of the node in the network, higher-order neighbors, etc.) in more detail.

Distance-Based Method

It is based on the shortest distance associated with a node. Examples are betweenness centrality and closeness centrality.

- Closeness centrality

The average distance d from node i to all other nodes can be obtained by **formula (2)**, where d_{ij} is the shortest distance between node i and the other node j.

$$d_i = \frac{1}{N}\sum_{j=1}^{N} d_{ij} \tag{2}$$

The closeness centrality CC_i of node i is equal to the reciprocal of d_i, so the relative importance of node i in the network can also be expressed by the relative size of closeness centrality. The formula of CC_i is as follows:

$$CC_i = \frac{1}{d_i} = \frac{N}{\sum_{j=1}^{N} d_{ij}} \tag{3}$$

If the shortest distance from a node to other nodes is very small, then the Closeness centrality CC_i of the node is high. This definition is more geometrically consistent with the concept of centrality than Degree centrality because the smallest average shortest distance to other nodes means that the node is geometrically centered in the graph.

- Betweenness centrality

Betweenness centrality stipulates that if a node appears on the shortest path of all node pairs in the network more frequently, then the node is more important. Its calculation formula is shown in **Eq. 4**, where n_{st} is the number of shortest paths between any node pair s and t that pass through node i, *and* g_{st} is the total number of shortest paths between any node pair s and t.

$$BC_i = \sum_{s \neq i \neq t} \frac{n_{st}^i}{g_{st}} \tag{4}$$

Betweenness centrality indicators perform well in the Internet protocol design, network optimization deployment, and network bottleneck detection. It reflects the influence of nodes on network flow, and through betweenness centrality can accurately identify important nodes with very large traffic in the network. The disadvantage of betweenness centrality is that the computational complexity, and especially in the big data environment, makes it restricted in practical applications.

Neighbor Importance-Based (Value Iteration) Method

The importance of a node depends on the importance of its neighboring nodes, such as the Eigenvector centrality and PageRank centrality. The computation of such algorithms can be performed by iterative algorithms.

- Eigenvector centrality

Eigenvector centrality argues that measuring the importance of a node by the number of neighboring nodes (the degree of the node) is too one-sided and should also focus on the importance of the neighboring nodes. Let EC_i be the importance of node i, then we have

$$\boldsymbol{EC_i} = \boldsymbol{c}\sum_{j=1}^{N}\boldsymbol{a_{ij}x_j} \tag{5}$$

where c is a constant, $A = (a_{ij})_{N\times N}$ is the adjacency matrix of the network, and $x = (x_1, x_2, \ldots, x_n)^T$ is the eigenvector corresponding to the eigenvalue of matrix A. The eigenvector indicator focuses on the interaction between nodes and is ideal for analyzing information propagation problems.

- PageRank centrality

PageRank, also known as page ranking, is a page ranking algorithm used by Google in web search. The basic idea is that the importance of a web page lies not only in the number of web pages pointing to it but also in the quality of the web pages pointing to it.

First, initialize the PageRank value of all nodes, and make sure that the sum of the PageRank values of all nodes is 1. Then, an iterative computation is performed to divide the PageRank value of each node equally among the nodes it points to at every step. Let the out-degree of node j be k_j^{out} at step *n-1*, then each node pointed by node j will get a PageRank value $\frac{PR_j(n-1)}{k_j^{out}}$.

Node Location-Based Method

If a network has the characteristics of hierarchical structure, a node in the core layer of the network often has a high influence even if the degree of the node is small.

- Coreness centrality

The coreness of a node can indicate the depth of the node in the core. The concept of coreness is given by Batagelj [21]: given a graph, by removing all the nodes with degrees less than k and their corresponding edge, get a remaining sub-graph, called *k-core*. If a node belongs to *k-core* and does not belong to (*k+1*)-core, then it has a coreness of k. The coreness of a node is a global property based on the location of the network. Its low computational complexity makes it suitable for large-scale networks.

Node Weights Based on Domain Name Resolution Business Attributes

The domain name resolution network is a business network established based on the domain name resolution path. The nodes participating in domain name resolution are connected to the network through the resolution chain. Combining with the actual domain name resolution process, this paper proposes a centrality algorithm to quantify the weights of each node [14]. For a node u, its weight reflects the sum of all other nodes' dependence on u in the resolution process, that is, the node weights here represent the resolution load of the node. The algorithm starts from the domain name nodes and follows the resolution chain to traverse each node in the name dependency graph and calculate its weight. The main idea of the algorithm is as follows:

Set the initial value of all domain name nodes to one and the initial value of the remaining nodes to 0. Node weight calculation is an iterative accumulation process. In *Name dependency graph model based on resolution path*, we define three types of edges. According to different edge types, the accumulative way of weight will be different.

1) Parent domain dependency

When a directed edge points from a node u to its parent domain node, the weight of the parent domain node will accumulate the weight of u. For a parent domain *parent_u*, let u represent a subdomain that depends on *parent_u,* and $Weight_u$ represents the weight of u. Then the weight of the parent domain *parent_u* in the network is

$$\boldsymbol{Weight_{parent\_u}} = \sum_{i\in N}\boldsymbol{Weight_{u_i}} \tag{6}$$

Where N is the number of subdomains that depend on the parent domain *parent_u*.

2) NS dependency

Multiple authoritative name servers can be deployed in a domain, and the domain can be resolved as long as one server is running properly. In this article, the load weight of a server node is equal to the weight of this domain divided by the number of name servers deployed in this domain, assuming that each name server has an equal chance of serving. If the server is set up as the authoritative name server for multiple domains, these values passed by each domain are accumulated as the weight for the server. Therefore, the weight of a name server in the network is

$$\boldsymbol{Weight_{name\_server}} = \sum_{i\in N_{domains}}\left(\frac{\boldsymbol{Weight}_{\boldsymbol{domain\_zone_i}}}{\boldsymbol{N_{server}}}\right) \tag{7}$$

Where *name_server* is the name server node, $domain\_zone_i$ is the domain where *name_server* is deployed, N_{server} is the number of all name servers in the domain $domain\_zone_i$, and $N_{domains}$ are the number of domains where *name_server* is deployed.

3) Alias dependency

If a domain name has a resource record of *CNAME* type, i.e., alias, the alias will inherit its weight. Because the resolution of this domain name has been transferred to the resolution of the alias. There is no cumulative calculation for the weight of aliases, so the weight of an alias node is

$$\mathbf{Weight_{alias}} = \mathbf{Weight_u} \quad (8)$$

Where *alias* represents the *CNAME* of a domain name *u*.

Multi-Indicators Node Importance Evaluation

In addition, there are also some other node centrality methods. However, all the above methods have their limitations. If only one of the metrics is used to evaluate the importance of nodes, appears to be ignored the overall characteristics of the network, resulting in inaccurate evaluation results.

To make the evaluation results more objective and accurate, we use multiple indicators to identify influential nodes. In the multi-indicators evaluation, there may be conflicts among multiple indicators, and appropriate rules should be adopted to resolve this problem. One method is to use the weighting method for determining the node importance [22, 23], assigning weight values to each indicator in advance. But, this method has uncertainties and is a partial subjective approach.

Therefore, we choose a more objective way to sort nodes by using partial order relation [24], which outputs the ranks of the node. The nodes in the first rank are more dominant than other nodes in each indicator. According to such a method, nodes are categorized into different ranks. This ensures that the node evaluation is comprehensive and diverse. It can always guarantee that if a node is better than another node in all indicators, then the importance of this node must be higher than that of another node.

In our multi-indicators evaluation based on the partial order, the set of all node characteristics vectors forms a partial order set based on the dominant relationship and the strict partial order relationship. Each characteristic of the node comes from the classic centrality algorithm for complex networks.

The definitions of partial order and dominance are given as follows.

Definition 1. partial order: Let R be a relation on a set A. If R is self-reflexive, antisymmetric, and transitive, then R is the partial order relation of set A, referred to as partial order, denoted as "≤". For $(a, b) \in R$, it is denoted as $a \leq b$.

For a given node a and b, $k_i\,(i = 1, 2...m)$ are the centrality indicators of the node, and m is the number of indicators. a_{k_i} denotes the value of node a on an indicator k_i.

Definition 2. dominance: Node a dominates node b, denoted as $a<b$, which is a strict partial order on A, and the following conditions shall be met:

a)

$$\forall \mathbf{i}:\ (\mathbf{a_{k_i}}'' \leq ''\mathbf{b_{k_i}})$$

b)

$$\exists \mathbf{i}:\ (\mathbf{a_{k_i}}'' < ''\mathbf{b_{k_i}})$$

Definition 3. dominance set: For a given set A, the priority set (PS) contains all nodes that are not dominated by other nodes, such as

$$\boldsymbol{PS} = \{\boldsymbol{u} | \exists \boldsymbol{a} \in \boldsymbol{A}:\ \boldsymbol{a}'' < ''\boldsymbol{u}\}$$

A node is said to be dominant if each of its metrics is greater than the metrics of other nodes. These dominant nodes are removed from the node-set to be sorted and their node ranks are output. The comparison of the vectors continues until all nodes are output. This process is an iterative process to obtain equivalence classes, and the pseudo-code representation of the partial sorting algorithm is in **Figure 6**. Suppose each node has n attributes illustrated from different perspectives, denoted by vector $f\,(k_1, k_2, \ldots, k_n)$. In each iteration, the nodes that are dominant in each indicator are searched for to obtain the dominant set. Each iteration process outputs the rank and the nodes that belong to this rank.

Figure 7 is a simple example to illustrate the correctness of the algorithm. The input is vectors of seven nodes, and the vector contains four attributes, i.e. k_1, k_2, k_3, k_4. The output is a partial sequence of nodes. In the first iteration of the algorithm, each component of node "*3*" is greater than the other nodes, then "*3*" dominates the remaining nodes. Next, in each round, the dominant node is selected among the remaining nodes, until the remaining nodes (0, 2, 4) do not dominate the other nodes.

Under the partial ordering, there may be multiple nodes at the same rank. This algorithm is adapted to identify nodes that

Multi-indicators evaluation algorithm based on partial order

Input: node set *N* of a *DNRN*,
node importance attribute vector *(k1, k2,... kn)*

Output: Rank, PS /*Output nodes and their ranking*/

```
PS = N;              /*PS is the dominant set, and the initial value is N*/
Remaining = Ø;  /*Remaining stores the selected non-dominant nodes*/
rank = 0;            /*node level, each level may have several nodes*/
while PS != Ø
    for each node ni ∈ PS
        if ∃ b∈PS, such that b(k1, k2, ···, kn) "<" ni (k1, k2,..., kn)
            Remaining = Remaining ∪{ ni };
        Endif
    endfor
    PS = PS - Remaining;
    Rank = Rank+1;
   Output Rank: PS;
   PS = Remaining;   Remaining = Ø ;
End while
```

FIGURE 6 | multi-indicators evaluation algorithm based on partial order.

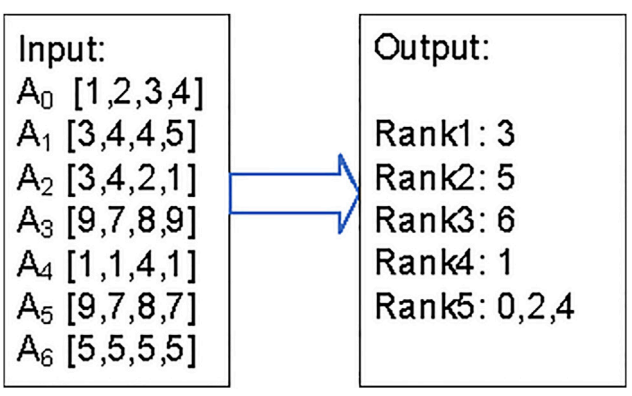


FIGURE 7 | a simple example to illustrate the multi-indicators evaluation algorithm.

perform well in all aspects. When the number of attributes in the vector is large, the number of ranks will be small, implying a higher number of nodes of the same rank, which affects the discrimination and identification of nodes. So, in practice, the selection of attributes is the key issue, and we need to select the relevant attributes according to the actual needs. It must be emphasized that if the selected indicators are different, the order relationship of nodes will be different.

EXPERIMENTS AND ANALYSIS

Test Data Set

We construct a resolution network DNRN for every 100,000 domain names selected in the order of the ranking of one million domain names, and a total of four networks (denoted as *N1, N2, N3, N4* respectively) have been constructed in this way. The domain name sets of these four networks are independent and have no intersection. The goal is to identify influential key nodes from a network. Of course, there is also the option to build a larger network. The number of 100,000 is chosen here as a compromise after considering experimental computing power. Moreover, the four networks are established for verification and comparison.

The DNRN is formed by the correlations among name dependency graphs of 100,000 domain names. In addition, the orphan domain name and small connected components are removed from it, and the maximum connected component is retained. Thus, *N1* to *N4* is a directed connected graph respectively.

The overall characteristics of these four networks, including the number of nodes, the number of edges, the degree-related features, the clustering coefficient, and the number of domain names contained in the network, are shown in **Table 1**. To compare the characteristics of each network, **Figure 8** shows a cross-sectional comparison of the data in **Table 1** for each feature, and it can be seen that the characteristics of the four networks are similar, as shown below:

1) In terms of the size of nodes and edges, *N1* is the largest. The other three networks are similar in scale and smaller than *N1*.
2) From the ratio of *M/N*, the four networks are extremely sparse.
3) The maximum degree of the four networks is above 1,000, but the average degree is around 8.
4) Nodes with the degree of one account for around 26% of the total number of nodes in *N1*, and the other three networks have roughly the same value, around 14%. This statistical value shows the number of edge nodes of the networks and allows us to foresee the overall structure of the network.
5) The average cluster coefficients (ACCs) of the four networks are 0.0016 and 0.026. ACC is used to characterize the robustness and redundancy of the network. If the ACC of a network is higher, the less chance it may be disconnected.

TABLE 1 | Characteristic parameters of *N1~N4*.

notation	*meaning*	*N1*	*N2*	*N3*	*N4*
N	number of nodes	237,848	210,526	210,092	212,385
M	number of edges	433,665	348,131	348,185	352,631
M/N	ratio of edges to nodes	2.206	1.654	1.657	1.660
max{d}	max degree	1,066	1,335	1,299	1,268
<d>	average degree	8.9	8.2	8.1	8.1
Pr [d = 1]	proportion of nodes with a degree of 1	26.07%	14.38%	14.20%	14.78%
<c>	average cluster coefficient	0.0026	0.0017	0.0021	0.0016
<domain names>	number of domain names	81,627	74,613	74,302	75,034

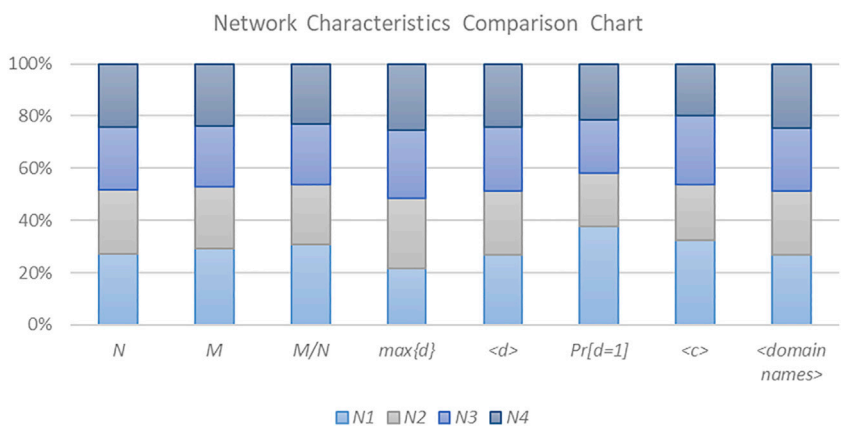


FIGURE 8 | Comparison of characteristic parameters of *N1-N4*.

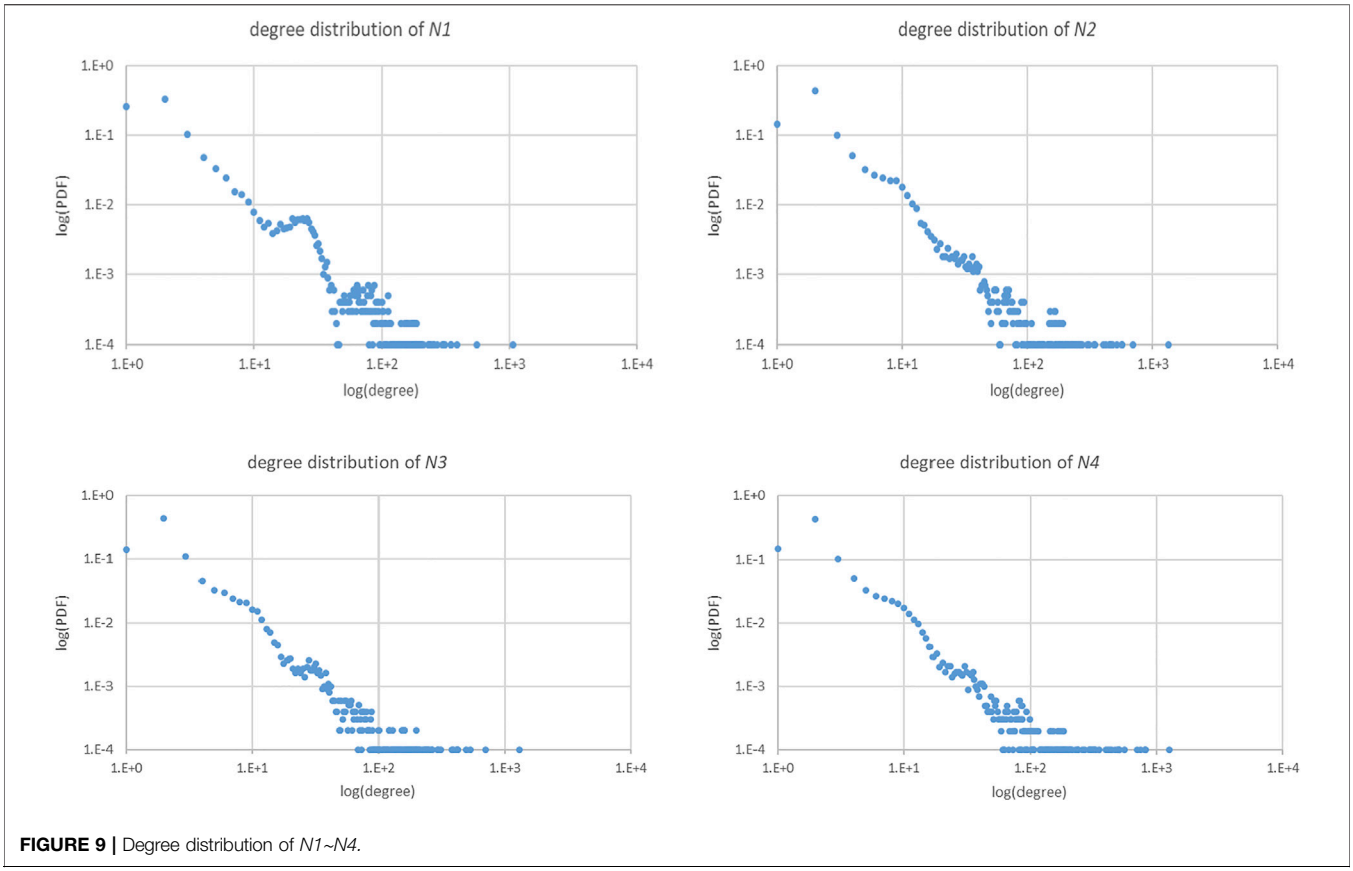


FIGURE 9 | Degree distribution of *N1~N4*.

Judging from the ACCs of the four networks, it also shows that network connections are relatively sparse.

6) *N1~N4* are the largest connected components in the networks formed by name dependency graphs of 100,000 domain names respectively. The number of domain names contained in network *N1* is 81,627, namely, the number of domain names included in the largest connected component is about 81.6% of the number of domain names included in the initial network construction. And *N2~N4* each contains nearly around 75%.

The general characteristics of the networks summarized above show that all four networks are sparse and extremely heterogeneous. In addition, degree distribution $P(k)$ is the ratio of the number of nodes with degree value k to the total number of nodes, which is an important measure to understand the overall structure of the network. The degree distribution of the four DNRNs is shown in **Figure 9**. The abscissa of the distribution graph represents the degree k of the nodes, and the ordinate represents the proportion of the number of nodes with degree k in the total number of nodes. The graph is displayed using double logarithmic coordinates. From the shape of the distribution curve, it shows that the distribution trends of the four networks are similar. The distribution has no peaks and presents an approximate diagonal line. The long-tail feature on the side with a higher k indicates that a few nodes have a large number of connections. In summary, from the distribution graph and its analysis, the DNRN has approximate scale-free network characteristics. For networks with such attributes, the existence of some ultra-high connectivity nodes greatly reduces the robustness of the network.

Identifying Influential SLD Authoritative Name Servers

We identify influential SLD authoritative name servers on the Internet by building DNRNs and the multi-indicator evaluation

TABLE 2 | Four kinds of node centrality attributes and partial order rules.

No	*Node Centrality attributes*	*Categories*	*Partial order rules*
k1	In-degree	Neighborhood-based	If In-degree(a)≥ In-degree(b), then a≤b
k2	Weight	name resolution business attributes	If Weight(a)≥ Weight(b), then a≤b
k3	PageRank	Value Iteration-based	If Eigenvector(a)≥Eigenvector(b), then a≤b
k4	Closeness	Distance-based	If Closeness(a)≥ Closeness(b), then a≤b

TABLE 3 | The partial order results of the four indicators (k1, k2, k3, k4) in *N1*.

Partial_sort_rank (k1, k2, k3, k4)	Number of all nodes	Number of server nodes
0	4	0
1	5	0
2	17	9
3	19	8
4	32	21
5	20	9
6	36	23
7	37	25
8	18	10
9	63	50

algorithm is used to identify influential nodes. Therefore, it is necessary to calculate the characteristics of a node as an indicator vector and input to the algorithm. We comprehensively evaluate the nodes to select the influential nodes in four aspects: node connectivity, node business attributes, node neighbor importance, and network structure centrality. Therefore, we choose the four centrality attributes, and define them into four rules, as shown in **Table 2**. The reason for selecting these four indicators is to comprehensively consider the different angles represented by each indicator and the operating efficiency of the algorithm on large networks. It must be emphasized that if the selected indicators are different, the order relationship of nodes will be different.

We use the above four centrality indicators to sort the network nodes in the partial order. There are 421 levels in the sorted result. **Table 3** shows the number of all nodes and the number of server nodes in the top 10 ranks (There are four different types of nodes in the network, including the main domain name nodes, domain nodes, alias nodes, and server nodes). In the first five ranks, there are 36 server nodes, and **Table 4** shows the detailed information of these influential SLD name servers. They all belong to DNS service providers. From **Table 4**, we can see the classification statistics based on some keywords of server names, divided into six categories, which can be considered as the six companies to which the

TABLE 4 | The influential SLD name servers of *N1* in the first five ranks.

name_key_word	Number	server_name (Pronoun)	In_degree	Weight	PageRank	Closeness
akam/ akamaiedge	21	*A1.server*	30	280.3	1.5919E-04	1.3266E-02
		A2.server	182	825.8	8.9228E-04	1.3031E-02
		A3.server	14	278.2	1.5575E-04	1.3180E-02
		A4.server	164	816.5	8.8081E-04	1.2862E-02
		A5.server	155	279.6	1.7973E-04	1.2726E-02
		A6.server	1	201.7	1.3228E-04	1.3116E-02
		A7.server	1	201.7	1.3228E-04	1.3116E-02
		A8.server	1	201.7	1.3228E-04	1.3116E-02
		A9.server	1	201.7	1.3228E-04	1.3116E-02
		A10.server	1	201.7	1.3228E-04	1.3116E-02
		A11.server	1	201.7	1.3228E-04	1.3116E-02
		A12.server	1	201.7	1.3228E-04	1.3116E-02
		A13.server	1	201.7	1.3228E-04	1.3116E-02
		A14.server	1	201.7	1.3228E-04	1.3116E-02
		A15.server	140	279.4	1.7983E-04	1.2695E-02
		A16.server	1	201.7	1.3228E-04	1.3116E-02
		A17.server	1	201.7	1.3228E-04	1.3116E-02
		A18.server	1	201.7	1.3228E-04	1.3116E-02
		A20.server	1	201.7	1.3228E-04	1.3116E-02
		A21.server	129	810.8	8.6999E-04	1.2744E-02
		A22.server	107	810.8	8.7061E-04	1.2721E-02
Cloudflare	5	*C1.server*	1	0.4	9.7556E-03	4.4182E-02
		C2.server	1	0.4	9.7556E-03	4.4182E-02
		C3.server	1	0.4	9.7556E-03	4.4182E-02
		C4.server	1	0.4	9.7556E-03	4.4182E-02
		C5.server	1	0.4	9.7556E-03	4.4182E-02
Parklogic	3	*P1.server*	1	1,249.5	1.6076E-03	3.0102E-03
		P2.server	1,064	427.1	7.5897E-04	5.9480E-03
		P3.server	1,065	427.6	7.5985E-04	5.9536E-03
Cscdns	3	*CS1.server*	227	1,133.1	1.3240E-03	2.7459E-03
		CS2.server	348	305.0	9.6762E-04	2.3114E-03
		CS3.server	348	305.0	9.6762E-04	2.3114E-03
dnsv2	2	*DV1.server*	1	741.7	1.2498E-03	2.2557E-03
		DV2.server	2	1974.6	1.9538E-03	3.1912E-03
Dnspod	2	*DP1.server*	1,065	581.9	9.5876E-04	5.8287E-03
		DP2.server	1,065	581.9	9.5876E-04	5.8287E-03

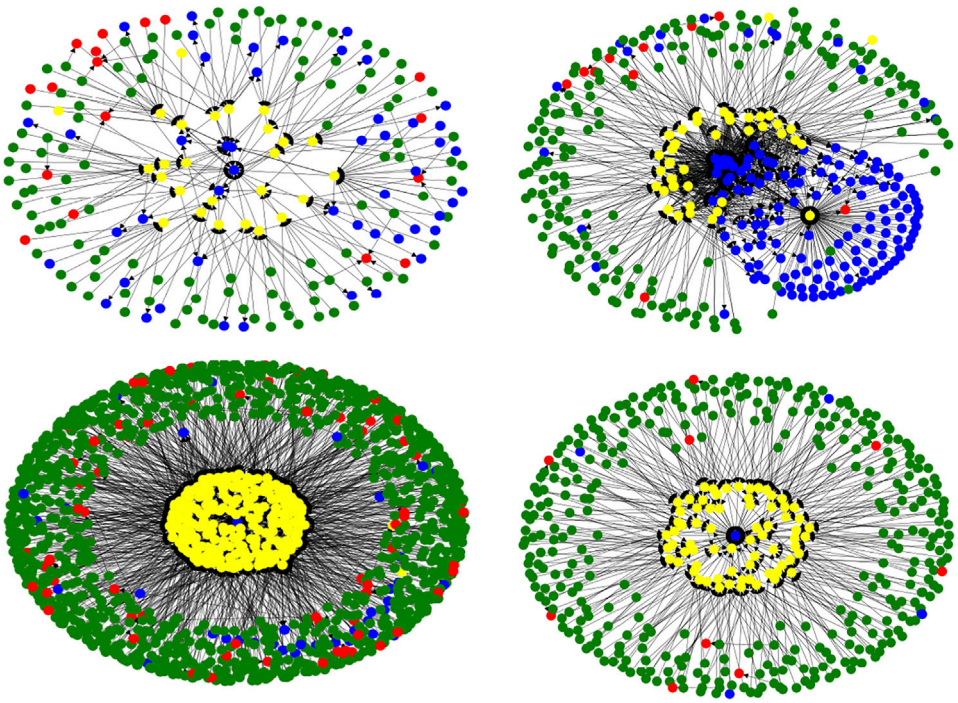


FIGURE 10 | network connections around four key name servers.

TABLE 5 | The partial order results of three sort combinations in *N1*.

Indicators	Number of ranks	Number of all nodes/server nodes in the top rank 3	Number of all nodes/server nodes in the top rank 5	Number of all nodes/server nodes in the top rank 10
k1, k2, k3, k4	421	26/9	77/38	251/155
k1, k2, k3	760	23/7	54/16	153/76
k1, k2	985	3/0	9/1	40/16

server belongs. The real names of the servers are hidden for security and privacy protection purposes. As can be seen from the four index values in the table, the results of the partial order sorting ensure the diversity of the nodes. The importance of such servers is self-evident, and their configuration and administration need to be highly valued. **Figure 10** shows the network connections around four key name servers respectively. For display purposes, some of the similar nodes with the same connections are merged in the figure. Different colors indicate different types of nodes: blue indicates server nodes, red indicates alias nodes, yellow indicates domain nodes, and green indicates primary domain name nodes. We can further develop easier-to-see visualization tools to zoom in on the connections around the core nodes so that administrators can easily check their configuration and security status.

This sort of ranking allows us to see the comprehensive evaluation of each node in the four aspects. It examines the comprehensive characteristics of the node and avoids the one-sidedness of a single indicator, but it also brings the disadvantage of weakening the advantage of the single feature of nodes.

The partial ordering of four indicators may have a coarser granularity, that is, there are multiple nodes at the same level. We can scale down the ranking metrics on this basis, leaving the more concerned node feature metrics to be ranked again. For comparison, several sets of ranking tests are conducted on the *N1* network. The ranking output and the number of top nodes are shown in **Table 5**. As the ranking index decreases, the number of output levels increases.

In practice, users can select multiple indicators that they consider important to sort the nodes in a partial order,

and finally classify the nodes according to their ranks. The nodes ranked in the first few levels dominate other nodes in all indicators. If more indicators are selected, fewer ranks may be output, i.e., the nodes are roughly classified. On this basis, the number of indicators can be reduced and then re-ordered, so that each node can be better distinguished.

CONCLUSION

This paper studies the resolution name dependency of the SLD in DNS and concludes that the security and robustness of the SLD are as important as the root and TLD. There are also influential servers in the SLDs, and they serve many domain names. Failure or compromise of the key servers will affect the normal resolution of a large number of domain names. Based on the actual detected domain name resolution data, this paper constructs domain name resolution networks and uses a multi-indicators node importance evaluation method based on partial order to identify influential servers and domains. This method can combine multiple centrality indicators to evaluate the comprehensive characteristics of nodes. This article has built several networks and combinations of multiple indicators to verify the effectiveness of the method [25, 26].

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

FUNDING

This work is supported by the National Science and Technology Major Project of China (Nos. 2018YFB0804703) and the Natural Science Foundation of Shandong Province of China (Nos. ZR2020KF009).

ACKNOWLEDGMENTS

Thanks to the Network and Information Security Technology Research Center in Harbin Institute of Technology, Weihai for the technical support.

REFERENCES

1. The Register. Bezos DDoS'd: Amazon Web Services' DNS Systems Knackered by Hours-Long Cyber-Attack (2021). Available at: https://www.theregister.com/2019/10/22/aws_dns_ddos/ (Accessed August 20, 2021).
2. Threatpost. Mirai-Fueled IoT Botnet behind DDoS Attacks on DNS Providers (2021) Available at: https://threatpost.com/mirai-fueled-iot-botnet-behind-ddos-attacks-on-dns-providers/121475/ (Accessed August 20, 2021).
3. Mockapetris P. *Domain Names - Concepts and Facilities*. USA: IETF (1987). RFC 1034.
4. Mockapetris P. *Domain Names - Implementation and Specification*. USA: IETF (1987). RFC 1035.
5. Alex (2020). Available at: https://www.alexa.com/topsites (Accessed August 1, 2020).
6. Afek Y, Bremler-Barr A, Shafir L. NXNSAttack: Recursive DNS Inefficiencies and Vulnerabilities. In: Proceedings of the 29th USENIX Security Symposium; Aug 12–14, 2020. ELECTR NETWORK: USENIX Association (2020). p. 631–48.
7. Ramasubramanian V, Sirer EG. Perils of Transitive Trust in the Domain Name System. In: Proceedings of Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement; Oct 19-21, 2005; Berkeley, CA, USA. Berkeley: USENIX Association (2005). p. 379–84. doi:10.1145/1330107.1330152
8. Deccio C, Sedayao J, Mohapatra P. Measuring Availability in the Domain Name System. In: Proceedings of 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies; Mar 15-19,2010; San Diego, CA, USA. NY: IEEE (2010). p. 76–80. doi:10.1109/INFCOM.2010.5462270
9. Deccio C, Sedayao J, Mohapatra P. Quantifying DNS Namespace Influence. *Computer Networks* (2012) 56(2):780–94. doi:10.1016/j.comnet.2011.11.005
10. Deccio C, Chen C-C, Mohapatra P, Sedayao J, Kant K. Quality of Name Resolution in the Domain Name System. In: Proceedings of the 17th IEEE International Conference on Network Protocols; Oct 13-16, 2009; Plainsboro, NJ, USA. NY: IEEE (2009). p. 113–22. doi:10.1109/ICNP.2009.5339693
11. Fujiwara K, Sato A, Yoshida K. DNS Traffic Analysis: Issues of IPv6 and CDN. In: Proceedings of the 12th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)/IEEE Annual Signature Conference on Computer Software and Applications; Jul 16-20, 2012; Izmir, TURKEY. NY: IEEE (2012). p. 129–37. doi:10.1109/SAINT.2012.26
12. Lars K, Jansen J, Vranken H. Resilience of the Domain Name System: A Case Study of the .Nl-Domain. *Comp Networks* (2018) 139-5:136–50. doi:10.1016/j.comnet.2018.04.015
13. Abhishta RVR, Nieuwenhuis LJM. Measuring the Impact of a Successful DDoS Attack on the Customer Behaviour of Managed DNS Service Providers. *Comp Commun Rev* (2018) 48-5:70–6. doi:10.1145/3229598.3229599
14. Xu H, Zhang Z, Yan J, Ma X. Evaluating the Impact of Name Resolution Dependence on the DNS. *Security Commun Networks* (2019) 2019:1–12. doi:10.1155/2019/8565397
15. Lü L, Chen D, Ren X-L, Zhang Q-M, Zhang Y-C, Zhou T. Vital Nodes Identification in Complex Networks. *Phys Rep* (2016) 650:1–63. doi:10.1016/j.physrep.2016.06.007
16. Sun PG, Miao Q, Staab S. Community-based K-Shell Decomposition for Identifying Influential Spreaders. *Pattern Recognition* (2021) 120:108130. doi:10.1016/j.patcog.2021.108130
17. Dong Z, Chen Y, Tricco TS, Li C, Hu T. Hunting for Vital Nodes in Complex Networks Using Local Information. *Sci Rep* (2021) 11(1):11. doi:10.1038/s41598-021-88692-9
18. Shang Q, Deng Y, Cheong KH. Identifying Influential Nodes in Complex Networks: Effective Distance Gravity Model. *Inf Sci* (2021) 577. 162–79. doi:10.1016/j.ins.2021.01.053
19. Veličković P, Cucurull G, Casanova A, Romero A, Pietro L, Bengio Y. *Graph Attention Networks [Preprint]*. arXiv:1710.10903 (2017). Available at: https://arxiv.org/abs/1710.10903v3 (Accessed June 15, 2021).
20. Li S, Jiang L, Wu X, Han W, Zhao D, Wang Z. A Weighted Network Community Detection Algorithm Based on Deep Learning. *Appl Maths Comput* (2021) 401:126012. doi:10.1016/j.amc.2021.126012

21. Yang H, An S. Critical Nodes Identification in Complex Networks. *Symmetry* (2020) 12(1):123. doi:10.3390/sym12010123
22. Crucitti P, Latora V, Marchiori M, Rapisarda A. Error and Attack Tolerance of Complex Networks. *Physica A: Stat Mech its Appl* (2004) 340-1:388–94. doi:10.1016/j.physa.2004.04.031
23. Batagelj V, Zaversnik M. An O(m) Algorithm for Cores Decomposition of Networks. *Comp Sci* (2003) 1(6):34–7. doi:10.1007/BF01074693
24. Hui Y, Liu Z, Li Y. Key Nodes in Complex Networks Identified by Multi-Attribute Decision-Making Method. *Acta Phys Sin* (2013) 62(2):46–54. doi:10.7498/aps.62.020204
25. Tian B, Hu J, Deng Y. Identifying Influential Nodes in Complex Networks Based on AHP. *Physica A: Stat Mech its Appl* (2017) 479:422–36. doi:10.1016/j.physa.2017.02.085
26. Zheng B, Li D, Chen G, Du W, Wang J. *Ranking the Importance of Nodes of Complex Networks by the Equivalence Classes Approach*. arXiv:1211.5484 (2012) Available at: https://arxiv.org/abs/1211.5484 (Accessed June 15, 2021).

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Physics

ORIGINAL RESEARCH
published: 29 November 2021
doi: 10.3389/fphy.2021.768006

Graph Embedding for Scholar Recommendation in Academic Social Networks

Chengzhe Yuan[1], *Yi He*[2], *Ronghua Lin*[2] *and Yong Tang*[2]*

[1]School of Electronics and Information, Guangdong Polytechnic Normal University, Guangzhou, China, [2]School of Computer Science, South China Normal University, Guangzhou, China

The academic social networks (ASNs) play an important role in promoting scientific collaboration and innovation in academic society. Accompanying the tremendous growth of scholarly big data, finding suitable scholars on ASNs for collaboration has become more difficult. Different from friend recommendation in conventional social networks, scholar recommendation in ASNs usually involves different academic entities (e.g., scholars, scientific publications, and status updates) and various relationships (e.g., collaboration relationship between team members, citations, and co-authorships), which forms a complex heterogeneous academic network. Our goal is to recommend potential similar scholars for users in ASNs. In this article, we propose to design a graph embedding-based scholar recommendation system by leveraging academic auxiliary information. First, we construct enhanced ASNs by integrating two types of academic features extracted from scholars' academic information with original network topology. Then, the refined feature representations of the scholars are obtained by a graph embedding framework, which helps the system measure the similarity between scholars based on their representation vectors. Finally, the system generates potential similar scholars for users in ASNs for the final recommendation. We evaluate the effectiveness of our model on five real-world datasets: SCHOLAT, Zhihu, APS, Yelp and Gowalla. The experimental results demonstrate that our model is effective and achieves promising improvements than the other competitive baselines.

Keywords: academic social networks, recommendation system, scholarly data, graph embedding, academic features

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Satyam Mukherjee,
Shiv Nadar University, India
Yun Yang,
Swinburne University of Technology, Australia

***Correspondence:**
Yong Tang
ytang@m.scnu.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *31 August 2021*
Accepted: *18 October 2021*
Published: *29 November 2021*

Citation:
Yuan C, He Y, Lin R and Tang Y (2021) Graph Embedding for Scholar Recommendation in Academic Social Networks. Front. Phys. 9:768006. doi: 10.3389/fphy.2021.768006

1 INTRODUCTION

Recent years have witnessed the fast-growing scholarly big data [1,2]. Against this background, academic social networks (ASNs) systems have aroused widespread attention; these systems provide scholars with an integrated platform to share their academic achievements and interact and collaborate with other scholars [3,4]. As a particular type of social networking, ASNs usually involve different academic entities and relationships. In ASNs, scientific collaboration plays an important role in promoting research and innovation. Scholar recommendation aims to help scholars in ASNs discover potential collaborators by measuring the correlation between scholars. Some research shows that collaboration is more likely to be undertaken between similar scholars [5].

Some studies explore scholarly recommendations in various real-world ASNs. For example, in [6], a user-based experimental approach to find experts on ResearchGate was proposed. Collaborator

recommendation on ScholarMate is based on network topology [7,8] or academic information [9,10]. However, recommending suitable scholars in ASNs is not a trivial task. As pointed out by [2], ASNs are complex heterogeneous networks, which contain multiple types of nodes (e.g., scholars and papers) and links (e.g., citations and co-authorships). Therefore, how to characterize this heterogeneous academic information in scholarly recommendation system becomes very important.

Previous works have tried to integrate various academic information (e.g., different entity and relationship data) in conventional recommendation systems. However, these methods usually lead to biased recommendation results and fail to apply in large-scale networks [11]. Recently, studies [12,13] have proved the success of graph embedding models in heterogeneous network-based recommendation systems, as they are able to learn the latent features of nodes in large-scale networks, which in turn facilitates recommendations. The effectiveness of graph embedding-based models has been applied in various recommendation scenarios, such as movie recommendations [14] and POI recommendations [15].

Different from friend recommendations in conventional social networks, we aim to recommend potential similar scholars to users in heterogeneous academic social networks. In this article, we propose a graph embedding-based scholar recommendation approach in ASNs called GESRec. We break down the scholar recommendation process of our model into three stages. First, we construct the enhanced academic social networks by combining two types of academic features (e.g., attributes features and textual features) from scholar's academic information with user-user relationships, and the correlations between the scholar's academic features are calculated by the similarity of the corresponding embedding vectors. Then, the refined feature representations of the scholars are obtained from the enhanced ASNs by the graph embedding framework. Finally, top-n potential scholars are generated based on the final recommendation. The main contributions of this work can be summarized as follows:

- We study an important yet challenging problem of recommending scholars in heterogeneous academic social networks that pose challenges beyond existing friend recommendation systems.
- We propose a graph embedding-based scholar recommendation system in ASNs by taking advantage of academic auxiliary information. The enhanced ASNs module integrates two types of academic features from scholars' academic information with user-user relationships, and the refined feature representations of the scholars are further obtained by graph embedding framework for scholar recommendation task.
- We conduct extensive experiments on three real-world datasets: SCHOLAT, Zhihu, and APS. The experimental results demonstrate that the proposed GESRec is effective and achieves promising improvements than the other competitive baselines.

The remainder of this article is organized as follows. **Section 2** introduces the related work. **Section 3** provides the problem formulation and components of our model in detail. Then, experimental evaluations and detailed analysis are discussed in **Sections 4**, **5**. Finally, the conclusions and the future work are presented in **Section 6**.

2 RELATED WORK

In this section, we review several existing studies related to our work. They are generally classified into three types: 1) academic social networks, 2) scholar recommendation, and 3) graph embedding-based recommendation.

2.1 Academic Social Networks

In recent years, academic social networks have received widespread attention due to the large volume of scholarly big data, e.g., entities (publications, scholars, etc.) and their relationships (citations, co-authorships, etc.) [2]. ASNs provide various research topics, such as community detection [16,17], knowledge sharing [18,19], and recommendation systems [20,21]. A line of research focuses on making use of ASNs for particular recommendations [21]. A course recommendation model in ASNs was proposed, which combines association rules algorithm with an improved multi-similarity algorithm of multi-source information. By constructing a weighted co-authorship network [22], a method called VOPRec is proposed, which utilizes the node embedding technology to explore text information and network structure information for papers recommendation. Overall, most of these studies focus on analyzing ASNs for the specific type of recommendation services.

2.2 Scholar Recommendation

Scholar recommendation is one of the main tasks in the scholarly recommendation, which can be useful in finding similar scholars for potential collaborations. A related research topic is the expert recommendation, which means trying to detect the most knowledgeable people in some specific topics [23]. For example, in [24], an expert recommendation system based on the Pearson correlation coefficient and the FP-growth association rule was proposed. A multilevel profile-based expert finding method for expert recommendation in online scientific communities is proposed in [25]. Some works focus on constructing expert recommendation systems based on social networks [26–28]. Besides, in [9], a context-aware researcher recommendation system to encourage university-industry collaboration on industrial R&D projects was introduced [6]. Moreover, a pathway-based approach was proposed that takes into consideration both pages and navigations, which aims to identify the right experts with relevant expertise or experience for a given topic.

There are some previous works that are similar to our study. For example, in [29], a research-fields-based graph in ASNs was built, and a community-based scholar recommendation model was proposed [30]. The previous study was extended by processing the problem of subset community through the GraphChi framework in parallel and recommending the

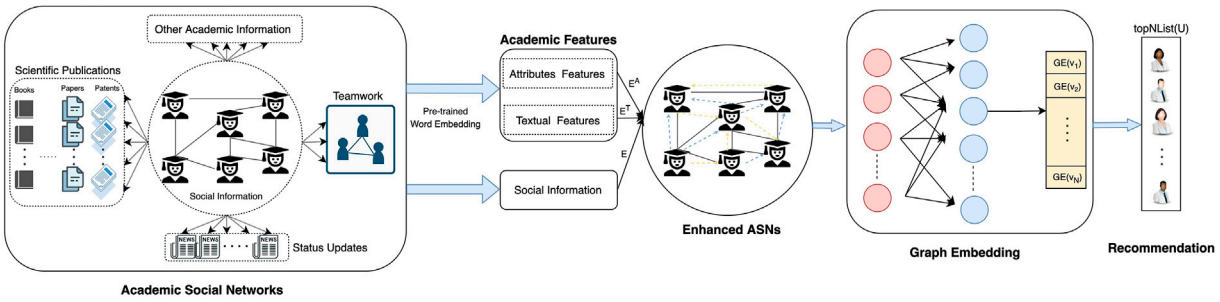


FIGURE 1 | Overview of the GESRec model.

scholars within the community according to the relevant recommendation rules [10]. Moreover, a heterogeneous network-based scholar recommendation approach was designed by integrating researchers' characteristic and relationship information [11]. Recommending potential friends for scholars in ASNs was proposed by considering both network topology and scholars' academic information.

As mentioned above, previous scholar recommendation methods usually fail to fully explore the auxiliary information in ASNs. In this article, we enhance the academic social networks with full auxiliary information for scholar recommendations.

2.3 Graph Embedding-Based Recommendation

Graph embedding techniques have attracted a great deal of attention. A line of research attempts to apply graph embedding in recommendation systems. One of the advantages of graph embedding is the ability to learn the low-dimensional representations of nodes in large-scale networks [10]. Graph embedding has been exploited in heterogeneous information networks for various recommendation scenarios, such as "co-author recommendation," "social recommendation," and "movie recommendation" [13]. A heterogeneous co-occurrence network was constructed to learn the latent representations of users and items. Some works focus on using meta-path-aware similarity between user and item for recommending items [31–33]. In [15], a graph embedding-based model for POI recommendation was proposed, by decomposing the high-order interrelationships among users, POIs and items to a set of pairwise interaction relationships to address data sparsity and cold start problems.

Different from the above methods, our study focuses on scholar recommendation in ASNs and we build enhanced ASNs by considering two types of academic features of scholars (attributes features and textual features), which in turn can be fully explored by graph embedding. Moreover, our work is complementary to the scholar recommendation task as our model can be extended to other graph embedding-based frameworks to recommend scholars in different scenarios.

3 METHODS

In this section, first, we present the formulation of the scholar problem in ASNs. Then, we introduce the details of the proposed model: Graph Embedding for Scholar Recommendation (GESRec) model in **Figure 1**, including enhancing academic social networks with auxiliary information modules and graph embedding for scholar recommendation framework. Finally, we show the overall algorithm and details.

3.1 Problem Formulation

We denote the academic social networks as $G = (V, E, F)$, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ denotes the set of scholars, $E = \{e_{ij} | v_i \in V, v_j \in V\}$, and the edge e_{ij} between nodes v_i and v_j indicates that there are academic connections between these scholars. $f_i = \{x_1, x_2, \ldots, x_k\}$ represents the scholar v_i that contains a set of k academic features. We define the set of academic features shared by all scholars together constituted as $F = \bigcup_{v_i \in V} f_i$. We define the correlation between scholars v_i and v_j as $p(v_i, v_j) = sim(w_i, w_j)$, where w_i denotes the embedding vector of scholar v_i learned by graph embedding technique and $sim(\bullet)$ means an assigned similarity calculation method.

The scholar recommendation in academic social networks problem takes the scholars $U = \{u_1, u_2, \ldots, u_{|U|}\}$ and academic social networks $G = (V, E, F)$ as input and then calculates the correlation $p(u_i, u_j | u_i \in U, u_j \in U, i \neq j)$ by iterations through all scholars in U. It outputs top-n similar scholars list for each scholar, where each list is ranked by the calculated correlation $p(\bullet)$.

3.2 Enhancing Academic Social Networks With Auxiliary Information

Different from other social networks platforms (e.g., Twitter and Facebook), academic social networks usually contain various types of academic data. Some recent researches have pointed out that additional academic data can be used as auxiliary information to extract better academic relationships between scholars and improve scholar recommendation performances [18]. In academic social networks, we divide the academic features into attributes features and textual features. In particular, attribute features include scholar's biographical information, research interests, and collaborative team;

these features can usually be expressed as lists of academic terminology. Textural features include a group of the scholar's academic achievements, such as scientific papers, patents, and research projects, and these features usually contain the title, abstract and other textual information of the academic achievement. In this study, we process the two features in separate ways, and the extracted academic relationships networks E' are added to the original user-user networks E to jointly construct enhanced academic social networks $E^C = E' \cup E$, and $G' = (V, E^C, F)$ is used for subsequent scholar recommendations.

Academic features can be expressed as the intuitive representation of the scholar's academic information, which is represented in the form of terminology vocabulary. To better mining the underlying semantics of these academic features, we utilize a pre-trained word embedding model [34] to get the academic features embedding vectors, which are further processed to calculate the similarity among scholars. The set of academic features $f_i = \{a_1, a_2, \ldots, a_k\}$ means that the scholar u_i contains k academic features, which are divided into attribute features ${f_i}^a$ and text features ${f_i}^t$.

For the attribute features, we get the representation vectors by the pre-trained word embedding model and then calculate the correlation between these vectors by the cosine similarity. Meanwhile, in order to avoid heavy computational burdens, we set a threshold θ_t to filter low correlations, and for reducing the centrality of high-frequency scholars, we also set a top-n list to balance the number of scholars. Besides, as to prevent the effect of differences in the number of scholar's attribute features, we define the correlation between scholar's attribute features as follows:

$$attrSim(u_i, u_j) = \max\left(cosSim\left(emb_{x_m}, emb_{x_n}\right) \mid x_m \in {f_i}^a, x_n \in {f_j}^a\right) \tag{1}$$

$${e_{ij}}^a = \begin{cases} attrSim(u_i, u_j), if\ attrSim(u_i, u_j) \geq \theta_a\ and\ attrSim(u_i, u_j)\ in\ topk_a \\ 0, else \end{cases} \tag{2}$$

where ${f_i}^a$ and ${f_j}^a$ denote the set of attributes features for scholars u_i and u_j, emb_{x_m} is the embedding vector representation of attribute feature x_m obtained by the pre-trained word vector model. By computing the similarity values of the embedding vectors between two sets of attribute features (${f_i}^a$ and ${f_j}^a$), we take the maximum value as the similarity between attribute features *attrSim* (u_i, u_j). If the value of *attrSim* (u_i, u_j) is greater than the threshold θ_a, and in the top-k_a list of similarity between attribute features, we add a new edge e_{ij}^a between scholar's attribute features correlation.

For the textual features, we process them by cleaning, tokenization, and stemming and combine all the processed text features. Similar to the attribute features, we also get the correlation between textual features by the cosine similarity of the corresponding representation vectors. We define the correlation between scholar's textual features as follows:

$$textSim(u_i, u_j) = cosSim\left(emb_{t_i}, emb_{t_j}\right), t_i = concat\left({f_i}^t\right) \tag{3}$$

$${e_{ij}}^t = \begin{cases} textSim(u_i, u_j), if\ textSim(u_i, u_j) \geq \theta_t\ and\ textSim(u_i, u_j)\ in\ topk_t \\ 0, else \end{cases} \tag{4}$$

where ${f_i}^t$ and ${f_j}^t$ denote the set of textual features for scholars u_i and u_j. By combining all the text features f_i^t of the scholar u_i, we get a document-level textual features t_i. *textSim* (u_i, u_j) means the cosine similarity between attribute features. We also use thresholds θ_t and k_t to determine whether to add an edge e_{ij}^t between scholar's textual features correlation.

By combining the multiple academic relationships between scholars with user-user connection, we define the enhanced academic social networks as follows:

$$E^C = E \cup E^A \cup E^T \tag{5}$$

$$G' = (V, E^C, W) \tag{6}$$

where $E^A = \{e_{ij}^a | u_i \in U, u_j \in U\}$ represents the set of scholar's attribute features correlations, E^T denotes the set scholar's textual features correlations, and G' is a directed weighted academic social networks.

3.3 Graph Embedding for Scholar Recommendation

In order to utilize multiple academic relationships for better scholar recommendation performance, we try to adopt the graph embedding method to learn the latent feature representations of the scholars for scholar recommendation. As we get enhanced academic social networks $G' = (V, E^C, F)$ from previous section, we aim to learn a low-dimensional embedding $\mathbf{v} \in \mathbb{R}^d$ for each node $v \in V$, where $d \ll |V|$ is the dimension of the representation space. The learned embeddings are considered to contain informative academic characteristics, which are useful in the scholar recommendation task. The whole algorithm framework is shown in Algorithm 1. First, we obtain two types of scholar's academic features (attributes features ${f_i}^a$ and textual features ${f_i}^t$). Second, we calculate the corresponding correlation between scholar's attribute features *attrSim* (u_i, u_j) and textual features *textSim* (u_i, u_j). Third, we construct enhanced ASNs by combining the multiple academic relationships between scholars with user-user connections in **Eqs 5**, **6**. Then, we get the refined feature representations $\mathbf{v}_i$ of the scholars by the graph embedding framework. Finally, the top-n list of (*sorSimList(U)*) similar scholars is recommended to the user u_i as the potential recommendation list.

For each node $v \in V$, we construct the neighborhood $\mathcal{N}_v$ by the skip-gram with negative sampling strategy. We formulate the overall objective of representation learning in our network as follows:

$$\max_f \sum_{v \in V} \log \Pr\left(\mathcal{N}_v \mid f(v)\right) \tag{7}$$

where $f: V \rightarrow \mathbb{R}^d$ is a mapping function from node to d-dimensional representation space and $\mathcal{N}_v \subset V$ is the neighborhood of node v. Following LINE [35], we define $\Pr\left(\mathcal{N}_v \mid f(v)\right)$ by the softmax function as follows.

Algorithm 1. Graph embedding for scholar recommendation model.

Input : the scholars U; the academic social networks $G=(V,E,A)$; the attribute features threshold θ_a, K_a; the textual features threshold θ_t, K_t
Output : $topNList(u_i)=u_{i_1},u_{i_2},\ldots,u_{i_N}, u_i \in U$

for $u_i \in U$ **do**
  $f_i=\{a_1,a_2,\ldots,a_k|a_i \in A\} \leftarrow u_i$;
  $\{f_i^a, f_i^t\} \leftarrow f_i$;
  for $x_m \in f_i^a$ **do**
    $emb_{x_m} \leftarrow x_m$;
  end
  for $x_n \in f_i^t$ **do**
    $emb_{x_n} \leftarrow x_n$;
  end
end
for $u_i \in U$ **do**
  for $u_j \in U$ **do**
    $attrSim(u_i,u_j) \leftarrow \langle u_i,u_j \rangle$;
    if $attrSim(u_i,u_j) \geq \theta_a$ & $attrSim(u_i,u_j)$ *in* $topK_a$ **then**
      $e_{ij}{}^a \leftarrow \langle v_i,v_j \rangle$, $w_{ij}{}^a = attrSim(u_i,u_j)$;
    end
    $textSim(u_i,u_j) \leftarrow \langle u_i,u_j \rangle$;
    if $textSim(u_i,u_j) \geq \theta_t$ & $textSim(u_i,u_j)$ *in* $topK_t$ **then**
      $e_{ij}{}^t \leftarrow \langle v_i,v_j \rangle$, $w_{ij}{}^t = textSim(u_i,u_j)$;
    end
  end
end
$E^C=\{E \cup E^A \cup E^T\}$;
$w_{ij}{}^c = w_{ij}+w_{ij}{}^a+w_{ij}{}^t$;
$G'=\{V,E^C,W\}$;
for $v_i \in V$ **do**
  $\mathbf{v}_i = GE(v_i)$; // Graph embedding for vector representation of v_i
  $\mathbf{v}_i \leftarrow u_i$;
end
for $u_i \in U$ **do**
  for $u_j \in U$ **do**
    $Sim(u_i,u_j)=Sim(vec_i,vec_j)$;
  end
  $sortSimList(u_i) \leftarrow Sim(u_i,u_j|u_j \in U, i \neq j)$;
end
return $topN(sortSimList(U))$;

$$\Pr(\mathcal{N}_v \mid f(v)) = \frac{exp(f(\mathcal{N}_v) \cdot f(v))}{\sum_{c \in V} exp(f(c) \cdot f(v))} \tag{8}$$

Considering the heavy cost of computing when the number of nodes in the network is relatively large, we employ negative sampling [36] to maximize the log probability of the softmax as follows:

$$\log \sigma(f(\mathcal{N}_v) \cdot f(v)) + \sum_{m=1}^{M} \mathbb{E}_{v_m \sim P(v)} [\log \sigma(-f(v_m) \cdot f(v))] \tag{9}$$

where M is the number of negative samples and $\sigma(\bullet)$ denotes the sigmoid function. $P(V)$ is the noise distribution of nodes. We apply stochastic gradient descent (SGD) to optimize this objective **(9)**.

Based on the graph embedding module, we get the low-dimensional embedding of all scholars U. We calculate the similarity between two scholars based on the cosine similarity of these two low-dimensional embedding vectors as follows:

$$\mathrm{Sim}(u_i,u_j) = \frac{\mathbf{v}_i \times \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|} \tag{10}$$

Finally, the top-n similar scholars are recommended to the selected scholar as the potential recommendation list.

4 EXPERIMENT SETUP

In this section, we first describe the statistics of five real-world datasets. Then, we describe the evaluation metrics used for experimental results. Finally, we conduct detailed experiments and a case study to demonstrate the effectiveness of the proposed model. The following subsections introduce the design of the experiment setup.

4.1 Datasets

We conduct experiments on five widely used real-world datasets: SCHOLAT, Zhihu, APS, Yelp, and Gowalla. These datasets are publicly accessible, real-world data with various sizes, sparsity and domains (e.g., academic social networks domain: SCHOLAT, Zhihu, and APS; popular social networks domain: Yelp and Gowalla). The statistics of the datasets are listed in **Table 1**.

- SCHOLAT[1] is an emerging vertical academic social networking system designed and built specifically for researchers in China. The main goal of SCHOLAT is to enhance collaboration and social interactions focused on scholarly and learning discourses among the community of scholars. In addition to social networking capabilities, SCHOLAT also incorporates various modules to encourage collaborative and interactive discussions, for example, chat, email, events, and news posts. We constructed the SCHOLAT dataset based on [37]. More specifically, first, we only select scholars with more than ten friends to ensure that the relationship network is not too sparse. Then, we clean and desensitize these data. Finally, we collected 19,841 scholars and 63,686 friends relations between them. We also extracted 3,309 academic attributes and 67,462 textual features for the scholars.
- Zhihu[2] is China's most popular Q&A platform. Users ask questions, express their opinions on different issues, and share, exchange, and discuss knowledge. We use the same version of the Zhihu dataset from [38], which contains 10,000 active users and 43,894 friends relations between them, and the context describing the user's topic of interest is used as the textual features. Since Zhihu is mainly a platform for users to ask and answer questions, users do not usually have academic attributes features.
- APS[3] (American Physical Society, APS) is a non-profit academic organization, which aims to promote the development of research in physics through academic journals, scientific conferences, and exhibitions. The APS database contains over 600,000 papers from 18 core physics journals, including paper metadata and citations. We select PRA (Physical Review A: Atomic, Molecular, and Optical Physics) journals to construct the APS dataset. First, we process the corresponding JSON metadata to extract co-author relationships between the authors as friend relationships. Then, we extract the authors' academic attributes by the glove.6B.200d[4] pre-trained word vector, which is obtained from the English corpus using the GloVe model [39]. In order to ensure that the relationship network is not too sparse, we also only select scholars with more than

[1]https://www.scholat.com/research/opendata/
[2]https://www.zhihu.com/
[3]https://www.aps.org/index.cfm
[4]https://www.kaggle.com/incorpes/glove6b200d

TABLE 1 | Statistics of the datasets.

	SCHOLAT	Zhihu	APS	—	Yelp	Gowalla
#Users	19,841	10,000	14,279	#Users	45,919	29,858
#Relationships	63,686	43,894	446,685	#Items	45,538	40,981
#Attributes features	3,309	\	6,221	#Interactions	1,185,065	1,027,370
#Textual features	67,462	10,000	81,088	Density	0.056%	0.084%

ten friends. Overall, we collect 14,279 scholars and 446,685 co-authorship relations between them. We also extract 6,221 academic attributes and 81,088 textual features for the scholars.

- Yelp[5] is a popular online directory for local business reviewing and social networking sites. The Yelp dataset is a subset of Yelp's businesses, reviews, and user data. This dataset consists of 16,239 users, 14,282 businesses, and 198,397 ratings ranging from 1 to 5. In addition, this dataset contains social relations and attribute information of businesses.
- Gowalla[6] is a location-based social networking website where users share their locations by checking in. Same as in [40], this dataset consists of 29,858 users, 40,981 items, and 1,027,370 interaction.

For each dataset, we train the academic attributes by pre-trained word vector models to obtain the corresponding embedding vectors. We pre-process the textual features by tokenization, lower-casing, and stemming, and then we transform the pre-processed textual features into the corresponding feature vectors by the pre-trained word vector models. We randomly split all the datasets into training, validation, and testing sets by the partition 80%:10%:10%, respectively.

4.2 Evaluation Metrics

In this researcher recommendation task, given a scholar, a practical recommendation model generates a top-n ranked list of researchers. We evaluate the performance of our model and other baseline models by four evaluation metrics: Precision@N, Recall@N, F1 score, and NDCG@N. The first three evaluation metrics are formulated as follows:

$$\textbf{Precision@N} = \frac{1}{m}\sum_{i=1}^{m}\frac{R_a \cap T_a}{R_a} \tag{11}$$

$$\textbf{Recall@N} = \frac{1}{m}\sum_{i=1}^{m}\frac{R_a \cap T_a}{T_a} \tag{12}$$

$$\textbf{F1@N} = \frac{2{*}Precision@N{*}Recall@N}{Precision@N + Recall@N} \tag{13}$$

where m, R_a, and T_a represent the number of scholars, the predicted list of items, and the ground-truth items associated with scholar i, respectively. F1 score is the harmonic mean of Precision and Recall. Besides, we also adopt NDCG@N (Normalized Discounted Cumulative Gain) as the evaluation metric to judge the quality of the top-n ranking list.

4.3 Comparison Methods

To verify the effectiveness of our proposed model, we employ the following methods as baselines:

- ItemPop: ItemPop is a simple method that recommends the most popular items. Items are ranked based on the observed frequency of each aspect in this item's historical reviews. All scholars are recommended with the same top-n item lists.
- ALS [41]: ALS (Alternating Least Squares) is a matrix factorization model that attempts to estimate the user-item rating matrix as the product of two lower-rank matrices. ALS minimizes two loss functions alternatively.
- BPR [42]: BPR (Bayesian Personalized Ranking) is a matrix factorization-based model that utilizes pairwise ranking loss, which is tailored to learn from implicit feedback.
- DeepWalk [43]: DeepWalk is a word2vec based network embedding model, which learns latent representations by predicting the local neighborhood of nodes sampled from random walks on the graph.
- HERec [31]: HERec is a heterogeneous information network embedding-based approach that utilizes auxiliary information for recommendation. This model designs a random walk strategy to filter the node sequence for network embedding.
- Multi-GCCF [40]: Multi-GCCF is a graph convolution-based recommendation framework, which employs a multi-graph encoding layer to integrate the information provided by the user-item, user-user, and item-item graphs.

GESRec_NW, GESRec_A, and GESRec_T are three variants of our proposed model GESRec. To verify the importance of academic relationships between scholars, GESRec_NW treats all relationships between scholars equally important. GESRec_A and GESRec_T only use attribute information and textual feature, respectively, as supplementary information to extend the interaction between scholars.

5 RESULTS AND DISCUSSIONS

In this section, we report and analyze the results of our experiments with Precision, Recall NDCG, and F1 metrics on five real-world datasets in various domains. In particular, we first

[5]https://www.yelp.com/dataset/download

[6]https://snap.stanford.edu/data/loc-gowalla.html

TABLE 2 | Experimental results on SCHOALT dataset with **Precision@N** and **Recall@N** metrics.

Metrics	N	ItemPop	ALS	BPR	DeepWalk	HERec	Multi-GCCF	GESRec_NW	GESRec_A	GESRec_T	GESRec
Precision	10	0.055	0.006	0.135	0.142	0.188	0.190	0.076	0.077	0.087	**0.192**
	20	0.046	0.007	0.115	0.120	0.140	0.158	0.060	0.059	0.066	**0.162**
	30	0.041	0.006	0.096	0.105	0.125	0.135	0.048	0.048	0.053	**0.143**
	40	0.039	0.006	0.062	0.087	0.096	0.105	0.041	0.041	0.044	**0.114**
	50	0.036	0.006	0.053	0.065	0.067	0.073	0.035	0.035	0.038	**0.088**
Recall	10	0.102	0.005	0.226	0.264	0.289	0.307	0.125	0.129	0.138	**0.318**
	20	0.145	0.013	0.297	0.325	0.364	0.374	0.185	0.188	0.198	**0.407**
	30	0.213	0.018	0.331	0.394	0.433	0.440	0.214	0.220	0.226	**0.442**
	40	0.240	0.023	0.363	0.426	0.465	0.47	0.233	0.241	0.242	**0.487**
	50	0.289	0.024	0.409	0.462	0.483	0.495	0.245	0.253	0.252	**0.501**

The best performance in each case is highlighted.

TABLE 3 | Experimental results on Zhihu dataset with Precision@N and Recall@N metrics.

Metrics	N	ItemPop	ALS	BPR	DeepWalk	HERec	Multi-GCCF	GESRec_NW	GESRec_A	GESRec_T	GESRec
Precision	10	0.024	0.001	0.038	0.056	0.068	0.096	0.085	—	-	**0.106**
	20	0.020	0.001	0.029	0.049	0.057	0.084	0.073	—	—	**0.089**
	30	0.018	0.001	0.020	0.041	0.051	0.076	0.065	—	—	**0.082**
	40	0.015	0.001	0.018	0.033	0.042	**0.072**	0.051	—	—	0.071
	50	0.011	0.001	0.016	0.029	0.036	**0.065**	0.042	—	—	0.063
Recall	10	0.130	0.003	0.159	0.213	0.296	0.347	0.245	—	—	**0.365**
	20	0.203	0.014	0.201	0.266	0.324	0.378	0.283	—	—	**0.398**
	30	0.244	0.017	0.247	0.315	0.372	0.411	0.311	—	—	**0.423**
	40	0.263	0.022	0.278	0.378	0.402	**0.462**	0.338	—	—	0.454
	50	0.271	0.027	0.286	0.401	0.427	**0.474**	0.347	—	—	0.469

The best performance in each case is highlighted.

TABLE 4 | Experimental results on APS dataset with **Precision@N** and **Recall@N** metrics.

Metrics	N	ItemPop	ALS	BPR	DeepWalk	HERec	Multi-GCCF	GESRec_NW	GESRec_A	GESRec_T	GESRec
Precision	10	0.136	0.002	0.327	0.333	0.341	0.356	0.326	0.316	0.340	**0.362**
	20	0.088	0.002	0.242	0.246	0.243	0.251	0.224	0.217	0.232	**0.257**
	30	0.070	0.002	0.193	0.182	0.185	0.189	0.170	0.165	0.175	**0.201**
	40	0.055	0.002	0.151	0.145	0.154	0.156	0.137	0.132	0.141	**0.168**
	50	0.042	0.002	0.133	0.130	0.133	0.133	0.114	0.111	0.117	**0.135**
Recall	10	0.184	0.002	0.446	0.451	0.466	0.494	0.456	0.415	0.484	**0.520**
	20	0.221	0.003	0.588	0.602	0.618	0.622	0.578	0.531	0.605	**0.630**
	30	0.248	0.005	0.612	0.644	0.664	0.688	0.629	0.582	0.656	**0.699**
	40	0.262	0.007	0.649	0.676	0.687	0.705	0.657	0.611	0.682	**0.726**
	50	0.267	0.009	0.665	0.698	0.711	0.720	0.673	0.629	0.699	**0.733**

The best performance in each case is highlighted.

provide detailed performance of our proposed model with six different baselines (for Yelp and Gowalla datasets, we evaluate our experiments with the same setting in this work [40]). Then, we analyze the impact of different factors of our model on scholar recommendation in academic social networks. Finally, we provide a visualization of scholars embedding vectors on the SCHOLAT dataset.

5.1 Performance Comparison

Tables 2–5 report the results of our proposed model and baselines across SCHOLAT, Zhihu, APS, Yelp, and Gowalla datasets. We adopt Precision@N, Recall@N, and NDCG@N as the evaluation metrics for these datasets. As can be seen, in academic social networks dataetsts, our model outperforms all of the baseline models on the Precision and Recall metrics on SCHOLAT and APS datasets and has comparable performance with the Multi-GCCF model [40] on the Zhihu dataset. In other popular social networks datasets. Our model has the best performance on the Recall and NDCG metrics on Yelp dataset and has comparable performance with the best baseline MultiGCCF model on the Gowalla dataset.

For the SCHOLAT dataset, in **Table 2**, it is clear that our GESRec model constantly outperforms all baselines with $N \in \{10, 20, 30, 40, 50\}$ in Precision and Recall metrics,

TABLE 5 | Experimental results on Yelp and Gowalla datasets with Recall@20 and NDCG@20 metrics.

	Yelp		Gowalla	
	Recall@20	**NDCG@20**	**Recall@20**	**NDCG@20**
ItemPop	0.0086	0.0117	0.0362	0.1081
ALS	0.0004	0.0007	0.0006	0.0008
BPR	0.0494	0.0662	0.1291	0.1878
DeepWalk	0.0534	0.0712	0.1395	0.1960
HERec	0.0599	0.0732	0.1547	0.2037
Multi-GCCF	0.0667	0.0810	**0.1595**	**0.2126**
GESRec	**0.0669**	**0.0816**	0.1587	0.2119

The best performance in each case is highlighted.

demonstrating the effectiveness of our model when applied to scholar recommendation task. According to the results, GESRec improves over the best baseline Multi-GCCF method by 0.2–1.5% in terms of Precision and 0.2–3.3% in terms of Recall. This suggests that by exploiting the auxiliary academic information with multi-relation graphs, our model can obtain higher performance in recommending scholars in ASNs. We notice that the GESRec model has better performance than other competitive deep graph embedding-based models (i.e., Multi-GCCF, HERec, and DeepWalk), which indicates that by fusing scholar's attributes features and textual features with user-user relationships, the GESRec model is able to have better academic learning representation vectors of the scholars. Besides, we also observe that deep graph embedding-based models are consistently superior to the traditional top-n recommendation methods (ItemPop, ALS, and BPR). One of the reasons might be that the user-user relations in the SCHOLAT are relatively too sparse, so they fail to obtain scholars' academic features for recommendations. Meanwhile, significant improvements are also observed over three variants models: GESRec_NW, GESRec_A, and GESRec_T. This demonstrates the effectiveness of utilizing multi-relation networks in graph embedding models. We further observe that the performance of GESRec_T is slightly better than GESRec_NW and GESRec_A. The possible reason is that academic textual features are more suitable than attribute features and user-user connections in expressing friendship preferences among scholars.

For the Zhihu dataset, in **Table 3**, we can notice that our model achieves the best performance in Precision and Recall metrics with $N \in \{10, 20, 30\}$ and is comparable to the Multi-GCCF model with $N \in \{40, 50\}$, which is one of the best baseline models in the SCHOLAT dataset. Since the Zhihu dataset has no attribute features, we can only use a small amount of textual information as auxiliary information to reconstruct the original user-user networks, and only one variant GESRec_NW is applied in this dataset. Compared to the best baseline Multi-GCCF, our model has slightly worse performance with $N \in \{40, 50\}$. A possible reason is that Zhihu is not a true ASN, and the motivation for users to establish relationships in common social networks might be different from that for academic social networks (i.e., scholars tend to follow other scholars who have similar academic features).

For the APS dataset, we observe similar results of our model in **Table 4**, which are similar to those of the SCHOLAT dataset. Although Multi-GCCF is the strongest baseline on all datasets, our model outperforms Multi-GCCF by up to 1.2 and 2.6% in terms of Precision and Recall. ALS performs the worst among all the models, as ALS is a matrix factorization-based model, but the user-user relations matrix is generally too sparse. We also notice that the overall experimental results on APS are better than Zhihu and SCHOLAT, which prove the validity of improving the performance of scholar recommendations by leveraging the academic features and relationships among scholars in ASNs. Due to the similarity of the SCHOLAT and APS datasets, we do not provide qualitative analysis in this experiment.

To further investigate the effectiveness of our proposed model, we also investigate the performance for all the models in other popular social networks domain datasets: Yelp and Gowalla. In **Table 5**, we find similar results as we observed in academic social networks datasets. ItemPop and ALS models give the worst performance on all datasets since these traditional recommendation methods fail to consider auxiliary information (i.e., the information of user-item interactions). We further notice that BPR outperforms ItemPop and ALS models because BPR is able to learn individual users' preference information. However, ItemPop is a simple model and ALS fails to deal with the sparse users-items interaction matrix.

We observe that the deep graph embedding-based models (GESRec, Multi-GCCF, HERec, and DeepWalk) consistently outperform the general methods (ItemPop, ALS, and BPR), which indicates the advantages of deep graph embedding-based model in processing graph structure. Compared to the best baseline, Multi-GCCF, our model achieves better performance in the Yelp dataset but has slightly worse performance in the Gowalla dataset. One reason could be that the Multi-GCCF model provides better representations of the users when the Gowalla is relatively dense than Yelp. However, our model is able to capture better representations of the users when the Yelp dataset contains more auxiliary information of user-item interactions.

The advantages of Precision@N and Recall@N scores on all academic social networks datasets suggest that our model is able to make better scholar recommendations in academic social networks by building better academic learning representation vectors of the scholars. In addition, the comparison results of Recall@20 and NGCG@20 on two popular social networks datasets further confirm the effectiveness of our proposed model.

5.2 Impact of Different Factors

GESRec model converts scholars' multiple academic data (i.e., attributes features and textual features) into academic relationships between scholars, which are further utilized as auxiliary information to enhance the original user-user relationships. GESRec shows obvious superiority against all the baselines. However, the differences between the embedding vectors of attributes features and textual features may affect the quality of academic relationships between scholars, which in turn may affect the effectiveness of recommendations.

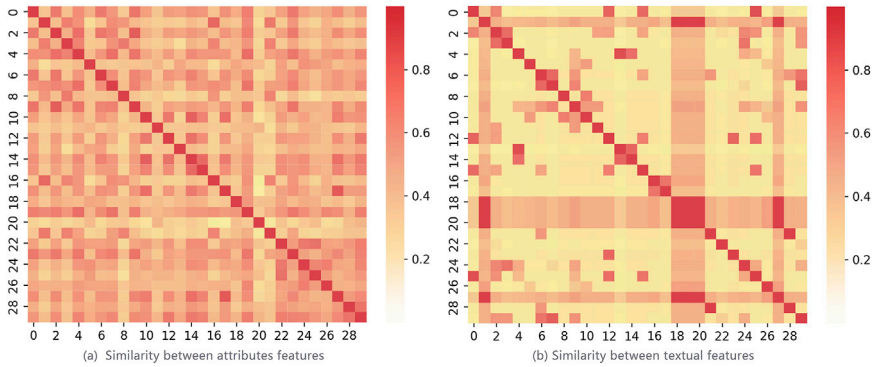


FIGURE 2 | Heatmap of the similarities between academic features from SCHOLAT. **(A)** Similarity between attributes features. **(B)** Similarity between textual features.

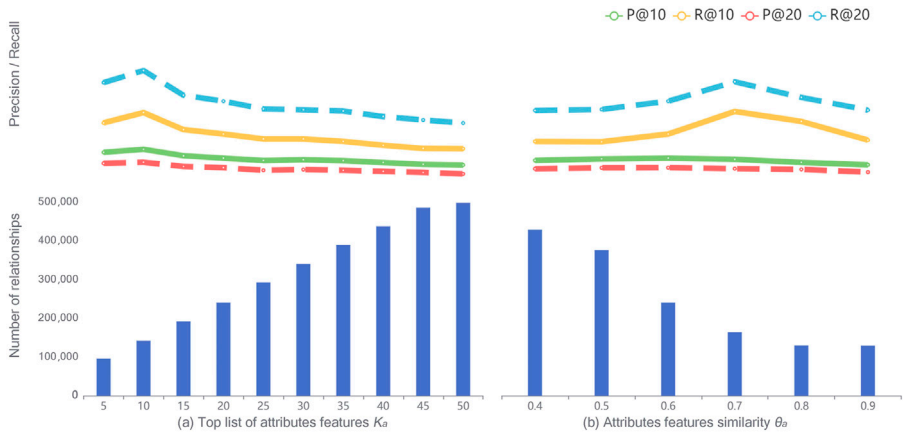


FIGURE 3 | Impact of the attributes features on SCHOLAT dataset. **(A)** Top list of attribute features K_a. **(B)** Attributes features of similarity θ_a.

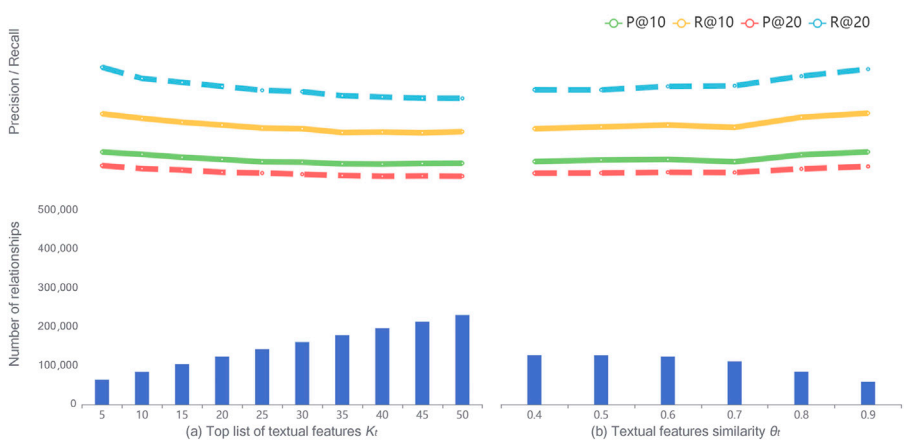


FIGURE 4 | Impact of the textual features on SCHOLAT dataset. **(A)** Top list textual features K_r. **(B)** Textual features of similarity θ_r.

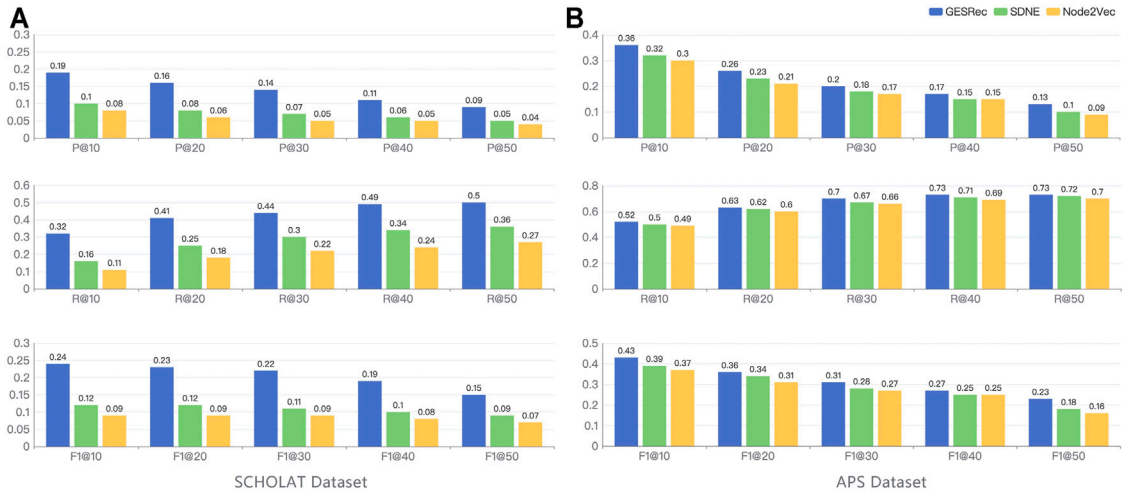


FIGURE 5 | Impact of the graph embedding on SCHOLAT and APS datasets. **(A)** SCHOLAT dataset. **(B)** APS dataset.

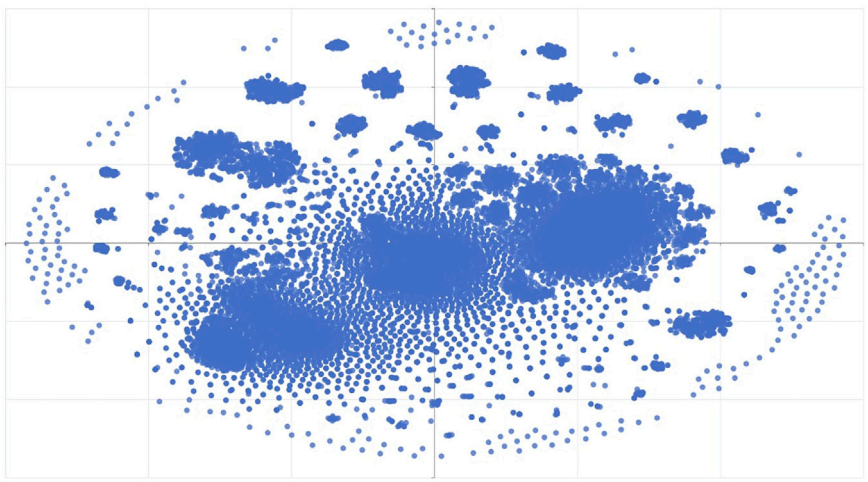


FIGURE 6 | Visualization of the scholar embedding vectors on SCHOLAT.

We take the real academic social networks SCHOLAT as an example. We select the 30 most frequent attributes features and textual features from SCHOLAT, respectively. We plot the heatmap of the similarity between these academic features as shown in **Figure 2**, where the darker the color is, the higher the similarities are. As we can observe from **Figure 2**, similarities between attributes features are higher than similarities between textual features, and the attributes features are more semantically related, indicating similar scholars share higher degrees of academic attributes features relevance. So attribute features-based user-user relationships could be better in building scholar's academic friendships.

Impact of attribute features: the threshold parameters of academic features could affect the effectiveness of building academic scholar's friendships, which in turn affects the results of scholar recommendations. Due to limited space, we only select Precision and Recall at $N = \{10, 20\}$. **Figure 3** shows how the attributes features affect the recommendation performance on the SCHOLAT dataset. We see that with the increase of k_a (threshold of the top list of attributes features), the number of relationships shows nearly linear growth. However, the Precision and Recall values are continuously declined with $k_a = \{15, 20, 25, 30, 35, 40, 45, 50\}$. The only exception is when k_a is 10. This result suggests that the proper number of relationships is able to boost the recommendation performance. When a threshold is breached, a larger number of relationships will bring more noise data, which in turn harms the recommendation performance. We also observe that the number of relationships drops with the increase of θ_a (threshold of attributes features similarity). The Recall values reach a turning point when $\theta_a = 0.7$. However, the Precision values slightly decline with the increase of parameter θ_a. Similarly,

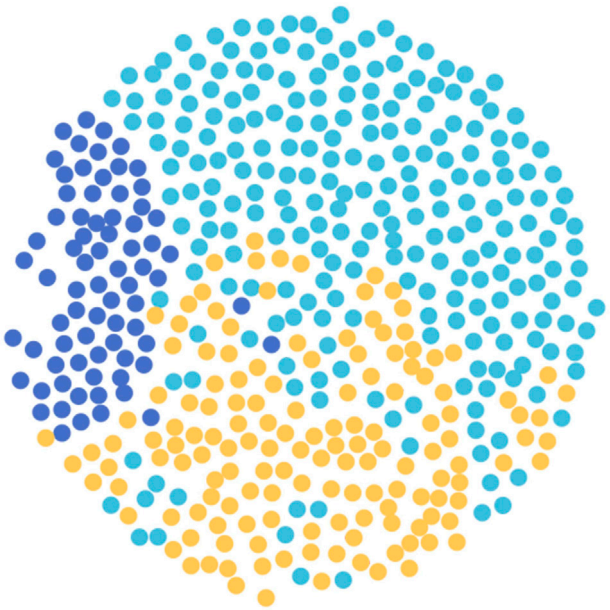


FIGURE 7 | Visualization of the scholar embedding vectors of research teams with different research interest. Blue: "data management," azure: "social networks," and yellow: "natural language processing."

we find that only a proper number (100,000–200,000) of relationships brings good recommendation performance.

Impact of textual features: in **Figure 4**, similar observations can be obtained for the impact of textual features on SCHOLAT datasets. We notice that there is a steady growth in the number of relationships but a smooth decline in the Precision and Recall values with the increase of k_t (threshold of the top list of textual features). These results suggest that the more the relationships, the worse the Precision and Recall results. We also find a large increase in decline in the number of relationships when the textual features similarity θ_t is over 0.7. On the contrary, the performance of Precision and Recall gets better with the increase of θ_t, except for a minor drop at $\theta_t = 0.7$. From **Figures 3**, **4**, we see that the impact of the attributes features threshold k_a is larger than the textual features threshold k_t due to the higher similarities between attributes features. This confirms previous findings in the heatmap of the similarities of academic features from SCHOLAT, where attribute features based on user-user relationships are more representative of scholars' academic relationships.

Impact of graph embedding module: to investigate the effectiveness of graph embedding module in scholar recommendation in academic social networks task, we implement two different network representation learning models[7], SDNE [44] and Node2Vec [45], as the replacement of our graph embedding module. **Figure 5** shows the impact of different graph embedding-based models on SCHOLAT and APS datasets. Recommendation performance on the SCHOLAT dataset shows that the GESRec model outperforms SDNE and Node2Vec in all Precision, Recall, and F1 score metrics. Similar results can be observed on the APS dataset. However, the advantages of performance have been decreased. The possible reason might be that the dataset is based on co-author relationships instead of rich academic correlations between scholars on the SCHOLAT dataset. Overall, these improvements on both datasets verify the effectiveness of the graph embedding in scholar recommendation in ASNs tasks.

[7]https://github.com/palash1992/GEM

5.3 Visualizing Scholar Embedding Vectors

In this section, we visualize the scholar embedding vectors extracted from the SCHOLAT dataset in **Figure 6**. Besides, we randomly select three research teams (e.g., 56 scholars in "data management," 149 scholars in "social networks," and 110 scholars in "natural language processing"). **Figure 7** provides a visualization of the scholar embedding vectors in three research teams with different research interests. Nodes with the same color mean they are from the same research team. Specifically, we first get the scholar embedding vectors by the GESRec model and then map these vectors into a low-dimensional space. Finally, we further map the low-dimensional vectors to a 2D space using the t-SNE [46]. In **Figure 6**, we observe that there are many clusters in which the scholars belong to the same cluster, meaning they share similar academic features. These scholar embedding vectors are used on the following scholar recommendation task. In **Figure 7**, we find that except for a few blue nodes, other nodes from the same research team tend to close one another. This is because scholars in the same research team usually share similar research interests, making their scholar embedding vectors have a higher academic similarity. Hence, these visualizations of the scholar embedding vectors show that our GESRec model is able to recommend potential similar scholars to users with high academic similarities.

6 CONCLUSION AND FUTURE WORK

In this article, we tackle the problem of recommending potential similar scholars for users in academic social networks (ASNs). We propose a novel graph embedding-based scholar recommendation system by leveraging academic auxiliary information. The proposed model consists of three steps: First, we construct the enhanced academic social networks by combining two types of academic features (e.g., attributes features and textual features) from scholar's academic information with user-user relationships, and the correlations between the scholar's academic features are calculated by the similarity of the corresponding embedding vectors. Then, the refined feature representations of the scholars are obtained from the enhanced ASNs by the graph embedding framework. Finally, top-n potential scholars are generated for final recommendation. In the future, we would like to consider more academic information to improve the performance of our model.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusion of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

All authors conceived the paper. CY drafted the initial manuscript. CY and RL developed the software and run the analyses. YH participated in the discussion of theories. YT supervised the study and edited the manuscript. All authors reviewed the article.

FUNDING

This work was supported by the National Natural Science Foundation of China (No. U1811263, 61772211), the Talent Research Start-Up Foundation of Guangdong Polytechnic Normal University (No. 2021SDKYA098).

REFERENCES

1. Xia F, Wang W, Bekele TM, Liu H Big Scholarly Data: A Survey. *IEEE Trans Big Data* (2017) 3:18–35. doi:10.1109/tbdata.2016.2641460
2. Kong X, Shi Y, Yu S, Liu J, Xia F Academic Social Networks: Modeling, Analysis, Mining and Applications. *J Netw Comp Appl* (2019) 132:86–103. doi:10.1016/j.jnca.2019.01.029
3. Jordan K From Social Networks to Publishing Platforms: A Review of the History and Scholarship of Academic Social Network Sites. *Front Digit Humanit* (2019) 6:5. doi:10.3389/fdigh.2019.00005
4. Wan H, Zhang Y, Zhang J, Tang J Aminer: Search and Mining of Academic Social Networks. *Data Intellegence* (2019) 1:58–76. doi:10.1162/dint_a_00006
5. Kyvik S, Reymert I Research Collaboration in Groups and Networks: Differences across Academic fields. *Scientometrics* (2017) 113:951–67. doi:10.1007/s11192-017-2497-5
6. Wu D, Fan S, Yuan F Research on Pathways of Expert Finding on Academic Social Networking Sites. *Inf Process Manage* (2021) 58:102475. doi:10.1016/j.ipm.2020.102475
7. Xu Y, Hao J, Lau RYK, Ma J, Xu W, Zhao D. A Personalized Researcher Recommendation Approach in Academic Contexts: Combining Social Networks and Semantic Concepts Analysis. In: Pacific Asia Conference on Information Systems; Taipei, Taiwan; July 2010. Taipei, Taiwan: PACISAISeL (2010). 9144–12.
8. Yang C, Sun J, Ma J, Zhang S, Wang G, Hua Z. In: Scientific collaborator recommendation in heterogeneous bibliographic networks. 48th Hawaii International Conference on System Sciences, HICSS 2015; January 5-8, 2015; Kauai, Hawaii, USA. IEEE Computer Society (2015). 552–61.
9. Wang Q, Ma J, Liao X, Du W A Context-Aware Researcher Recommendation System for university-industry Collaboration on R&D Projects. *Decis Support Syst* (2017) 103:46–57. doi:10.1016/j.dss.2017.09.001
10. Xu Y, Zhou D, Ma J Scholar-friend Recommendation in Online Academic Communities: An Approach Based on Heterogeneous Network. *Decis Support Syst* (2019) 119:1–13. doi:10.1016/j.dss.2019.01.004
11. Zhang C, Wu X, Yan W, Wang L, Zhang L Attribute-aware Graph Recurrent Networks for Scholarly Friend Recommendation Based on Internet of Scholars in Scholarly Big Data. *IEEE Trans Ind Inf* (2020) 16:2707–15. doi:10.1109/tii.2019.2947066
12. Goyal P, Ferrara E Graph Embedding Techniques, Applications, and Performance: A Survey. *Knowledge-Based Syst* (2018) 151:78–94. doi:10.1016/j.knosys.2018.03.022
13. Zhao Z, Zhang X, Zhou H, Li C, Gong M, Wang Y Hetnerec: Heterogeneous Network Embedding Based Recommendation. *Knowledge-Based Syst* (2020) 204:106218. doi:10.1016/j.knosys.2020.106218
14. Forouzandeh S, Berahmand K, Rostami M Presentation of a Recommender System with Ensemble Learning and Graph Embedding: a Case on Movielens. *Multimed Tools Appl* (2021) 80:7805–32. doi:10.1007/s11042-020-09949-5
15. Hu X, Xu J, Wang W, Li Z, Liu A A Graph Embedding Based Model for fine-grained POI Recommendation. *Neurocomputing* (2021) 428:376–84. doi:10.1016/j.neucom.2020.01.118
16. Khan HU, Daud A, Ishfaq U, Amjad T, Aljohani N, Abbasi RA, et al. Modelling to Identify Influential Bloggers in the Blogosphere: A Survey. *Comput Hum Behav* (2017) 68:64–82. doi:10.1016/j.chb.2016.11.012
17. Wang Y, Han X Attractive Community Detection in Academic Social Network. *J Comput Sci* (2021) 51:101331. doi:10.1016/j.jocs.2021.101331
18. Zhao P, Ma J, Hua Z, Fang S Academic Social Network-Based Recommendation Approach for Knowledge Sharing. *SIGMIS Database* (2018) 49:78–91. doi:10.1145/3290768.3290775
19. Porcel C, Ching-López A, Lefranc G, Loia V, Herrera-Viedma E Sharing Notes: An Academic Social Network Based on a Personalized Fuzzy Linguistic Recommender System. *Eng Appl Artif Intelligence* (2018) 75:1–10. doi:10.1016/j.engappai.2018.07.007
20. Brandão MA, Moro MM, Lopes GR, de Oliveira JPM. Using Link Semantics to Recommend Collaborations in Academic Social Networks. In: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume. International World Wide Web Conferences Steering Committee/ACM (2013). 833–40. doi:10.1145/2487788.2488058
21. Huang X, Tang Y, Qu R, Li C, Yuan C, Sun S, et al. Course Recommendation Model in Academic Social Networks Based on Association Rules and Multi-similarity. In: 22nd IEEE International Conference on Computer Supported Cooperative Work in Design; May 9-11, 2018; Nanjing, China. CSCWDIEEE (2018). 277–82. doi:10.1109/cscwd.2018.8465266
22. Kong X, Mao M, Wang W, Liu J, Xu B Voprec: Vector Representation Learning of Papers with Text Information and Structural Identity for Recommendation. *IEEE Trans Emerg Top Comput.* (2021) 9:226–37. doi:10.1109/tetc.2018.2830698
23. Nikzad-Khasmakhi N, Balafar MA, Feizi-Derakhshi M. The State-Of-The-Art in Expert Recommendation Systems. *Eng Appl Artif Intell* (2019) 82:126–47.
24. Feng W, Zhu Q, Zhuang J, Yu S An Expert Recommendation Algorithm Based on pearson Correlation Coefficient and Fp-Growth. *Cluster Comput* (2019) 22:7401–12. doi:10.1007/s10586-017-1576-y
25. Yang C, Ma J, Silva T, Liu X, Hua Z A Multilevel Information Mining Approach for Expert Recommendation in Online Scientific Communities. *Comp J* (2015) 58:1921–36. doi:10.1093/comjnl/bxu033
26. Davoodi E, Afsharchi M, Kianmehr K A Social Network-Based Approach to Expert Recommendation System. In: Hybrid Artificial Intelligent Systems - 7th International Conference, HAIS 2012; Salamanca, Spain; March 28-30th, 2012, 7208. Salamanca, Spain: Lecture Notes in Computer Science (2012). 91–102. Proceedings, Part I (Springer). doi:10.1007/978-3-642-28942-2_9
27. Ding J, Chen Y, Li X, Liu G, Shen A, Meng X Unsupervised Expert Finding in Social Network for Personalized Recommendation. In: Web-Age Information Management - 17th International Conference, WAIM; Nanchang, China; June 3-5, 2016, 9658. Nanchang, China: Lecture Notes in Computer Science (2016). 257–71. Proceedings, Part I (Springer). doi:10.1007/978-3-319-39937-9_20
28. Ge H, Caverlee J, Lu H. TAPER: A Contextual Tensor-Based Approach for Personalized Expert Recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems; Boston, MA, USA; September 15-19, 2016. Boston, MA, USA: ACM (2016). 261–8.
29. Chen J, Tang Y, Li J, Mao C, Xiao J. Community-based Scholar Recommendation Modeling in Academic Social Network Sites. In: Web Information Systems Engineering - WISE 2013 Workshops - WISE 2013 International Workshops BigWebData, MBC, PCS, STeH, QUAT, SCEH, and STSC 2013; Nanjing, China; October 13-15, 2013, 8182. Nanjing, China: Lecture Notes in Computer Science (2013). 325–34. Revised Selected Papers (Springer).

30. Mao D, Li C, Li J, Tang Y, Chen M, Yang X. Academic Social Network Scholars Recommendation Model Based on Community Division. In: 22nd IEEE International Conference on Computer Supported Cooperative Work in Design; May 9-11, 2018; Nanjing, China. CSCWDIEEE (2018). p. 265–70. doi:10.1109/cscwd.2018.8465149
31. Shi C, Hu B, Zhao WX, Yu PS Heterogeneous Information Network Embedding for Recommendation. *IEEE Trans Knowl Data Eng* (2019) 31: 357–70. doi:10.1109/tkde.2018.2833443
32. Huang X, Qian S, Fang Q, Sang J, Xu C. Meta-path Augmented Sequential Recommendation with Contextual Co-attention Network. *ACM Trans Multim Comput Commun Appl* (2020) 16(1–52):5224. doi:10.1145/3382180
33. Xie F, Zheng A, Chen L, Zheng Z Attentive Meta-Graph Embedding for Item Recommendation in Heterogeneous Information Networks. *Knowledge-Based Syst* (2021) 211:106524. doi:10.1016/j.knosys.2020.106524
34. Li S, Zhao Z, Hu R, Li W, Liu T, Du X. Analogical Reasoning on Chinese Morphological and Semantic Relations. In. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018; Melbourne, Australia; July 15-20, 2018: editors, I Gurevych Y Miyao, 2. Melbourne, Australia: Short Papers Association for Computational Linguistics (2018). 138–43. doi:10.18653/v1/p18-2023
35. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. LINE: Large-Scale Information Network Embedding. In: Proceedings of the 24th International Conference on World Wide Web; Florence, Italy; May 18-22, 2015. Florence, Italy: WWW 2015ACM (2015). 1067–77.
36. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed Representations of Words and Phrases and Their Compositionality. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems; Lake Tahoe, Nevada, United States, 5-8. Lake Tahoe, Nevada, United States: Proceedings of a meeting held December (2013). 3111–9.
37. Xu Q, Qiu L, Lin R, Tang Y, He C, Yuan C. An Improved Community Detection Algorithm via Fusing Topology and Attribute Information. CSCWD 2021. In: 24th IEEE International Conference on Computer Supported Cooperative Work in Design; May 5-7, 2021; Dalian, China. IEEE (2021). p. 1069–74. doi:10.1109/cscwd49262.2021.9437681
38. Tu C, Liu H, Liu Z, Sun M. CANE: Context-Aware Network Embedding for Relation Modeling. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017; Vancouver, Canada; July 30 - August 4, 1. Vancouver, Canada: Long Papers Association for Computational Linguistics (2017). 1722–31. doi:10.18653/v1/p17-1158
39. Pennington J, Socher R, Manning CD. Glove: Global Vectors for Word Representation. In: A Moschitti, B Pang, W Daelemans, editors. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014; October 25-29, 2014; Doha, Qatar. Doha, Qatar: A meeting of SIGDAT, a Special Interest Group of the ACL ACL (2014). p. 1532–43. doi:10.3115/v1/d14-1162
40. Sun J, Zhang Y, Ma C, Coates M, Guo H, Tang R, et al. Multi-graph Convolution Collaborative Filtering. *CoRR* ***abs/***(2020) 2001–00267.
41. Zhou Y, Wilkinson DM, Schreiber R, Pan R. Large-scale Parallel Collaborative Filtering for the Netflix Prize. In: Algorithmic Aspects in Information and Management, 4th International Conference, AAIM 2008; Shanghai, China; June 23-25, 2008, 5034. Shanghai, China: Lecture Notes in Computer Science (2008). 337–48. Proceedings (Springer).
42. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. BPR: Bayesian Personalized Ranking from Implicit Feedback. *CoRR* ***abs/*** (2012) 1205:2618.
43. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online Learning of Social Representations. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; New York, NY, USA; August 24 - 27, 2014. New York, NY, USA: KDD '14ACM (2014). 701–10.
44. Wang D, Cui P, Zhu W. Structural Deep Network Embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; San Francisco, CA, USA; August 13-17, 2016. San Francisco, CA, USA: ACM (2016). p. 1225–34. doi:10.1145/2939672.2939753
45. Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; San Francisco, CA, USA; August 13-17, 2016, 2016. San Francisco, CA, USA: ACM (2016). 855–64. KDD. doi:10.1145/2939672.2939754
46. van der Maaten L, Hinton G. Visualizing Data Using T-Sne. *J Machine Learn Res* (2008) 9:2579–605.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Physics

REVIEW
published: 29 November 2021
doi: 10.3389/fphy.2021.766540

Adversarial Machine Learning on Social Network: A Survey

Sensen Guo[1,2], *Xiaoyu Li*[1,2]* *and Zhiying Mu*[1,2]

[1]Research & Development Institute of Northwestern Polytechnical University in Shenzhen, Shenzhen, China, [2]School of Cybersecurity, Northwestern Polytechnical University, Xi'an, China

In recent years, machine learning technology has made great improvements in social networks applications such as social network recommendation systems, sentiment analysis, and text generation. However, it cannot be ignored that machine learning algorithms are vulnerable to adversarial examples, that is, adding perturbations that are imperceptible to the human eye to the original data can cause machine learning algorithms to make wrong outputs with high probability. This also restricts the widespread use of machine learning algorithms in real life. In this paper, we focus on adversarial machine learning algorithms on social networks in recent years from three aspects: sentiment analysis, recommendation system, and spam detection, We review some typical applications of machine learning algorithms and adversarial example generation and defense algorithms for machine learning algorithms in the above three aspects in recent years. besides, we also analyze the current research progress and prospects for the directions of future research.

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Xinghua Li,
Xidian University, China
Gui-Quan Sun,
North University of China, China

***Correspondence:**
Xiaoyu Li
lixiaoyu@nwpu.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *29 August 2021*
Accepted: *14 October 2021*
Published: *29 November 2021*

Citation:
Guo S, Li X and Mu Z (2021) Adversarial Machine Learning on Social Network: A Survey. Front. Phys. 9:766540. doi: 10.3389/fphy.2021.766540

Keywords: social networks, adversarial examples, sentiment analysis, recommendation system, spam detection

1 INTRODUCTION

In recent years, with the rapid development of internet technology, social networks have played an increasingly important role in people's lives [1]. Among them, social networks such as Facebook, Twitter, and Instagram have shortened the distance between people and changed the way that people get information. For example, more and more people are willing to share new things happening around them with friends through social networks, and government agencies release the latest policy information to the public through social networks. With the rapid popularization of social networks, the role of social networks is not limited to providing people with a channel to communicate with friends. For example, users can be profiled according to its timelines, then the system can recommend friends, topics, information, and products that users may be interested in, which can greatly enrich people's leisure life. Filtering useless spam and robot accounts can not only reduce the time that users spend on browsing spam but also protect users from phishing website attacks. Besides, research on social network information dissemination [2, 3] can not only facilitate social network marketing but also effectively predict and control public opinion. The study of the interaction between disease and disease information on complex networks [4] has played an important role in understanding the dynamics of epidemic transmission and the interaction between information dissemination. Therefore, how to use social networks to achieve various functions has become a research hotspot in recent years.

With significant improvement in the performance of computers and the widespread application of GPUs, Machine learning (ML) especially Deep Learning (DL) has been widely used in various industries (such as automatic driving, computer vision, machine translation, recommendation

systems, cybersecurity, etc.). In terms of social networks, many scholars also use machine learning algorithms to implement functions such as friend or information recommendation, user interest analysis, and spam detection. However, it can't be ignored that machine learning algorithms are vulnerable to adversarial examples, that is, adding perturbations that are not perceptible to the human eye can mislead the classifier to output a completely different classification result. After the concept of adversarial examples was proposed, many studies have shown that no matter how the machine learning model is adjusted, it can always be successfully broken by new adversarial example generation methods. In recent years, the research on the generation and defense of adversarial examples has spread from the field of computer vision [5] to social networks, cybersecurity [6], natural language processing [5, 7], audio and video processing [8], graph data processing [5], etc. Therefore, the ability to effectively defend against adversarial examples has become a key factor of whether machine learning algorithms can be applied on a large scale.

In this paper, we focus on adversarial machine learning in the field of social networks, that is, adversarial example generation and defense technology in the field of social networks. Firstly, we reviewed the recent research progress of machine learning algorithms in social networks in terms of sentiment analysis, recommendation systems, and spam detection, and then we summarized the latest research on adversarial example generation and defense algorithms in recent years. Next we sorted out some research progress of adversarial example generation and defense algorithms in social networks. Finally, we summarized the advantages and disadvantages of existing algorithms, and prospects for its future research directions.

The rest of this paper is organized as follows. In **section 2**, the application of machine learning algorithms in social networks in recent years is reviewed. **Section 3** reviews the security issues faced by machine learning algorithms and the robust reinforcement strategies against different attacks. **Section 4** summarizes the attack and defense algorithms for machine learning in social networks. **Section 5** analyzes the problems of adversarial example generation and defense algorithms in the field of social networks, prospect the future research direction, and concludes this paper.

2 MACHINE LEARNING IN SOCIAL NETWORKS

While social network such as Twitter, Facebook, and Instagram facilitate people's communication, they also change people's lifestyles to a great extent. The application of machine learning in social networks also promotes the vigorous development of social networks to a large extent. The main applications of machine learning in social networks are as follows: sentiment analysis, recommendation system, spam detection, community detection [9], network immunization [10], user behavior analysis [11, 12], and other aspects. In this paper, we mainly review the application of machine learning in social networks from three aspects: sentiment analysis, recommendation system, and spam detection.

2.1 Sentiment Analysis

Millions of users have posted various opinions on social networks every day, involving daily life, news, entertainment, sports, and other aspects. The emotional of user's comment on different topics can be divided into positive, neutral, and negative categories. With the user's emotional tendency on different topics, we can learn the user's personality, value tendency, and other information. And then more targeted strategies can be used for specific users in activities such as topic dissemination and product promotion. Some researches of machine learning in sentiment analysis are shown in **Table 1**.

Wang et al. [26] introduced a multi-head attention-based LSTM model to perform aspect-level sentiment analysis, they carry out their experiment on the dataset of SemEval 2014 Task 4 [27], the results of the experiment show that their model is advantageously competitive in aspect-level classification. Based on this, Long et al. [15] introduced an improved method with bidirectional LSTM network and multi-head attention mechanism, they utilize the multi-head attention to learn the relevant information from a different representation subspace, and achieved 92.11% accuracy on comment dataset from Taobao.

To perform aspect-based sentiment analysis of Arabic Hotels' reviews, both SVM and deep RNN were used in Al-Smadi et al. [13]'s works, respectively. They evaluated their method on Arabic Hotels' reviews dataset. The results show that the performance of SVM is superior to the other deep RNN approach in the aspect category identification, opinion target expression extraction, and the sentiment polarity identification, but inferior to RNN approaches in the execution time required for training and testing.

By using the API provided by Twitter, Hitesh et al. [14] collected 18,000 tweets without retweets on the term Indian elections. Based on these data, they proposed a model that combined with word2vec and random forest model to perform sentiment analysis, and they used a Word2Vec feature selection model to extract features and then train a random forest model for sentiment analysis, and their final accuracy reaches 86.8%.

Djaballah et al. [16] proposed a method to detect content that incites terrorism on Twitter, they collected tweets related to terrorism in Arabic and manually classified these tweets in "tweets not inciting terrorism" and "tweets inciting terrorism". Based on Google's Word2vec method [17], they introduce a method of Word2vec by the weighted average to generate tweets feature vectors, then SVM and Random Forest classifiers were used for the prediction of sentiments. The experiments results show that their method can improve the prediction results of the Word2vec method [17] slightly.

Ho et al. [18] proposed a two-stage combinatorial model to perform sentiment analysis. In the first stage, they trained five machine learning algorithms: logistic regression, naive Bayes, multilayer perceptron, support vector machine and random forest with the same dataset. In the second stage, a combinatorial fusion is used to combine a subset of these five algorithms, and experiment results show that the combination of these algorithms can achieve better performance.

To capture precise sentiment expressions in aspect-based sentiment analysis for reasoning, Liu et al. [19] introduced a

TABLE 1 | Maching learning in sentiment analysis.

Authors	Introduced methods	Year	Datasets	Baseline
Al-Smadi et al. [13]	SVM and Deep RNN	2018	Arabic Hotels' reviews	—
Hitesh et al. [14]	Word2Vec & Random forest	2019	Twitter	BOW, TF-IDF
Long et al. [15]	BiLSTM-MHAT	2019	Taobao	CNN, BiLSTM, Attention-BiLSTM
Djaballah et al. [16]	SVM, Random Forest	2019	Twitter	Word2vec [17]
Ho et al. [18]	Combinatorial model	2019	Kaggle	LR, NB, RF, SVM, MLP
Liu et al. [19]	AS-Reasoner	2019	SemEval-2014, SemEval-2015	LSTM, TD-LSTM, TD-LSTM.etc
Yao et al. [20]	DSSA-H	2020	Twitter	SVM, RF
Umer et al. [21]	CNN-LSTM	2021	Twitter	CNN [22], LSTM [23]
Lv et al. [24]	CAMN	2021	SemEval-2014, Twitter	CEA, DAuM, TNet-AS,etc
Rawat et al. [25]	SMODT	2021	Twitter	KNN, SVM, DT, SMO

TABLE 2 | Maching learning in recommendation system.

Authors	Introduced methods	Year	Datasets	Baseline
Fan et al. [28]	GraphRec	2019	Epinions, Ciao	GC-MC [29], DeepSoR [30], NeuMF [31]
Gui et al. [32]	Cooperative Multi-Agent Approach	2019	Dataset Containing 50 Historical Tweets Per User	LSTM, Attention methods, Independent Q-Learning, Random sampling
Guo et al. [33]	GNN-SoR	2020	pinions [34], Yelp [35], Flixster [36]	SocialMF [37], TrustSVD [38], TrustMF [39], AutoRec [40]
Huang et al. [41]	MAGRM	2020	Meetup, MovieLens-1M	DPMF-CNN [42], AGR [43], AGREE [44]
Pan et al. [45]	CoDAE	2020	Epinions, Ciao	CDAE [46], TDAE [47]
Zheng et al. [48]	ITRA	2021	Delicious [49], FilmTrust [50], CiaoDVD [51]	CDAE [46], SAMN [52], CAVE [53]
Ni et al. [54]	RM-DRL	2021	Netflix, BookCrossing, Movielens-20M, Movielens-1M, HetRec 2011-Movielens	ConvMF [55], DRMF [56], GNN [57], AFM [58], RACMF [59], HRAM [60], DAINN [61]
Tahmasebi et al. [62]	SRDNet	2021	MovieTweetings, Open Movie Database	AutoRec [40], MRS-RBM [63], PP-CF [64], et

method named Attention-based Sentiment Reasoner (AS-Reasoner). In their model, an intra attention and a global attention mechanism was designed, respectively. The intra attention computes weights by capturing the sentiment similarity between any two words in a sentence, and the global attention computes weights by a global perspective. They carried out an experiment on various datasets, and the results show that the AS-Reasone is language-independent, and it also achieves state-of-the-art macro-F1 and accuracy for aspect-based sentiment analysis.

Umer et al. [21] proposed a deep learning model which is combined with CNN and LSTM network to perform sentiment analysis on Twitter. The CNN layer is used to learn the higher-level representation of sequences from original data and feed it to the LSTM layers. They carry out their experiment on three Twitter dataset which includes a women's e-commerce dataset, an airline sentiment dataset, and a hate speech dataset, and the accuracy on three datasets is 78.1, 82.0, and 92.0%, respectively, which is markedly superior to singly use of CNN [22] and LSTM [23].

2.2 Recommendation System

The social network recommendation system is an important part of the social network system. Recommendation systems such as friend recommendation, content recommendation, and advertising delivery greatly enrich people's social life while also create huge economic benefits. Recommending friends and article content that users may be interested in will extend the time users surf the social networks; Pushing advertising information to users reasonably and effectively can not only creating significant economic benefits but also facilitate users' lives. As shown in **Table 2**, with the rapid development of machine learning, many scholars have also carried out research on social network recommendation system based on machine learning.

Fan et al. [28] try to perform social recommendation with graph neural networks, and they introduced a model named GraphRec (Graph Neural Network Framework), which is composed of the user modeling, the item modeling, and the rating prediction. Both the user modeling and the item modeling used graph neural network and attention network to learn user latent factors ($\mathbf{h_i}$) and the learn item latent factors ($\mathbf{z_j}$) from the original data, respectively, the rating prediction concatenate the user latent factors and the item latent factors and feed into a multilayer perceptron neural network for rating prediction. They evaluated the GraphRec with two representative datasets Epinions and Ciao, and the results show that the GraphRec can outperform GC-MC (Graph Convolutional Matrix Completion) [29], DeepSoR (Deep Neural Network Model on Social Relations for Recommendation) [30], NeuMF (Neural Matrix Factorization) [31], and some other baseline algorithms.

Guo et al. [33] hold that the feature space of social recommendation is composed of user features and item feature, the user feature is composed of inherent preference and social influence, and the item feature include attribute contents, attribute correlations, and attribute concatenation. They introduced a framework named GNN-SoR (Graph

Neural Network-based Social Recommendation Framework) to exploit the correlations of item attributes. In their framework, two graphs neural network methods are used to encode the user feature space and the item feature space, respectively. Then, the encoded two spaces are regarded as two potential factors in the matrix factorization process to predict the unknown preference ratings. They conducted experiments on real-world datasets Epinions [34], Yelp [35] and Flixster [36] respectively, and the experimental results indicated that the perform of GNN-SoR is superior to four baselines algorithm such as: SocialMF (Matrix Factorization based Social Recommendation Networks) [37], TrustSVD [38], TrustMF [39], and AutoRec [40].

Huang et al. [41] introduced a model named MAGRM (Multiattention-based Group Recommendation Model) to perform group recommendation, and the MAGRM is consists of two multiattention based model: the VR-GF (vector representation for group features) and the PL-GI (preference learning for groups on items). The VR-GF is used for getting the deep semantic feature for each group. Based on VR-GF, the PL-GI is used for predicting groups' ratings on items, the experiment with two real-world dataset Meetup and MovieLens-1M, and the performance of MAGRM outperforms AGR [43], AGREE (Attentive Group Recommendation) [44] and other algorithms.

Pan et al. [45] introduced a model named CoDAE (Correlative Denoising Autoencoder) to perform top-k recommendation task, which learn user features by modeling user with truster, roles of rater, and trustee with three separate denoising autoencoder model. They carried out an experiment on Ciao and Epinions datasets, they found that their method is superior to CDAE (Collaborative Denoising Auto-Encoders) [46], TDAE [47], and some other baseline algorithms. Similar to [45], Zheng et al. [48]. proposed a model named ITRA (Implicit Trust Relation-Aware model) which is based on Variational Auto-Encoder to learn the hidden relationship between huge amounts of graph data. They evaluated their model on three dataset: Delicious [49], FilmTrust [50], and CiaoDVD [51], where the performance of ITRA was markedly superior to SAMN (Social Attention Memory Networ) [52], CVAE [53], and CDAE [46] in the top-n item recommendation task.

By capturing the semantic features of users and items effectively, Ni et al. [54] proposed a model named RM-DRL (Recommendation Model based on Deep Representation Learning). According to the authors, firstly, they used a CNN network to learn the semantic feature vector of the item from its primitive feature vectors. Next, they used an Attention-Integrated Gated Recurrent Unit to learn user semantic feature vector from a series of user features such as the user preference history, semantic feature vectors, primitive feature vector and so on. Finally, the users' preferences on the items were calculated with the semantic feature vectors of the items and the users. They conduct their experiments on five datasets, and the results show that the performance of RM-DRL is superior to ConvMF [55], AFM (Attentional Factorization Machines) [58], GNN [57], HRAM (Hybrid Recurrent Attention Machine) [60], etc.

2.3 Spam Detection

Social networking is one of the main channels for people to acquire information. However, the overwhelming spam and network phishing links also bring great troubles to people's work and life. Therefore, how to detect spam on social networks effectively is an important issue. As shown in **Table 3**, many scholars have proposed various methods to solve this problem in recent years.

Karakasli et al. [65] tried to detect spam users with machine learning algorithms. Firstly, they collect twitter user data with software named CRAWLER. Then, 21 features in total was extracted from the original Twitter data. Next, a dynamic feature selection method was used to reduce the model complexity. Finally, they used SVM and KNN algorithm to perform spam user detection, and the success detects rate for KNN was 87.6 and 82.9% for SVM.

Aiming at the problem of difficult spam detection caused by the short text and large semantic variability on social networks, by combining the convolutional neural network (CNN) with long short term memory neural network (LSTM), Jain et al. [66] introduced a deep learning spam detection architecture named Sequential Stacked CNN-LSTM (SSCL). Firstly, it uses the CNN network to extract feature sequences from original data, then it feed the feature sequences to the LSTM network, and finally the sigmoid function was used to classify the label as spam or none-spam. They evaluated the performance of SSCL on two dataset: SMS and Twitter, and its precision, accuracy, recall, and F1 score achieved 85.88, 99.01, 99.77, and 99.29%, respectively.

Zhao et al. [68] introduced a semi-supervised graph embedding model to detect spam bot for the directed social network, where they used the attention mechanism and graph neural network to detect spam bot based on the retweet relationship and the following relationship between users. They experimented with the Twitter 1KS-10KN dataset [69] which was collected on Twitter, compared with GCN, GraphSAGE, and GAT, their method achieved the best performance in Recall, Precision, and F1-score.

Focusing on the uneven distribution of spam data and non-spam data on Twitter, Zhang et al. [70] proposed an algorithm named I2RELM (Improved Incremental Fuzzy-kernel-regularized Extreme Learning Machine), which adopt fuzzy weights (each input data is provided with a weight s_i, which is in the interval of (0,1] and assigned by the ratio of spam users to non-spam users in the whole dataset) to improve the detection accuracy of the model on the non-uniformly distributed dataset. They evaluated their method with the data obtained from Twitter, and the performance of I2RELM on the accuracy, TRP, precision, and F-measure was superior to SVM, DT, RF, BP, RBF, ELM, and XG-Boost.

To perform spam detection for movie reviews, Gao et al. [71] proposed an attention mechanism based machine learning model named adCGAN. Firstly, they used SkipGram to extract word vectors from all reviews, and extended SIF algorithm [80] to generate sentence embedding. Then, they combined the encoded movie features and sentence vectors, and used attention driven generate adversarial network to perform review spam detection. They evaluated their method with the review data collected from

TABLE 3 | Machine learning in spam detection.

Authors	Introduced methods	Year	Datasets	Baseline
Karakasli et al. [65]	SVM and KNN	2019	Twitter	—
Jain et al. [66]	SSCL	2019	SMS and Twitter	KNN, NB, RF, SVM etc.
Tajalizadeh et al. [67]	INB-DenStream	2019	Twitter	DenStream, StreamKM++, CluStream
Zhao et al. [68]	Attention + GNN	2020	Twitter 1KS-10KN [69]	GCN, GraphSAGE, and GAT
Zhang et al. [70]	I2RELM	2020	Twitter	SVM, DT, RF, BP, RBF, ELM, XG-Boost
Gao et al. [71]	adCGAN	2020	Douban	MCSVM [72], VAE [73]
Zhao et al. [74]	Ensemble Learning	2020	[75]	CSDNN and WSNN [76]
Alom et al. [77]	Text-based & Combined classifier	2020	Twitter Social Honeypot, Twitter 1KS-10KN [69]	Blacklist-based Approach [78]
Neha et al. [79]	LSTM + Attention	2021	Twitter	Bi-LSTM, K Neighbor, Random forest, Decision tree, Naive Bayes

Douban, the accuracy of adCGAN achieved 87.3%, which was markedly superior to MCSVM [72], VAE [73], and some other baseline algorithms.

Aiming at the problem of class imbalances in the spam detection task, Zhao et al. [74] proposed an ensemble learning framework which based on heterogeneous stacking. Firstly, six different machine learning algorithms including SVM, CART, GNB (Gaussian Naive Bayes), KNN, RF, and LR were used to perform classification tasks separately. Then, feed the output of six machine learning algorithm to cost-sensitive learning based neural network to get the spam detect result. They experimented with the dataset collected by Chen et al. [75], and its performance was markedly superior to CSDNN and WSNN [76].

3 SECURITY IN MACHINE LEARNING

The concept of adversarial example was first introduced by Szegedy et al. [81], they found that the machine learning classifier would get completely different results by adding perturbation that hardly perceptible by the human eye to the original picture, Szegedy believes that the discontinuity of mapping between input and output caused by the highly nonlinear machine learning model is the main cause for the existence of adversarial examples. While Goodfellow et al. [82] and Luo et al. [83] believe that the machine learning model are vulnerable to adversarial examples is mainly due to its linear part, in the high-dimensional linear space, the superposition of multiple small perturbations in the network will cause a great change in the output. Glimer et al. [84] believe that adversarial examples are caused by the high dimensionality of the input data, while Ilyas et al. [85] believe that the adversarial example is not bugs but features, since the attributes of the dataset include robustness and non-robustness features, when we delete non-robust features from the original training set, we can obtain a robust model through training, the adversarial examples are generated due to its non-robust features, and have little relation with machine learning algorithms.

3.1 Attacks to Machine Learning Models

The generation process of adversarial examples is to mislead the target machine learning model by adding perturbation η that are imperceptible to the human eye on the original data, which can be expressed as [86]:

$$\min_{x^{adv}} J\left(f\left(x^{adv}\right), y^{adv}\right) \quad \text{s.t.} \quad \begin{cases} \|\eta\|_p \le \varepsilon, \\ f(x) = y, \\ y \ne y^{adv}, \end{cases} \tag{1}$$

where $J(\cdot)$ is the loss function, $f(\cdot)$ is the target machine learning model, x^{adv} is the adversarial example, η is the adversarial perturbation added to original data x, ε is a normal used to limit the size of η.

According to the degree of understanding to the target model, attacks to machine learning models can be divided into white-box attacks and black-box attacks. White-box attacker obtains all information such as the structure and parameters of the target model, on the contrary, the black-box attacker know nothing about the structural information of the target model, and can only query the output of the target model based on the input [87].

3.1.1 White-Box Attacks

Szegedy et al. [81] first introduced a white-box attack method named L-BFGS, which try to craft adversarial examples by defining the search for the smallest possible attack perturbation as an optimization problem, it can be expressed as:

$$\min_{x'} c\|\eta\| + J_\theta\left(x', l'\right) \quad \text{s.t.} \quad x' \in [0, 1] \tag{2}$$

where c is a constant, η is the perturbation, $J(\cdot)$ is the loss function.

Although L-BFGS has a high attack success rate, its computational complexity is expensive; Similar to Szegedy, based on optimization method, Carlini et al. [88] also proposed an adversarial example generate method named C&W. The research made by Carlini et al. showed that the algorithm can effectively attack most of the existing models [89, 90]; Combining C&W and Elastic Net, Chen et al. [91] introduced a method named EAD to craft adversarial examples, compared with C&W, the adversarial examples generated by EAD have stronger transferability.

To reduce the computation complexity of L-BFGS, Goodfellow et al. [82] introduced a method named FGSM

(Fast Gradient Sign Method), which is a single-step attack that adds perturbation along the direction of gradient, and the perturbation is calculated as $\eta = \varepsilon \text{sign}(\nabla_x J(\theta, x, y))$, where $J(\cdot)$ is the loss function, θ is the parameters of target model, and ε is the size of the perturbation; Based on FGSM, Kurakin et al. [92, 93]) introduced a method named BIM (Basic Iterative Method), which used an iterative method to generate adversarial examples, and they also used the real pictures to evaluate the effectiveness of BIM; Based on BIM, by limiting the size of the perturbation in each iteration strictly, Madry et al. [94] introduced a method named PGD (Projected Gradient Descent), the experiment result shows that the adversarial examples crafted by PGD have better transferability; Similar to Madry et al. [94], Dong et al. [95] introduced a method named MI-FGSM, which integrated the momentum term into the iterative process to craft adversarial examples, compare with BIM, the MI-FGSM can effectively escape from poor local maxima during the iterations.

In order to find the minimal perturbations that are sufficient to mislead the target machine learning model, based on the iterative linearization of the classifier, Moosavi-Dezfooli et al. [96] proposed the DeepFool algorithm, which helps the attacker to craft adversarial examples with minimal perturbations.

Without calculating the gradient of the target model, Baluja et al. [97] introduced a method named ATN (Adversarial Transformation Network) to perform white-box or black-box attacks by training an adversarial transformer network, which can transformer the input data into the target or untargeted adversarial examples. Similar to ATN, Xiao et al. [98] introduced advGan to craft adversarial examples, based on generative adversarial network, the generator of advGAN is used to generate the perturbation to the input data, and the discriminator is used to distinguish the original data from adversarial examples generated by the generator. Besides, Bai et al. [99] proposed AI-GAN to crafted adversarial examples. The above methods [97–99] only need to query the target model during the stage of model training stage, which is fast and efficient.

To find the strongest attack policy, Mao et al. [100] proposed Composite Adversarial Attacks (CAA). They adopted the NSGA-II genetic algorithm to find the best combination of attack algorithms from a candidate pool composed of 32 base attackers. The attack policy of CAA can be expressed as:

$$s: \mathcal{A}_N^s\left(\mathcal{A}_2^s\left(\mathcal{A}_1^s\left(x, \mathcal{F}; \epsilon_{s_1}, t_{s_1}\right), \mathcal{F}; \epsilon_{s_2}, t_{s_2}\right)\right) \ldots, \mathcal{F}; \epsilon_{s_N}, t_{s_N}\right) \quad (3)$$

where $\mathcal{A}_i(\cdot)$ is one of the attack algorithm in attack pool, ϵ_{s_i} and t_{s_i} is the hyperparameter of $\mathcal{A}_i(\cdot)$, $\mathcal{F}$ is the target model.

3.1.2 Black-Box Attacks

During the processes of black-box attack, the attacker know nothing about the target model, and the mainstream approach is based on gradient estimation and substitute model.

3.1.2.1 Based on Gradient Estimation

In this scenario, the attacker estimates the gradient information of the target model by feeding data into the target model and querying its output. Chen et al. [101] extended the C&W [88] algorithm and proposed Zeroth Order Optimization (ZOO) algorithm to perform black-box adversarial examples generation. Although the ZOO algorithm has a high success rate in generating adversarial examples, it requires a large amount of queries on the target model. To reduce the number of queries to the target model, Ilyas et al. [102] used the variant of NES algorithm [103] to estimate the gradient of the target model, which significantly reduces the query complexity to the target model. Tu et al. [104] proposed a framework named AutoZOOM, which adopts an adaptive random gradient estimation strategy and dimension reduction techniques to reduce the query count, compared with the ZOO [101], under the premise of achieving the same attack effect, AutoZOOM can significantly reduce the query complexity. Du et el [105]. also train a meta attacker mode to reduce the query count. Bai et al. [106] proposed the NP-attack algorithm, which also greatly reduces the query complexity by exploring the distribution of adversarial examples around benign inputs. Besides, Chen et al. [107] proposed the HopSkipJumpAttack algorithm, which applies binary information at the decision boundary to estimate gradient direction.

3.1.2.2 Based on Substitute Model

Based on the transferability of the adversarial examples, the attack usually trains a substitute model and uses the white-box attack algorithm to craft adversarial examples on the substitute model. Papernot et al. [108] first used substitute model to generate adversarial examples. Their research also shows that the attacker can perform black-box attack based on the transferability of adversarial examples, even if the structure of the substitute model is completely different from the target model. Zhou et al. [109] proposed a data-free substitute model train method (DaST) to train a substitute model for adversarial attack without any real data. By efficiently using the gradient of the substitute model, Ma et al. [110] proposed a highly query-efficient black-box adversarial attack model named SWITCH. Zhu et al. [111] used the PCIe bus to learn the information of machine learning models in the model-privatization deployments and proposed the Hermes Attack algorithm to fully reconstruct the target machine learning model. By focusing on the training strategy of the substitute model on the data distributed near the decision boundary, Wang et al. [112] improve the transferability of adversarial examples between the substitute model and the target model significantly. Based on meta-learning, Ma et al. [113] train a generalized substitute model named Simulator to mimic any unknown target model, which significantly reduces the query complexity to the target model.

3.2 Defense Against Adversarial Examples

The defense of adversarial examples is an important component of machine learning security. Many scholars have also proposed different adversarial example defense strategies in recent years. The strategies are divided into input data transformer, adversarial example detection, and model robust enhance.

3.2.1 Input Data Transformer

Since perturbation of adversarial examples are usually visually imperceptible, by compressing away these pixel manipulation,

Das et al. [114] introduced a defense framework based on JPEG compression. Cheng et al. [115] adopt a self-adaptive JPEG compression algorithm to defend against adversarial attacks of the video.

Based on generative adversarial network, Samangouei et al. [116] introduced the Defense-GAN to defend against adversarial attacks. By learning the distribution of unperturbed images, the Defense-GAN can generate the clean sample that approximates the perturbed images. Although the Defense-GAN could defend against most commonly attack strategies, its hyper-parameters is hard to train. Hwang et al. [117] also introduced a Purifying Variational Autoencoder (PuVAE) to purify adversarial examples, which is 130 times faster than Defense-GAN [116] in inference time. Besides, Lin et al. [118] introduced InvGAN to speed up Defense-GAN [116]. Zhang et al. [119] proposed an image reconstruction network based on residual blocks to reconstruct adversarial examples into clean images, in addition, adding random resize and random pad layer to the end of the reconstruction network is very effective in eliminating the perturbations introduced by iterative attacks.

3.2.2 Adversarial Example Detection

Just as the name implies, the adversarial example detection algorithms enhance the robustness of the machine learning model by filtering out adversarial examples in a large number of data sets, it detects adversarial examples mainly by learning the differences in characteristics and distribution between the adversarial examples and the normal data.

Among many works, Liu et al. [120] used the gradient amplitude to estimate the probability of modifications caused by adversarial attacks and applied steganalysis to detect adversarial examples. The experiment indicated that their method can accurately detect adversarial examples crafted by FGSM [82], BIM [92], DeepFool [96], and C&W [88]. Wang et al. [121] proposed a SmsNet to detect adversarial examples, which introduced a "SmsConnection" to extract statistical features and proposed a dynamic pruning strategy to prevent overfitting. The experiment indicated that the performance of SmsNet was superior to ESRAM (Enhanced Spatial Rich Model) [120] on detecting adversarial examples crafted by various attacks algorithms.

Noticing the sensitivity of adversarial examples to the fluctuations occurring at the highly-curved region of the decision boundary, Tian et al. [122] proposed Sensitivity Inconsistency Detector (SID) to detect adversarial examples, which achieved detection performance in detecting adversarial examples with small perturbation. Besides, based on the feature that adversarial examples are more sensitive to channel transformation operations than clean examples, Chen et al. [123] proposed a light-weighted adversarial examples detector based on adaptive channel transformation named ACT-Detector. The experiments show that the ADC-detector can defend against most adversarial attacks.

To lessen the dependence on prior knowledge of attacks algorithms, Sutanto et al. [124] proposed a Deep Image Prior (DIP) network to detect adversarial examples, they used a blurring network as the initial condition to train the DIP network only using normal noiseless images. In addition, it is applicable for real-time AI systems due to its faster detection speed for real images. Liang et al. [125] consider the perturbation crafted by adversarial attacks as a kind of noise, They use scalar quantization and smoothing spatial filter to implement an adaptive noise reduction for input images.

3.2.3 Model Robust Enhancement

Model robust enhancement mainly includes adversarial training and certified training. Adversarial training improves the model's immunity to adversarial examples by adding adversarial examples in its training set [126]. Certified training enhances model robustness by constraining the output space of each layer of the neural network under specific inputs during the training process.

3.2.3.1 Adversarial Training

Adversarial training is one of the effective methods to defend against the attacks from adversarial example, the process of adversarial training can be approximated by the following minimum-maximum optimization problem [94].

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}}\left[\max_{\delta\in S} L\left(f_{\theta}(x+\delta), y\right)\right] \tag{4}$$

where D is the set of training data, $f(\cdot)$ is the target neural network, θ is the parameter of $f(\cdot)$, L is the loss function, and δ is the adversarial perturbation.

Szegedy et al. [81] first introduced the concept of adversarial training by training the neural network on the dataset composed of clean data and adversarial examples. Goodfellow et al. [82] also tried to enhance the robustness of the machine learning model by adding adversarial examples crafted by FGSM algorithm to the training set. Although it is more effective in defending against attacks from the FGSM algorithm, it is helpless against attacks from more aggressive algorithms such as C&W [88] and PGD [94]. Madry et al. [94] tried to enhance the robustness of neural networks from the lens of robust optimization, they used the saddle point formula to optimize the parameters of the network model, thereby reducing the loss of the model on adversarial examples. Although adversarial training with PGD [94] algorithm can significantly enhance the robustness of the model, the computational complexity is very expensive when training on large-scale datasets.

To reduce the computational complexity of adversarial training, Shafahi et al. [127] introduced a speed adversarial training method by updating both the model parameters and images perturbations for each update step, and its training speed is 3–30 times than that of PGD [94]. Zhang et al. [128] found that the adversarial perturbation was only coupled with the first layer of the neural network, based on this, they proposed an adversarial training algorithm named YOPO(You Only Propagate Once), by focusing adversary computation only on the input layer of the neural network, experiment indicated that the training efficiency of YOPO was 4–5 times than that of original PGD training [94]. Besides, the research of Wong et al. [129] shown that the combination of FGSM [82] and random initialization in

TABLE 4 | Attack to machine learning in social network.

Authors	Year	Method	Dataset	Baseline	Attack type		Aspect		
					Black-box	White-box	SA	SD	RS
Gao et al. [133]	2018	DeepWordBug	Enron spam emails, IMDB	Projected FGSM, Random + DeepWordBug Transformer	✓		✓	✓	
Vijayaraghavan et al. [134]	2019	AEG	IMBA, AG News	DeepWordBug [133], NMT-BT [135]	✓		✓		
Ren et al. [136]	2020	Lage Scale Adversarial Attack	IMBA, Rotten Tomatoes Movie Reviews	FGSM [82], DeepFool [96], Textbugger [137]		✓	✓		
Li et al. [138]	2020	BERT-Attack	AG News, IMDB, Yelp, FAKE, SNLI, MNLI	TextFooler [139], Genetic attack [140]	✓		✓	✓	
Nuo et al. [141]	2020	WordChange	Ctrip, JD.com	TF-IDF, TextRank	✓		✓	✓	
Li et al. [142]	2020	CLARE	Yelp, AG News, MNLI, QNLI	TextFooler [139], TextFooler + LM, BERTAttack	✓		✓		
Garg et al. [143]	2020	BAE	Amazon Yelp, IMDB, MR	TextFooler [139]	✓		✓		
Jin et al. [139]	2020	TextFooler	AG News, FAKER, MR, Yelp, IMDB	Textbugger [137]	✓		✓		
Maheshwary et al. [144]	2021	Hard Label Attack	AG News, Yahoo Answers, MR, IMDB, Yelp, SNLI, MNLI	TextFooler [139], PSO [145], AEG [134]	✓		✓		
Yang et al. [146]	2017	Co-visitation attack	YouTube, eBay, Amazon, Yelp	Popular-item-attack, Random-item-attack		✓			✓
Fang et al. [147]	2018	Graph Poisoning Attack	MovieLens-100K, Amazon Instant Video	Co-visitation attack [146]		✓			✓
Christakopoulou et al. [148]	2019	Oblivious Recommender System Attack	MovieLens-100K, MovieLens-1M	—		✓			✓
Sun et al. [149]	2020	NIPA	Cora, Citeseer, Pumbed	Random, Preferential, PGA		✓			✓
Song et al. [150]	2020	PoisonRec	Steam, MovieLens-1M and Amazon	Popular Attack, Random Attack, Middle Attack, Power Item Attack, ConsLOP	✓				✓
Chang et al. [151]	2020	GF-Attack	Cora, Citeseer, Pubmed	Random, Degree, RL-S2V, Aclass	✓				✓
Fang et al. [152]	2020	TNA	Yelp, Amazon, Digital Music	PGA [153], SGLD [153]		✓			✓
Lin et al. [154]	2020	AUSH	MovieLens-100K, Amazon, FilmTrus	Random, Segment, Bandwagon, DCGAN		✓			✓
Fan et al. [155]	2021	CopyAttack	MovieLens-10M & Flixster, MovieLens-20M & Netflix	RL-Generative, RandomAttack, TargetAttack	✓				✓
Zhan et al. [156]	2021	BBGA	Cora, Citesser, Cora-ML	DICE-BB, Random, Mettack, Aclass	✓				✓
Finkelshtein et al. [157]	2021	Single-Node Attack	Cora, CiteSeer, PubMed, Twitter-Hateful-Users	EdgeGrad	✓	✓			✓
Huang et al. [158]	2021	Poisoning Attack	Movielens-100K, Movielens-1M, Last.fm	Random, Bandwagon, MF		✓			✓
Wu et al. [159]	2021	TrialAttack	Movielens-100K, Movielens-1M, FilmTrust	Random, Average, PGA [153], TNA [152], AUSH [154]		✓			✓

adversarial training can significantly reduce training cost while achieving similar effects to original PDG training [94].

3.2.3.2 Certified Training

Gowal et al. [130] proved that the robustness to PGD [94] attack was not a true measure of robustness. They focus on research on formal verification, and they proposed a neural network verified training algorithm named IBP (Interval Bound Propagation). Although the IBP algorithm is not only computationally cheaper but also significantly reduce the verified error rate, its training process is unstable, especially in the initial stages of training, to enhance the stability of IBP. Zhang et al. [131] combined the IBP [130] algorithm and the tight linear relaxation algorithm named CROWN [132], and proposed a verified training algorithm named IBP-CROWM. The experiment results shown that both standard errors and verified errors of IBP-CROWN were outperforming than IBP [130].

4 SECURITY OF MACHINE LEARNING IN SOCIAL NETWORKS

Most of the existing researches related to adversarial examples are focusing on the field of image classification. However, the generation of adversarial examples in social networks needs to process data like text and graph, unlike images, text and graph are discrete in feature distribution, which makes it more difficult to craft adversarial examples with text or graph. In this section, we mainly review the researches of adversarial examples in sentiment analysis (SA), spam detection (SD), and recommendation systems (RS), as well as some researches on question and

TABLE 5 | Defend against to adversarial examples in social network.

Authors	Year	Method	Dataset	Baselines	Attacks	Aspect		
						SA	SD	RS
Pruthi et al. [160]	2019	Robust Word Recognition	SST, IMBA, Stanford Sentiment Treebank	data augmentation [154], adversarial training [82]	Swap, Drop, Keyboard, Add	✓	✓	
Jia et al. [161]	2019	Certified Robustness Rraining	IMDB, SNLI	Standard Training, Data Augmentation	Genetic attack [140]	✓		
Zhou et al. [162]	2019	DISP	SST-2, IMDB	Adversarial Data Augmentation (ADA), Adversarial Training (AT), Adversarial Training (AT)	Insertion, Deletion, Swap, Random, Embed	✓		
Si et al. [163]	2020	AMDA	SST-2, IMDB	Adversarial Data Augmentation (ADA)	TextFooler [139], PWWS [164]	✓		
Wang et al. [165]	2020	MUDE	Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993)	Enchant 3 spell checker, scRNN	Permutation, Insertion, Deletion, Substitution	✓	✓	
Shi et al. [166]	2020	Transformers Robustness Verify	Yelp, SST	IBP	—	✓		
Ye et al. [167]	2020	Safer	IMDB, Amazon	Certified Robustness Rraining [161], IBP	Genetic attack [140]	✓		
Mozes et al. [168]	2020	FGWS	SST-2, IMDB	DISP [162]	Genetic attack [140], PWWS [164]	✓		
Zeng et al. [169]	2021	RanMASK	AG News, SST-2	Safer [167]	TextFooler [139], Bert-Attack [138], DeepWordBug [133]	✓		
Wang et al. [170]	2021	TextFirewall	IMBA, Yelp	Adversarial Training, Spelling Check and Recovery (SCR), RSE	Deepwordbug [133], Genetic attack [140], PWWS [164]	✓	✓	
Karimi et al. [171]	2021	BAT	SemEval 2014 task 4, SemEval 2016 task 5	BERT [172]	Gradient attack [173]	✓		
Du et al. [174]	2019	FNCF	Movielens-100K, Movielens-1M	Distillation [175]	C&W [88]			✓
Tang et al. [176]	2019	AMR	Pinterest, Amazon	POP, MF-BPR, DUIF, VBPR	FGSM [82]			✓
Manotumruksa et al. [177]	2020	SAO	MovieLens, Beauty, Video, Foursquare, Brightkite, Yelp	MostPop, BPR, APR, SASRec, ASASRec	—			✓
Li et al. [178]	2020	SACRA	Yelp, Foursquare	WRMF, MMMF, BPRMF, CofiRank, CLiMF, USG, GeoMF, etc.	FGSM [82]			✓
Wang et al. [179]	2020	ATMBPR	Movielens-100K, Yelp	BPR, CDAE, MPR, AMF, MLP, NeuMF, LRML, JRL, etc.	FGSM [82]			✓
Shahrasbi et al. [180]	2020	Semi-Supervised Attack Detection	Instacart grocery	LSTM	—			✓
Wu et al. [181]	2021	APT	FilmTrust, MovieLens-100K, MovieLens-1M, Yelp	Adversarial Training (AT), PCMF	AUSH [154], TNA [152], PGA [153]			✓
Yi et al. [182]	2021	DAVE	Yelp, Digital Music, MovieLens-1M, MovieLens-100K, Pinterest	NeuMF, CDAE, CFGAN, APR, ACAE, AVB, VAEGAN, CVAE-GAN, RecVAE	AAE			✓

answer robot and neural machine translation. Since the data used in sentiment analysis and spam detection are both texts, they are similar in the generation and defense of examples, we will review them in one subsection, due to the relative lack of research on question and answering robots and neural machine translation, we will review them in one subsection. **Table 4** and **Table 5** has shown some algorithms in sentiment analysis, spam detection, and machine translation of adversarial example generation and defense studies in recent years.

4.1 Security in Sentiment Analysis and Spam Detection

4.1.1 Adversarial Attacks

The goal of the attack against sentiment analysis and spam detection system is to craft a text x' that is semantically similar to the original text x but can mislead the target classifier. It can be expressed as:

$$\min_{x'} S(x, x') \quad \text{s.t.} \quad F(x) \neq F(x') \tag{5}$$

where function $S(\cdot)$ is used to compute the semantic similarity between x and x', $F(\cdot)$ is the target model.

The adversarial example generation algorithm for texts is mainly by finding the keywords in the whole sentence that have a greater impact on the classification results and then adding perturbation to these keywords.

Gao et al. [133] proposed an effectively black-box text adversarial example generate method named DeepWordBug, and they introduced temporal tail score (TTS) and temporal score (TS) to evaluate the importance of words in the sentence. According to the authors, firstly, they calculate the TTS and TS by

querying the output of the target model after shielding some words, and then combine the value of TTS and TS to calculate the importance of every word in the whole sentence. It can be expressed as:

$$\begin{aligned} \mathrm{TS}(x_i) &= F(x_1, x_2, \ldots, x_{i-1}, x_i) - F(x_1, x_2, \ldots, x_{i-1}) \\ \mathrm{TTS}(x_i) &= F(x_i, x_{i+1}, x_{i+2}, \ldots, x_n) - F(x_{i+1}, x_{i+2}, \ldots, x_n) \\ Score(x_i) &= \mathrm{TS}(x_i) + \lambda(\mathrm{TTS})(x_i) \end{aligned} \quad (6)$$

where, $F(\cdot)$ is target machine learning model, and x_i is the i-th word in the sentence. Finally, they modify some characters in the keywords to generate text adversarial examples. Experiments have proved that although DeepWordBug can generate text adversarial examples with a high success rate, it will introduce grammatical errors and can be easily defended by grammar detection tools.

Vijayaraghavan et al. [134] proposed an Adversarial Examples Generator (AEG) based on reinforcement learning to craft none-target text adversarial examples, according to authors, they evaluated the effectiveness of the AEG algorithm on two target sentiment analysis convolutional neural networks: CNN-Word and CNN-Char, the experiment showed that the AEG model was able to fool the target sentiment analysis models with high success rates while preserving the semantics of the original text.

Li et al. [138] also proposed a word-level text adversarial examples generate algorithm named BERT-Attack. According to the authors, firstly, different from [133], they tried to find the vulnerable words in a sentence by masking each word and the query the target model for correct label. Then they used a pre-trained model Bert to replace vulnerable vocabulary with grammatically correct and semantically similar words. The process of calculating the vulnerability of each word can be expressed as:

$$I_{w_i} = F(S) - F(S_{\setminus w_i}) \quad (7)$$

where $F(\cdot)$ is target machine learning model, $S = [w_0, \ldots, w_i \ldots,]$ is the input text, and $S_{\setminus w_i} = [w_0, \ldots, w_{i-1}, [\mathrm{MASK}], w_{i+1}, \ldots,]$ is the text that replace the w_i with $[MASK]$.

Based on multiple modification strategies, Nuo et al. [141] proposed a black-box Chinese text adversarial example generate method named WordChange. Similar to the algorithm for calculating *TS* in **Eq. 6**, they search for keywords by gradually deleting a certain vocabulary in the sentence and then querying whether the output of the model has changed, and then applying "insert" and "swap" strategies on these keywords, thereby generating Chinese text adversarial examples that can fool the machine learning model.

Jin et al. [139] proposed a text adversarial examples generate method named TextFooler, which craft adversarial examples by finding the words that have the greatest impact on the output of the target model in the whole sentence and replacing it with words that share similar meanings with the original words. Although the replaced words in the adversarial examples generated by TextFooler are similar to the original words, it may not fit overall sentence semantics. To make the text adversarial examples more natural and free of grammatical errors, Similar to [138] Garg et al. [143] proposed a text adversarial example generation algorithm named BAE. According to the authors, firstly, they calculate the importance of each word in the text, and then choose a certain word and replace it with MASK or insert a MASK adjacent to it according to the importance of each word. Finally, they use the pre-trained language model BERT-MLM [183] to replace the mask with a word that fits the context. Similar to BAE [143], Li et al. [142] also introduced a pre-trained language model based text adversarial example generation algorithm named CLARE (ContextuaLized AdversaRial Example). Compared with BAE [143], CLARE has richer attack strategies and can generate text adversarial examples with varied lengths. Experiment showed that the text adversarial examples generated by BAE [143] and CLARE [142] were more fluent, natural and grammatical.

To attack text neural networks in hard label black-box setting where the attacker can only get the label output by the target model, Maheshwary et al. [144] utilized a Genetic Algorithm (GA) to craft text adversarial examples that share similar semantics with the original text. Experimental results show that on sentiment analysis tasks, their method can generate text adversarial examples with a higher success rate using smaller perturbation than algorithms such as TextFooler [139], PSO (Particle Swarm Optimization) [145], AEG [134], etc.

In addition, different from generating examples by replacing some words or characters in the text, Ren et al. [136] introduced a white-box text adversarial example generate model to generate text adversarial examples on large scale without inputting the original text, and their model is composed of a vanilla VAE-based generator and a series of discriminators. The generator is used to generate text adversarial examples, and the discriminators are used to make the adversarial examples of different labels crafted by *G* look more realistic. Their experiment showed that the proposed model could deceive the target neural network with high confidence.

4.1.2 Defense Against Adversarial Attacks

The current defense strategies for text adversarial examples are mainly divided into two aspects: adversarial example processing and model robustness enhancement. The adversarial example processing method mainly includes identifying the adversarial examples by detecting the misspellings and unknown words contained in the text and performing partial vocabulary replacement of the adversarial examples to convert them into clean text; The model robustness enhancement method enhances the model's defense ability against adversarial examples through methods such as adversarial training and formal verification.

4.1.2.1 Adversarial Example Processing

Adversarial example detection is an important way to detect adversarial examples in sentiment analysis and spam detection. Pruthi et al. [160] proposed RNN-based word recognizers to detect adversarial examples by detecting misspellings in the sentences, but it is hard to defend word-level attacks. By calculating the influence of words in texts, Wang et al. [170] proposed a general text adversarial examples detection algorithm named TextFirewall. They used it to defend the adversarial attacks from Deepwordbug [133], Genetic attack [140], and PWWS (Probability Weighted Word Saliency) [164], and the

average attack success rate decreased on Yelp and IMDB are 0.73 and 0.63%, respectively. Mozes et al. [168] also proposed adversarial example detection method named FGWS (Frequency-Guided Word Substitutions), and they tried to detect text adversarial example with the frequency properties of adversarial words and achieved a higher F1 score than DISP [162] in SST-2 and IMDB dataset. Besides, Wang et al. [165] also proposed a framework named MUDE (Multi-Level Dependencies) to detect adversarial word by taking advantage of both character and word level dependencies.

Zhou et al. [162] also introduced a framework named DISP (Discriminate Perturbations) to transform the text adversarial examples into clean text data. According to the authors, firstly, they identified the perturbed tokens in the input text with a perturbation discriminator, and then replaced the perturbed token with an embedding estimator. Finally, they recovered these tokens into a clean text with a KNN(k-nearest neighbors) algorithm. The experiment indicated that the DISP was outperforming the Adversarial Data Augmentation (ADA), Adversarial Training (AT), and Spelling Correction (SC) in terms of the efficiency and semantic integrity of the text adversarial examples.

4.1.2.2 Model Robustness Enhancement

As mentioned above, the algorithms to enhance the robustness of the NLP model mainly include adversarial training and formal verification. In terms of adversarial training, Si et al. [163] introduced a method named AMDA (Adversarial and Mixup Data Augmentation) to cover the larger proportion of the attack space during the process of adversarial training by crafting large amount of augmented training adversarial examples and feeding them to the machine learning model. They used AMDA to defend against attacks from PPWS [164] and TextFooler [139] on the data sets SST-2, AG News and IMBD, and achieved significant robustness gains in both Targeted Attack Evaluation (TAE) and Static Attack Evaluation (SAE). For large pre-training model BERT, Karimi et al. [171] introduced a method named BAT to fine-tuned the BERT model by using normal and adversarial text at the same time to obtain a model with better robustness and generalization ability. The experiment indicated that the BERT model trained with BAT was more robust than the traditional BERT model in aspect-based sentiment analysis task.

In terms of formal verification, Jia et al. [161] proposed certified robustness training by using interval bound propagation to minimize the upper bound on the worst-case loss. Facts have proved that this method can effectively resist word substitution attacks from Genetic attack [140]. Shi et al. [166] proposed a transformers robustness verify method to verify the robustness transformers network, compared with the interval boundary propagation algorithm, their method could achieve much tighter certified robustness bounds. Ye et al. [167] proposed a structure-free certified robustness framework named SAFER, which only needs to query the output of the target model when verifying its robustness, so it can be applied to neural network models with any structure, but it is only suitable for word substitutions attacks. Zeng et al. [169] proposed a smoothing-based certified defense method named RanMASK, it could defend against both defense method against both the character and word substitution-based attacks.

4.2 Security in Social Recommendation System

4.2.1 Adversarial Attacks

The poisoning attack affects the recommendation list of the target recommendation system by feeding fake users into the recommendation system, which has occupies the dominant position in adversarial attacks against machine learning-based recommendation systems.

Yang [146] performed promotion and demotion poisoning attacks by taking attacks as constrained linear optimization problems, and they verified their method on real social network recommendation systems, such as YouTube, eBay, Amazon, Yelp, etc., and achieved a high success attack rate. Similar to Yang [146], Fang et al. [147] also formulates the poisoning attacks to graph-based recommendation system as an optimization problem, and performs poison attacks by solving these optimization problems. Christakopoulou et al. [148] proposed a two-step white-box poisoning attack framework to fool the machine learning-based recommendation system. Firstly, they utilize a GAN network to generate faker users, and then craft the profiles of fake users with projected gradient method. Fang et al. [152] performed attacks to matrix factorization based social recommendation system by optimizing the ratings of a fake user with a subset of influential users. Huang et al. [158] also tried to poison the deep learning based recommendation system by maximizing the hit rate of a certain item appearance in the top-n recommendation list predicted by target recommendation system.

To effectively generate fake user profile with strong attack power for poisoning attacks, Wu et al. [159] introduced a flexible poisoning framework named TrialAttack, the TrialAttack is based on GAN network and consists of three parts: generator G, influence module I, and discriminator D, the generator is used to generate fake user profile that is close to the real user and has attack influence, the influence model is used to guide the generator to generate fake users profile with greater influence, and the discriminator is used to distinguish the faker profiles generated by the generator from the real.

The above attack methods are all white-box-based attack algorithms, that is, the attacker needs to fully understand the parameter information of the target model, but this is unrealistic to the recommendation system in the real social network. In terms of black-box attacks, Fan et al. [155] introduced a framework named CopyAttack to perform a black-box adversarial attack to recommendation system in social network, they used reinforcement learning algorithms to select users from the original domain and inject them into the target domain to improve the hit rate of the target item in the top-n recommendation list.

Song et al. [150] proposed an adaptive data poisoning framework named PoisonRec, it leverages reinforcement learning to inject false user data into the recommendation

system, which can automatically learn effective attack strategies for various recommendation systems with very limited knowledge.

To attack the graph embedding models with limited knowledge, Chang et al. [151] introduced an adversarial attacker framework named GF-Attack, which formulated the graph neural network as a general graph signal processing with corresponding graph filters, and then attacked the graph filters through the feature matrix and adjacency matrix. To minimize the modification of the original graph data in the attack, Finkelshtein et al. [157] introduced a single-node attack to perform adversarial attack to graph neural networks, which could fool the target model by only modifying a single arbitrary node in the graph.

4.2.2 Defense Against Adversarial Attacks

The current defense algorithms for recommendation systems are mainly divided into two aspects: model robustness enhancement and abnormal data detection. Among them, model robustness enhancement is based on adversarial training, and abnormal data detection improves the robustness of the recommendation systems by recognizing pollution data.

In terms of adversarial training, Tang et al. [176] proposed an adversarial training method named AMR (Adversarial Multimedia Recommendation) to defend against adversarial attack. According to the authors, the process of adversarial training could be interpreted as playing a minimax game. On the one hand, continuously generate perturbations that can maximize the loss function of target model. On the other hand, continuously optimize the parameters of target model to identify these perturbations.

By combining knowledge distillation with adversarial training, Du et al. [174] produced a more robust collaborative filtering model based on neural network to defend against adversarial attacks. The experiments indicated that their model can effectively enhance the robustness of the recommendation system under the attack of the C&W [88] algorithm.

Manotumruksa et al. [177] also proposed a recommendation system robust enhancement framework named SAO (Sequential-based Adversarial Optimization) to enhance the robustness of the recommendation system by generating a sequence of adversarial perturbations and adding it into the training set during the training process.

Li et al. [178] introduced a framework named SACRA (Self-Attentive prospective Customer Recommendation Framework) to perform prospective customer recommendation. Similar to Manotumruksa [177], the SACRA enhances its robust by adding adversarial perturbation into the training set dynamically to make the recommend system immune to these perturbations.

Wu et al. [181] used the influence function proposed by Koh et al. [184] to craft fake users, and then injected these fake users into the training set to enhance the robustness of the recommendation system. They named their method as adversarial poisoning training (APT), they used five poisoning attack algorithms to evaluate the effectiveness of the APT. The experiment indicated that APT can enhance the robustness of the recommendation system effectively.

By combining the advantages of adversarial training and VAE (Variational Auto-Encoder), Yi et al. [182] proposed a robust recommendation model named DAVE (Dual Adversarial Variational Embedding), which is composed of User Adversarial Embedding (UserAVE), User Adversarial Embedding (ItemAVE) and Neural Collaborative Filtering Network, UserAVE and ItemAVE generate user and item embedding according to user interaction vector and item interaction vector, respectively. Then the user and item embedding are fed into the Collaborative Filtering Network to predict and recommend results. During the training process of the DAVE, it reduces the impact of adversarial perturbation by adaptively generating a unique embedding distribution for each user and item.

In terms of abnormal data detection, Shahrasbi et al. [180] proposed a GAN-based pollution data detection method. According to the authors, firstly, they convert the clean user session data to embedding sequences with a Doc2Vec language model. Then, during the training process of GAN, the generator is trained to learn the distribution of real embedding sequences, and the discriminator is trained to learn the distinguish the real embedding sequences and the sequence generated by the generator. Based on this, when the training of GAN network is completed, the pollution data can be identified from the whole dataset.

4.3 Security in Other Aspects of Social Networks

In this subsection, we mainly review some research on adversarial examples from the aspects of question and answer robot and neural machine translation.

4.3.1 Question and Answer Robot

Xue et al. [185] introduced a text adversarial example craft method named DPAGE (Dependency Parse-based Adversarial Examples Generation) to perform black-box adversarial attack to Q&A robots. They extract the keywords of the sentence based on the dependency relation of the sentences and then replace these keywords with the adversarial word that are similar to these keywords to craft adversarial questions. They evaluated the performance of DPAGE with two Q&A robots: DrQA and Google Assistant, and the results shown that the adversarial examples crafted by DPAGE cannot affect both the correct answer and the top-k candidate answers output by the Q&A robot. Similar to [185] Deng et al. [186] proposed a method named APE (Attention weight Probability Estimation) to extract keywords from the dialogue and fool the target Q&A system by replaced these keywords with synonyms. The experiment results show that their method can attack the Q&A system with a high success rate.

4.3.2 Neural Machine Translation

The NMT model is also vulnerable to attacks from adversarial examples. Ebrahimi et al. [187] proposed a white-box gradient-based optimization text adversarial example generation method to perform targeted adversarial attacks to NMT models. The

experiment results have shown that their method can attack the target NMT model with a high success rate, and the robustness of the model can improve significantly after robust training. Besides, in the study of poison attacks, Wang et al. [188] can successfully implement the poison attack by injecting only 0.02% of the total data into the data set.

To enhance the robustness of the NMT model, Cheng et al. [189] craft text adversarial examples with a white-box gradient-based method and then used it to enhance the robustness of the model. Experiments on English-German and Chinese-English translation tasks have shown that their method can significantly improve the robustness and performance of the NMT model. In another study by Cheng et al. [190], they also try to enhance the robustness of the NMT models by augmenting the training data with an adversarial augmentation technique.

5 DISCUSSION AND CONCLUSION

5.1 Discussion

Although the generation and defense algorithms of adversarial examples have made great achievements on unstructured data in social networks, there are still many key issues that have not been resolved.

5.1.1 Constraints for Attacks on Real Systems

Many adversarial example generation algorithms in social networks do not consider the restrictions on attacks on real systems. In terms of text adversarial generation, many studies [133, 138, 139, 141] try to get the keywords in the sentence by frequently querying the target model, however, the action of the frequent query is easy to be detected and defended when the attack is performed on the real system. In terms of adversarial example generation in the recommendation system, the attacker poisons the recommendation system by modifying the edge and attribute information of some nodes in the social network graph [146, 147, 150, 151], however, in the real social network, the node that the attacker chooses to modify may be a node that is not controlled by the attacker. Therefore, in the subsequent research on adversarial example generation algorithms in the field of social networks, more consideration should be given to the limitations in real scenarios.

5.1.2 The Security of Social Network Data

To adapt to the complex and changeable user structure on social networks, the rapid change and short timeliness of cyber language, the AI models applied to social networks need to frequently fine-tune its parameters based on real data from social networks. Therefore, poisoning attacks must be effectively avoided during the process of online training. Since adversarial examples are usually difficult to find visually, on the one hand, there is currently little research on adversarial examples detection for unstructured data such as graphs. On the other hand, with the continuous evolution of attack methods, the existing data enhancement and cleaning technologies cannot effectively detect malicious data in all data. Therefore, how to accurately detect poisoning data in social networks will become a focus of future research.

5.1.3 Robustness Evaluation of Social Network Models

Due to the poor interpretability of machine learning algorithms, it is difficult to analyze and prove the robustness of machine learning mathematically. Therefore, the current robustness evaluation of machine learning algorithms mainly depends on the defensive ability of specific adversarial attacks, however, the robustness conclusions obtained by this method are difficult to apply to the latest attack algorithms. In the field of computer vision, some researches [130, 131] have tried to use formal verification to analyze the robustness of machine learning algorithms. In terms of social networks, some researches [161, 166, 167, 169] also try to use formal verification algorithms to analyze the robustness of text classification machine learning models, but it is great affected by the model structure and data types, in terms of recommendation systems, the research on the robustness analysis for machine learning algorithm is still blank. Therefore, the robustness analysis for machine learning algorithm will also be a research focus in social networks.

5.2 Conclusion

Although machine learning algorithms have made significant developments in many fields, it cannot be ignored that machine learning algorithms are vulnerable to attacks from adversarial examples. That is, adding perturbations that are not detectable by the human eye to the original data may cause the machine learning algorithm to make a completely different output with a high probability. In this paper, we review the application of machine learning algorithms in the field of social networks from aspects of sentiment analysis, recommendation systems, and spam detection, as well as the research progress of the generation of adversarial examples and defense algorithms in social networks.

Although the data processed by machine learning models that are used in social networks is usually unstructured data such as text or graphs, the adversarial example generation algorithm for images in the field of computer vision is also applicable to unstructured data such as text and graphs after extension. Therefore, how to use machine learning algorithms to implement various functions in social networks while ensuring the robustness of the algorithm itself is one of the hotspots for studying. Besides, to improve the robustness of machine learning algorithms in the field of the social network, in terms of adversarial example generation, more focus should be put on the adversarial example generation algorithms that can be applied to real online social network machine learning models, so as to enhance the robustness of online machine learning models. In terms of adversarial example defense, while strengthening robustness against specific attacks, more research on active defense algorithms such as certified training should also be carried out to defend against adversarial attacks.

AUTHOR CONTRIBUTIONS

SG: Investigation, Analysis of papers, Drafting the manuscript, Review; XL: Review, Editing; ZM: Review, Editing.

FUNDING

This work was supported by National Key R&D Program of China (Grant No. 2020AAA0107700), Shenzhen Fundamental Research Program (Grant No. 20210317191843003), the Shaanxi Provincial Key R&D Program (Grant No. 2021ZDLGY05-01), the Natural Science Basic Research Plan in Shaanxi Province of China (Grant No. 2020JQ-214).

REFERENCES

1. Clark JL, Algoe SB, Green MC. Social Network Sites and Well-Being: the Role of Social Connection. *Curr Dir Psychol Sci* (2018) 27(1):32–7. doi:10.1177/0963721417730833
2. Liu C, Zhou N, Zhan X-X, Sun G-Q, Zhang Z-K. Markov-based Solution for Information Diffusion on Adaptive Social Networks. *Appl Maths Comput* (2020) 380:125286. doi:10.1016/j.amc.2020.125286
3. Li L, Zhang J, Liu C, Zhang H-T, Wang Y, Wang Z. Analysis of Transmission Dynamics for Zika Virus on Networks. *Appl Maths Comput* (2019) 347: 566–77. doi:10.1016/j.amc.2018.11.042
4. Zhan X-X, Liu C, Zhou G, Zhang Z-K, Sun G-Q, Zhu JJH, et al. Coupling Dynamics of Epidemic Spreading and Information Diffusion on Complex Networks. *Appl Maths Comput* (2018) 332:437–48. doi:10.1016/j.amc.2018.03.050
5. Han X, Yao M, Liu H-C, Deb D, Liu H, Tang J-L, et al. Adversarial Attacks and Defenses in Images, Graphs and Text: A Review. *Int J Automation Comput* (2020) 17(2):151–78. doi:10.1007/s11633-019-1211-x
6. Guo S, Zhao J, Li X, Duan J, Mu D, Xiao J. A Black-Box Attack Method against Machine-Learning-Based Anomaly Network Flow Detection Models. *Security Commun Networks* (2021) 2021:1–13. doi:10.1155/2021/5578335
7. Zhang WE, QuanSheng Z, F Alhazmi AA, Li C. *Generating Textual Adversarial Examples for Deep Learning Models: A Survey* (2019). arXiv preprint arXiv:1901.06796.
8. Yao Q, Carlini N, Cottrell G, Goodfellow I, Raffel C. Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. In: 36th International Conference on Machine Learning, ICML 2019. Long Beach, CA, United states: PMLR (2019). p. 5231–40.
9. Li S, Jiang L, Wu X, Han W, Zhao D, Wang Z. A Weighted Network Community Detection Algorithm Based on Deep Learning. *Appl Maths Comput* (2021) 401:126012. doi:10.1016/j.amc.2021.126012
10. Li S, Zhao D, Wu X, Tian Z, Li A, Wang Z. Functional Immunization of Networks Based on Message Passing. *Appl Maths Comput* (2020) 366:124728. doi:10.1016/j.amc.2019.124728
11. Han W, Tian Z, Huang Z, Li S, Jia Y. Topic Representation Model Based on Microblogging Behavior Analysis. *World Wide Web* (2020) 23(6):3083–97. doi:10.1007/s11280-020-00822-x
12. Nie Y, Jia Y, Li S, Zhu X, Li A, Zhou B. Identifying Users across Social Networks Based on Dynamic Core Interests. *Neurocomputing* (2016) 210:107–15.
13. Al-Smadi M, Qawasmeh O, Al-Ayyoub M, Jararweh Y, Gupta B. Deep Recurrent Neural Network vs. Support Vector Machine for Aspect-Based Sentiment Analysis of Arabic Hotels' Reviews. *J Comput Sci* (2018) 27: 386–93. doi:10.1016/j.jocs.2017.11.006
14. Hitesh MSR, Vaibhav V, Kalki YJA, Kamtam SH, Kumari S. Real-time Sentiment Analysis of 2019 Election Tweets Using Word2vec and Random forest Model. In: 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT). IEEE (2019). p. 146–51. doi:10.1109/icct46177.2019.8969049
15. Long F, Zhou K, Ou W. Sentiment Analysis of Text Based on Bidirectional Lstm with Multi-Head Attention. *IEEE Access* (2019) 7:141960–9. doi:10.1109/access.2019.2942614
16. Djaballah KA, Boukhalfa K, Omar B. Sentiment Analysis of Twitter Messages Using Word2vec by Weighted Average. In: 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS). IEEE (2019). p. 223–8. doi:10.1109/snams.2019.8931827
17. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed Representations of Words and Phrases and Their Compositionality. In: *Advances in Neural Information Processing Systems* (2013), Lake Tahoe, NV, United states. p. 3111–9.
18. Ho J, Ondusko D, Roy B, Hsu DF. Sentiment Analysis on Tweets Using Machine Learning and Combinatorial Fusion. In: 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE (2019). p. 1066–71. doi:10.1109/dasc/picom/cbdcom/cyberscitech.2019.00191
19. Liu N, Shen B, Zhang Z, Zhang Z, Mi K. Attention-based Sentiment Reasoner for Aspect-Based Sentiment Analysis. *Human-centric Comput Inf Sci* (2019) 9(1):1–17. doi:10.1186/s13673-019-0196-3
20. Yao F, Wang Y. Domain-specific Sentiment Analysis for Tweets during Hurricanes (Dssa-h): A Domain-Adversarial Neural-Network-Based Approach. *Comput Environ Urban Syst* (2020) 83:101522. doi:10.1016/j.compenvurbsys.2020.101522
21. Umer M, Ashraf I, Mehmood A, Kumari S, Ullah S, Sang Choi G. Sentiment Analysis of Tweets Using a Unified Convolutional Neural Network-long Short-term Memory Network Model. *Comput Intelligence* (2021) 37(1): 409–34. doi:10.1111/coin.12415
22. Conneau A, Schwenk H, Barrault L, Lecun Y. Very Deep Convolutional Networks for Text Classification. In: 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017-Proceedings of Conference. Valencia, Spain (2016). p. 1107–16. doi:10.18653/v1/e17-1104
23. Cliche M. Bb_twtr at Semeval-2017 Task 4: Twitter Sentiment Analysis with Cnns and Lstms. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017. Vancouver, Canada: Association for Computational Linguistics (2017). p. 573–80. doi:10.18653/v1/S17-2094
24. Lv Y, Wei F, Cao L, Peng S, Niu J, Yu S, et al. Aspect-level Sentiment Analysis Using Context and Aspect Memory Network. *Neurocomputing* (2021) 428: 195–205. doi:10.1016/j.neucom.2020.11.049
25. Rawat R, Mahor V, Chirgaiya S, Shaw RN, Ghosh A. Sentiment Analysis at Online Social Network for Cyber-Malicious post Reviews Using Machine Learning Techniques. *Computationally Intell Syst their Appl* (2021) 950: 113–30. doi:10.1007/978-981-16-0407-2_9
26. Wang Y, Huang M, Zhu X, Zhao L. Attention-based Lstm for Aspect-Level Sentiment Classification. In: Proceedings of the 2016 conference on empirical methods in natural language processing (2016). p. 606–15. doi:10.18653/v1/d16-1058
27. Pontiki M, Galanis D, Papageorgiou H, Androutsopoulos I, Manandhar S, Al-Smadi M, et al. Semeval-2016 Task 5: Aspect Based Sentiment Analysis. In: International workshop on semantic evaluation (2016). p. 19–30. doi:10.18653/v1/s16-1002
28. Fan W, Yao M, Li Q, Yuan H, Zhao E, Tang J, et al. Graph Neural Networks for Social Recommendation. In: The World Wide Web Conference (2019). p. 417–26. doi:10.1145/3308558.3313488
29. van den Berg R, Kipf TN, Welling M. *Graph Convolutional Matrix Completion* (2017). arXiv preprint arXiv:1706.02263.
30. Fan W, Li Q, Cheng M. Deep Modeling of Social Relations for Recommendation. In: *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).
31. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S. Neural Collaborative Filtering. In: Proceedings of the 26th international conference on world wide web (2017). p. 173–82. doi:10.1145/3038912.3052569
32. Gui T, Liu P, Zhang Q, Zhu L, Peng M, Zhou Y, et al. Mention Recommendation in Twitter with Cooperative Multi-Agent Reinforcement Learning. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (2019). p. 535–44. doi:10.1145/3331184.3331237
33. Guo Z, Wang H. A Deep Graph Neural Network-Based Mechanism for Social Recommendations. *IEEE Trans Ind Inform* (2020) 17(4):2776–83. doi:10.1109/tii.2020.2986316
34. Massa P, Avesani P. Controversial Users Demand Local Trust Metrics: An Experimental Study on Epinions. Com Community. *AAAI* (2005) 5: 121–6. doi:10.5555/1619332.1619354
35. Hegde S, Satyappanavar S, Setty S. Restaurant Setup Business Analysis Using Yelp Dataset. In: 2017 International Conference on Advances in Computing,

Communications and Informatics (ICACCI). IEEE (2017). p. 2342–8. doi:10.1109/icacci.2017.8126196
36. Yang X, Steck H, Guo Y, Liu Y. On Top-K Recommendation Using Social Networks. In: Proceedings of the sixth ACM conference on Recommender systems (2012). p. 67–74. doi:10.1145/2365952.2365969
37. Jamali M, Ester M. A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks. In: Proceedings of the fourth ACM conference on Recommender systems (2010). p. 135–42. doi:10.1145/1864708.1864736
38. Guo G, Zhang J, Yorke-Smith N. Trustsvd: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings. In: Proceedings of the AAAI Conference on Artificial Intelligence, volume 29 (2015).
39. Yang B, Lei Y, Liu J, Li W. Social Collaborative Filtering by Trust. *IEEE Trans Pattern Anal Mach Intell* (2016) 39(8):1633–47. doi:10.1109/TPAMI.2016.2605085
40. Sedhain S, Menon AK, Scott S, Xie L. Autorec: Autoencoders Meet Collaborative Filtering. In: Proceedings of the 24th international conference on World Wide Web (2015). p. 111–2.
41. Huang Z, Xu X, Zhu H, Zhou M. An Efficient Group Recommendation Model with Multiattention-Based Neural Networks. *IEEE Trans Neural Netw Learn Syst.* (2020) 31(11):4461–74. doi:10.1109/tnnls.2019.2955567
42. Wang H, Dong M. Latent Group Recommendation Based on Dynamic Probabilistic Matrix Factorization Model Integrated with Cnn. *J Comput Res Dev* (2017) 54(8):1853. doi:10.7544/issn1000-1239.2017.20170344
43. Tran LV, Pham T-AN, Tay Y, Liu Y, Gao C, Li X. Interact and Decide: Medley of Sub-attention Networks for Effective Group Recommendation. In: Proceedings of the 42nd International ACM SIGIR conference on research and development in information retrieval (2019). p. 255–64.
44. Cao D, He X, Miao L, An Y, Yang C, Hong R. Attentive Group Recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (2018). p. 645–54. doi:10.1145/3209978.3209998
45. Pan Y, He F, Yu H. A Correlative Denoising Autoencoder to Model Social Influence for Top-N Recommender System. *Front Comput Sci* (2020) 14(3): 1–13. doi:10.1007/s11704-019-8123-3
46. Wu Y, DuBois C, Zheng AX, Ester M. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In: Proceedings of the ninth ACM international conference on web search and data mining (2016). p. 153–62. doi:10.1145/2835776.2835837
47. Wang M, Wu Z, Sun X, Feng G, Zhang B. Trust-aware Collaborative Filtering with a Denoising Autoencoder. *Neural Process Lett* (2019) 49(2):835–49. doi:10.1007/s11063-018-9831-7
48. Zheng Q, Liu G, Liu A, Li Z, Zheng K, Zhao L, et al. Implicit Relation-Aware Social Recommendation with Variational Auto-Encoder. *World Wide Web*. Springer (2021). p. 1–16.
49. Cantador I, Peter B, Kuflik T. Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (Hetrec2011). In: Proceedings of the fifth ACM conference on Recommender systems (2011). p. 387–8. doi:10.1145/2043932.2044016
50. Guo G, Zhang J, Yorke-Smith N. A Novel Bayesian Similarity Measure for Recommender Systems. In: Twenty-third international joint conference on artificial intelligence, (2013).
51. Guo G, Zhang J, Thalmann D, Yorke-Smith N. Etaf: An Extended Trust Antecedents Framework for Trust Prediction. In: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014). IEEE (2014). p. 540–7. doi:10.1109/asonam.2014.6921639
52. Chen C, Zhang M, Liu Y, Ma S. Social Attentional Memory Network: Modeling Aspect-And Friend-Level Differences in Recommendation. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (2019). p. 177–85.
53. Liang D, Krishnan RG, Hoffman MD, Jebara T. Variational Autoencoders for Collaborative Filtering. In: Proceedings of the 2018 world wide web conference (2018). p. 689–98. doi:10.1145/3178876.3186150
54. Ni J, Huang Z, Cheng J, Gao S. An Effective Recommendation Model Based on Deep Representation Learning. *Inf Sci* (2021) 542:324–42. doi:10.1016/j.ins.2020.07.038
55. Kim D, Park C, Oh J, Lee S, Yu H. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In: Proceedings of the 10th ACM conference on recommender systems (2016). p. 233–40. doi:10.1145/2959100.2959165
56. Huang Z, Xu X, Ni J, Zhu H, Wang C. Multimodal Representation Learning for Recommendation in Internet of Things. *IEEE Internet Things J* (2019) 6(6):10675–85. doi:10.1109/jiot.2019.2940709
57. Liu J, Wu C, Wang J. Gated Recurrent Units Based Neural Network for Time Heterogeneous Feedback Recommendation. *Inf Sci* (2018) 423:50–65. doi:10.1016/j.ins.2017.09.048
58. Xiao J, Ye H, He X, Zhang H, Wu F, Chua T-S. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI-17). Melbourne, VIC, Australia (2017). p. 3119–25. doi:10.24963/ijcai.2017/435
59. Zeng B, Shang Q, Han X, Zeng F, Zhang M. Racmf: Robust Attention Convolutional Matrix Factorization for Rating Prediction. *Pattern Anal Applic* (2019) 22(4):1655–66. doi:10.1007/s10044-019-00814-2
60. Khattar D, Kumar V, Varma V, Gupta M. Hram: A Hybrid Recurrent Attention Machine for News Recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018). p. 1619–22.
61. Zhang L, Liu P, Gulla JA. Dynamic Attention-Integrated Neural Network for Session-Based News Recommendation. *Mach Learn* (2019) 108(10):1851–75. doi:10.1007/s10994-018-05777-9
62. Tahmasebi H, Ravanmehr R, Mohamadrezaei R. Social Movie Recommender System Based on Deep Autoencoder Network Using Twitter Data. *Neural Comput Applic* (2021) 33(5):1607–23. doi:10.1007/s00521-020-05085-1
63. Behera DK, Das M, Swetanisha S. Predicting Users' Preferences for Movie Recommender System Using Restricted Boltzmann Machine. In: *Computational Intelligence in Data Mining*. Springer (2019). p. 759–69. doi:10.1007/978-981-10-8055-5_67
64. Polatidis N, GeorgiadisGeorgiadis CK, Pimenidis E, Mouratidis H. Privacy-preserving Collaborative Recommendations Based on Random Perturbations. *Expert Syst Appl* (2017) 71:18–25. doi:10.1016/j.eswa.2016.11.018
65. Salih Karakaşlı M, Aydin MA, Yarkan S, Ali B. Dynamic Feature Selection for Spam Detection in Twitter. In: International Telecommunications Conference. Springer (2019). p. 239–50.
66. Jain G, Sharma M, Agarwal B. Spam Detection in Social media Using Convolutional and Long Short Term Memory Neural Network. *Ann Math Artif Intell* (2019) 85(1):21–44. doi:10.1007/s10472-018-9612-z
67. Tajalizadeh H, Boostani R. A Novel Stream Clustering Framework for Spam Detection in Twitter. *IEEE Trans Comput Soc Syst* (2019) 6(3):525–34. doi:10.1109/tcss.2019.2910818
68. Zhao C, Xin Y, Li X, Zhu H, Yang Y, Chen Y. An Attention-Based Graph Neural Network for Spam Bot Detection in Social Networks. *Appl Sci* (2020) 10(22):8160. doi:10.3390/app10228160
69. Yang C, Harkreader R, Zhang J, Shin S, Gu G. Analyzing Spammers' Social Networks for Fun and Profit: a Case Study of Cyber Criminal Ecosystem on Twitter. In: Proceedings of the 21st international conference on World Wide Web (2012). p. 71–80.
70. Zhang Z, Hou R, Yang J. Detection of Social Network Spam Based on Improved Extreme Learning Machine. *IEEE Access* (2020) 8:112003–14. doi:10.1109/access.2020.3002940
71. Gao Y, Gong M, Xie Y, Qin AK. An Attention-Based Unsupervised Adversarial Model for Movie Review Spam Detection. *IEEE Trans multimedia* (2020) 23:784–96.
72. Quinn M, Olszewska JI. British Sign Language Recognition in the Wild Based on Multi-Class Svm. In: 2019 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE (2019). p. 81–6.
73. An J, Cho S. Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability. *Spec Lecture IE* (2015) 2(1):1–18.
74. Zhao C, Xin Y, Li X, Yang Y, Chen Y. A Heterogeneous Ensemble Learning Framework for Spam Detection in Social Networks with Imbalanced Data. *Appl Sci* (2020) 10(3):936. doi:10.3390/app10030936
75. Chen C, Zhang J, Chen X, Yang X, Zhou W. 6 Million Spam Tweets: A Large Ground Truth for Timely Twitter Spam Detection. In: 2015 IEEE international conference on communications (ICC). IEEE (2015). p. 7065–70. doi:10.1109/icc.2015.7249453
76. Sze-To A, Wong AKC. A Weight-Selection Strategy on Training Deep Neural Networks for Imbalanced Classification. In: International Conference Image

Analysis and Recognition. Springer (2017). p. 3–10. doi:10.1007/978-3-319-59876-5_1
77. Alom Z, Carminati B, Ferrari E. *A Deep Learning Model for Twitter Spam Detection*, 18. Online Social Networks and Media (2020). p. 100079. doi:10.1016/j.osnem.2020.100079A Deep Learning Model for Twitter Spam Detection*Online Soc Networks Media*
78. Swe MM, Myo NN. Fake Accounts Detection on Twitter Using Blacklist. In: 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS). IEEE (2018). p. 562–6. doi:10.1109/icis.2018.8466499
79. Neha MV, Nair MS. A Novel Twitter Spam Detection Technique by Integrating Inception Network with Attention Based Lstm. In: 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI). IEEE (2021). p. 1009–14. doi:10.1109/icoei51242.2021.9452825
80. Arora S, Li Y, Liang Y, Ma T, Risteski A. A Latent Variable Model Approach to Pmi-Based Word Embeddings. *Tacl* (2016) 4:385–99. doi:10.1162/tacl_a_00106
81. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, et al. Intriguing Properties of Neural Networks. 2nd International Conference on Learning Representations, ICLR 2014-Conference Track Proceedings (2013), Banff, AB, Canada: Elsevier Inc.
82. Goodfellow IJ, Shlens J, Szegedy C. *Explaining and Harnessing Adversarial Examples* (2014). 3rd International Conference on Learning Representations, ICLR 2015-Conference Track Proceedings, San Diego, CA, United states: Elsevier Inc.
83. Luo Y, Boix X, Roig G, Poggio T, Qi Z. *Foveation-based Mechanisms Alleviate Adversarial Examples* (2015). arXiv preprint arXiv:1511.06292.
84. Gilmer J, Metz L, Faghri F, Schoenholz SS, Raghu M, Martin W, et al. *Adversarial Spheres* (2018). arXiv preprint arXiv:1801.02774.
85. Ilyas A, Santurkar S, Tsipras D, Engstrom L, Tran B, Madry A. *Adversarial Examples Are Not Bugs, They Are Features* (2019). arXiv preprint arXiv:1905.02175.
86. Yuan X, He P, Zhu Q, Li X. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Trans Neural Netw Learn Syst.* (2019) 30(9):2805–24. doi:10.1109/tnnls.2018.2886017
87. Ji J, Du TY, Li JF, Shen C, Li B. Application of Artificial Intelligence Technology in English Online Learning Platform. *J Softw* (2021) 32(1): 41–9. doi:10.1007/978-3-030-89508-2_6
88. Carlini N, Wagner D. Towards Evaluating the Robustness of Neural Networks. In: 2017 ieee symposium on security and privacy (sp). IEEE (2017). p. 39–57. doi:10.1109/sp.2017.49
89. Carlini N, Wagner D. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In: Proceedings of the 10th ACM workshop on artificial intelligence and security (2017). p. 3–14.
90. Carlini N, Wagner D. *Magnet and" Efficient Defenses against Adversarial Attacks" Are Not Robust to Adversarial Examples* (2017). arXiv preprint arXiv:1711.08478.
91. Chen P-Y, Sharma Y, Zhang H, Yi J, Hsieh C-J. Ead: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. In: Thirty-second AAAI conference on artificial intelligence (2018).
92. Kurakin A, Goodfellow I, Bengio S. *Adversarial Examples in the Physical World* (2016).
93. Kurakin A, Goodfellow I, Bengio S. *Adversarial Machine Learning at Scale* (2016). arXiv preprint arXiv:1611.01236.
94. Madry A, Makelov A, Schmidt L, Tsipras D, Adrian V. *Towards Deep Learning Models Resistant to Adversarial Attacks*. In: 6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings (2017), Vancouver, BC, Canada.
95. Dong Y, Liao F, Pang T, Su H, Zhu J, Hu X, et al. Boosting Adversarial Attacks with Momentum. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2018). p. 9185–93. doi:10.1109/cvpr.2018.00957
96. Moosavi-Dezfooli S-M, Fawzi A, Pascal F. Deepfool: a Simple and Accurate Method to Fool Deep Neural Networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016). p. 2574–82. doi:10.1109/cvpr.2016.282
97. Baluja S, Fischer I. Learning to Attack: Adversarial Transformation Networks. In: Thirty-second aaai conference on artificial intelligence (2018).
98. Xiao C, Li B, Zhu J-Y, He W, Liu M, Song D. Generating Adversarial Examples with Adversarial Networks. In 2021 IEEE International Conference on Image Processing (ICIP). Stockholm, Sweden: International Joint Conferences on Artificial Intelligence Organization (2018).
99. Bai T, Zhao J, Zhu J, Han S, Chen J, Li B, et al. Ai-gan: Attack-Inspired Generation of Adversarial Examples. In: 2021 IEEE International Conference on Image Processing (ICIP). IEEE (2021), 35(10):2543–7. doi:10.1109/icip42928.2021.9506278
100. Mao X, Chen Y, Wang S, Su H, Yuan H, Xue H. *Composite Adversarial Attacks* (2020). arXiv preprint arXiv:2012.05434.
101. Chen P-Y, Zhang H, Sharma Y, Yi J, Hsieh C-J. Zoo: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models. In: Proceedings of the 10th ACM workshop on artificial intelligence and security (2017). p. 15–26.
102. Ilyas A, Engstrom L, Athalye A, Lin J. Black-box Adversarial Attacks with Limited Queries and Information. In: International Conference on Machine Learning. Stockholm, Sweden: PMLR (2018). p. 2137–46.
103. Tim Salimans TH, Jonathan S, Chen X, Sidor S, Sutskever I. *Evolution Strategies as a Scalable Alternative to Reinforcement Learning* (2017). arXiv preprint arXiv:1703.03864.
104. Tu C-C, Ting P, Chen P-Y, Liu S, Zhang H, Yi J, et al. Autozoom: Autoencoder-Based Zeroth Order Optimization Method for Attacking Black-Box Neural Networks. *Aaai* (2019) 33:742–9. doi:10.1609/aaai.v33i01.3301742
105. Du J, Zhang H, Zhou JT, Yang Y, Feng J. *Query-efficient Meta Attack to Deep Neural Networks* (2019). arXiv preprint arXiv:1906.02398.
106. Yang B, Zeng Y, Jiang Y, Wang Y, Xia S-T, Guo W. Improving Query Efficiency of Black-Box Adversarial Attack. In: Computer Vision–ECCV 2020: 16th European Conference; August 23–28, 2020; Glasgow, UK. Springer (2020). p. 101–16.
107. Chen J, Jordan MI, Hopskipjumpattack MJW. A Query-Efficient Decision-Based Attack. In: 2020 ieee symposium on security and privacy (sp). IEEE (2020). p. 1277–94.
108. Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A. Practical Black-Box Attacks against Machine Learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security (2017). p. 506–19. doi:10.1145/3052973.3053009
109. Zhou M, Wu J, Liu Y, Liu S, Zhu C. Dast: Data-free Substitute Training for Adversarial Attacks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020). p. 234–43. doi:10.1109/cvpr42600.2020.00031
110. Ma C, Cheng S, Chen L, Zhu J, Junhai Y. *Switching Transferable Gradient Directions for Query-Efficient Black-Box Adversarial Attacks* (2020). arXiv preprint arXiv:2009.07191.
111. Zhu Y, Cheng Y, Zhou H, Lu Y. Hermes Attack: Steal {DNN} Models with Lossless Inference Accuracy. In: 30th {USENIX} Security Symposium ({USENIX} Security 21) (2021).
112. Wang W, Yin B, Yao T, Zhang L, Fu Y, Ding S, et al. Delving into Data: Effectively Substitute Training for Black-Box Attack. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021). p. 4761–70. doi:10.1109/cvpr46437.2021.00473
113. Ma C, Chen L, Jun-Hai Y. Simulating Unknown Target Models for Query-Efficient Black-Box Attacks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021). p. 11835–44. doi:10.1109/cvpr46437.2021.01166
114. Das N, Shanbhogue M, Chen S-T, Hohman F, Li S, Chen L, et al. Shield: Fast, Practical Defense and Vaccination for Deep Learning Using Jpeg Compression. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2018). p. 196–204.
115. Cheng Y, Wei X, Fu H, Lin S-W, Lin W. Defense for Adversarial Videos by Self-Adaptive Jpeg Compression and Optical Texture. In: Proceedings of the 2nd ACM International Conference on Multimedia in Asia (2021). p. 1–7. doi:10.1145/3444685.3446308
116. Samangouei P, Kabkab M, Chellappa R. Defense-gan: Protecting Classifiers against Adversarial Attacks Using Generative Models. 6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings, Vancouver, BC, Canada (2018).
117. Hwang U, Park J, Jang H, Yoon S, Cho NI. Puvae: A Variational Autoencoder to Purify Adversarial Examples. *IEEE Access* (2019) 7:126582–93. doi:10.1109/ACCESS.2019.2939352

118. Lin W-A, Balaji Y, Samangouei P, Chellappa R. *Invert and Defend: Model-Based Approximate Inversion of Generative Adversarial Networks for Secure Inference* (2019). arXiv preprint arXiv:1911.10291.
119. Zhang S, Gao H, Rao Q. Defense against Adversarial Attacks by Reconstructing Images. *IEEE Trans Image Process* (2021) 30:6117–29. doi:10.1109/tip.2021.3092582
120. Liu J, Zhang W, Zhang Y, Hou D, Liu Y, Zha H, et al. Detection Based Defense against Adversarial Examples from the Steganalysis point of View. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019). p. 4825–34. doi:10.1109/cvpr.2019.00496
121. Wang J, Zhao J, Yin Q, Luo X, Zheng Y, Shi YQ, et al. SmsNet: A New Deep Convolutional Neural Network Model for Adversarial Example Detection. *IEEE Trans Multimedia* (2021), IEEE.
122. Tian J, Zhou J, Li Y, Jia D. Detecting Adversarial Examples from Sensitivity Inconsistency of Spatial-Transform Domain. *Proc. AAAI Conf. Art. Intel.* (2021) 3511:9877–85. doi:10.1016/j.ins.2021.01.035
123. Chen J, Zheng H, Shangguan W, Liu L, Ji S. Act-detector: Adaptive Channel Transformation-Based Light-Weighted Detector for Adversarial Attacks. *Inf Sci* (2021) 564:163–92. doi:10.1016/j.ins.2021.01.035
124. Evan Sutanto R, Lee S. Real-time Adversarial Attack Detection with Deep Image Prior Initialized as a High-Level Representation Based Blurring Network. *Electronics* (2021) 10(1):52. doi:10.3390/electronics10010052
125. Liang B, Li H, Su M, Li X, Shi W, Wang X. Detecting Adversarial Image Examples in Deep Neural Networks with Adaptive Noise Reduction. *IEEE Trans Dependable Secure Comput* (2018) 18(1):72–85. doi:10.1109/TDSC.2018.2874243
126. Bai T, Luo J, Zhao J, Wen B, Wang Q. *Recent Advances in Adversarial Training for Adversarial Robustness* (2021). arXiv preprint arXiv:2102.01356.
127. Ali S, Najibi M, Amin G, Xu Z, John D, Studer C, et al. *Adversarial Training for Free!* (2019) Vancouver, BC, Canada, 32.
128. Zhang D, Zhang T, Lu Y, Zhu Z, Dong B. *You Only Propagate once: Accelerating Adversarial Training via Maximal Principle* (2019) Vancouver, BC, Canada, 32.
129. Wong E, Rice L, Kolter JZ. *Fast Is Better than Free: Revisiting Adversarial Training* (2020). arXiv preprint arXiv:2001.03994.
130. Gowal S, Dvijotham K, Stanforth R, Bunel R, Qin C, Uesato J, et al. *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models* (2018). arXiv preprint arXiv:1810.12715.
131. Zhang H, Chen H, Xiao C, Gowal S, Stanforth R, Li B, et al. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. *Adv. Neural Inform. Process. Syst.* (2019) 2018(10495258):4939–4948.
132. Zhang H, Weng T-W, Chen P-Y, Hsieh C-J, Daniel L. Efficient Neural Network Robustness Certification with General Activation Functions. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Virtual, United states: Association for Computational Linguistics (2018), 6066–80.
133. Gao J, Lanchantin J, Lou Soffa M, Qi Y. Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. In: 2018 IEEE Security and Privacy Workshops (SPW). IEEE (2018). p. 50–6. doi:10.1109/spw.2018.00016
134. Vijayaraghavan P, Roy D. Generating Black-Box Adversarial Examples for Text Classifiers Using a Deep Reinforced Model. *Mach. Learn. Knowl. Disc. Datab.* (2019). p. 711–726.
135. Iyyer M, John W, Gimpel K, Zettlemoyer L. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In: NAACL HLT 2018-2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies-Proceedings of the Conference. New Orleans, LA, United states (2018). p. 1875–85.
136. Ren Y, Lin J, Tang S, Zhou J, Yang S, Qi Y, et al. Generating Natural Language Adversarial Examples on a Large Scale With Generative Models. *Front. Artif. Intell. App.* (2020) 325(09226389):2156–63. doi:10.3233/FAIA200340
137. Li J, Ji S, Du T, Li B, Wang T. Textbugger: Generating Adversarial Text against Real-World Applications. *IEEE Access* (2020) 8:79561–72.
138. Li L, Ma R, Guo Q, Xue X, Qiu X. *Bert-attack: Adversarial Attack against Bert Using Bert* (2020). arXiv preprint arXiv:2004.09984.
139. Jin D, Jin Z, Zhou JT, Szolovits P. Is Bert Really Robust? a strong Baseline for Natural Language Attack on Text Classification and Entailment. *Aaai* (2020) 34:8018–25. doi:10.1609/aaai.v34i05.6311
140. Alzantot M, Sharma Y, Ahmed E, Ho B-J, Srivastava M, Chang K-W. Generating Natural Language Adversarial Examples in. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018. Virtual, United states (2018), 2890–96.
141. Cheng N, Chang G-Q, Gao H, Pei G, Zhang Y. Wordchange: Adversarial Examples Generation Approach for Chinese Text Classification. *IEEE Access* (2020) 8:79561–72.
142. Li D, Zhang Y, Peng H, Chen L, Brockett C, Sun M-T, et al. Contextualized Perturbation for Textual Adversarial Attack in. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics (2020), 5053–69.
143. Garg S, Bae GR. Bert-based Adversarial Examples for Text Classification. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2020). p. 6174–81.
144. Maheshwary R, Maheshwary S, Pudi V. Generating Natural Language Attacks in a Hard Label Black Box Setting. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence (2021).
145. Yuan Z, Qi F, Yang C, Liu Z, Zhang M, Liu Q, et al. *Word-level Textual Adversarial Attacking as Combinatorial Optimization* (2019). arXiv preprint arXiv:1910.12196.
146. Yang G, Gong NZ, Cai Y. *Fake Co-visitation Injection Attacks to Recommender Systems*. NDSS (2017).
147. Fang M, Yang G, Gong NZ, Jia L. Poisoning Attacks to Graph-Based Recommender Systems. In: Proceedings of the 34th Annual Computer Security Applications Conference (2018). p. 381–92. doi:10.1145/3274694.3274706
148. Christakopoulou K, Banerjee A. Adversarial Attacks on an Oblivious Recommender. In: Proceedings of the 13th ACM Conference on Recommender Systems (2019). p. 322–30. doi:10.1145/3298689.3347031
149. Sun Y, Wang S, Tang X, Hsieh T-Y, Honavar V. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. *Proc Web Conf* (2020) 2020:673–83. doi:10.1145/3366423.3380149
150. Song J, Zhao L, Hu Z, Wu Y, Li Z, Li J, et al. Poisonrec: an Adaptive Data Poisoning Framework for Attacking Black-Box Recommender Systems. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE). IEEE (2020). p. 157–68. doi:10.1109/icde48307.2020.00021
151. Chang H, Yu R, Xu T, Huang W, Zhang H, Cui P, et al. *Adversarial Attack Framework on Graph Embedding Models with Limited Knowledge* (2021). arXiv preprint arXiv:2105.12419.
152. Fang M, Gong NZ, Jia L. Influence Function Based Data Poisoning Attacks to Top-N Recommender Systems. In: Proceedings of The Web Conference (20202020). p. 3019–25. doi:10.1145/3366423.3380072
153. Li B, Wang Y, Singh A, Vorobeychik Y. Data Poisoning Attacks on Factorization-Based Collaborative Filtering. *Adv Neural Inf Process Syst* (2016) 29:1885–93.
154. Lin C, Chen S, Li H, Xiao Y, Li L, Yang Q. Attacking Recommender Systems with Augmented User Profiles. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management (2020). p. 855–64. doi:10.1145/3340531.3411884
155. Fan W, Tyler D, Zhao X, Yao M, Liu H, Wang J, et al. Attacking Black-Box Recommendations via Copying Cross-Domain User Profiles. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). IEEE (2021). p. 1583–94. doi:10.1109/icde51399.2021.00140
156. Zhan H, Pei X. *Black-box Gradient Attack on Graph Neural Networks: Deeper Insights in Graph-Based Attack and Defense* (2021). arXiv preprint arXiv:2104.15061.
157. Ben F, Baskin C, Zheltonozhskii E, Alon U. *Single-node Attack for Fooling Graph Neural Networks* (2020). arXiv preprint arXiv:2011.03574.
158. Huang H, Mu J, Gong NZ, Qi L, Liu B, Xu M. *Data Poisoning Attacks to Deep Learning Based Recommender Systems* (2021). arXiv preprint arXiv:2101.02644.
159. Wu C, Lian D, Ge Y, Zhu Z, Chen E. Triple Adversarial Learning for Influence Based Poisoning Attack in Recommender Systems. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (2021). p. 1830–40. doi:10.1145/3447548.3467335
160. Pruthi D, Dhingra B, Lipton ZC. Combating Adversarial Misspellings with Robust Word Recognition in. ACL 2019-57th Annual Meeting of the

Association for Computational Linguistics, Proceedings of the Conference. Florence, Italy (2020), 5582–91.
161. Jia R, Raghunathan A, Göksel K, Liang P. Certified Robustness to Adversarial Word Substitutions in. EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference. Hong Kong, China (2019), 4129–42.
162. Zhou Y, Jiang J-Y, Chang K-W, Wang W. Learning to Discriminate Perturbations for Blocking Adversarial Attacks in Text Classification in. EMNLP-IJCNLP 2019-2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference. Hong Kong, China (2019), 4904–13.
163. Si C, Zhang Z, Qi F, Liu Z, Wang Y, Liu Q, et al. *Better Robustness by More Coverage: Adversarial Training with Mixup Augmentation for Robust fine-tuning* (2020). arXiv preprint arXiv:2012.15699.
164. Ren S, Deng Y, He K, Che W. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In: Proceedings of the 57th annual meeting of the association for computational linguistics. Florence, Italy (2019). p. 1085–97. doi:10.18653/v1/p19-1103
165. Wang Z, Liu H, Tang J, Yang S, Huang GY, Liu Z. Learning Multi-Level Dependencies for Robust Word Recognition. *Proc. AAAI Conf. Artif. Intell.* (2020) 34:9250–7. doi:10.1609/aaai.v34i05.6463
166. Shi Z, Zhang H, Chang K-W, Huang M, Hsieh C-J. Robustness Verification for Transformers in. *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics (2020), 164–171.
167. Ye M, Gong C, Liu Q. Safer: A Structure-free Approach for Certified Robustness to Adversarial Word Substitutions. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics. Virtual, Online, United states (2020). p. 3465–75.
168. Mozes M, Stenetorp P, Bennett K, LewisGriffin D. Frequency-guided Word Substitutions for Detecting Textual Adversarial Examples in. EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference. Virtual (2020), 171–86.
169. Zeng J, Zheng X, Xu J, Li L, Yuan L, Huang X. *Certified Robustness to Text Adversarial Attacks by Randomized [mask]* (2021). arXiv preprint arXiv: 2105.03743.
170. Wang W, Wang R, Ke J, Wang L. Textfirewall: Omni-Defending against Adversarial Texts in Sentiment Classification. *IEEE Access* (2021) 9: 27467–75. doi:10.1109/access.2021.3058278
171. Karimi A, Rossi L, Prati A. Adversarial Training for Aspect-Based Sentiment Analysis with Bert. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE (2021). p. 8797–803. doi:10.1109/icpr48806.2021.9412167
172. Hu X, Liu B, Shu L, Philip SY. Bert post-training for Review reading Comprehension and Aspect-Based Sentiment Analysis in. NAACL HLT 2019-2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference. Minneapolis, MN, United states (2019), 2324–35.
173. Miyato T, Dai AM, Goodfellow I. Adversarial Training Methods for Semi-supervised Text Classification in. 5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings. Toulon, France (2016).
174. Du Y, Fang M, Yi J, Xu C, Cheng J, Tao D. Enhancing the Robustness of Neural Collaborative Filtering Systems under Malicious Attacks. *IEEE Trans Multimedia* (2018) 21(3):555–65. doi:10.1109/tmm.2018.2887018
175. Papernot N, McDaniel P, Wu X, Jha S, Swami A. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. In: 2016 IEEE symposium on security and privacy (SP). IEEE (2016). p. 582–97. doi:10.1109/sp.2016.41
176. Tang J, Du X, He X, Yuan F, Qi T, Chua T-S. Adversarial Training towards Robust Multimedia Recommender System. *IEEE Trans Knowledge Data Eng* (2019) 32(5):855–67. doi:10.1109/TKDE.2019.2893638
177. Manotumruksa J, Yilmaz E. Sequential-based Adversarial Optimisation for Personalised Top-N Item Recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020). p. 2045–8. doi:10.1145/3397271.3401264
178. Li R, Wu X, Wang W. Adversarial Learning to Compare: Self-Attentive Prospective Customer Recommendation in Location Based Social Networks. In: Proceedings of the 13th International Conference on Web Search and Data Mining (2020). p. 349–57.
179. Wang J, Han P. Adversarial Training-Based Mean Bayesian Personalized Ranking for Recommender System. *IEEE Access* (2019) 8:7958–68.
180. Shahrasbi B, Mani V, Arrabothu AR, Sharma D, Kannan A, Kumar S. *On Detecting Data Pollution Attacks on Recommender Systems Using Sequential gans* (2020). arXiv preprint arXiv:2012.02509.
181. Wu C, Lian D, Ge Y, Zhu Z, Chen E, Yuan S. Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (2021). p. 1074–83. doi:10.1145/3404835.3462914
182. Yi Q, Yang N, Yu P. Dual Adversarial Variational Embedding for Robust Recommendation. *IEEE Trans Knowledge Data Eng* (2021). doi:10.1109/tkde.2021.3093773
183. Shi Z, Huang M. *Robustness to Modification with Shared Words in Paraphrase Identification* (2019). arXiv preprint arXiv:1909.02560.
184. PangKoh W, Liang P. Understanding Black-Box Predictions via Influence Functions. In: International Conference on Machine Learning. Sydney, NSW, Australia: PMLR (2017). p. 1885–94.
185. Xue M, Yuan C, Wang J, Liu W. Dpaeg: A Dependency Parse-Based Adversarial Examples Generation Method for Intelligent Q&a Robots. *Security and Communication Networks*Hindawi (2020). p. 2020.
186. Deng E, Qin Z, Meng L, Ding Y, Qin Z. Attacking the Dialogue System at Smart home. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing. Springer (2020). p. 148–58.
187. Ebrahimi J, Daniel L, Dou D. Efficient Neural Network Robustness Certification with General Activation Functions in. Proceedings of the 27th International Conference on Computational Linguistics. Santa Fe, New Mexico, USA: Association for Computational Linguistics (2018), 653–63.
188. Wang J, Xu C, Guzmán F, El-Kishky A, Tang Y, Rubinstein BIP, et al. Putting Words into the System's Mouth: A Targeted Attack on Neural Machine Translation Using Monolingual Data Poisoning *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021* (2021), 1463–73.
189. Cheng Y, Jiang L, Macherey W. Robust Neural Machine Translation with Doubly Adversarial Inputs in. ACL 2019-57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference. Florence, Italy (2019), 4324–33.
190. Cheng Y, Jiang L, Macherey W, Jacob E. Advaug: Robust Adversarial Augmentation for Neural Machine Translation in. Proceedings of the Annual Meeting of the Association for Computational Linguistics. Virtual, United states (2020), 5961–70. doi:10.18653/v1/2020.acl-main.529

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Physics

ORIGINAL RESEARCH
published: 06 December 2021
doi: 10.3389/fphy.2021.768181

Efficient Targeted Influence Maximization Based on Multidimensional Selection in Social Networks

*Dong Jing and Ting Liu**

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

The influence maximization problem over social networks has become a popular research problem, since it has many important practical applications such as online advertising, virtual market, and so on. General influence maximization problem is defined over the whole network, whose intuitive aim is to find a seed node set with size at most k in order to affect as many as nodes in the network. However, in real applications, it is commonly required that only special nodes (target) in the network are expected to be influenced, which can use the same cost of placing seed nodes but influence more targeted nodes really needed. Some research efforts have provided solutions for the corresponding targeted influence maximization problem (TIM for short). However, there are two main drawbacks of previous works focusing on the TIM problem. First, some works focusing on the case the targets are given arbitrarily make it hard to achieve efficient performance guarantee required by real applications. Second, some previous works studying the TIM problems by specifying the target set in a probabilistic way is not proper for the case that only exact target set is required. In this paper, we study the Multidimensional Selection based Targeted Influence Maximization problem, MSTIM for short. First, the formal definition of the problem is given based on a brief and expressive fragment of general multi-dimensional queries. Then, a formal theoretical analysis about the computational hardness of the MSTIM problem shows that even for a very simple case that the target set specified is 1 larger than the seed node set, the MSTIM problem is still NP-hard. Then, the basic framework of RIS (short for Reverse Influence Sampling) is extended and shown to have a $1 - 1/e - \epsilon$ approximation ratio when a sampling size is satisfied. To satisfy the efficiency requirements, an index-based method for the MSTIM problem is proposed, which utilizes the ideas of reusing previous results, exploits the covering relationship between queries and achieves an efficient solution for MSTIM. Finally, the experimental results on real datasets show that the proposed method is indeed rather efficient.

Keywords: targeted, influence maximization, index, sampling, multidimensional selection

OPEN ACCESS

Edited by:
Chengyi Xia,
Tianjin University of Technology, China

Reviewed by:
Peican Zhu,
Northwestern Polytechnical University, China
Zhen Wang,
Hangzhou Dianzi University, China

***Correspondence:**
Ting Liu
tliu@ir.hit.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *31 August 2021*
Accepted: *09 November 2021*
Published: *06 December 2021*

Citation:
Jing D and Liu T (2021) Efficient Targeted Influence Maximization Based on Multidimensional Selection in Social Networks. Front. Phys. 9:768181. doi: 10.3389/fphy.2021.768181

1 INTRODUCTION

As the social applications and graph structured data become more and more popular, many fundamental research problems over social networks have increased the interests of researchers. Influence maximization problem is a typical one of such problems, which aims to find a set of nodes with enough influential abilities over the whole network. One typical application of influence maximization problem is virtual marketing, which utilizes the method of pushing advertisements to special users and encourages them to propagate the advertisements to more users by their social relationships. Recently, the problem has been focused by lots of research works, which spreads over several areas such as network, information management and so on. Also, in different applications, the corresponding variants of the influence maximization problem have been proposed and studied.

One important variant of the general influence maximization problem is called targeted influence maximization, TIM for short. In the general definition, given a network G, the influence maximization problem is to compute a set S of k seed nodes such that S can influence the most nodes in G. Different from the general one, the aim of targeted influence maximization problem is to influence the nodes in a special subset $T \subseteq V_G$ (target set) as many as possible but not the whole node set of G. Obviously, in the definition of TIM, how to define the target set T is a key step. In [1,2], the target set is chosen arbitrarily, whose definition is independent from the application settings and in the most general way. In Li et al. (2015), it is given by a topic-aware way, where each node is associated with several topics and the target is specified by a topic list. Given the topic list (query), a measure about the closeness between each node and the query can be computed. As a consequence, the optimizing goal of TIM can be defined by a weighted sum of all nodes in G. In fact, the definition used by [3] assigns each node a probability of appearing in the target set and solves the corresponding influence maximization problem by using a modified optimizing goal.

There are two main drawbacks of previous works focusing on the TIM problem. First, providing abilities of quick feedback for the influence maximization applications [2,3] is very important, however, the general definition of TIM taken by previous works like [1] makes it hard to improve the performance of TIM algorithms by utilizing previous efforts on computing for other target sets, since the targets are usually given randomly and independent and caching the related information will cause huge costs. Second, previous works like [3] study the TIM problems by specifying the target set by topic-ware queries, query based specification of target sets

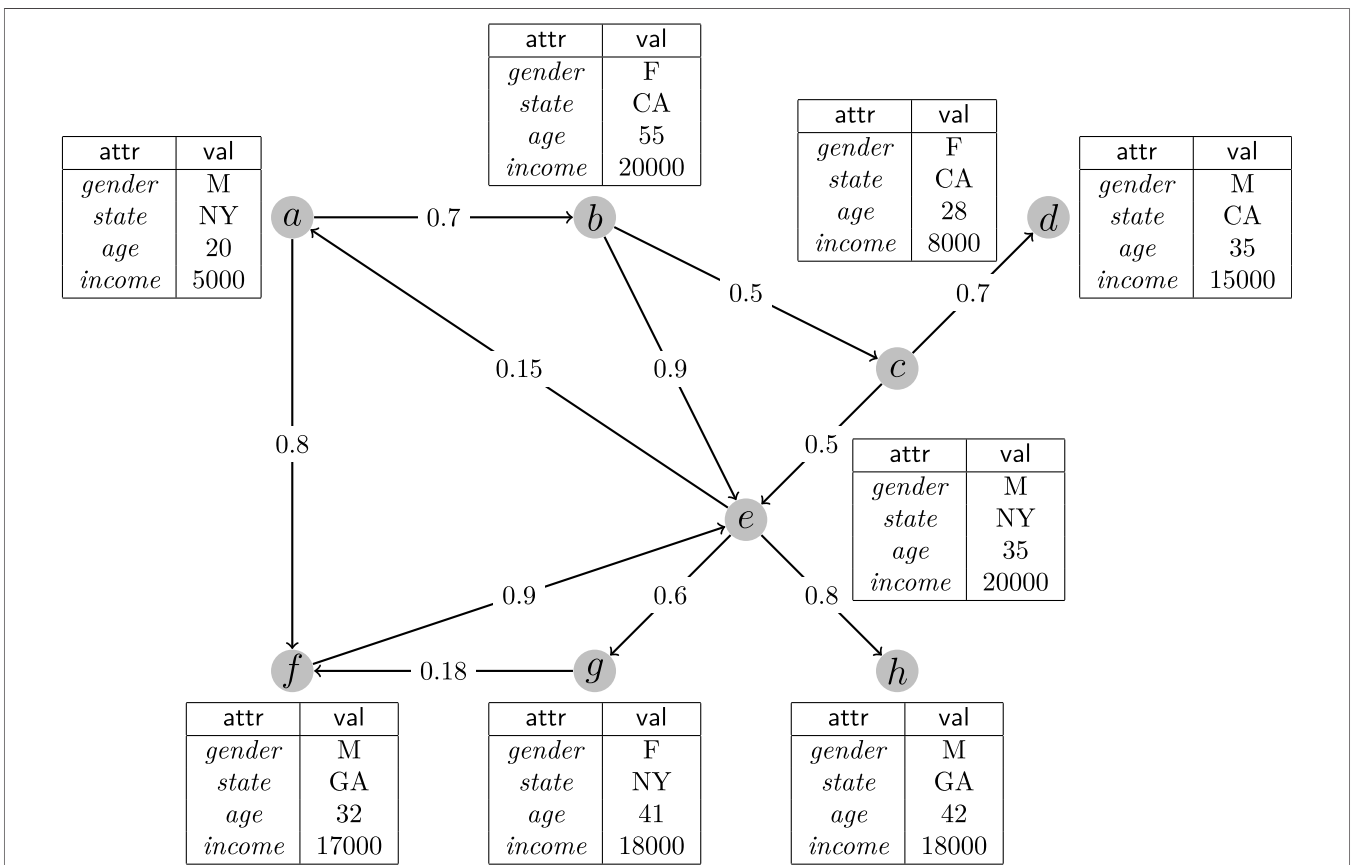


FIGURE 1 | A motivated example of multidimensional selection based targeted influence maximization.

make it possible to index relatively less information and answer an arbitrary TIM problem defined on topics efficiently by reusing the information indexed. However, as shown by the following example, in many applications, users may expect the target set can be specified in a more exact way, and the definition used in [3] will be not proper.

Example 1. As shown in **Figure 1**, there is a social network whose relationships can be represented by the graph structure. The node labelled by "*a*" maintains the information about a man aged 20 lived in "NY" whose salary is 5,000 per month. Also, the information associated with other nodes in the graph can be explained similarly. Each directed edge between two nodes u and v means that u can influence v. The edge between a and f is labelled by 0.8, it means that when receiving a message from a the probability that f will accept and transform the message is 0.8. That is, the value in the middle of each edge is the corresponding influence probability.

Let us consider a simple example. Suppose there is only one seed node b during the information propagation, since there is only one path between b and d, the probability that d will be influenced at last will be $p_{b,d} = 0.5 \times 0.7 = 0.35$. The general influence maximization problem is to find a seed node set such that after the information propagation procedure the expected number of nodes influenced is maximized.

Now, consider a case that the user want to select some nodes to help him to make an advertisement of an expensive razor. In this application, the target set may be naturally expected to be the male persons with high salary (no less than 15,000). That is, only the seed nodes with high influences to the nodes in $\{d, e, f, h\}$ should be considered. Moreover, to specify the target set exactly and briefly, multi-dimensional range query is a proper choice, which can express the above requirements by a statement q: $(gender = \text{"M"}) \wedge (income \geq 15000)$.

As far as known by us, there are no previous works focusing on the targeted influence maximization problem based on multi-dimensional queries. To provided quick response to the targeted influence maximization problem based on multi-dimensional queries, there are at least two challenges. 1) Different from previous methods, since the target set is specified in a non-trivial way, efficient techniques for collecting the exact target set for an ad-hoc query must be developed. 2) To return the seed node set efficiently, the idea of using previously cached results should be well exploited. Although in the area of topic aware influence maximization [3] has investigated such method, it is not proper for the cases when the target set is specified by multi-dimensional queries.

Therefore, in this paper, we address the problem of Multidimensional Selection based Targeted Influence Maximization, MSTIM for short. To support efficient evaluation of queries specifying the target set, an index based solution is utilized to reuse the previous query results. While to compute the corresponding influence maximization problem efficiently, a sample based method is developed based on previous works, and it is extended to an index based solution which can reuse the samples obtained before and improve the performance significantly. The main contributions of this paper can be summarized as follows.

1. We identify the effects of multi-dimensional queries to specify the target set in the influence maximization problem, and propose the formal problem definition of Multidimensional Selection based Targeted Influence Maximization (MSTIM for short) based on a brief and expressive fragment of general multi-dimensional queries.
2. We show the computational hardness of the MSTIM problem, in fact, even if a very simple case that the target set specified is 1 larger than the seed node set, the MSTIM problem is still NP-hard.
3. Based on the Reverse Influence Sampling (RIS for short) method previously proposed, for the MSTIM problem, the basic framework of RIS is extended and shown to have a $1 - 1/e - \epsilon$ approximation ratio when a sampling size is satisfied.
4. The index-based solution for the MSTIM problem is proposed. Using indexes of queries previously maintained, the performance of evaluating multi-dimensional queries are improved by reusing the results computed before. Sophisticated techniques for handling searching query predicates are designed and well studied. By the help of indexes of previous samples and the inverted index between nodes and samples, the MSTIM problem can be solved efficiently.
5. The experimental results on real datasets show that the proposed method is indeed rather efficient.

The rest parts of the paper are organized as follows. In **section 2**, some background information are introduced. Then, in **section 3**, the theoretical analysis of the MSTIM problem is given. **Section 4** provides the basic framework of the sampling based approximation solution for the MSTIM problem. In **section 5**, the index version is proposed and introduced in details. **Section 6** shows the experimental results. Related works are discussed in **section 7**, and the final part is the conclusion.

2 RELATED WORK

The influence maximization is an important and classical problem in the research area of online social networking, which has many applications such as viral marketing, computational advertising and so on. It is firstly studied by Domingo and Richardson [4,5], and the formalized definitions and comprehensive theoretical analysis are given in [6]. Different models have been formally defined to simulate the information propagation processes with different characteristics, the two most popular models are the Independent Cascade (IC for short) and Linear Threshold (LT for short) models. In [6], the influence maximization problems under both IC and LT models are shown to be NP-hard problems and the problem of computing the exact influence of given nodes set is shown to be ♯P-hard problem in [7].

After the problem is proposed, many research efforts have been made to find the node set with maximum influence [6]. Proposed an algorithm for influence maximization based on greedy ideas which has constant approximation ratio $(1 - 1/e)$, whose time cost is usually expensive for large networks. To

overcome the shortcomings of greedy based algorithms [8], proposed CELF (Cost-Effective Lazy-Forward) algorithm, which can improve the performance of greedy based algorithms for influence maximization by reducing the times of evaluations of influence set of given seed set [9]. Proposed SIMPATH algorithm in LT model which improve the performance of greedy based influence maximization algorithm in LT model. Similar works focusing on improve the performance of influence maximization algorithms can be found also, such as [10–12] and so on. Recently, a series of sampling based influence maximization algorithms such as [13–15] are proposed and well developed, which have improved the practical performance greatly by involving a tiny loss on the approximation ratio. However, as shown by [16,17], the efficiency problem is still challenging for applying influence maximization algorithms in real applications.

The work most related with ours is [3], which focuses on the topic aware targeted influence maximization problem. In the topic aware setting, each node is associated with a list of interesting topics and the query is specified in the form of topic list. After calculating the similarities between users and the query, a weight can be assigned to the node such that the general optimizing goal of IM problem is extended to the weighted case. A sampling based solution under the help of indexes are given in [3]. Indexes are built for each keyword refereed in the topic setting. When a random query is given, the topic list associated with the query will be weighted and the computations of the similarities will be assigned to each keyword, and finally the samples are collected by combining the samples maintained for each keyword. Different from this paper, the work is not proper for the case that an exact subset of the whole network is expected to be the target set. Considering the case that the target set can be specified in an arbitrary way [1,2] studies the most general targeted influence maximization problems and provides efficient solutions. However, the general method adopted by them makes it almost impossible to provide efficient solutions using previous results with acceptable space cost. While this paper considers a more specific case that the target set can be described by a multi-dimensional query and utilizes the characteristics of those queries to develop sophisticated index based solutions. Therefore, the paper studies a variant of targeted influence maximization problem which is different from previous works.

There are also many works which try to extend the classic influence maximization methods to other application settings [18]. Studies the problem of influence maximization under location based social networks. In those networks, one node can be influenced by the other node if and only if they are neighbours according to their location informations, and [18] focus on the problem of finding k users which can affect maximum users in the location based social network [19]. Identifies the relation types during propagating the information and formally defines the problem of influence maximization by considering different types of relationships between nodes. A key idea is that given certain information which needs to be propagate the influence set of some node set can be computed more efficiently by reducing those edges belonging to some certain types [20]. Studies the problem of influence maximization under topic-aware applications. As shown by [21], the influence probabilities between users with special triangle structures are obviously higher than others. The above research efforts focus on totally different problems, compared with this paper, but their ideas on developing efficient influence maximization algorithms are helpful for us.

3 PRELIMINARY

3.1 Classical Influence Maximization

The general description of information diffusion can be explained to be a propagating procedure of information over some special network. A network is denoted by a graph $G(V, E)$. Given an information diffusion model M, the model will describe how the nodes influences others in network. In an instant state of the network, nodes in the network will be labelled by active or inactive. According to the model M, the inactive nodes may become active because of the existence of special active neighbours, whose rule is defined by M.

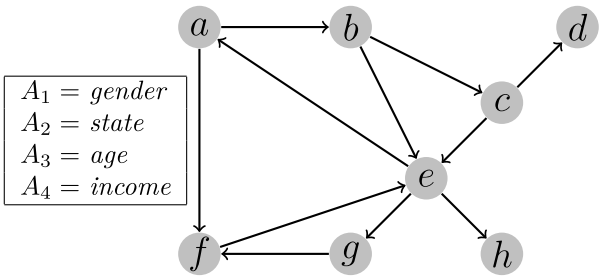
There are two classical methods to define the information propagation model, linear threshold and independent cascade model. This paper focuses on the independent cascade model (IC for short). In this model, for each edge (u, v) a probability p_{uv} is given to describe that u can activate v with probability p_{uv}. After initializing an active node set S_0, in the ith step, every node will try to activate their neighbours. In detail, for each node $u \in S_{i-1}$ and node $v \in V \setminus S_{i-1}$, if $(u, v) \in E$, v will be activated once in probability p_{uv}. If v indeed becomes active, it will be added to S_i and not be further considered in current step. Repeat this procedure until that no new nodes are added. Obviously, under a specific information propagation model, given an initial active set S over a network G, we can obtain a node set I_S which can be activated when the propagation procedure is finished. Therefore, an expected value $\mathbb{E}[I_S]$ can be defined according to the probabilistic distributions of the possible information propagation graphs, whose details can be found in [6].

Definition 1 (Influence Maximization, IM for short). Given a propagation graph $G = (V, E)$ such that there is an associated probability p_{uv} for each edge $(u, v) \in E$, and an integer $k > 0$, the goal is to compute a node set S such that $\mathbb{E}[I_S]$ is maximum.

3.2 Multidimentional Selection

To support multidimensional selections over social networks, it is necessary to consider an extended model of the general network for information propagation.

For each node $v \in V_G$, there are m attributes $A = \{A_1, A_2, \ldots, A_m\}$ $(m \in \mathbb{N})$ associated. Let N_i be the number of distinct values in the attribute A_i. For ordered attributes, wlog, it can be assumed that the domain of A_i is $Dom(A_i) = \{1, 2, \ldots, N_i\}$. While, for general attributes, we can represent the values of A_i as $Dom(A_i) = \{a_1, a_2, \ldots, a_i\}$. In practical applications, most of semantic information related to nodes in the network can be represented by the associated attributes. For example, in social



nodeID	A_1	A_2	A_3	A_4
a	M	NY	20	5000
b	F	CA	55	20000
c	F	CA	28	8000
d	M	CA	35	15000
e	M	NY	35	20000
f	M	GA	32	17000
g	F	NY	41	18000
h	M	GA	42	18000

FIGURE 2 | An example of network with attributes.

networks, the vertex related information such as age, birthplace, interests, and so on can be represented by the attributes associated with vertices. Specially, we use $v.A_i$ to represent the value of node v on attribute A_i. Formally, such a network can be represented by $G = (V, E, A)$, where for each node v there is $v.A_i \in Dom(A_i)$ for every attribute $A_i \in A$.

Then, we can define some basic concepts of multidimensional selection queries.

Definition 2 (1-Dimension Set). A 1-dimension set of a general attribute A_i is a set $s = \{v_1, v_2, \ldots, v_l\}$ satisfying $v_j \in Dom(A_i)$ for each j between 1 and l.

Definition 3 (1-Dimension Range). A 1-dimensional range $\mathsf{r} = [l_i, u_i]$ satisfying the constraint $l_i < u_i$ of an ordered attribute A_i defines a 1-dimensional set $[l_i, u_i] = \{l_i, l_i + 1, \ldots, u_i\}$.

Similarly, we can define the 1-dimension range (l_i, u_i), $(l_i, u_i]$ and $[l_i, u_i)$, where the round bracket means that it excludes the boundary value.

Then, a 1-dimensional selection query q can be represented by (A_i, p) where p is a 1-dimension set s or a 1-dimension range r, the predict p essentially defines a function p: $Dom(A_i) \mapsto \{0, 1\}$. For a node $v \in V$, v satisfies a 1-dimensional query $q = (A_i, s)$ or $q = (A_i, \mathsf{r})$, represented by $v \in q(V)$, if and only if $v.A_i \in s$ or $v.A_i \in \mathsf{r}$. It should be noted that a 1-dimensional selection query q defines a function q: $V \mapsto \{0, 1\}$.

To be more general, we can define k-dimensional selection based on the definitions above.

Definition 4 (k-Dimensional Selection Query). A k-dimensional selection query Q, which defines a function $V \mapsto (0, 1)$, is composed of a set of k 1-dimensional query $\{q_1, \ldots, q_k\}$. For each node $v \in V$, $Q(v) = 1$ or $v \in Q(V)$ if and only if we have $q_i(v) = 1$ for all q_i $(1 \le i \le k)$.

Here, given a k-dimensional selection query Q, let $\mathbb{Q}$ be the vector which includes all associated 1-dimensional selection queries of Q and for each $i \in (1, k)$ the query q_i is stored in $\mathbb{Q}^i$. Then, the condition of $Q(v) = 1$ will be equivalent with the fact that $\mathbb{Q}^i(v) = 1$ for all $i \in (1, k)$. In the followings, to be convenient, we will use k-dimensional query and 1-dimensional query to denote k-dimensional selection query and 1-dimensional query, respectively.

Then, based on the concepts above, for a specific node set V, we can give a formal definition of the selection result of query Q as $Q(V) = [v | v \in V \text{ and } Q(v) = 1]$, which is used in an informal way before.

Example 2. Continue with Example 1, the network shown in **Figure 1** can be transformed into the form shown in **Figure 2**, where each node in the network is associated with four attributes which are listed on the left. For each node, the corresponding values are shown on the right in the form of a table. Given a 2-dimensional query Q: $[A_1 = M, A_2 \in (20, 40)]$, the query result $Q(V)$ will include the nodes a, d, e, and f.

3.3 Multidimensional Selection Based Targeted Influence Maximization

Given a k-dimensional query Q representing the target users, it can be used to determine whether a node v is interested by checking whether $Q(v) = 1$. For a node set $S \subseteq V$, after a specific information propagation procedure, only the nodes activated which belong to the result of query Q are really interested by the users. Then, we can define the selection query based targeted influence as follows.

Definition 5 (Multidimensional Selection based Targeted Influence). Given a network graph $G = (V, E, A)$, a k-dimensional query Q and a node set $S \subseteq V$, if the influence of S in classic influence maximization model is denoted by I_S, the targeted influence based on Q can be represented by $F_S = I_S \cap Q(V)$.

Similarly, we can define the expected targeted influence $\mathbb{E}[F_S]$ based on the probabilistic distributions generated by the information diffusing procedures, and the formal definition of targeted influence maximization problem can be given as follows.

Definition 6 (Multidimensional Selection based Targeted Influence Maximization, MSTIM for short). Given a network graph $G = (V, E, A)$, a k-dimensional query Q and an integer k, the problem is to find a subset $S \subseteq V$ satisfying $|S| \le k$ such that the expected size of $\mathbb{E}[F_S]$ is maximized.

4 THE COMPUTATIONAL COMPLEXITY OF MSTIM

Obviously, the general influence maximization problem is a special case of the MSTIM problem. Therefore, it can be known that the MSTIM problem is NP-*hard* when the query Q simply returns all nodes in V. Then, we can have the following result without detailed proofs.

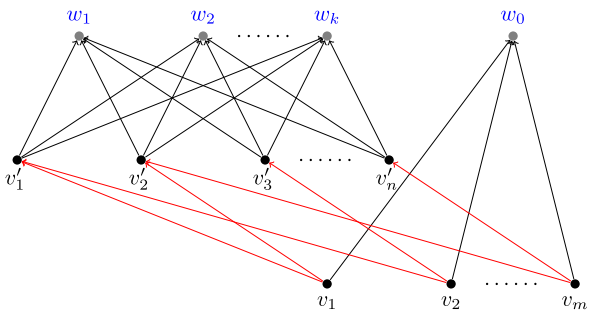


FIGURE 3 | An example of the reduction, where the edge with activation probability 1 is in red and the other edges have probability c.

Proposition 4.1. The MSTIM problem is NP-hard.

Of course, the above general case is very special and our interesting point is whether the MSTIM problem can be solved more efficiently when a practical query Q is met.

Intuitively, the definition of MSTIM is based on limiting the nodes influenced within a range defined by the query Q. An extreme case is that the result size of $Q(V)$ is very small, and it is interested to study whether or not there exists an efficient algorithm for such cases. Obviously, when $|Q(V)| \leq k$, the MSTIM problem can be solved efficiently simply by choosing the nodes in $Q(V)$ into S, since the query result $Q(V)$ can be solved by scanning every node $v \in V$ and checking the dimensional predicates which will produce an algorithm in linear time. Then, it is meaningful to study whether such an algorithm can be extended to solve more cases for the MSTIM problem.

Next, we can prove that even for very limited but not trivial cases, it is still hard to solve the MSTIM problem efficiently.

Theorem 1. Given a network graph $G = (V, E, A)$, a k-dimensional query Q and an integer k, the MSTIM problem is still NP-hard even for the case $|Q(V)| = k + 1$.

Proof. Consider an instance of the Set Cover problem, which is a well-known NP-complete problem, whose input includes a collection of subsets $S_1, S_2, \ldots, S_m$ of a ground set $U = \{u_1, u_2, \ldots, u_n\}$. The question is whether there exist k of the subsets whose union is equal to U.

In [6], the Set Cover problem is shown to be a special case of the classical influence maximization problem, whose following results is that the classical influence maximization problem is NP-hard. While, in this paper, by the following reduction, it can be utilized to show that the MSTIM problem is NP-hard even for the case $|Q(V)| = k + 1$.

Given an arbitrary instance of the Set Cover problem, we first define a corresponding directed bipartite graph with $n + m$ nodes like done in the proof of [6]. Suppose the bipartite graph constructed is $G_1 = (V_1, E_1)$. For each set S_i, there is a corresponding node v_i, and there is a node v'_j for each element u_j. If $u_j \in S_i$, there is an edge (v_i, v'_j) with activation probability $p_{v_i v'_j} = 1$. Then, G_2 is built based on G_1 by adding k nodes $\{w_1, w_2, \ldots, w_k\}$ into G_1 and building an edge (v_i, w_j) with activation probability $p_{v_i w_j} = c$ for each $i \in (1, m)$ and $j \in (1, k)$, where c is a constant between 0 and 1. Then, G_3 is built based on G_2 by adding one node w_0 and inserting an edge (v'_j, w_0) with activation probability c for each $j \in (1, n)$. Then, we will build the associated attributes for the graph G_3 as follows. Let the attribute set $A = (A_1)$. For each node v of G_3, if $v \in \{w_0, w_1, \ldots, w_k\}$, set $v.A_1 = 1$, otherwise, set $v.A_1 = 2$. Let $Q = (q_1)$ where q_1: $A_1 = 1$. Finally, we require that the constant c used above is larger than $\frac{k-1}{k}$. An example of the reduction can be found in **Figure 3**.

The following facts are essential for verifying the correctness of the reduction above.

- For the query Q, we have $|Q(V)| = k + 1$, and we can obtain an easy lower bound by selecting arbitrary k nodes from $\{w_0, \ldots, w_k\}$. Observing that there are no output edges of the nodes in $(w_0, \ldots, w_k)$, such a method can produce a seeding node set with expected influence k exactly.
- Obviously, S should not choose nodes in $\{v'_j\}$ $(j \in [1, n])$. In that case, an alternative node in (v_i) $[i \in (1, m)]$ can be used to replace those nodes without decreasing the influence.
- Suppose S contains nodes in both (w_i) and (v_j) and there exists a set cover (S_t) of size k, we can always increase the influence by utilizing nodes in (v_j) to replace nodes in (w_i). Assume that there are x nodes of $\{v_j\}$ and $y > 0$ nodes of $\{w_i\}$ in S, and the x nodes can cover $n - 1$ nodes in $\{v'_l\}$ which is the best situation. The expected influence in $(w_1, \ldots, w_k)$ obtained by S can be calculated by the following formula.

$$\mathbb{E}[S] = y + (k - y)\left(1 - (1 - c)^{n-1}\right) \tag{1}$$

Let S' be the nodes of $\{v_i\}$ corresponding to the cover $\{S_t\}$.

$$\mathbb{E}[S'] = k\left(1 - (1 - c)^n\right) \tag{2}$$

Then, we have the following results.

$$\mathbb{E}[S] \leq \mathbb{E}[S'] \tag{3}$$

$$\Leftrightarrow y + (k - y)\left(1 - (1 - c)^{n-1}\right) \leq k\left(1 - (1 - c)^n\right) \tag{4}$$

$$\Leftrightarrow (k - y)(1 - c)^{n-1} \geq k(1 - c)^n \tag{5}$$

$$\Leftrightarrow c \geq \frac{y}{k} \tag{6}$$

$$\Leftrightarrow \frac{k-1}{k} \geq \frac{y}{k} \tag{7}$$

$$\Leftrightarrow k - 1 \geq y \tag{8}$$

Obviously, we have $\mathbb{E}[S] \leq \mathbb{E}[S']$ always. Moreover, consider the node w_0, it can only increase its influence when choosing more nodes in $\{v_i\}$. Therefore, if there exists a set cover $\{S_t\}$ of size k, an optimal solution can be built by choosing the corresponding k nodes in $\{v_i\}$.

Then, according to the facts above, it is easy to check that there exists a solution of the set cover problem if and only if we can find k seeding nodes such that the influence obtained is at least $k + 1 - k(1 - c)^n - (1 - c)^k$.

Finally, the MSTIM problem is NP-hard even for the case $|Q(V)| = k + 1$.

5 THE BASIC SAMPLING ALGORITHM FOR MSTIM

5.1 Reverse Influence Sampling

RIS (Reverse Influence Sampling) based methods are the state-of-the-art techniques for approximately solving influence

maximization problem. In this part, we introduce this kind of methods first. First, we introduce the concept of Reverse Reachable (RR) Set and Random RR Set.

Definition 7 (Reverse Reachable Set, [22]). Let v be a node in G, and $\hat{G}$ be a graph obtained by removing each edge e in G with $1 - p(e)$ probability. The reverse reachable (RR) set for v in $\hat{G}$ is the set of nodes in $\hat{G}$ that can reach v. (That is, for each node u in the RR set, there is a directed path from u to v in $\hat{G}$).

Definition 8 (Random RR Set, [22]). Let $\mathcal{G}$ be the distribution of $\hat{G}$ induced by the randomness in edge removals from G. A random RR set is an RR set generated on an instance of $\hat{G}$ randomly sampled from $\mathcal{G}$, for a node selected uniformly at random from $\hat{G}$.

Based on the two definitions above, a typical RIS based method can be described as follows.

- Generate multiple random RR sets based on G.
- Utilize the greedy based algorithm for max-coverage problem shown in [23] to find a node set A satisfying $|A| \leq k$ such that as many random RR sets can be covered by A as possible. The solution is a $(1 - 1/e)$-approximation result.

Obviously, since the second step is just a standard method for solving maximum coverage problem, to guarantee the $(1 - 1/e)$ approximation ratio, enough samples should be gathered in the first round of the algorithm. As shown in [24], there have been several research works providing such sampling based influence maximization algorithm with $1 - 1/e - \epsilon$ approximation ratio within time cost $O(\frac{k(|E|+|V|)\log|V|}{\epsilon^2})$.

Here, assuming that the sample size is presented by θ, a result shown in [22] is utilized to explain the principles of our method. Since there have been always research efforts focusing on improving the sampling size, it is easy to check that the improved version can be easily applied to the method proposed by us.

Theorem 2 [22]. If θ is at least $(8 + 2\epsilon) \cdot |V| \cdot \frac{\ln \frac{1}{\delta} + \ln\left(\frac{|V|}{k}\right) + \ln 2}{OPT_k \cdot \epsilon^2}$, the typical RIS method will return a solution with $1 - 1/e - \epsilon$ approximation ratio with high probability $1 - \delta$.

5.2 RIS Based Algorithm for MSTIM

As shown in **Figure 1**, a conceptual algorithm is given to solve the MSTIM problem. Compared with the original version of RIS method, the random RR set is not generated by starting from an arbitrary node in G, but from a node in $Q(V)$, where $Q(V)$ is the nodes in the selection result of query Q.

Algorithm 1. RIS-MSTIM.

Input: A network graph $G = (V, E, A)$, a k-dimensional query Q and an integer k
Output: A seeding node set S
1: function RIS-MSTIM(G, Q, k)
2: Let T be the set of nodes $Q(V)$;
3: $R \leftarrow \emptyset$;
4: Generate θ random RR sets for nodes in T, and insert them into R;
5: $S \leftarrow \emptyset$;
6: for $i \in [1, k]$ do
7: Let v_i be the node covering the most random RR sets in R;
8: $S \leftarrow S \cup \{v_i\}$;
9: Remove all random RR sets covered by v_i from R;
10: return S;

Obviously, the verify the correctness of the above algorithm, it is only needed to show that the random RR set obtained is a proper estimator of $\mathbb{E}[F_S]$ and θ can take a large enough size such that the quality of the final seeding node set can be guaranteed.

Theorem 3. Given a set $S \subseteq V$, for a random RR set e of $Q(V)$, we have $Pr[e \cap S \neq \varnothing] = \frac{\mathbb{E}[F_S]}{|Q(V)|}$.

Proof. According to the definition of $\mathbb{E}[F_S]$, given a special possible graph $\hat{G}$ let $F_S^{\hat{G}}$ be the influenced node size of S, then we have

$$\mathbb{E}[F_S] = \sum_{\hat{G} \sim \mathcal{G}} \left(Pr\left[\hat{G}\right] \cdot F_S^{\hat{G}} \right) \tag{9}$$

$$= \sum_{\hat{G} \sim \mathcal{G}} \left(Pr\left[\hat{G}\right] \cdot \sum_{v \in Q(V)} \mathbf{I}[v \in I_S] \right) \tag{10}$$

$$= \sum_{\hat{G} \sim \mathcal{G}} \left(Pr\left[\hat{G}\right] \cdot \sum_{v \in Q(V)} \mathbf{I}[e_v \cap S \neq \varnothing] \right) \tag{11}$$

$$= \sum_{v \in Q(V)} \sum_{\hat{G} \sim \mathcal{G}} \left(Pr\left[\hat{G}\right] \cdot \mathbf{I}[e_v \cap S \neq \varnothing] \right) \tag{12}$$

$$= \sum_{v \in Q(V)} Pr[e_v \cap S \neq \varnothing] \tag{13}$$

$$= |Q(V)| \sum_{v \in Q(V)} \frac{Pr[e_v \cap S \neq \varnothing]}{|Q(V)|}. \tag{14}$$

Since the starting node v of $Q(V)$ is randomly selected, for each of them the probability of being selected is $\frac{1}{|Q(V)|}$. In addition, the selection of S and v is independent from each other. Therefore, we have $Pr[e \cap S \neq \varnothing] = \sum_{v \in Q(V)} \frac{Pr[e_v \cap S \neq \varnothing]}{|Q(V)|}$. Finally, we have $Pr[e \cap S \neq \varnothing] = \frac{\mathbb{E}[F_S]}{|Q(V)|}$.

Theorem 4. If θ is at least $(8 + 2\epsilon) \cdot |Q(V)| \cdot \frac{\ln \frac{1}{\delta} + \ln\left(\frac{|V|}{k}\right) + \ln 2}{OPT_k^Q \cdot \epsilon^2}$, the RIS-MSTIM method will return a solution with $1 - 1/e - \epsilon$ approximation ratio with high probability $1 - \delta$, where OPT_k^Q is the largest influence of k seeding set in the MSTIM problem.

Proof. Let ρ be the probability $Pr[e \cap S \neq \varnothing]$ and X_i be a Bernoulli variable defined based on ρ, considering the sum of all X_is corresponding to the generated RR sets, we only need to guarantee the following condition.

$$Pr\left[|X_i \cdot |Q(V)| - \mathbb{E}[F_S]| \geq \frac{\epsilon}{2} \cdot OPT_k^Q \right] \leq \frac{\delta}{\binom{|V|}{k}} \tag{15}$$

$$\Leftrightarrow Pr\left[\left| \sum_{i=1}^{\theta} X_i - \rho\theta \right| \geq \frac{\epsilon}{2} \cdot OPT_k^Q \cdot \frac{\theta}{|Q(V)|} \right] \leq \frac{\delta}{\binom{|V|}{k}} \tag{16}$$

Then, similar with the analysis in [22], it can be shown that when $\theta \geq (8 + 2\epsilon) \cdot |Q(V)| \cdot \frac{\ln \frac{1}{\delta} + \ln\left(\frac{|V|}{k}\right) + \ln 2}{OPT_k^Q \cdot \epsilon^2}$, the RIS-MSTIM method will return a solution with $1 - 1/e - \epsilon$ approximation ratio with high probability $1 - \delta$.

6 INDEX-BASED SOLUTION FOR MSTIM

6.1 Indexing for the Range Query

Since the query Q required is holistic, the result $Q(V)$ cannot be predicated well. For the step 2 of the conceptual level algorithm RIS-MSTIM, the query result $T = Q(V)$ is extracted to help the

following sampling steps, however, evaluating the query Q may be an expensive procedure because of the multi-selection predicates [25].

A possible solution is to utilize sophisticated index to improve the query performance, such as R-Tree [26], k-d Tree [27] and so on. However, the above indexes will cause large storage overhead and usually the selection time cost using the index will increase as storage cost increases, especially when the data distribution is seriously skewed. For the worst cases, even the scanning method may become more time and space efficient [28].

Since the evaluation of the range queries is a preprocess of the whole RIS-MSTIM algorithm and most space cost will be expected to be improve the sampling performance as shown by the follows, inspired by the method used by [28], an adaptive indexing method for the range queries are utilized here to improve the performance of evaluating range queries.

There are mainly two index structures, resultPool and queryPool, utilized to improve the performance of the range query evaluation.

6.1.1 The ResultPool Index Structure

The resultPool index maintains the information about the query results of the queries processed previously, whose function is to provide a physical cache for the results such that the queries selecting a subset of some previous query can be processed efficiently. Assuming that each query can be identified by an unique queryID, as shown in **Figure 4**, for each q_i, four parts of information are maintained in resultPool.

a) The query statement is the formal representation of the query q_i.
b) The query results are the IDs of the nodes in V which belong to the set $q_i(V)$, and the nodes are listed in the ascending order of their IDs. Furthermore, the node set V can be stored in an array indexed by the nodeIDs, such that the attribute values of some node v can be accessed in constant time cost using the ID of v.
c) The dimension size k represents how many predicates are utilized in q_i.
d) The result size is the size of $q_i(V)$.

The queries stored in resultPool come from two parts. One part includes the queries processed before, and the other part includes some queries maintained in previous, which are represented by q_is and q_i^*s, respectively. Intuitively, since the queries appear in an ad-hoc way, if only q_is are maintained, a totally new query will cause poor performance, therefore, some typical queries which can improve the performance of more general queries are also maintained. The details of how to choose the queries q_i^*s will be introduced in the following.

Example 3. Continue with Example 2, given the graph shown in **Figure 2**, as shown in **Figure 4**, the index structure related with a special query history $\{q_1, q_2, \dots\}$ contains two parts, q_is and q_i^*s. The query q_1 is requested by the users before, according to *resultPool*, it can be known that 1) q_1 is a 2-dimensional range query whose statement is $(A_2 = NY) \wedge (20 \le A_3 \le 30)$, and 2) the result is $q_1(V) = \{a\}$. The query q_1^* do not need to be requested by the previous users, but the related information is still maintained. According to resultPool, it can be know that 1) q_1^* is a 1-dimensional range query with statement $0 \le A_1 \le 20$, and 2) the result set $q_1^*(V)$ is of size 1 and only contains the node a.

6.1.2 The QueryPool Index Structure

Algorithm 2. IndexOperations.

```
1:  function SearchQueryPool(Q)
2:      candidate ← ∅;
3:      for i from |Q| to 1 do
4:          for each predicate q ∈ Q do
5:              Search and add all keys covering q of queryPool with k = i into MatchKeys;
6:          for each keyid kid in MatchKeys do
7:              Initialize a pointer p_kid to the first query of the corresponding qlist;
8:          while there are at least i items in MatchKeys do
9:              qID ← the query with smallest ID among all queries pointed by p_kid s;
10:             cnt ← 0;
11:             for each keyid kid in MatchKeys do
12:                 if p_kid ==qID then
13:                     cnt ← cnt + 1;
14:                     if p_kid is the last one in qlist then
15:                         Remove kid from MatchKeys;
16:                     else
17:                         Move p_kid to the next query;
18:             if cnt ≥ i then
19:                 candidate ← candidate ∪ qID;
20:         if candidate ≠ ∅ then
21:             break;
22:     if candidate ≠ ∅ then
23:         Remove redundant queries from candidate;
24:         return An arbitrary one of candidate;
25:     else
26:         return null;
```

The resultPool makes it possible to utilize previous query answers to accelerate the evaluation of current query. To make it work, given a special query q, it still needs to support efficient extraction of helpful queries of q from all queries maintained by resultPool. Before introducing the index structure queryPool with the above abilities, the concepts of query covering and redundant queries are explained first.

Given two range queries q_1 and q_2, q_1 is contained by q_2, denoted by $q_1 \subseteq q_2$, if for every data instance D there is $q_1(D) \subseteq q_2(D)$. Specially, if $q_1 \subseteq q_2$ and q_2 is a 1-dimensional range query, a.k.a. a predicate, we say q_2 covers q_1. For a set of queries $\{q_1, \dots, q_n\}$, if there is $q_i \subseteq q_j$ $(i \neq j)$, q_j is called to be a redundant one in the following.

The queryPool index maintains the relations between keys and queries, where a key is some predicate utilized in the queries. For all queries q_is previously processed, queryPool organizes those predicates into groups by the dimension size. Each key is assigned with an unique keyid. For each special key p, qlist maintains the queryIDs of the queries which contain p. Moreover, the queryIDs are sorted into the ascending order and stored, which will facilitate the process of searching queries. The following example will explain how to search the related queries utilizing queryPool.

Example 4. Continue with Example 3, the corresponding queryPool index is shown in **Figure 5B**. The rows with $k = 2$ maintains the information about the 2-dimensional queries previously used. For the predicate with statement $A_2 = NY$, its keyid is *k2* and the corresponding *qlist* contains two queries q_1 and q_3, which are sorted in the ascending order. For the predicate $A_3 \in [20, 30]$, although it appears in both q_1 and q_5, its corresponding qlist only contains q_1, because q_1 is a 2-dimensional query but q_5 is not. The related information for the queries q_i^*s are stored in the group with $k = 0$ of queryPool.

queryID	query statement	results	$k =$ dimension size	result size
q_1	$A_2 = NY,\ A_3 \in [20, 30]$	a	2	1
q_2	$A_1 = M,\ A_2 \in \{NY, GA\}$	a, e, f, h	2	4
q_3	$A_3 \in [25, 50],\ A_2 = NY$	a, e, g	2	3
q_4	$A_3 \in [25, 50],\ A_4 \in [10000, 20000]$	d, e, f, g, h	2	5
q_5	$A_3 \in [20, 30]$	a, c	1	2
q_6	$A_4 \in [10000, 20000]$	a, b, c, d, e, f, g, h	1	8
...	...	...	...	...
q_1^*	$A_3 \in [0, 20]$	a	1	1
q_2^*	$A_3 \in (20, 40]$	c, d, e, f	1	4
q_3^*	$A_3 \in (40, 60]$	b, g, h	1	3
q_4^*	$A_3 \in (60, 80]$	–	1	0
q_5^*	$A_3 \in (80, 100]$	–	1	0
...	...	...	...	...

FIGURE 4 | Index structure resultPool of queries.

6.1.3 The SearchQueryPool Procedure

Next, the principles of using the two index structures to improve the performance of range query evaluation are introduced.

The SEARCHQUERYPOOL function is shown in **Algorithm 2**, which will return a query Q' for given a query Q such that the result $Q(V)$ can be obtained efficiently based on $Q'(V)$. First, a set candidate is initialized to be empty (line 2), which will be used to store the candidates of Q'. Then, an iteration from the group with $k = |Q|$ to the one with $k = 1$ is done to generate the queries in candidate (line 3–21). For each fixed $i \in [1, |Q|]$, a merge-style method is used to extract the queries containing Q efficiently by the following steps. (a) The keys covering some predicate of Q are collected, and a pointer is initialized at the front of the corresponding qlist for each key found (line 4–7). The details of obtaining keys covering some predicate will be introduced later. (b) Using the pointers initialized above, the merge-style method works by counting the appearing times of the current smallest queryID and inserting the query appearing no less than i times to the candidate set (line 8–19). At the end of each iteration, if the candidate is not empty, the iterations will stop. Finally, after the iterations, if the candidate set is not empty, an arbitrary non-redundant query in candidate is returned (line 22–24). Otherwise, null is returned (line 26).

As shown in **Figure 5B**, to efficiently search the keys covering some predicate, for the numerical attribute we can use a tree based search structure to achieve a $O(\log n)$ time cost for search operation where n is the number of distinct values appearing in the query predicates, and for the category values a hash table can be utilized to achieve an expected $O(1)$ time cost.

- For each numerical attribute A, a standard binary search tree is built. The search keys are chosen from the set of all distinct values appearing in the queries. In each leaf node with search key m, the keyID of each predicate q_A over A satisfying $m \in q_A$ is stored. For example, as shown in **Figure 5B**, the leaf node with search key 25 includes K4 and K5 whose corresponding predicates in the *queryPool* are $A_3 \in [20, 30]$ and $A_3 \in [25, 50]$ respectively. Then, given a predicate with range *[lbound, rbound]*, the leaf node *lnode* is obtained by finding the smallest search key no less than *lbound*, and the leaf node *rnode* is obtained by finding the largest search key no larger than *rbound*. Finally, the intersection of the two keyID sets associated with *lnode* and *rnode* is returned.
- For each category attribute A, a standard hash map is built by using the values as the hashing keys. Similarly, each item x obtained by the hash map is associated the keyIDs of the predicates in the form $A = x$. For example, as shown in **Figure 5B**, the hashed item of "NY" is associated with K2 and K3, which are keyIDs of predicates $A_2 = NY$ and $A_2 \in \{NY, GA\}$.

Example 5. Continue with Example 4, given a query Q: $(A_2 = NY) \wedge (25 \le A_3 \le 40)$, the candidate queries which contain Q can be found as follows. First, Q can be divided into two predicates q_1: $A_3 \in [25, 40]$ and q_2: $A_2 = NY$. Then, for the group of keys with $k = 2$ in the queryPool index, as shown in **Figure 5B**, q_1 can be processed in the search tree structure with keys 25 and 40, and the query obtained is $\{K4,K5\} \cap \{K5\} = \{K5\}$. Similarly, the queries obtained for q_2 are $\{K2K3\}$ using the hash map structure. After that, the corresponding qlists of $\{K2,K3,K5\}$ in queryPool are extracted and merged as the following steps.

$$\begin{aligned}\{\underline{q_1}\, q_3\}\{\underline{q_2}\}\{\underline{q_3}\, q_4\} &\Rightarrow \{\underline{q_3}\}\{\underline{q_2}\}\{\underline{q_3}\, q_4\} \Rightarrow \{\underline{q_3}\}\{\underline{q_3}\, q_4\} \\ &\Rightarrow \{\}\{\underline{q_4}\} \Rightarrow \{\}\{\}\{\} \end{aligned} \tag{17}$$

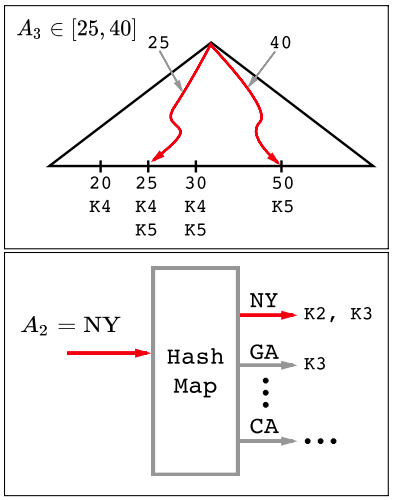
Here, the underline indicates the position of the related pointer for each list. During the merge procedure, the query q_3 will be inserted into the candidate set, since in the third step q_3 appears at the front of two lists. To verify the correctness, it can be found

A

k	keyid	key	qlist
2	K1	$A_1 = M$	q_2
	K2	$A_2 = NY$	q_1 q_3
	K3	$A_2 \in \{NY, GA\}$	q_2
	K4	$A_3 \in [20, 30]$	q_1
	K5	$A_3 \in [25, 50]$	q_3 q_4
	K6	$A_4 \in [10000, 20000]$	q_4
1	K7	$A_3 \in [20, 30]$	q_5
	K8	$A_4 \in [10000, 20000]$	q_6
0	...	...	...
	–	$A_3 \in [0, 20]$	q_1^*
	–	$A_3 \in (20, 40]$	q_2^*
	–	$A_3 \in (40, 60]$	q_3^*
	–	$A_3 \in (60, 80]$	q_4^*
	...	...	...

Index structure queryPool of queries.

B



Index structures of predicates.

FIGURE 5 | Index structures for the range queries.

that the statement of q_3 is $(A_3 \in [25, 50]) \wedge (A_2 = NY)$ and obviously we have $Q \subseteq q_3$.

6.1.4 Index Based Range Query Evaluation

Algorithm 3. *IndexRQE* (Index Based Range Query Evaluation).

Input: A network graph $G = (V, E, A)$, a k-dimensional query Q
Output: The result $Q(V)$
1: function IndexRQE(G, Q)
2: $q \leftarrow$ SearchQueryPool(Q);
3: $qres \leftarrow \emptyset$;
4: if $q \neq$ null then
5: $qres \leftarrow$ resultPool[q].results;
6: else
7: Build the set $\{\delta_A = \frac{|rbound - lbound|}{|Dom_A|} | (lbound \leq A \leq rbound) \in Q\}$;
8: Let X be the attribute with the smallest δ_X;
9: Let $[l_x, r_x]$ be the predicate over X in Q;
10: for each key $t : X \in [l_t, r_t]$ in the group with $k = 0$ in queryPool do
11: if $[l_x, r_x] \cap [l_t, r_t] \neq \emptyset$ then
12: $qres = qres \cup$resultPool[t.qlist].results;
13: for each result node $v \in qres$ do
14: if v does not satisfy any predicate in Q then
15: continue;
16: $S \leftarrow S \cup \{v\}$;
17: if $|S|/|qres| < \alpha$ then
18: InsertQuery(Q, S);
19: return S;

Now, we will show how to obtain the exact result of a given query based on the techniques shown above. As shown in **Algorithm 3**, given a k – *dimensional* query Q, the function SEARCHQUERYPOOL is first invoked to search a candidate query set q whose item will contain the query Q (line 2). The structure *qres* is utilized to maintain a superset of $Q(V)$ and initialized to be empty (line 3). If q is not null, the results of q will be collected into *qres* (line 5), otherwise, *qres* will be built based on the queries q_i^*s as follows (line 7–12). The selectivity δ_A of each attribute A involved in Q is calculated based on the assumption that all appearing values of A are chosen in an uniform random way. Here, δ_A is defined to be $\frac{|rbound - lbound|}{|Dom_A|}$, where *lbound* and *rbound* is the range of A in Q and Dom_A is the domain size of A. Intuitively, δ_A can tell us how many items can be filtered using the predicate of A in Q. The attribute X with the smallest δ_X will be chosen (line 8), since it is expected to filter the largest part of the data. Then, the predicates of X maintained in the group with $k = 0$ in queryPool will be scanned, and the results of the predicates having intersections with Q will be collected together into *qres* (line 9–12). Then, the items in *qres* will be checked one by one to obtain $S = Q(V)$ (line 13–16). Finally, if $\frac{|S|}{|qres|}$ is smaller than a predefined threshold α, the query Q will be inserted into the index resultPool and queryPool (line 17–18), where the details are omitted here since it can be implemented trivially. Essentially, the value $\frac{|S|}{|qres|}$ can represent the ratio of truly useful items in the set *pres*, and a smaller value means that more useless items are collected in the previous step.

Example 6. Continue with Example 5, suppose the query Q': $A_3 \in [10, 40]$ is given. Obviously, there are no predicates covering Q', therefore, the function SearchQueryPool(Q') will return null. Then, according to the function IndexRQE, the results of q_1^* and q_2^* maintained in resultPool will be collected into *qres*. Finally, only the nodes *a, c, d, e* and *f* will be checked, but not all nodes in V.

6.2 Indexing for the Random RR Sets

6.2.1 The Indexing Structures

To maintain the related information of random RR sets, three index structures are utilized, *NodeToRR*, *RRSet*, and *RRInvertedList*.

In the RRSet, for each random RR set rr_i, there is an unique RRid, and all nodes contained in rr_i are stored as nodeList. Within RRSet, the items are maintained in the ascending order of RRid. In the NodeToRR, for each node $v \in V$, the nodeID of v is stored and the corresponding rrList includes the list of RRids of the random RR sets which are obtained by sampling from node v. In

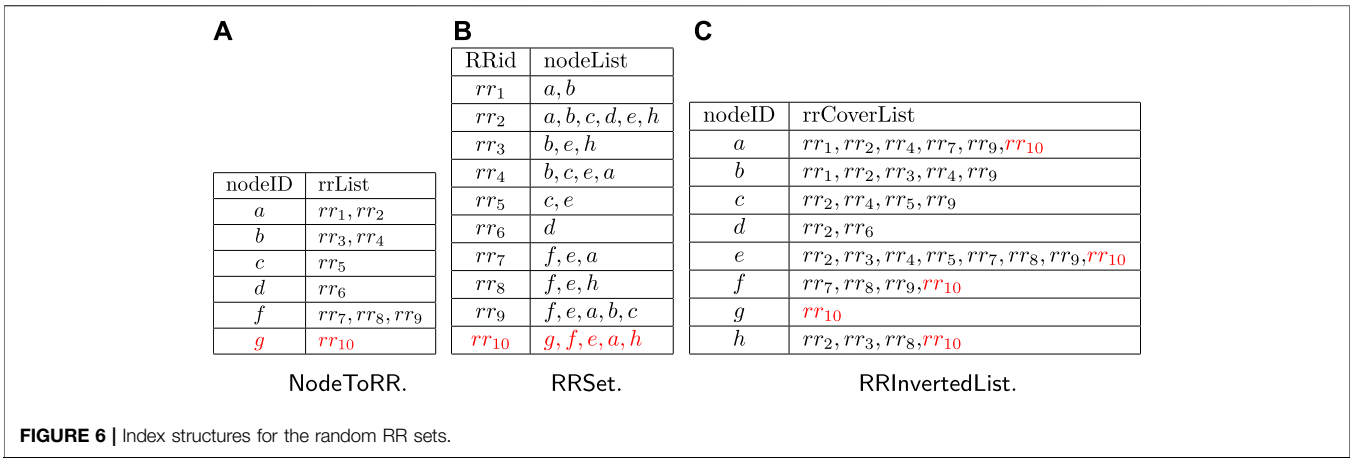


FIGURE 6 | Index structures for the random RR sets.

the RRInvertedList, for each node $v \in V$, the nodeID of v is stored and the corresponding rrCoverList maintains the list of RRids of the random RR sets which cover the node v.

Example 7. As shown in **Figure 6B**, a set of samples are listed in the RRSet, where there are 9 samples in total and the first random RR set is $rr_1 = \{a, b\}$. According to the information in RRSet, as shown in **Figure 6A**, the NodeToRR structure maintains the list of samples beginning from some special node. There are totally two samples rr_1 and rr_2 in RRSet beginning from the node a, then, the corresponding rrList of nodeID a includes rr_1 and rr_2. In the RRInvertedList structure, the rrCoverList of nodeID a includes 5 items rr_1, rr_2, rr_4, rr_7 and rr_9, each of them contains the nodeID a in the corresponding nodeList.

6.2.2 Adaptive Sampling Using Index

Algorithm 4. AdaptiveSampling.

Input: A node set T, the network graph $G = (V, E, A)$, and a sampling size θ
Output: A set R of sampled random RR sets.
1: function AdaptiveSampling(G, T, θ)
2: $R \leftarrow \emptyset$;
3: for each node $v \in T$ do
4: $cnt_v \leftarrow 0$;
5: for i from 1 to θ do
6: Randomly select a node v from T;
7: $cnt_v + +$;
8: for each node $v \in T$ do
9: if NodeToRR[v].rrList.size() $\geq cnt_v$ then
10: Randomly select cnt_v samples from NodeToRR[v].rrList and add them to R;
11: else
12: $\Delta_v \leftarrow$ NodeToRR[v].rrList.size() $- cnt_v$;
13: Generate Δ_v random RR sets for v, and insert them to NodeToRR[v].rrList;
14: Update the RRInvertedList;
15: Add all samples in NodeToRR[v].rrList to R;
16: return R;

In this part, we will explain how to sample the random RR sets adaptively under the help of indexes introduced above. As shown in **Algorithm 4**, the inputs of ADAPTIVESAMPLING include the network graph G, a target node set T and a sampling size θ, and the output R is a sample set of random RR sets. For each node $v \in T$, a variable cnt_v will be used to record the number of samples needed starting from v and initialized to be 0 (line 3–4). Then, θ nodes are randomly selected from T with replacement, different from the commonly used sampling methods this step does not start the random walking but just increases the counter variable cnt_v to remember that one sample is needed (line 5–7). After that, we will know the number of samples needed for each node, all left we need to do is to reuse the samples maintained in RRSet to build R and collect more samples adaptively by random walking when there are no enough samples in RRSet. The details can be explained as follows. For each node $v \in T$, if NodeToRR *[v]*.rrList.size() is as large as cnt_v, the samples maintained in RRSet is enough and no further real sampling operations are needed (line 9–10). Otherwise, we use Δ_v to represent the difference between NodeToRR*[v]*.rrList.size() and cnt_v (line 12), and Δ_v times random walking will be executed to collect extra samples needed which will be inserted into NodeToRR at the same time (line 13). After updating the RRInvertedList (line 14), those new samples will be inserted into R (line 15). Since the RRInvertedList is a commonly used inverted list, the update can be implemented in a natural way whose details are omitted here.

Example 8. Let us consider the case that cnt_g and cnt_f are assigned to be 1 and 2, respectively, when running AdaptiveSampling. As shown in **Figure 6**, the information shown in black is the index structures before invoking AdaptiveSampling. Since $cnt_g = 1$ and there are no items with *nodeID* = *g*, one random walking will be made to obtain a new sample rr_10. Then, the parts in **Figure 6** shown in red will be added. For the node *f*, since $cnt_f = 2$ and there are three *RRid*s in the rrList of the item with *nodeID* = *f* in NodeToRR, no more random walking are needed, AdaptiveSampling will choose two samples from $\{rr_7, rr_8, rr_9\}$.

TABLE 1 | Statistics of graph datasets.

Dataset	Type	#Vertices	#Edges
google	Social network	23,628	39,242
wikivote	Social network	7,115	103,689
twitter	Social network	465,017	834,797
youtube	Social network	1,138,499	4,942,297

6.3 RIS-MSTIM-Index Algorithm

Finally, we introduce the main procedure of index based solution for the MSTIM problem. As shown in **Algorithm 5**, the RIS-MSTIM-

INDEX function is the index version of **Algorithm 1**. Different from **Algorithm 1**, RIS-MSTIM-INDEX utilizes the INDEXRQE method to collect the results of *Q(V)* (line 1). Then, during the sampling procedure, ADAPTIVESAMPLING is invoked to obtain the sample set with enough random RR sets (line 4). After that, to utilize the RRInvertedList structure to compute the greedy based approximation solution of the optimal k seeds, for each node v maintained in RRInvertedList(nodeID), a list structure $rrTmpList_v$ is used to store the random RR sets during the computation (line 7–13). There are k round in total (line 7–12), each of them chooses the current node with maximum *RR* sets covered. Within one round, for each node v, the size of its *rrCoverList* can measure how many RR sets it can cover in the left, therefore, the one with largest *rrCoverList.size*() is chosen to be the seed node in that round (line 8, 12). Since the *rrCoverList* needs to be maintained dynamically to show the number of RR sets each node can cover, within each round, if some node v_i is chosen, the RR sets it covers should be removed from the *rrCoverList* and maintained temporally in *rrTmpList* (line 9–11). Finally, before returning the final seed set (line 14), the RRInvertedList is restored by merging with *rrTmpList* (line 13).

Algorithm 5. RIS-MSTIM-Index.

Input: A network graph $G = (V, E, A)$, a k-dimensional query Q and an integer k
Output: A seeding node set S
1: function RIS-MSTIM-Index(G, Q, k)
2: $T \leftarrow$ IndexRQE(G, Q);
3: $R \leftarrow \emptyset$;
4: $R \leftarrow$ AdaptiveSampling(G, T, θ);
5: $S \leftarrow \emptyset$;
6: Initialize an $rrTmpList_v$ to be $\emptyset$ for each $v \in$ RRInvertedList[$nodeID$];
7: for $i \in [1, k]$ do
8: $v_i \leftarrow$ the node with largest $rrCoverList$.size() in RRInvertedList;
9: Move all items of RRInvertedList[v_i].$rrCoverList$ to $rrTmpList_{v_i}$;
10: for each node $u \in$ RRInvertedList[$nodeID$] do
11: Move all items in (RRInvertedList[u].$rrCoverList$) $\cap$ ($rrTmpList_{v_i}$) to $rrTmpList_u$;
12: $S \leftarrow S \cup \{v_i\}$;
13: Restore RRInvertedList by merging with $rrTmpList$[];
14: return S;

7 EXPERIMENTAL EVALUATION

In this part, experiments on real datasets are conducted to evaluate the efficiency and performance of the targeted influence maximization algorithm defined based on multidimensional queries.

7.1 Experiment Setup

We ran our experiments on four real datasets, Google, Youtube, Twitter, and WikiVote, which are collected from Konect (http://konect.uni-koblenz.de/) and SNAP (http://snap.stanford.edu/data/) respectively. All of them are social network datasets. Typically, the characteristic information of the four datasets are shown in **Table 1**. In these four social networks, nodes represent users and edges represent friendships between users. All experiments were executed on a PC with 3.40 GHz Intel Core i7 CPU and 32 GB of DDR3 RAM, running Ubuntu 20.04.

We compare two versions of the influence maximization algorithm proposed. For RIS algorithm, it means that the baseline method shown in **Algorithm 1**, where a trivial method is utilized to obtain the query result first and the commonly used RIS method is utilized to solve the corresponding targeted influence maximization problem. While, for RIS-MSTIM-Index algorithm, it means that the index version of **Algorithm 1**, which utilizes the resultPool to improve the performance of query evaluation and the RRSet index to save the sampling costs. There have been several sophisticated influence maximization algorithms (such as [1,13,22,29]) within the framework of RIS. In this paper, we implement the method in [13] and use it as the standard RIS algorithm. In fact, the method proposed by [1] is the state of the art, which is denoted by BCT here. There are two reasons that we choose [13] as the standard RIS method. First, the method in [13] is simple and easy to be integrated to the framework of selection based targeted influence maximization. Second, the cost of computing seed nodes for targeted influence maximization is highly dominated by the cost of evaluating multi-dimensional queries, choosing different RIS methods does not produce a significant impact on the total cost, as verified by the following experimental results.

For each dataset, the dimensions and values of the nodes are generated randomly. For each node, 5 category dimensions and 10 numerical dimension are associated and the corresponding values are randomly selected in an uniform way within the value domain. To evaluate the query based targeted influence maximization algorithm, multidimensional range queries are generated in the following way. For each dataset, 10 group of queries are generated randomly, within each group, there are 50 queries in total. The dimension number of each query is chosen between 1 and 5 according to a normal distribution, most of the queries have three or four predicates. A query is only allowed to contain at most one predicate on each dimension. The range predicates are also randomly generated by controlling the selectivities to be about 20%. Usually, to be simple, we use the number of queries and random RR sets to control the index size. It should be remarked that the controlling method is not an accurate way since the size of each query or sample is not known, but it can avoid complex calculating the index size which may affect the performance of the main algorithm.

7.2 Experimental Results and Analysis

The experimental results are conducted to verify the efficiencies of the influence maximization algorithm proposed from several aspects.

7.2.1 Efficiency of RIS-MSTIM Algorithm

To study the efficiencies of the RIS-MSTIM algorithm proposed, for each real dataset, both the RIS-MSTIM algorithms with and without indexes are executed to solve the targeted influence maximization problem. To measure the performance of RIS-MSTIM fairly when considering different queries, 10 range query groups, each of which contain 50 queries, are generated randomly. For each group of queries, RIS-MSTIM algorithm is invoked to solve the corresponding targeted influence maximization problem. Within each group, the performance of RIS-MSTIM will become better as the increase of the number of queries processed, assuming that there are enough space costs to maintain the corresponding indexes. For each same

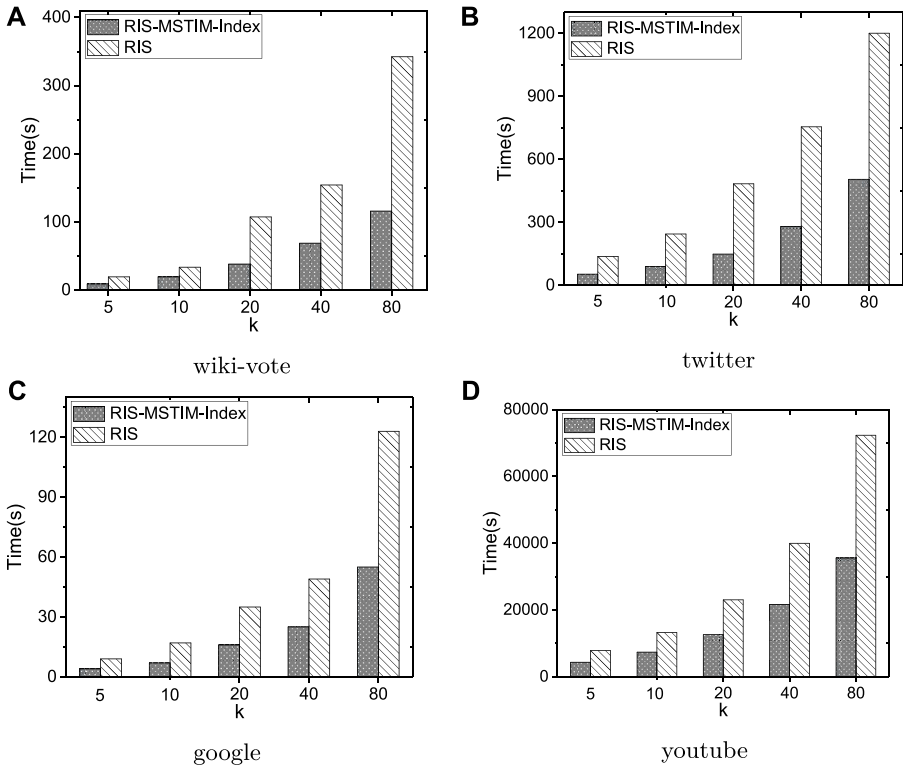


FIGURE 7 | The average time cost of targeted influence maximization algorithm for evaluating the randomly generated query groups over four datasets.

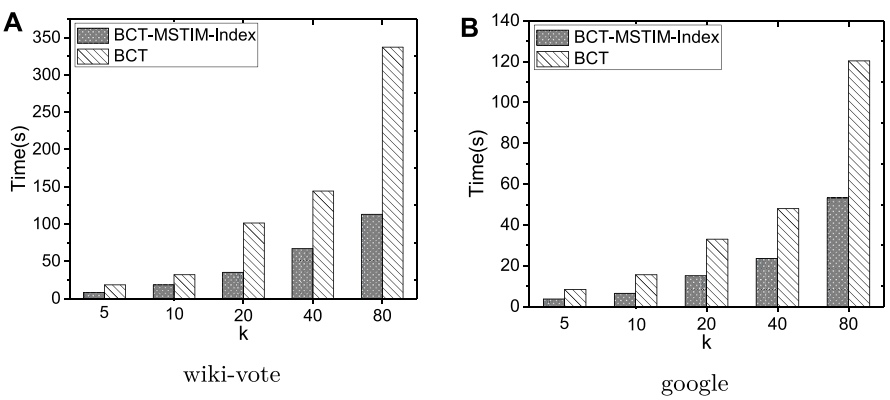


FIGURE 8 | The average time cost of BCT based targeted influence maximization algorithm for evaluating the randomly generated query groups over two datasets.

setting, the algorithm is ran 5 times and the average time costs are recorded.

As shown in **Figure 7**, executing the RIS-MSTIM algorithms with and without indexes over the four datasets, when the seed size k is increased from 5 to 80, the average time costs for evaluating each group of queries generated are reported. Here, for the previous three datasets, the average time cost of 10 group of queries are reported, while for the youtube dataset the average time cost of 3 group of queries are reported since it will take much longer time than other datasets. Based on the above results, we have two observations. The first one is that as the seed size increases in an exponential speed both the index and no-index versions of RIS-MSTIM can scale well, where it should be noted that the seed size is enlarged twice each time. The second one is that using the indexes the RIS-MSTIM algorithm performances much better, where the indexes can help the reduce the time costs to about 50% for all datasets. It is verified that the index based idea of improving the performance of RIS-MSTIM is effective and can be utilized in rather diverse settings for which within the experiments there are about 500 queries and five different seed

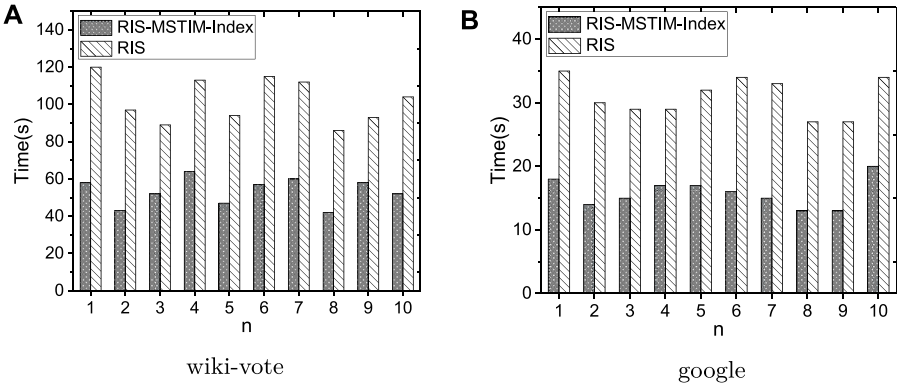


FIGURE 9 | The time costs of each query group over wiki-vote and google datasets.

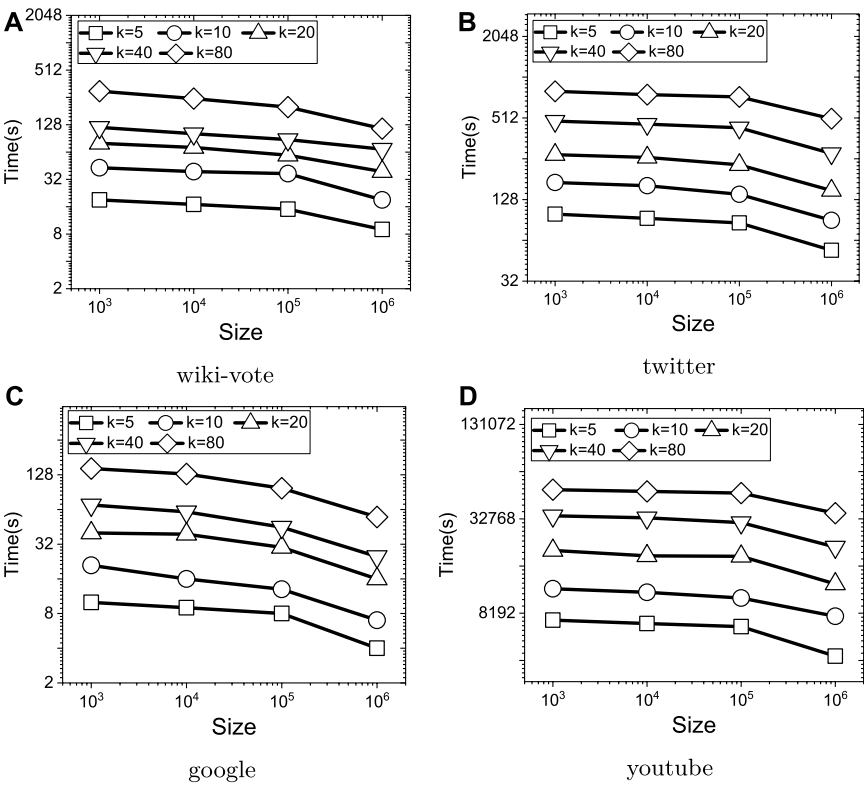


FIGURE 10 | The effects of adjusting the index sizes.

sizes. Moreover, since the datasets used here have different characteristics, it has been also verified that our index based method is proper for different types of data.

Also, it is verified that the index based solution can be applied to other sophisticated method within RIS framework also. As shown in **Figure 8**, using BCT method in [1] which is the state of the art and the same setting, for wiki-vote and google datasets, the average time costs of BCT methods with and without index are compared. It can be observed that the index solution proposed by this paper can improve the performance of BCT based method significantly also. The second key observation is that, although BCT is better than the standard RIS method, the total time costs caused by BCT and RIS methods are nearly same. The reason is that the cost of performing targeted influence maximization is highly dominated by the cost of evaluating multi-dimensional queries.

As shown in **Figure 9**, fixing the seed size to be 20, the time costs for evaluating each query group are reported. In **Figure 9A**, over the wiki-vote dataset, the results for 10 groups of queries are shown, where as discussed above each of the group contains 50

TABLE 2 | Space costs of indexes.

Index type	$k = 5$	$k = 10$	$k = 20$	$k = 40$	$k = 80$	Dataset
query-result	1.54 MB	1.96 MB	1.54 MB	1.68 MB	1.69 MB	google
random-RR	134 MB	163 MB	182 MB	183 MB	184 MB	
query-result	448 KB	565 KB	639 KB	538 KB	537 KB	wiki-vote
random-RR	29.5 MB	64.5 MB	164 MB	163 MB	169 MB	
query-result	101 MB	104 MB	105 MB	106 MB	115 MB	twitter
random-RR	434 MB	437 MB	432 MB	445 MB	434 MB	
query-result	307 MB	312 MB	310 MB	340 MB	320 MB	youtube
random-RR	6.0 GB	5.8 GB	5.9 GB	5.9 GB	6.1 GB	

TABLE 3 | Influence of the seeds.

Dataset	SeedSize	RIS-MSTIM + index	RIS-MSTIM	QueryID
wiki-vote	5	20	20	q_{26}
	10	22	22	q_{24}
	20	133	134	q_{44}
	40	78	79	q_{17}
	80	195	193	q_{16}
google	5	2302	2305	q_{21}
	10	3275	3273	q_{6}
	20	2609	2618	q_{35}
	40	4050	4063	q_{45}
	80	4075	4039	q_{9}
twitter	5	17832	17824	q_{1}
	10	18484	18456	q_{18}
	20	3856	3883	q_{14}
	40	53494	53430	q_{1}
	80	23690	23708	q_{47}
youtube	5	4833	4824	q_{9}
	10	11246	11263	q_{26}
	20	6038	6097	q_{32}
	40	24179	24185	q_{4}
	80	8816	8834	q_{4}

queries and the label n of x-axis means the group id. Similarly, the experimental results over google dataset is shown in **Figure 9B**. It can be observed that over those two datasets the performance of RIS-MSTIM method can be improved by using indexes for all the query group randomly generated.

7.2.2 Effectiveness of Adjusting Index Sizes

To evaluate the effects of index sizes, the memory size used by the indexes are controlled by limiting the total number of queries and random RR sets cached by the indexes. The total size is changed between 1 and 1000 K, where each time it is increased by 10 times. For each dataset, the average time costs of evaluating the targeted influence maximization algorithm for the 10 groups of queries generated are recorded. As shown in **Figure 10**, when increasing the index size, the time costs over all four datasets are reduced. Since increasing the index size can enlarge the probability that a random chosen query share random RR sets with previous queries, such an observation is just what are expected. Therefore, it can be verified that the indexes using by the algorithm is the essential part for improving the performance of RIS-MSTIM, and when being allowed, the threshold for controlling index size should be assigned to a value as large as possible.

7.2.3 The Space Cost of Indexes

Intuitively, to let the index based method more general and usable, the space cost of the indexes used should be well controlled. Intuitively, when it is assumed that the memory is enough large to contain all possible index items, without considering the maintaining and searching costs of the indexes, the performance of the influence maximization algorithms can only become better when more items are indexed. Although, the maintaining and searching costs are relatively small comparing with the total time cost of RIS-MSTIM, it still needs huge space cost to store the sampled random RR sets temporally. If the indexes consume too many space costs, there may be only few space to store the new samples needed and the algorithm will perform very bad because of no enough memory space. In this part, fixing the index control size as 1000 K, for the four datasets, we run RIS-MSTIM algorithm over them for the parameter settings $k \in \{5, 10, 20, 40, 80\}$. Then, the sizes of indexes for query results and random RR sets are reported. As shown in **Table 2**, the space cost of RIS-MSTIM with index over the youtube dataset is the largest and the cost over the wiki-vote dataset is the smallest, which is expected based on the observation about the execution time costs. Generally speaking, when the seed size becomes larger, the space cost increases also. For the google and wiki-vote datasets, when the seed size is rather smaller, because that the space cost of all samples generated is still smaller than the threshold, the cost labelled by random-RR is significantly smaller than the case with larger k value. Moreover, it can be observed that the cost of query-result is much smaller than the cost of random-RR, which is as expected since each random RR set is essentially a node set.

7.2.4 Comparison of Influence Obtained

Since the two versions of the RIS-MSTIM algorithm are the same one in principle, the effects of solving targeted influence maximization problem should be the same also. However, the key idea utilized in the index version is to reuse the samples obtained before, therefore, the detailed execution of the two RIS-MSTIM algorithms may be different. In this part, we should verify that the difference discussed above is very tiny such that we can ignore them when considering the qualities of the solutions obtained.

Since for each parameter setting, 50 queries in total are ran, we randomly select one query for each setting, record the seed node set, evaluate and compare their corresponding influence obtained. The results are shown in **Table 3**. It can be observed that there are tiny differences between the influence values obtained by RIS-MSTIM with and without indexes. This is as expected since the algorithm is a randomized one which only makes sure that a $(1 - 1/e - \epsilon)$ approximation solution is obtained with at least $(1 - \delta)$ probability. Also, it can be observed that the differences are acceptable for each dataset when considering the total dataset sizes.

8 CONCLUSION

In this paper, the problem of multidimensional selection based Targeted Influence Maximization is studied. The MSTIM problem is shown to be NP-hard even when the target set is rather small. The RIS framework is extended to the MSTIM case, based on a careful analysis of the sampling size, it is shown that the MSTIM problem admits a $1 - 1/e - \epsilon$ approximation algorithm based on reverse influence sampling. To answer the MSTIM problem efficiently, an index based solution is proposed. To improve the performance of evaluating multi-selection queries, an inverted list style index for query predicates is presented, and efficient index based query evaluation method is developed. To improve the performance of the sampling procedure, using the idea of sharing samples as much as possible, an adaptive sampling strategy based on index is introduced and the corresponding influence maximization algorithm is designed. The experimental results show that the method proposed is efficient.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: http://snap.stanford.edu/data/ http://konect.uni-koblenz.de/.

AUTHOR CONTRIBUTIONS

DJ completes the main work and updates the manuscript of this paper. TL gave the main idea of the key method, designed the study, and helped to draft the manuscript. All authors read and approved the final manuscript.

FUNDING

This work is supported by the National Key Research and Development Program of China (2018AAA0101901), the National Natural Science Foundation of China (NSFC) *via* Grant 61976073 and 61702137.

REFERENCES

1. Nguyen HT, Thai MT, Dinh TN. A Billion-Scale Approximation Algorithm for Maximizing Benefit in Viral Marketing. *Ieee/acm Trans Networking* (2017) 25:2419–29. doi:10.1109/tnet.2017.2691544
2. Epasto A, Mahmoody A, Upfal E. Real-time Targeted-Influence Queries over Large Graphs. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017; July 31 - August 03, 2017; Sydney, Australia (2017). p. 224–31. doi:10.1145/3110025.3110105
3. Li Y, Zhang D, Tan K-L. Real-time Targeted Influence Maximization for Online Advertisements. *Proc VLDB Endow* (2015) 8:1070–81. doi:10.14778/2794367.2794376
4. Domingos P, Richardson M. Mining the Network Value of Customers. In: KDD '01 (2001). p. 57–66. doi:10.1145/502512.502525
5. Richardson M, Domingos P. Mining Knowledge-Sharing Sites for Viral Marketing. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; New York, USA: ACM (2002). p. 61–70. KDD '02. doi:10.1145/775047.775057
6. Kempe D, Kleinberg J, Tardos E. Maximizing the Spread of Influence through a Social Network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; New York, USA: ACM (2003). p. 137–46. KDD '03. doi:10.1145/956750.956769
7. Chen W, Wang C, Wang Y. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In: KDD '10; New York, USA: ACM (2010). p. 1029–38. doi:10.1145/1835804.1835934
8. Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen J, Glance N. Cost-effective Outbreak Detection in Networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; New York, USA: ACM (2007). p. 420–9. KDD '07. doi:10.1145/1281192.1281239
9. Goyal A, Lu W, Lakshmanan LVS. Simpath: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model. In: ICDM '11; Washington, DC, USA, (2011). p. 211–20.
10. Chen W, Wang Y, Yang S. Efficient Influence Maximization in Social Networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; New York, USA: ACM (2009). p. 199–208. KDD '09. doi:10.1145/1557019.1557047
11. Chen Y-C, Peng W-C, Lee S-Y. Efficient Algorithms for Influence Maximization in Social Networks. *Knowl Inf Syst* (2012) 33:577–601. doi:10.1007/s10115-012-0540-7
12. Jiang Q, Song G, Cong G, Wang Y, Si W, Xie K. Simulated Annealing Based Influence Maximization in Social Networks. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press (2011). p. 127–32. AAAI'11.
13. Tang Y, Shi Y, Xiao X. Influence Maximization in Near-Linear Time. In: TK Sellis, SB Davidson, ZG Ives, editors. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne; May 31 - June 4, 2015; New York, USA: ACM (2015). p. 1539–54. doi:10.1145/2723372.2723734
14. Huang K, Wang S, Bevilacqua G, Xiao X, Lakshmanan LVS. Revisiting the Stop-And-Stare Algorithms for Influence Maximization. *Proc VLDB Endow* (2017) 10:913–24. doi:10.14778/3099622.3099623
15. Huang K, Tang J, Han K, Xiao X, Chen W, Sun A, et al. Efficient Approximation Algorithms for Adaptive Influence Maximization. *VLDB J* (2020) 29:1385–406. doi:10.1007/s00778-020-00615-8
16. Arora A, Galhotra S, Ranu S. Influence Maximization Revisited: The State of the Art and the Gaps that Remain. In: M Herschel, H Galhardas, B Reinwald, I Fundulaki, C Binnig, Z Kaoudi, editors. Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019; March 26-29, 2019; Lisbon, Portugal. Konstanz, Germany: OpenProceedings.org (2019). p. 440–3. doi:10.5441/002/edbt.2019.40
17. Arora A, Galhotra S, Ranu S. Debunking the Myths of Influence Maximization. In: S Salihoglu, W Zhou, R Chirkova, J Yang, D Suciu, editors. Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017; May 14-19, 2017; New York, USA: ACM (2017). p. 651–66. doi:10.1145/3035918.3035924

18. Li G, Chen S, Feng J, Tan K, Li W. Efficient Location-Aware Influence Maximization. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data; New York, USA: ACM (2014). p. 87–98. SIGMOD '14. doi:10.1145/2588555.2588561
19. Tang S, Yuan J, Mao X, Li X, Chen W, Dai G. Relationship Classification in Large Scale Online Social Networks and its Impact on Information Propagation. In: INFOCOM 2011 (2011). p. 2291–9. doi:10.1109/infcom.2011.5935046
20. Chen S, Fan J, Li G, Feng J, Tan K-l., Tang J. Online Topic-Aware Influence Maximization. *Proc VLDB Endow* (2015) 8:666–77. doi:10.14778/2735703.2735706
21. Zhang J, Tang J, Zhong Y, Mo Y, Li J, Song G, et al. Structinf: Mining Structural Influence from Social Streams. In: AAAI'17 (2017). p. 73–80.
22. Tang Y, Xiao X, Shi Y. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. In: SIGMOD 2014 (2014). p. 75–86.
23. Khuller S, Moss A, Naor J. The Budgeted Maximum Coverage Problem. *Inf Process Lett* (1999) 70:39–45. doi:10.1016/s0020-0190(99)00031-9
24. Li Y, Fan J, Wang Y, Tan K-L. Influence Maximization on Social Graphs: A Survey. *IEEE Trans Knowl Data Eng* (2018) 30:1852–72. doi:10.1109/tkde.2018.2807843
25. Gaede V, Günther O. Multidimensional Access Methods. *ACM Comput Surv* (1998) 30:170–231. doi:10.1145/280277.280279
26. Guttman A. R-trees: A Dynamic index Structure for Spatial Searching. In: SIGMOD'84, Proceedings of Annual Meeting; June 18-21, 1984. Boston, Massachusetts, USA (1984). p. 47–57.
27. Bentley JL. Multidimensional Binary Search Trees Used for Associative Searching. *Commun ACM* (1975) 18:509–17. doi:10.1145/361002.361007
28. Lamb A, Fuller M, Varadarajan R, Tran N, Vandiver B, Doshi L, et al. The Vertica Analytic Database. *Proc VLDB Endow* (2012) 5:1790–801. doi:10.14778/2367502.2367518
29. Borgs C, Brautbar M, Chayes J, Lucier B. Maximizing Social Influence in Nearly Optimal Time. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms; USA: Society for Industrial and Applied Mathematics. Philadelphia, PA. SODA '14 (2014). p. 946–57. doi:10.1137/1.9781611973402.70

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Physics

ORIGINAL RESEARCH
published: 03 January 2022
doi: 10.3389/fphy.2021.766615

Identifying Multiple Influential Spreaders in Complex Networks by Considering the Dispersion of Nodes

*Li Tao [1], Mutong Liu [2], Zili Zhang [1] and Liang Luo [1]**

[1]School of Computer and Information Science, Southwest University, Chongqing, China, [2]Department of Computer Science, Hong Kong Baptist University, Kowloon, Hong Kong, SAR, China

Identifying multiple influential spreaders, which relates to finding k ($k > 1$) nodes with the most significant influence, is of great importance both in theoretical and practical applications. It is usually formulated as a node-ranking problem and addressed by sorting spreaders' influence as measured based on the topological structure of interactions or propagation process of spreaders. However, ranking-based algorithms may not guarantee that the selected spreaders have the maximum influence, as these nodes may be adjacent, and thus play redundant roles in the propagation process. We propose three new algorithms to select multiple spreaders by taking into account the dispersion of nodes in the following ways: (1) improving a well-performed local index rank (LIR) algorithm by extending its key concept of the local index (an index measures how many of a node's neighbors have a higher degree) from first-to second-order neighbors; (2) combining the LIR and independent set (IS) methods, which is a generalization of the coloring problem for complex networks and can ensure the selected nodes are non-adjacent if they have the same color; (3) combining the improved second-order LIR method and IS method so as to make the selected spreaders more disperse. We evaluate the proposed methods against six baseline methods on 10 synthetic networks and five real networks based on the classic susceptible-infected-recovered (SIR) model. The experimental results show that our proposed methods can identify nodes that are more influential. This suggests that taking into account the distances between nodes may aid in the identification of multiple influential spreaders.

Keywords: identification of multiple influential spreaders, dispersion of nodes, location index rank algorithm, independent set algorithm, susceptible-infected-recovered model

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Jiajin Huang,
Beijing University of Technology, China
Kevin Du,
The University of Hong Kong, Hong Kong, SAR China

****Correspondence:***
Liang Luo
luoliang@swu.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *29 August 2021*
Accepted: *12 November 2021*
Published: *03 January 2022*

Citation:
Tao L, Liu M, Zhang Z and Luo L (2022) Identifying Multiple Influential Spreaders in Complex Networks by Considering the Dispersion of Nodes. Front. Phys. 9:766615. doi: 10.3389/fphy.2021.766615

1 INTRODUCTION

Many real-world problems involve the identification of multiple influential nodes in complex networks, such as finding a few individuals who are critical to the spread of information on the internet, or who may speed up the transmission process of pestilence in crowds once infected [1]. The problem of identifying multiple influential nodes differs from that of discovering the most influential nodes. The latter refers to finding the k ($k > 1$) most influential spreaders, which is commonly addressed by ranking the influence of individual nodes. The former involves the identification of a set of k nodes with the maximum influence as a whole. That is, identifying multiple influential nodes should take into account the different roles that nodes play in the propagation process rather than just evaluating their individual influence [2].

Methods to identify multiple influential spreaders fall in three categories. The first regards this as an influence maximization (IM) problem. Some well-known methods include the greedy [3], new greedy [4], community-based greedy [5], k-medoid [6], two-phase influence maximization [7], and collective influence [8] algorithms. However, as the IM problem is NP-hard, these algorithms are challenged by increasing network sizes, and thus are not applicable to huge real networks.

Methods in the second category attempt to identify multiple influential nodes by ranking their influences, which are calculated according to various topology-based centrality measures: 1) classic topological centrality metrics, such as degree centrality [9], betweenness centrality [10], and closeness centrality [10]; 2) centrality measures that take into account multiple (global or local) network features, such as KED centrality [11], efficiency centrality (EC) [12], composite centrality based on analytic hierarchy process [13], and classified neighbors centrality [14]; and 3) local-information-based iterative algorithms such as PageRank [15], LeaderRank [16], and VoteRank [17]. However, the ranking approach may not always find a set of nodes with the maximum influence [18], possibly because they separately measure the influence of each node, and thus omit overlapping effects of topologically adjacent top-ranked nodes.

Algorithms in the third category consider the distance between nodes when evaluating node importance. For instance, the local index rank (LIR) algorithm [19] is based on the local index (*LI*) value of a node, which represents the number of neighbors whose degree exceeds that of the focus node. Spreaders are selected from nodes whose *LI* values are 0 (i.e., 0-*LI* nodes). However, the LIR method cannot avoid some adjacent 0-LI nodes, and sometimes there are not enough 0-*LI* nodes to be selected as spreaders. Another example is the independent set (IS) algorithm [20], which divides nodes into independent sets by the Welsh-Powell coloring algorithm and selects spreaders in the largest independent set to ensure that selected nodes are non-adjacent. However, special situations may occur, such as not enough spreaders in the largest independent set; meanwhile directly selecting rest spreaders in following independent sets may derogate the advantages brought by independent set.

We propose three methods with different degrees of dispersion to identify multiple spreaders. The first one is LIR-2 method which extends the concept of the local index to second-order neighbors and does not restrict the spreaders' selection from the 0-LI nodes. By doing so, this method enlarges the distance between the 0-LI nodes and can guarantee to select enough spreaders. The second one is IS-LIR method which hybrids LIR and IS to ensure that nodes in the same independent set are non-adjacent. The third one is IS-LIR-2 method, which hybrids the improved second-order LIR method and IS method so that the selected spreaders are more dispersed. Comparing the proposed three methods with traditional methods for multiple spreader identification on 10 synthetic networks and five real networks based on the SIR propagation model, we find our methods more effective in maximizing the size of the spreading coverage, and that a higher dispersion of the selected multiple spreaders helps to amplify the spreading.

The rest of this paper is organized as follows.**Sec. 2** introduces work relating to the identification of multiple influential spreaders. **Sec. 3** formalizes the research problem and proposes our method. **Sec. 4** describes our experiments, including baseline methods, the SIR propagation model, evaluation metrics, parameter settings for experiments, and datasets. **Sec. 5** provides the experimental results and discusses why diversity should be considered when we select a set of influential spreaders. We summarize our work in **Sec. 6**.

2 RELATED WORK

Identifying a set of influential nodes in a network is important for designing network immunization [21], system control strategy [22] and improving the network robustness [23,24]. Work about multiple spreader identification falls in three categories. The first regards it as an influence maximization problem [3], and thus utilizes optimization algorithms to directly identify a set of spreaders. The greedy algorithm [3] is a classic example. These algorithms are accurate but time-consuming, and thus do not suit large-scale networks. Some researchers employ information about network structures to reduce the time complexity while maintaining the high accuracy of classic optimization algorithms. The NewGreedy algorithm [4] removes edges that do not contribute to propagation, so as to speed up the simulation process. The community-based greedy algorithm (CGA) [5] mines the top-*k* spreaders from detected communities so as to reduce the running time. Another algorithm [6] constructs an information transfer probability matrix and uses the k-medoid clustering algorithm to find the most centrally located nodes in clusters as spreaders. Two-phase influence maximization (TIM) [7] includes the phases of parameter estimation and node selection to reduce time complexity.

Methods in the second category select the top-ranked spreaders, whose influence is calculated based on network topological information. Classic indicators such as degree centrality [9], betweenness centrality [10], closeness centrality [10], and coreness centrality [25], have been utilized to estimate the influence of spreaders. Some researchers take into account multiple (global or local) network features when measuring the importance of spreaders [26]. For instance, KED centrality [11] combines the number and diversity of paths. Composite centrality based on the analytic hierarchy process (AHP) [13] combines degree, betweenness, and closeness centrality. Classified neighbors centrality (CNC) [14] classifies the neighbors of a focal node into four groups according to the removal order in the process of k-shell decomposition, weights each class differentially, and sums them to characterize the spreading capacity of the node. PageRank [15], LeaderRank [16], and VoteRank [17] all consider the importance of a node itself and its connections with other nodes to identify influential nodes. These rank-based algorithms often have simple forms and low time complexity and can effectively mine a single important node. However, they may not efficiently find multiple important spreaders because they seldom consider interactions between spreaders, i.e., they ignore the

overlapping effects of top-ranked nodes if they are topologically adjacent.

Algorithms in the third category attempt to minimize the overlapping effects of spreaders during selection. The SuperNode algorithm [27] uses the Blondel community detection algorithm to get the community division in the network, and selects important nodes from the communities according to size so that the selected nodes have some distance. An independent set (IS)-based partitioned ranking algorithm [20] divides nodes into independent sets by the Welsh-Powell coloring algorithm, then selects the top-ranked nodes in the largest independent set based on certain centrality indicators. The local index rank (LIR) algorithm [19] selects spreaders from nodes with 0-*LI* values, i.e., those whose direct neighbors have lower degrees than themselves. However, there may not be enough 0-*LI* nodes to be selected as spreaders in some cases, and the selection of adjacent nodes cannot be avoided. We seek to overcome the above deficiencies by extending LIR methods to two-layer neighbors and integrating them with IS methods.

3 METHODS

We formalize the problem of multiple influential spreader identification and propose the LIR-2, IS-LIR, and IS-LIR-2 algorithms, which consider the diversity of nodes to different degrees.

3.1 Formulation of Research Problem

Given a graph $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ denotes the node-set and whose size is N, and $E = \{e_1, e_2, \ldots, e_M\}$ denotes the edge-set, whose size is M. A method to address the problem of multiple influential node identification can be regarded as a function $f(\cdot)$ to select a node subset $S \subseteq V$ with a given k ($1 < k < N$) nodes, which should have the maximum influence on graph G, i.e., $S^* = \arg\max_S f(S, G)$.

3.2 LIR-2 Method

LIR-2 improves on LIR [19], where the local index (*LI*) of node v_i is the number of its first-order neighbors of greater degree, i.e., $LI(v_i) = \sum_{v_j \in N(v_i)} Q(d_j - d_i)$, where d_i is the degree of node v_i, $N(v_i) = \{v_j | v_j \in V\ (v_j, v_i) \in E\}$ contains the neighbors of v_i, and $Q(x) = 1$ when $x > 0$, and otherwise $Q(x) = 0$. Nodes with *LI* values of zero (i.e., 0-*LI* nodes) are ranked by degree, and the top-ranked nodes are selected as spreaders.

LIR-2 extends the neighbors of node v_i from first to second order. The second-order local index LI_2 of node v_i is defined as

$$LI_2(v_i) = \sum_{v_k \in \{N(v_i) \cup N(N(v_i))\}} Q(d_k - d_i), \quad (1)$$

where $N(v_i)$ and $N(N(v_i))$ denote the first- and second-order neighbors, respectively, of node v_i, $Q(x) = 1$ when $x > 0$, and otherwise $Q(x) = 0$. According to the definition, the LI_2 value of node v_i is the number of its first- and second-order neighbors of greater degree.

The LIR-2 method sorts nodes by LI_2 values within degrees, and selects those of top rank as spreaders, as described in **Algorithm 1**.

Algorithm 1. LIR-2

Require: Network $G(V, E)$, k
Ensure: k spreaders
Calculate the two-layer neighbor local index (LI_2) value of each node
Put the same LI_2 value nodes in the LI_2-Set
LI_2-**Set-List** ← Sort LI_2-**Set**s in ascending order based on LI value
FinalList ← [ ]
for LI_2-**Set** in LI_2-**Set-List do**
  sorted-LI_2-**Set** ← Sort nodes in LI_2-**Set** in descending order based on degree
  FinalList.append(**sorted-**LI_2-**Set**)
end for
Output top-k nodes in **FinalList**

Figures 1A,B illustrate LIR and LIR-2, respectively, on a toy network with 20 nodes and 41 edges. **Figure 1C** shows a single 0-*LI* node (node 20). Therefore, 0-*LI* nodes are insufficient for the selection of multiple spreaders. As LIR-2 is not limited to the selection of top-ranked spreaders from nodes with 0 LI_2 values, they can select spreaders as required.

3.3 IS-LIR Method

The LIR method cannot avoid the selection of adjacent nodes. We combine LIR with the IS method to ensure that nodes in the same independent set are non-adjacent. The proposed IS-LIR method uses the Welsh-Powell algorithm to divide nodes into different independent sets, then calculates LI for nodes in independent sets that are ranked in descending order. Nodes are selected from the ranked independent sets, one by one, based on the LIR method. The IS-LIR algorithm is outlined in **Algorithm 2**.

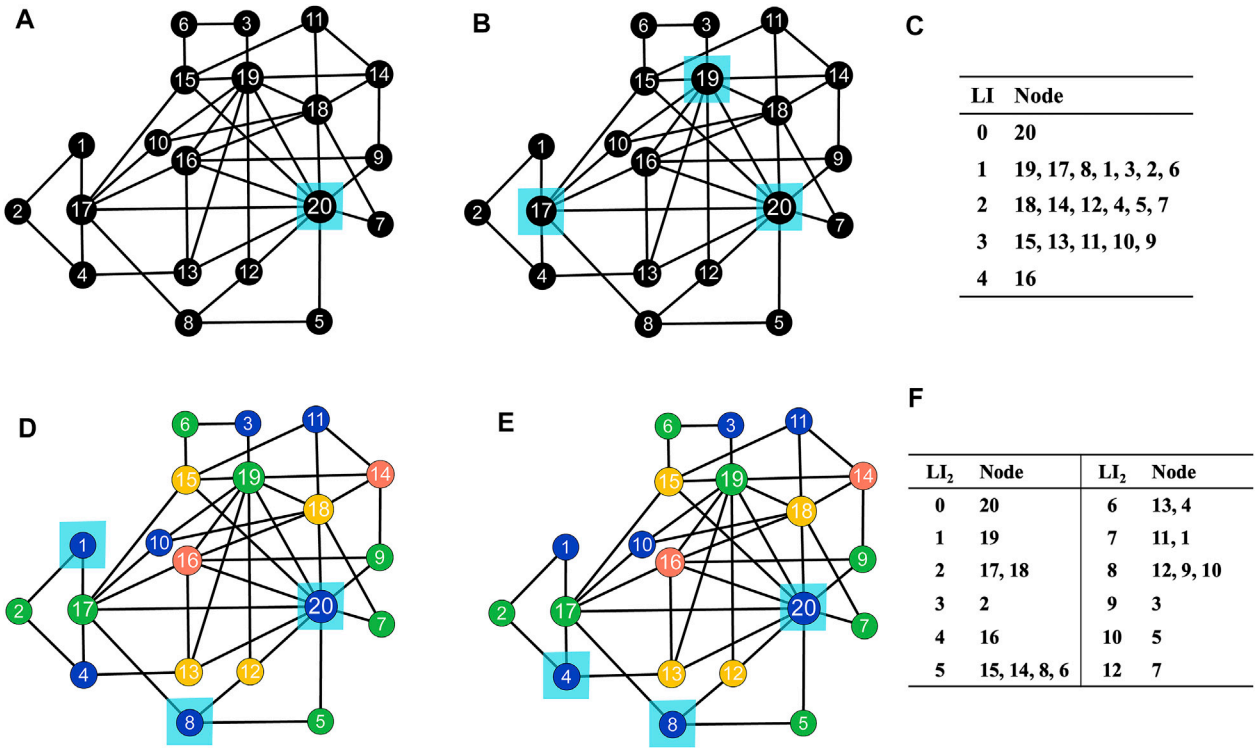
Algorithm 2. IS-LIR

Require: Network $G(V, E)$, k
Ensure: k spreaders
Color every node by Welsh-Powell algorithm
Put nodes with the same color in an **independent set (IS)**
IS-List ← Sort **IS**s in descending order based on the size of **IS**
FinalList ← [ ]
for IS in **IS-List do**
  Calculate the local index (LI) value of each node in **IS**
  Place nodes with the same LI value in a **sub independent set (subIS)**
  subIS-List ← Sort **subIS**s in ascending order based on LI value
  for subIS in **subIS-List do**
    sorted-subIS ← Sort nodes in **subIS** in descending order based on degree of node
    FinalList.append(**sorted-subIS**)
  end for
end for
Output top-k nodes in **FinalList**

Figure 1D illustrates the IS-LIR method on a toy network as an example, first using the Welsh-Powell algorithm to color all nodes in four colors (blue, green, yellow, and pink). Nodes of the same color constitute independent sets, which are sorted by node size, and nodes are sorted by degree within each independent set. We now have a node list, whose top members are selected as the influential spreaders. For instance, using the IS-SIR method, if we seek three effective spreaders on the toy network, we will select nodes 20, 8, and 1 in the blue set.

3.4 IS-LIR-2 Method

IS-LIR-2 combines IS and LIR-2 to select spreaders from more dispersed candidates. Its process, as shown in **Algorithm 3**, is similar to that of IS-LIR, but nodes in each independent set are ranked based on LI_2 values.



LI	Node
0	20
1	19, 17, 8, 1, 3, 2, 6
2	18, 14, 12, 4, 5, 7
3	15, 13, 11, 10, 9
4	16

LI_2	Node	LI_2	Node
0	20	6	13, 4
1	19	7	11, 1
2	17, 18	8	12, 9, 10
3	2	9	3
4	16	10	5
5	15, 14, 8, 6	12	7

FIGURE 1 | Illustrations of LIR [subfigure **(A)**], LIR-2 [subfigure **(B)**], IS-LIR [subfigure **(D)**], and IS-LIR-2 [subfigure **(E)**] methods on a toy network with 20 nodes and 41 edges. The selected spreaders are highlighted in light blue color. Nodes with different colors in subfigures **(D)** and **(E)** belong to different independent sets, which are generated by the Welsh-Powell algorithm. The values of index LI and LI2 of nodes are represented in subfigure **(C)** and subfigure **(F)**.

Algorithm 3. IS-LIR-2

Require: Network $G(V, E)$, k
Ensure: k spreaders
Color every node by Welsh-Powell algorithm
Put the same color nodes in a set, called **independent set (IS)**
IS-List ← Sort **IS**s in descending order based on the size of **IS**
FinalList ← [ ]
for IS in **IS-List do**
  Calculate the two-layer neighbor local index (LI_2) value of each node in **IS**
  Put the same LI_2 value nodes in a set, called **sub independent set (subIS)**
  subIS-List ← Sort **subIS**s in ascending order based on LI_2 value
  for subIS in **subIS-List do**
    sorted-subIS ← Sort nodes in **subIS** in descending order based on the degree of node
    FinalList.append(**sorted-subIS**)
  end for
end for
Output the top-k nodes in **FinalList**

Figure 1E illustrates how IS-LIR-2 runs on the toy network. Like the IS-LIR method (**Figure 1D**), nodes are colored with four colors. Three spreaders, nodes 20, 8, and 4, are selected according to their LI_2 values, as shown in **Figure 1F**.

4 EXPERIMENT SETTINGS

We introduce the classic SIR model, which will be utilized to simulate epidemic spreading, and present two evaluation metrics to compare the performance of the proposed methods with eight baseline methods: degree centrality ranking (DC) [9], LIR [19], degree centrality ranking based independent set (IS-DC) [20], eigenvector centrality ranking based independent set (IS-EV) [20], neighborhood centrality ranking based independent set (IS-ND) [20], and VoteRank [17]. We describe the synthetic and real networks used in our experiments, and discuss parameter settings.

4.1 SIR Model

The SIR model classifies each node in a propagation process into the three states of susceptible, infected, and recovered. All nodes are initially susceptible, except a few in infected states. In our simulations, the infected nodes at time step $t = 0$ are those identified as influential nodes by our proposed methods and the baseline methods for comparisons. At each time step, infected nodes at the end of the previous time step randomly select a neighbor node, which, if susceptible, will be infected with probability μ. All infected nodes recover with probability β. Recovered nodes cannot be infected again, and cannot affect susceptible neighbor nodes. Simulations end when there are no infected nodes in the network.

4.2 Evaluation Metrics

We use two measures to evaluate the performance of our methods in identifying effective influential spreaders. The outbreak size proportion [28] at time step T is

$$F(T) = \frac{n_{R(T)} + n_{I(T)}}{N}, \tag{2}$$

where $n_{R(T)}$ and $n_{I(T)}$ are the numbers of susceptible and infected nodes, respectively, at the end of the time step T, and N is the total number of nodes.

TABLE 1 | Key parameter settings in generating synthetic networks. *N* is the number of nodes; *p* is a random reconnection probability; $< k >$ is the average degree; *m* is the number of new edges in every iteration; $\tau 1$ is the exponent of the degree sequence; $\tau 2$ is the exponent of the community size distribution; μ is a mixing parameter that is the average ratio of the external and total degrees; *MD* is the maximum degree of the network.

Network	*N*	*P*	< k >	*m*	$\tau 1$	$\tau 2$	μ	*MD*
WS1	5,000	0.001	4	—	—	—	—	—
WS2	5,000	0.01	4	—		—	—	—
WS3	5,000	0.1	4	—	—	—	—	
BA1	5,000	—	—	1	—	—	—	—
BA2	5,000	—	—	2	—	—	—	—
BA3	5,000	—	—	3	—	—	—	—
BA4	5,000	—	—	4	—	—	—	—
LFR1	5,000	—	6	—	–2.5	–2.5	0.1	50
LFR2	5,000	—	6	—	–2.5	–2.5	0.3	50
LFR3	5,000	—	6	—	–2.5	–2.5	0.5	50

TABLE 2 | Topological features of synthetic networks. *N* is the number of nodes; *M* is the number of edges; $< k >$ is the average degree; *L* is the average shortest path length; *D* is the network diameter; *C* is the average clustering coefficient.

Network	N	M	< k >	L	D	C
WS1	5,000	10,000	4	192.677	536	0.498
WS2	5,000	10,000	4	43.25	117	0.487
WS3	5,000	10,000	4	11.295	22	0.37
BA1	5,000	4,999	2	7.756	20	0
BA2	5,000	9,996	3.998	4.768	8	0.008
BA3	5,000	14,991	5.996	4.502	7	0.01
BA4	5,000	19,984	7.994	3.663	6	0.0011
LFR1	5,000	14,535	5.841	7.47	21	0.575
LFR2	5,000	15,091	6.036	5.232	11	0.319
LFR3	5,000	14,613	5.845	4.769	9	0.116

The average shortest path length of the identified spreaders represents the dispersion among them [28], and is defined as

$$L = |S|\,(|S| - 1)\frac{1}{\sum_{u,v\in S}\ \sum_{u\neq v}\ \frac{1}{l(u,v)}}, \tag{3}$$

where $l\,(u, v)$ is the shortest path length between nodes u and v; when $|S| = 1$, $L = 0$. A larger L indicates a smaller overlapping neighbor area between nodes in the spreader set.

4.3 Synthetic and Real Networks

To evaluate the effectiveness of our proposed methods in identifying influential spreaders on networks with different topological structures, we compare them with benchmark algorithms on 10 synthetic networks and four real networks. The synthetic networks include three small-world networks generated based on the *Watts-Strogtaz* (WS) small-world network model [29], four scale-free networks generated based on the *Barabási-Albert* scale-free network model [30], and three networks with community structures generated by the LFR community network model [31]. **Table 1** presents key parameter settings for the 10 synthetic networks, and **Table 2** summarizes their basic topological features.

The five real networks used in this study include a football network [32], a collaboration network [33] between jazz musicians (referred to as jazz network), a contact network between high school students (referred to as high-school network) [34], an email network [35], and a power network [29]. The football network includes United States college Division I football games in 2000, where nodes represent teams, and edges are regular-season games between two connected teams [32]. The jazz network describes collaborations between jazz musicians, where each node represents a jazz musician, and an edge denotes that two musicians have played together in a band. The high-school network shows contacts between high school students in specific classes (called “classes préparatoires” in Lycée Thiers, France). The email network presents email communications at the University Rovira i Virgili in Tarragona, Spain, in 2003. Nodes are users, and each edge represents that at least one email was sent. The power network is a topological representation of the Western States Power Grid in the United States, where an edge denotes a power supply line and a node can be a generator, transformer, or substation. **Table 3** summarizes the basic topological features of the five real networks.

TABLE 3 | Basic topological features of five real networks. N is the number of nodes; M is the number of edges; $< k >$ is the average degree; L is the average shortest path length; D is the network diameter; C is the average clustering coefficient.

Network	N	M	< k >	L	D	C
Football	115	613	10.661	2.508	4	0.403
arenas-jazz	198	2,742	27.679	2.235	6	0.633
hschool0	312	2,242	14.37	2	5	0.4
arenas-email	1,133	5,451	9.62	3.606	8	0.254
Power	4,941	6,594	2.669	18.989	46	0.107

4.4 Parameter Settings

Our experiments based on the SIR model explored the proportion of final outbreak size F (t_{end}) with respect to the effective infected probability λ and proportion of spreaders p. We set the parameter of the recovered probability $\beta = \frac{1}{< k >}$ used by He et al. [27].

We also carried out a sensitivity analysis on the size of the spreader set p, varying it between 0.01 and 0.15, i.e., $p \in [0.01, 0.15]$, with a step of 0.01, and the effective infected probability λ was fixed at 2.0.

In addition, we explored the final outbreak size proportion F (t_{end}) while varying the effective infected probability λ, where $\lambda \in [1.5, 2.5]$ with a step of 0.1, and fixed the scales of spreaders at $p = 0.08$. Results were averaged over 1,000 independent runs.

5 RESULTS AND DISCUSSION

We present the experimental results evaluating the proposed methods, and determine whether identified spreaders are effective while varying the infection probability λ. We show

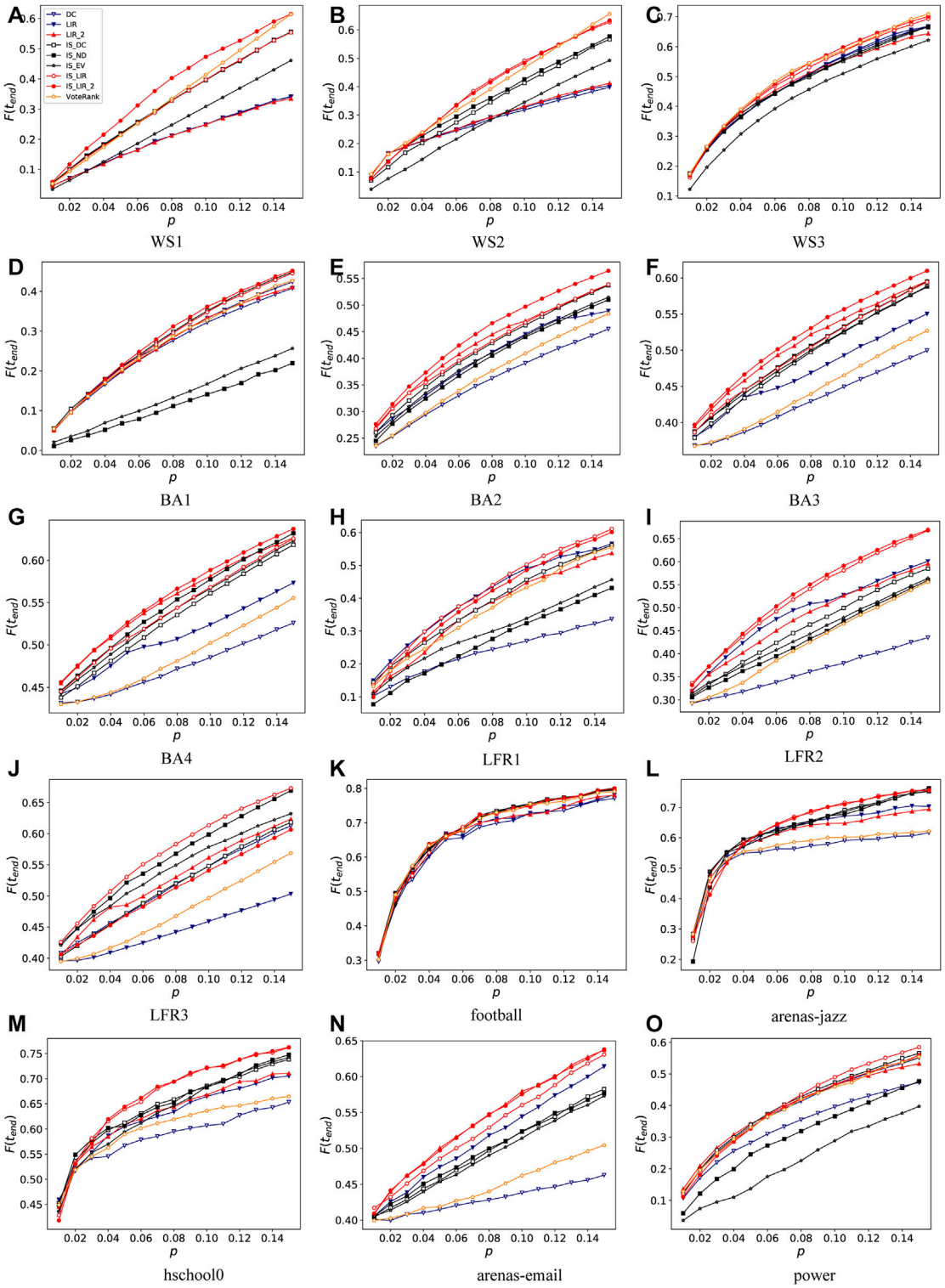


FIGURE 2 | Proportion of final outbreak size $F(t_{end})$ with respect to different proportions p of selected spreaders with respect to nine algorithms on 10 synthetic networks [subfigures **(A–J)**] and five real networks [subfigures **(K–O)**]. Infected probability in SIR model is $\lambda = 2.0$; SIR recovered probability $\beta = \frac{1}{<k>}$ and results are averaged over 1,000 independent runs.

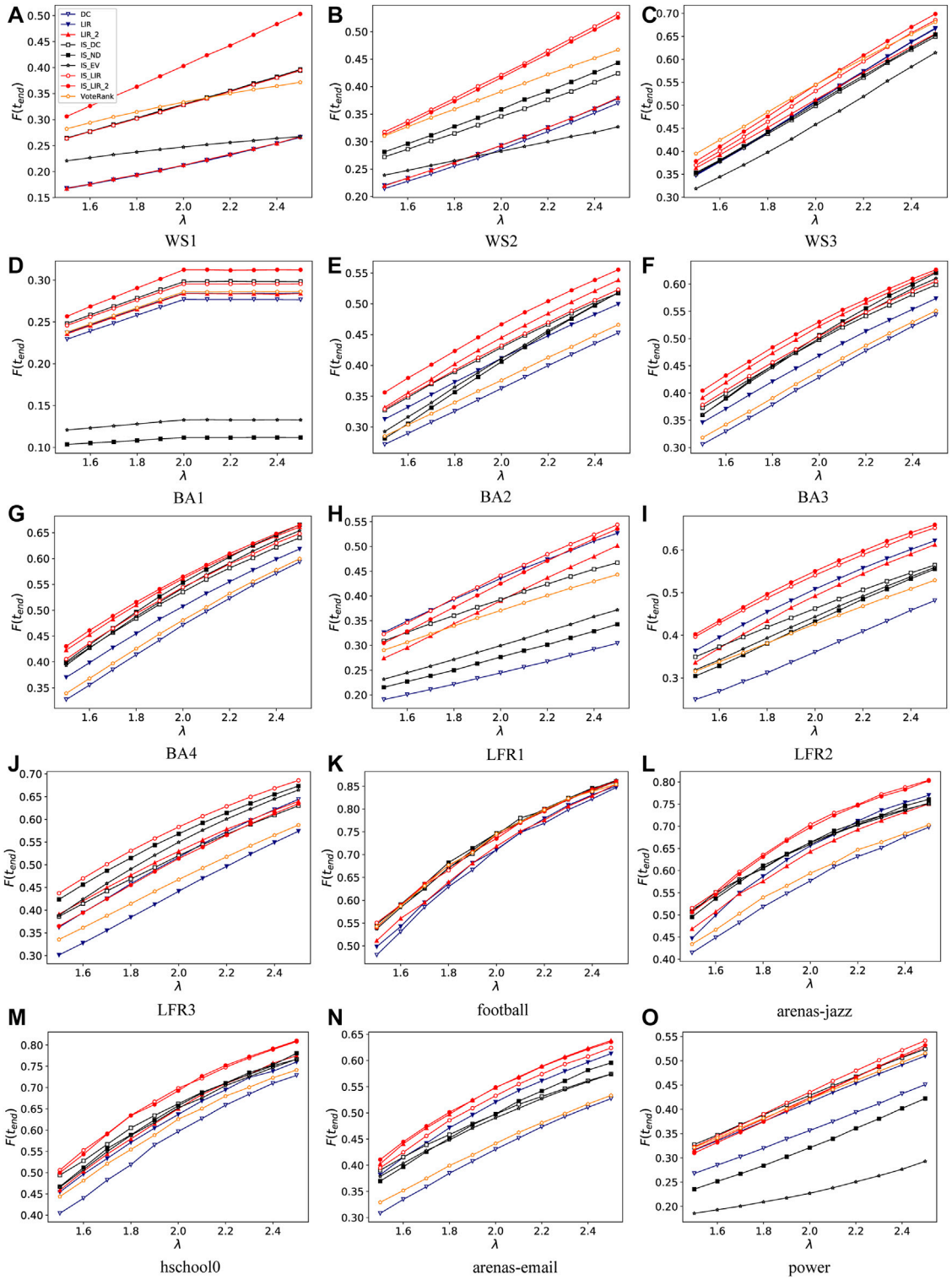


FIGURE 3 | Final outbreak size proportion $F(t_{end})$ at different effective infected probabilities λ with respect to nine algorithms on ten synthetic networks [subfigures **(A–J)**] and five real networks [subfigures **(K–O)**]. SIR recovered probability $\beta = \frac{1}{<k>}$, proportion of spreaders $\rho = 0.08$, and results are averaged over 1,000 independent runs.

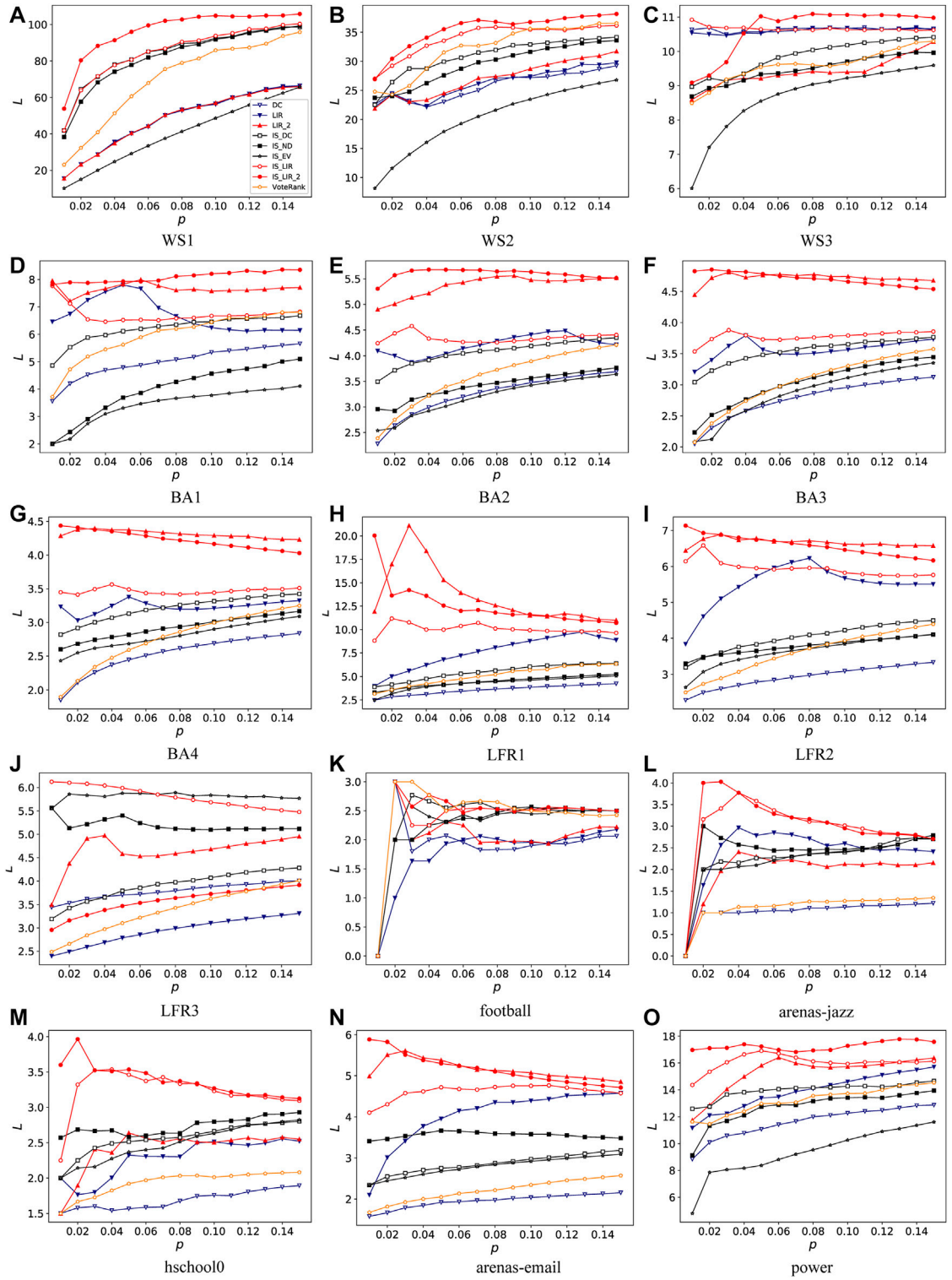


FIGURE 4 | Harmonic mean of average shortest path length L between any two nodes with respect to different proportions of multiple spreaders p with respect to nine algorithms on ten synthetic networks [subfigures **(A–J)**] and five real networks [subfigures **(K–O)**].

the relationships between the dispersion and the effectiveness of influential spreaders identified by our proposed methods and the baseline methods.

Figure 2 displays the final outbreak size F (t_{end}) for different numbers of spreaders (denoted by the proportion of selected spreaders p) identified by different methods based on SIR simulations, and shows that our proposed methods generally outperform the baseline methods on synthetic and real networks. On WS networks, IS-LIR-2 has the largest final outbreak scale on WS1. IS-LIR-2, IS-LIR, and VoteRank perform similarly to or better than other algorithms on WS2 and WS3. On BA networks, the performance of IS-LIR-2 and LIR-2 is superior to the other methods, especially on BA2, BA3, and BA4. On LFR networks, IS-LIR performs better than the other methods, and IS-LIR-2 performs better on LFR2 but not so well on LFR3. In experiments on real networks, IS-LIR and IS-LIR2 could identify more influential spreaders in most cases on almost all five real networks. However, LIR-2 was not significantly superior on real networks except the arenas-email network. LIR-2, IS-LIR, and IS-LIR2 had obvious advantages selecting multiple spreaders in most cases. This implies that to take into account the dispersion of selected nodes can improve performance.

As the infected probability in the SIR model is a key parameter that may affect the final break size of infections, we explored the performance (represented by the final outbreak size proportion F (t_{end})) of our proposed methods with different values of the infected probability λ. As shown in **Figure 3**, whether λ is small or large, IS-LIR and IS-LIR-2 have significant advantages over baseline methods on most of the synthetic and real networks. Specifically, on WS networks, as the infected rate increases, the performance of IS-LIR-2 increases significantly on WS1 and WS2, and IS-LIR performs best on WS2. On BA networks, IS-LIR-2 and LIR-2 are consistently superior to other algorithms on most BA networks. Focusing on LFR networks, we can see that IS-LIR is always better than the baseline methods except on the LFR2 network, where IS-LIR-2 performs better. On real networks, we can see that IS-LIR and IS-LIR-2 maintain their advantages whether λ is small or large.

Figure 4 presents the structural characteristics of influential nodes identified by LIR-2, IS-LIR, and IS-LIR2 and the baseline methods, and shows that spreaders identified by IS-LIR-2, IS-LIR, and LIR-2 have the largest harmonic mean of the average shortest path length L between any two nodes in most cases, except the LFR3 and football networks. On LFR3, multiple spreaders identified by IS-LIR, IS-ND, and IS-EV have the top three average shortest path lengths (as shown in **Figure 4J**). On the football network, vote-rank and IS-DC identified spreaders with larger average shortest path lengths than our proposed method in a few cases (as shown in **Figure 4K**). These results may explain why our proposed methods outperform the baseline methods in identifying multiple influential spreaders (as shown in **Figure 2**): if the identified spreaders have a larger mean shortest path length, they may result in a more heavier infection spreading. This implies that taking into account the dispersion of nodes can help find the most influential spreaders.

6 CONCLUSION

To effectively identify a set of influential spreaders is important in infectious disease prevention or information dissemination. To address this problem, inspired by the LIR method [19] and IS method [20], we proposed the LIR-2, IS-LIR, IS-LIR-2 algorithms, which take into account the dispersion of selected spreaders in different ways. In evaluation experiments on 10 synthetic networks and five real networks, our proposed methods, especially IS-LIR and IS-LIR-2, were more effective than six baseline methods at identifying more influential spreaders. One potential reason is that the spreaders found by our methods have a larger average shortest path length, i.e., the selected spreaders are more dispersed, so as to reduce the opportunity to infect the same nodes in the propagation process. IS-LIR, LIR-2, and IS-LIR-2 achieved a good balance between expanding the final spreading range of the spreaders on the SIR model and increasing the topological distance between them. However, we merely studied static, undirected, and unweighted networks. How to extend our methods to other types of networks, and how to investigate their sensitivity to specific network characteristics are two interesting questions to be addressed in future work.

DATA AVAILABILITY STATEMENT

The data used in the study are all available *via* the cited references, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

Proposed and implemented algorithms: LT, ML, and LL. Performed the experiments: LT, ML, and LL. Wrote the paper: LT, ML, ZZ, and LL.

FUNDING

This work is supported by National Natural Science Foundation of China (No. 61976181) and Fundamental Research Funds for the Central Universities (XDJK2019C122).

REFERENCES

1. Su Z, Gao C, Liu J, Jia T, Wang Z, Kurths J. Emergence of Nonlinear Crossover under Epidemic Dynamics in Heterogeneous Networks. *Phys Rev E* (2020) 102:052311. doi:10.1103/PhysRevE.102.052311
2. Gao C, Su Z, Liu J, Kurths J. Even central Users Do Not Always Drive Information Diffusion. *Commun ACM* (2019) 62:61–7. doi:10.1145/3224203
3. Kempe D, Kleinberg J, Tardos É. Maximizing the Spread of Influence through a Social Network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-KDD '03; 2003 August 24–27; Washington, D.C. New York, NY, USA (2003). p. 137–46. doi:10.1145/956750.956769
4. Chen W, Wang Y, Yang S. Efficient Influence Maximization in Social Networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-KDD '09; 2009 June 28–July 1; Paris, France. New York, NY, USA (2009). p. 199–208. doi:10.1145/1557019.1557047
5. Wang Y, Cong G, Song G, Xie K. Community-based Greedy Algorithm for Mining Top-K Influential Nodes in mobile Social Networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD '10; 2010 July 24–28; Washington, D.C. New York, NY, USA (2010). p. 1039–48. doi:10.1145/1835804.1835935
6. Zhang X, Zhu J, Wang Q, Zhao H. Identifying Influential Nodes in Complex Networks with Community Structure. *Knowledge-Based Syst* (2013) 42:74–84. doi:10.1016/j.knosys.2013.01.017
7. Tang Y, Xiao X, Shi Y. Influence Maximization. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data SIGMOD '14; 2014 June 22–27; Snowbird, UT. New York, NY, USA (2014). p. 75–86. doi:10.1145/2588555.2593670
8. Morone F, Makse HA. Influence Maximization in Complex Networks through Optimal Percolation. *Nature* (2015) 524:65–8. doi:10.1038/nature14604
9. Bonacich P. Factoring and Weighting Approaches to Status Scores and Clique Identification. *J Math Sociol* (1972) 2:113–20. doi:10.1080/0022250X.1972.9989806
10. Freeman LC. Centrality in Social Networks Conceptual Clarification. *Social Networks* (1978) 1:215–39. doi:10.1016/0378-8733(78)90021-7
11. Chen D-B, Xiao R, Zeng A, Zhang Y-C. Path Diversity Improves the Identification of Influential Spreaders. *EPL* (2014) 104:68006. doi:10.1209/0295-5075/104/68006
12. Wang S, Du Y, Deng Y. A New Measure of Identifying Influential Nodes: Efficiency Centrality. *Commun Nonlinear Sci Numer Simul* (2017) 47:151–63. doi:10.1016/j.cnsns.2016.11.008
13. Bian T, Hu J, Deng Y. Identifying Influential Nodes in Complex Networks Based on AHP. *Phys A: Stat Mech Appl* (2017) 479:422–36. doi:10.1016/j.physa.2017.02.085
14. Li C, Wang L, Sun S, Xia C. Identification of Influential Spreaders Based on Classified Neighbors in Real-World Complex Networks. *Appl Maths Comput* (2018) 320:512–23. doi:10.1016/j.amc.2017.10.001
15. Brin S, Page L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Comput Networks ISDN Syst* (1998) 30:107–17. doi:10.1016/S0169-7552(98)00110-X
16. Lü L, Zhang Y-C, Yeung CH, Zhou T. Leaders in Social Networks, the Delicious Case. *PLoS One* (2011) 6:e21202–9. doi:10.1371/journal.pone.0021202
17. Zhang J-X, Chen D-B, Dong Q, Zhao Z-D. Identifying a Set of Influential Spreaders in Complex Networks. *Sci Rep* (2016) 6:27823. doi:10.1038/srep27823
18. Gu J, Lee S, Saramäki J, Holme P. Ranking Influential Spreaders Is an Ill-Defined Problem. *EPL* (2017) 118:68002. doi:10.1209/0295-5075/118/68002
19. Liu D, Jing Y, Zhao J, Wang W, Song G. A Fast and Efficient Algorithm for Mining Top-K Nodes in Complex Networks. *Sci Rep* (2017) 7:43330. doi:10.1038/srep43330
20. Zhao X-Y, Huang B, Tang M, Zhang H-F, Chen D-B. Identifying Effective Multiple Spreaders by Coloring Complex Networks. *EPL* (2015) 108:68005. doi:10.1209/0295-5075/108/68005
21. Li S, Zhao D, Wu X, Tian Z, Li A, Wang Z. Functional Immunization of Networks Based on Message Passing. *Appl Maths Comput* (2020) 366:124728. doi:10.1016/j.amc.2019.124728
22. Yu D, Long J, Chen CLP, Wang Z. Adaptive Swarm Control within Saturated Input Based on Nonlinear Coupling Degree. *IEEE Trans Syst Man Cybern Syst* (2021) 1–12. doi:10.1109/TSMC.2021.3102587
23. Gao C, Fan Y, Jiang S, Deng Y, Liu J, Li X. Dynamic Robustness Analysis of a Two-Layer Rail Transit Network Model. *IEEE Trans Intell Transport Syst* (2021) 1–16. doi:10.1109/TITS.2021.3058185
24. Li S, Lu D, Wu X, Han W, Zhao D. Enhancing the Power Grid Robustness against Cascading Failures under Node-Based Attacks. *Mod Phys Lett B* (2021) 35:2150152. doi:10.1142/s0217984921501529
25. Kitsak M, Gallos LK, Havlin S, Liljeros F, Muchnik L, Stanley HE, et al. Identification of Influential Spreaders in Complex Networks. *Nat Phys* (2010) 6:888–93. doi:10.1038/nphys1746
26. Gao C, Zhong L, Li X, Zhang Z, Shi N. Combination Methods for Identifying Influential Nodes in Networks. *Int J Mod Phys C* (2015) 26:1550067. doi:10.1142/S0129183115500679
27. He J-L, Fu Y, Chen D-B. A Novel Top-K Strategy for Influence Maximization in Complex Networks with Community Structure. *PLoS One* (2015) 10: e0145283. doi:10.1371/journal.pone.0145283
28. Sun H-l., Chen D-b., He J-l., Ch'ng E. A Voting Approach to Uncover Multiple Influential Spreaders on Weighted Networks. *Physica A: Stat Mech its Appl* (2019) 519:303–12. doi:10.1016/j.physa.2018.12.001
29. Watts DJ, Strogatz SH. Collective Dynamics of 'small-World' Networks. *Nature* (1998) 393:440–2. doi:10.1038/30918
30. Baraba´si A-L, Albert R. Emergence of Scaling in Random Networks. *Science* (1999) 286:509–12. doi:10.1126/science.286.5439.509
31. Lancichinetti A, Fortunato S, Radicchi F. Benchmark Graphs for Testing Community Detection Algorithms. *Phys Rev E* (2008) 78:046110. doi:10.1103/PhysRevE.78.046110
32. Girvan M, Newman MEJ. Community Structure in Social and Biological Networks. *Proc Natl Acad Sci* (2002) 99:7821–6. doi:10.1073/pnas.122653799
33. Gleiser PM, Danon L. Community Structure in Jazz. *Advs Complex Syst* (2003) 06:565–73. doi:10.1142/S0219525903001067
34. Mastrandrea R, Fournet J, Barrat A. Contact Patterns in a High School: a Comparison between Data Collected Using Wearable Sensors, Contact Diaries and friendship Surveys. *PLoS One* (2015) 10:e0136497–26. doi:10.1371/journal.pone.0136497
35. Guimerà R, Danon L, Díaz-Guilera A, Giralt F, Arenas A. Self-similar Community Structure in a Network of Human Interactions. *Phys Rev E* (2003) 68:065103. doi:10.1103/PhysRevE.68.065103

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Physics

ORIGINAL RESEARCH
published: 05 January 2022
doi: 10.3389/fphy.2021.787185

HetInf: Social Influence Prediction With Heterogeneous Graph Neural Network

Liqun Gao[1], *Haiyang Wang*[1], *Zhouran Zhang*[2], *Hongwu Zhuang*[1] *and Bin Zhou*[1]*

[1]*Key Lab of Software Engineering for Complex Systems, School of Computer, National University of Defense Technology, ChangSha, China,* [2]*Department of Materials Science and Engineering, National University of Defense Technology, ChangSha, China*

With the continuous enrichment of social network applications, such as TikTok, Weibo, Twitter, and others, social media have become an indispensable part of our lives. Web users can participate in their favorite events or pay attention to people they like. The "heterogeneous" influence between events and users can be effectively modeled, and users' potential future behaviors can be predicted, so as to facilitate applications such as recommendations and online advertising. For example, a user's favorite live streaming host (user) recommends certain products (event), can we predict whether the user will buy these products in the future? The majority of studies are based on a homogeneous graph neural network to model the influence between users. However, these studies ignore the impact of events on users in reality. For instance, when users purchase commodities through live streaming channels, in addition to the factors of the host, the commodity is also a key factor that influences the behavior of users. This study designs an influence prediction model based on a heterogeneous neural network HetInf. Specifically, we first constructed the heterogeneous social influence network according to the relationship between event nodes and user nodes, then sampled the user heterogeneous subgraph for each user, extracted the relevant node features, and finally predicted the probability of user behavior through the heterogeneous neural network model. We conducted comprehensive experiments on two large social network datasets. Furthermore, the experimental results show that HetInf is significantly superior to the previous homogeneous neural network methods.

Keywords: social network analysis, social influence analysis, heterogeneous neural network, user behavior prediction, deep learning

OPEN ACCESS

Edited by:
Chengyi Xia,
Tianjin University of Technology, China

Reviewed by:
Li Pan,
Shanghai Jiao Tong University, China
Di Jin,
Tianjin University, China

***Correspondence:**
Bin Zhou
binzhou@nudt.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *30 September 2021*
Accepted: *17 November 2021*
Published: *05 January 2022*

Citation:
Gao L, Wang H, Zhang Z, Zhuang H and Zhou B (2022) HetInf: Social Influence Prediction With Heterogeneous Graph Neural Network. Front. Phys. 9:787185. doi: 10.3389/fphy.2021.787185

1 INTRODUCTION

Nowadays, social networks are everywhere around us in our daily lives. Social influence occurs when we get information from social networks, which means that network events (such as network news, trending topics, publishing papers, or other events) or network users we are interested in constantly influence us through social media, and both of them can induce us to engage in social action (including retweet, comment, like, publish, and purchase). For example, live commerce is very popular nowadays, and we will choose our favorite live streaming host to buy necessary commodities. From another perspective, both the live streaming host (user) and the commodities (event) have a substantial impact on the target user's behavior. Similar to the definition of "event" in the study

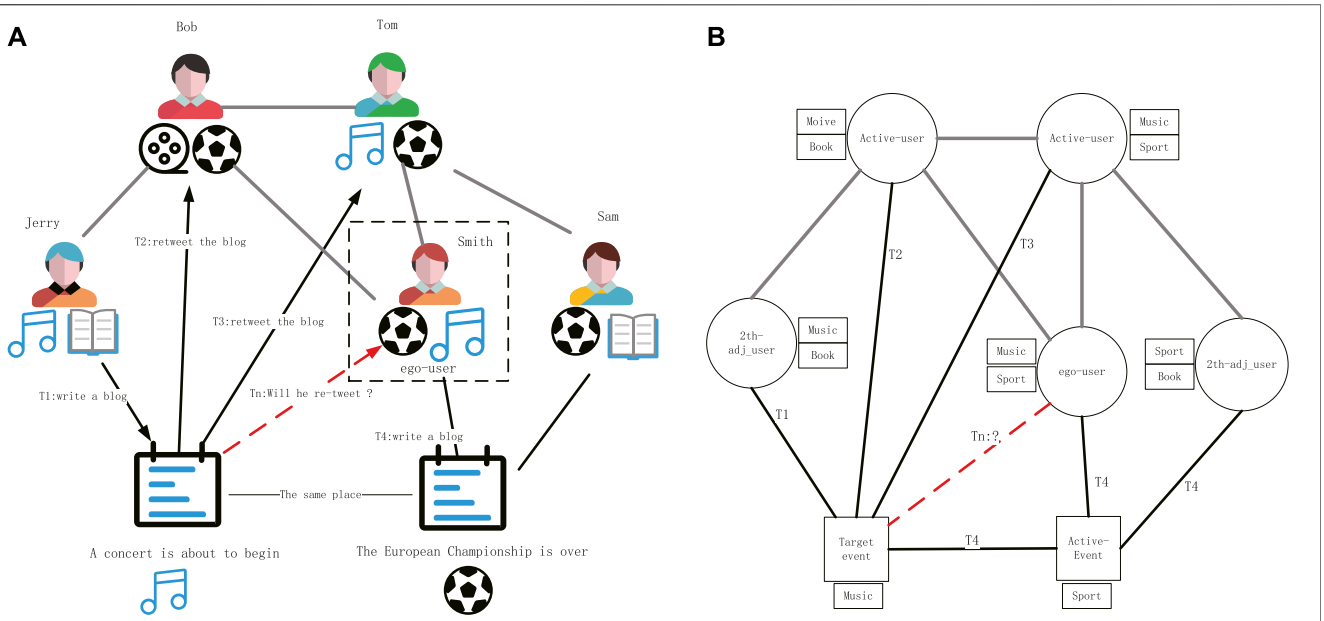


FIGURE 1 | Problem illustration of mining user–event influence in heterogeneous networks and predicting user behaviors. **Figure 1A** shows an instantiated prediction case, and the goal is to predict whether Smith (ego-user) will forward the concert blog (whether the red line will occur in the future). **Figure 1B** is obtained by abstracting **Figure 1A**, given 1) the relationship between the user and the event within the observable time (the connection in b includes three relationships: u-u, u-e, and e-e); 2) the embedding representation of the attributes of different nodes in the observable time (the rectangle next to the node in b); 3) the activation state of neighbor nodes (active or inactive); and 4) the embedding representation of each node in the network, then we predicted whether ego-user will participate in the target event.

mentioned in reference 1, social events can be regarded as a complete semantic unit in which network users participate and understand. One of the key computational problems is to predict the user's social behavior in social influence analysis. How to model the influence relationship to predict the behavior of network users on events is one of the key computational problems in user-level social influence prediction. This problem is applied to many fields, including but not limited to election [2], network marketing [3,4], recommendation [5], rumor detection [6], and information dissemination[7,8].

There are a large number of research studies on the role of the heterogeneity of nodes in social networks in social influence [9–12]. This kind of study mainly focuses on user nodes' interest in event nodes and predicts user behavior by capturing the influence of event nodes' topic level. The study in reference 10 improves the traditional cascade propagation mode and applies the topic distribution methods to an independent cascade model and linear threshold model. The study in reference 9 uses a graph generation method to predict user behavior through the relationship between event topics and network users. These methods use traditional machine learning models to predict users' social behavior through the manual feature representation of learning nodes. However, they do not consider the association between different types of network nodes in heterogeneous social networks, such as the dual impact of users and events on target users, which leads to the limited ability to capture the incentives that really affect user behavior.

Due to the progress of the graph neural network [13], the nodes of network have stronger representation ability. Many studies use graph neural network to model the problem of social influence prediction and make plenty of progress. The study in reference 14 uses the user's local network as the input of the graph neural network to learn the user's potential social representation and uses both network structures and user-specific features in convolutional neural and attention networks. Based on DeepInf, the study mentioned in reference 15 applies the multi-view impact prediction network to solve the social influence prediction problem. However, these methods are based on assumptions that users are only affected by other users, in order to model the relationship between users (homogeneous network), which lacks the analysis of the influence between heterogeneous nodes. Real social networks (such as Twitter, Digg, and Citation network) are heterogeneous and contain different types of entities [16], for example, user nodes and event nodes (stories, tweets, papers, and other objects), which inevitably interact with each other. For example, in **Figure 1A**, Bob may forward the concert event because he is affected by the user Jerry (because he is not interested in music), and Tom may forward the concert event because he is affected by the event (he likes music).

To tackle these challenges, we focused on user-level behavior prediction in a heterogeneous network. This network contains two types of nodes: user and event. It aims to construct a heterogeneous influence network of event nodes and user nodes based on attributes, and we use graph neural networks to model the influence between these nodes so as to better mine the inducement of user behavior. Inspired by the latest research on heterogeneous neural network [13], the local modeling of

heterogeneous networks can capture both structure and content heterogeneity and provide more reliable heterogeneous node representation ability for downstream tasks. Therefore, we combine the benefits of heterogeneous neural networks and semantic representation methods to model the influence network of local neighbor nodes based on heterogeneous network graph. For example, in **Figure 1A**, with the aim of learning the influence of different types of nodes on him through historical semantic information and influence relationship, we input the heterogeneous neighbor local graph network with Smith as the ego-user, so as to predict his future behavior (whether to participate in the discussion of concert events).

Specifically, we proposed HetInf, a heterogeneous network influence prediction model based on two types of nodes. First, based on the influence relationship of network nodes, we constructed a heterogeneous relationship graph composed of them and hoped to build a more accurate influence model. Second, we sampled ego-user neighbor subgraphs. Specifically, an innovative heterogeneous network sampling strategy, based on restart random walk (RWR) [17], is used to sample the topology features and the semantic features (including event topics and user interests) of the heterogeneous nodes in ego-user neighbor subgraphs. Subsequently, an end-to-end heterogeneous neural network influence model is built, the historical topic features of events and the historical interest features of users based on semantics is embedded using Word2Vec [18], the node representation through the node semantics is aggregated, and the heterogeneous graph neural network model is used to learn the node relationship of event–user heterogeneous network. Finally, we learned the influence of different neighbor nodes on ego-user node through the graph attention networks [19], so as to predict whether users will have the social behavior of participating in events in the future.

Summarizing, our contributions are given as follows:

(1) We applied the heterogeneous graph neural network method to predict the influence of users at the micro-level in social networks. Specifically, we extend the deep learning method of homogeneity social influence networks and analyze the dynamic propagation mode of heterogeneous networks to infer more accurate influence network.
(2) As respect to heterogeneous networks, we design a local sampling method in line with time sequence process, established the influence relationship between events and users, and applied an innovative end-to-end heterogeneous graph neural network model to more accurately predict users' social behavior.
(3) Therefore, we tested two real large-scale social network data: Digg and Weibo. The experimental results show that HetInf exhibit significant improved accuracy when constructing a heterogeneous network compared with several state-of-the-art baselines.

The rest of this article is organized as follows: **Section 2** formulates social influence locality problem. **Section 3** introduces the proposed framework in detail. **Section 4** describes extensive experiments with two datasets, **Section 5** summarizes related study, and **Section 6** concludes this study.

2 PROBLEM FORMULATION

In this section, we introduced several related definition and then formally formulated the problem of heterogeneous social network influence locality.

2.1 Definition 1: R-Neighbors and r-Heterogeneous Neighbor Subgraph

A heterogeneous network $\mathcal{G} = (V, E; O_V, R_E, A_V)$ with two types of nodes V, three types of links E, and node attributes A_V is defined. In **Figure 1B**, O_V includes user node U (round), event nodes E (square), and $V = U \cup E$; R_E includes user–event relation R_{ue}, event–event relation R_{ee}, user–user relation R_{uu}, and $E = R_{ue} \cup R_{ee} \cup R_{uu}$. A_V is the attribute feature of the node, including the semantic attribute A_V^S(rectangle) and topology attribute A_V^T. For user u, its r-neighbors are defined to be $\Gamma_u^r = \{v: d(v,u) <= r\}$, where $d(v, v_u)$ is the distance (number of hops) from v to u, and v is different types of nodes in heterogeneous subgraph G. The r-ego heterogeneous subgraph of user u is the local heterogeneous subgraph induced by Γ_u^r and denoted by G_u^r.

2.2 Definition 2: Social Action

Social action refers to the behavior of users in the social network events, such as social network users retweet tweets (events) or publish papers (events). Formally, a social action can be regarded as the action of user u on event e at time t in the heterogeneous graph G_u. We define social action as a binary problem. Action status S_{ue}^t belongs to (0,1). When $S_{ue}^t = 1$, it means that user u has social action for an event e after time t; when $S_{ue}^t = 0$, it means that no social action has occurred.

2.3 Problem 1: Heterogeneous Social Network Influence Locality

Social influence locality models the probability of social action when ego-user u_i is influenced by neighbor nodes on his r-ego network $G_{u_i}^r$; formally, given a 6-tuple $\{u, e, t, G, A, S\}$, social influence locality aims to quantify the activation probability of user u_i's social action in response to event e after a given time t in G with attribute feature A and action status S as follows:

$$prob(S_{u_i}^{t+\Delta t} | e_j^t, G_{u^i}^r, A_G^t, S_G^t), \tag{1}$$

where u_i represents the ego-user, e_j^t represents the event e_j at time t, $G_{u_i}^r$ represents r-heterogeneous neighbor subgraph, A_G^t represents the node attributes by time t, including semantic attribute A^S, topology attribute A^T, and S_G^t represents the action state of the subgraph node before timestamp t.

After determining the problem, we sample N 6-tuple samples through preprocessing data. Similar to the definition of DeepInf [14], we regard social influence locality as a binary classification problem and calculate the model parameters by minimizing the negative log likelihood

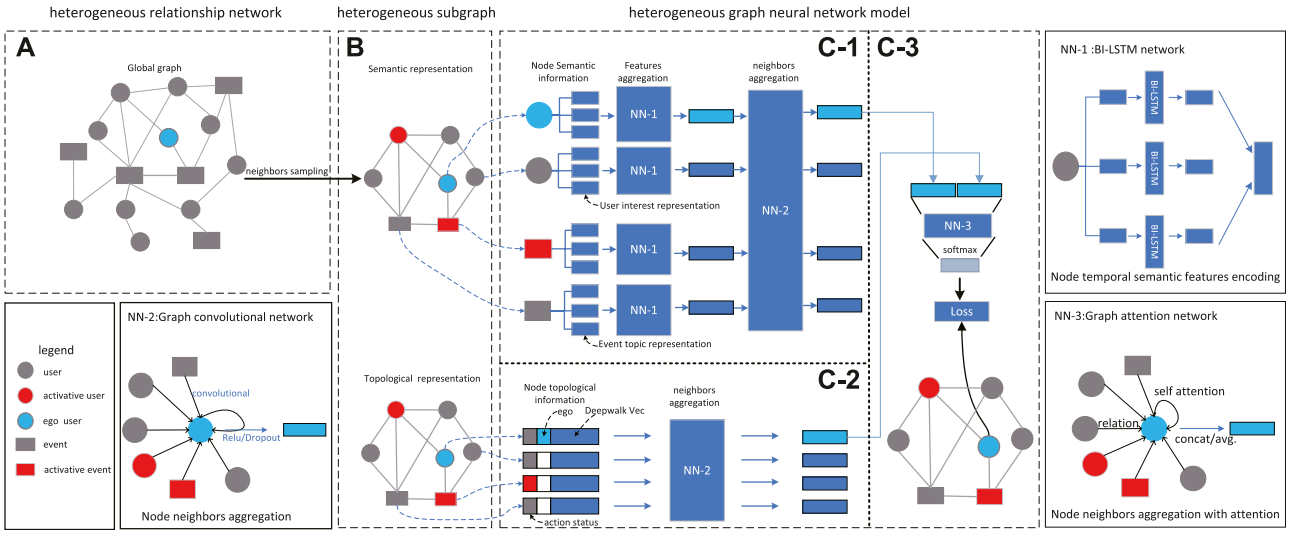


FIGURE 2 | Model framework of HetInf: It first **(A)** constructs heterogeneous relational networks with attributes), next **(B)** samples fix sized heterogeneous neighbors for each node, then aggregates the attribute features and topological features of the sampled heterogeneous neighbors *via* (C-1) and (C-2), and finally optimizes the model *via* (C-3). NN-1: Bi-LSTM, used to aggregate semantic features based on time series; NN-2: GCN network, used to aggregate heterogeneous node attributes; and NN-3: GAT network, used to learn the weights of interaction between nodes.

objective method. We use the following objective with parameters θ:

$$L(\theta) = -\sum_{j=1}^{M}\sum_{i=1}^{N}(S_{u_i}^{t+\Delta t}|e_j^t, G_{u^i}^r, A_G^t, S_G^t). \tag{2}$$

Similarly, we assume that time Δt is infinite, that is, we only predict user action outside the observation time window.

3 MODEL FRAMEWORK

Our goal is to design a heterogeneous graph network model based on the interaction between events and users, named HetInf, which aims to learn the dynamic preferences of individuals and the influence of heterogeneous neighbors in detail. Building the HetInf model needs three steps: 1) constructing heterogeneous relational networks with attributes; 2) sampling r-heterogeneous neighbor subgraph; and 3) building heterogeneous graph neural network model. **Figure 2** shows the framework of HetInf.

3.1 Construct Heterogeneous Graph

3.1.1 Constructing Heterogeneous Relational Networks

Considering the heterogeneous network based on influence prediction user action, an intuitive way is to construct heterogeneous influence graph [20] to obtain the influence between nodes from the heterogeneous graph. We first obtain two types of nodes, including users and events. The event node can be regarded as a specific network event, such as a hashtag in social network dataset or a "story" in Digg dataset.

Then we establish the relationship between nodes according to the data characteristics, including the relationship between user–user, user–event, and event–event. Specifically, we use follow relationship and interact relationship as the user–user relationship; the user–event relationship between users and events can be determined by the user's historical behavior, for example, the user has participated in the event; and the event–event relationship can use co-occurrence association [21] or semantic association [22]. In this way, we construct the global heterogeneous relationship network $\mathcal{G}$ (Definition 2.1), as shown in **Figure 2A**.

3.1.2 Extract Node Attributes

For different types of nodes, we select two features as the initial node representation of the heterogeneous relationship network:

Semantic features: Since the user's behavior is more influenced by the semantic information of the social event [9], the semantic feature A_v^S of each node was used to indicate the "bias" in the semantic level. For the event nodes, TF-IDF [23] was used to sample K keywords $Ew_i, i \in K$ in each e and to distinguish between different events. For the user nodes, the stopwords (non-meaningful) was first removed, then the most frequent I keywords Uw_i $(i \in I)$ of a user was sampled to represent the user's interest, and finally, these keywords with timestamp was sorted to represent the semantic evolution of nodes over time. Formalization is given as follows:

$$A_v^S = \begin{cases} Sort\_by\_time(Ew_i) & \text{if} v \in E \\ Sort\_by\_time(Uw_i) & \text{if} v \in U. \end{cases} \tag{3}$$

Topology features: In addition, inspired by the study in reference 14, the DeepWalk [24] algorithm was used to obtain

the topology feature $A_v^T, A_v^T \in R^d$, where d represents the embedding dimension, $v \in V$.

3.2 Sample r-Heterogeneous Neighbor Subgraph

Generally, the graph neural network uses the feature information of the node's n-order neighbor (e.g., first older) for the aggregation process of node features, such as GAT [19]. Therefore, the breadth first search (BFS) strategy [25] used in the graph localization process will make the weight between users and events too large in hot events. For example, in popular events, due to a large number of related events, most of the neighbor nodes sampled by a user are event type nodes, which will lead to the event type nodes becoming the dominant influence, and ignore the influence between users. In order to solve this problem, an improved random walk strategy was used to comply with the law of information dissemination. The sampling strategy includes two steps:

1. Sample a fixed-length random walk sequence. We took each user $u \in U$ as the starting point, utilizing the RWR [17] method to sample a fixed number of N_r neighbor nodes.
2. Use the meta-path method to perform sub-sampling in the sequence of step 1. We used random walk probability and u-e-u (user publishes an event, which is then forwarded by other users) and U-U (users directly forward through other users) meta-paths, sampled neighbor nodes with a fixed length of $N < N_R$, and then used these neighbor nodes to construct a subgraph G_u.

This strategy satisfies the law of information dissemination and helps avoid the problem of too little sampling of some types of nodes. Similar to the definition in DeepInf [14], a positive instance of a local heterogeneous subgraph was generated if a user has social action with an event after the timestamp t, and a negative instance was generated if the user is not observed in the watch window to be associated with the event.

3.3 Build Heterogeneous Graph Neural Network Model

In this way, 6-tuple (2.3) was used as a set of examples and a deep learning model was designed to predict the action state $S_u^{t+\Delta t}$ of the ego-user. Our neural network model consists of three parts: a semantic feature aggregation module (shown in **Figure 2c-1**), topological feature aggregation module (shown in **Figure 2c-2**), and heterogeneous multi-attribute hidden layer aggregation module (shown in **Figure 2c-3**). In this section, the different modules are introduced to express the process of model building.

3.3.1 Semantic Feature Aggregation Module

A neural network module was designed to learn the deep association between user and event semantics. The module uses the node semantic attribute A_v^S (obtained in 3.1.2) and the local heterogeneous subgraph G_u as input and realizes the aggregation function of semantic features through a neural network $H_S'(v)$. Specifically, we denoted the semantic feature as W_v^i, indicating the ith semantic feature of node v, and utilized Word2Vec [18] to pre-train W_v^i as x_i. Inspired by the study in reference 13, the neural network structure of Bi-LSTM was used to capture the association between semantic features of nodes at a deeper level, and the average value of all hidden states was used to represent the general aggregation embedding as follows:

$$H_S(v) = \frac{\sum_{i \in W_v^i} \left[\overrightarrow{LSTM}_\theta(x_i) \| \overleftarrow{LSTM}_\theta(x_i) \right]}{|W_v|}, \tag{4}$$

where $H_S(v) \in R^{d \times 1}$ (semantic feature embedding dimension), v represents a node in subgraph G_u, and x_i represents the ith feature word of node v (refer to [13]). $\overrightarrow{LSTM}$ represents the forward LSTM network, $\overleftarrow{LSTM}$ represents the backward LSTM network, θ represents the neural network parameter, and operator $\|$ represents concatenation. Bi-LSTM can learn the potential evolution process of node semantics, leading to a strong expression capability [26].

Then the GCN [27] framework was used to aggregate semantic nodes of $H_S(5)$ to learn the influence relationship between different nodes, which is formally expressed as follows:

$$\begin{aligned} H_S'(v) &= GCN(H_S(v)) = g(A(G_u)H_S(v)W^T + b) \\ A(G_u) &= D^{-1/2}AD^{-1/2} \end{aligned}, \tag{5}$$

where $W \in R^{d \times d}$, $b \in R^d$ are model parameters, g is a non-linear activation function, A is the adjacency matrix of $G(u)$, and D represents $diag(A)$. Since the number of subgraph nodes is fixed, A (G_u) can be calculated efficiently.

3.3.2 Topological Feature Aggregation Module

A topological feature can represent the importance of nodes in the network [28]. To aggregate topological feature embeddings of heterogeneous neighbors for each node, a layer of the GCN model was used for feature aggregation; in particular, the input vector consists of topological features A_v^T and node state S_v^t, which is inspired by the study in reference 14. Then the concatenated vector into the GCN layer was input to generate the node topological features hidden layer vector $H_T'(v)$. The formalization is given as follows:

$$\begin{aligned} H_T(v) &= A_v^T \| S_v^t \\ H_T'(v) &= GCN(H_T(v)) = g(A(G_u)H_T(v)W^T + b). \\ A(G_u) &= D^{-1/2}AD^{-1/2} \end{aligned} \tag{6}$$

Eq. 6 is the same as **Eq. 5**, except for the input and model parameters. We used the GCN to aggregate the topological embeddedness of all heterogeneous neighbors. Obviously, GCN has excellent performance in relation aggregation capability [27].

3.3.3 Heterogeneous Multi-Attribute Hidden Layer Aggregation Module

We can obtain the semantic feature embedding $H_T'(v)$ and structural feature embedding $H_S'(v)$ of each node in the heterogeneous subgraph G_u. To combine these features based on neural network module for each node v of

TABLE 1 | Statistics of the datasets.

Statistics	Weibo	Digg
Num of users	410764	279631
Num of events	7124	3155
User–user	92819	1731657
User–event	1196489	3018196
Event–event	22518	—
Num/dim of user interest topics	20/32	—
Num/dim of event historical topics	20/32	—
Dim of node topology embedding	128	128
Num of samples	52130	34113

subgraph G, the graph attention network [19] was employed. The advantage of this is that since different nodes have different influence contributions to the results, the multi-head GAT learns the influence between different attributes of heterogeneous nodes.

First, we concatenated the hidden layer results of the previous steps, then, following the study in reference 14, we used multi-head GAT and calculated the normalized attention coefficients $H'_f(v)$ as follows:

$$\begin{aligned} H_f(v) &= H'_T(v) \| H'_S(v) \\ H'_f(v) &= \sigma\left(\frac{1}{K}\sum_{k=1}^{1}\sum_{v\in G_u}\alpha_{iv}^{k}\mathbf{W}^{k}H_f(v)\right) \\ \alpha_{iv} &= \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{T}\left[\mathbf{W}H_f(i)\|\mathbf{W}H_f(v)\right]\right)\right)}{\sum_{v\in G_u}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{T}\left[\mathbf{W}H_f(i)\|\mathbf{W}H_f(v)\right]\right)\right)}, \end{aligned} \tag{7}$$

where $\mathbf{a}$ is the attention parameter, $\mathbf{a} \in R^{2d}$, $\mathbf{W} \in R^{d\times d}$ is model parameters, $\cdot^{T}$ represents transposition, $\|$ is the concatenation operation, α_{iv} indicates the importance of node i to node v, and K represents the number of heads.

3.3.4 Output Layer and Loss Function

As shown in **Figure 2c-3**, the full connection layer (FC layer [27]) was used to output the two-dimensional representation of each node, then the current ego-user result was taken out, the ground truth was compared with, and **formula 2** was optimized as the loss function used in our study.

4 EXPERIMENTS

In this section, extensive experiments were conducted with the aim of answering the following research questions:

- (RQ1) How does HetInf perform vs state-of-the-art baselines for influence prediction tasks?
- (RQ2) How do different components, for example, heterogeneous multi-attribute hidden layer aggregation module or semantic feature aggregation module, affect the model performance? How much performance gain is added to these modules?
- (RQ3) How do various hyper-parameters, for example, embedding dimension of keywords or the size of sampled heterogeneous neighbors set, impact the model performance?

4.1 Datasets

Following the previous studies [14], experiments on two public datasets were conducted to quantitatively evaluate our proposed model. The detailed statistics are presented in **Table 1**.

Digg [29]: The Digg dataset is a story collector, which contains the data of stories that were promoted to Digg home page within 1 month in 2009. For each story, this dataset collects the voting lists of all users, the voters' friendship links, and the timestamp of each vote. This dataset comes from the study in reference 30. In our experiment, we took the story as the "event" node and "voting" as the user action to build a heterogeneity graph. Due to the lack of text data in the dataset, the deep framework (**Figure 2c-2**) of semantic information was not used in this dataset.

Weibo [31]: Weibo is the most popular social networking platform in China. This dataset contains 3,000,000 original tweets and retweets and comments of the original tweets from September 28, 2012 to October 29, 2012. At the same time, the dataset also contains the follow relationship between users who participate in these tweets. The dataset comes from in the study in reference 32. In our experiment, we extracted hashtags as events and built the heterogeneous graph of users and events, and the behavior of users participating in events (comment or retweet) is regarded as user action.

4.2 Data Preparation

In view of the imbalance in the number of active neighbors proposed by DeepInf, we set a threshold $n > 5$ (n is the sum of the number of active users and active events).Therefore, less active observation samples are removed, and thus, the sample characteristics involved in training are significantly related to social influence [31]. In order to solve the problem of data skew, the down-sampling method was used to control the positive and negative ratio of samples at 1:3.

Compared with the previous study, the preparation has the following differences:

(1) For the choice of events, due to the shortcomings of the number of participants lack significance, we excluded some events with fewer participants and set the threshold of the number of participants to the top 30% of the distribution of the number of events so as to extract the total number of events in **Table 1**.
(2) In the Weibo dataset, we established the event–event relationship (edge) through the semantic correlation of events. Specifically, the historical text of each event was collected, the tweets text collection was sampled in the time window t, and then the semantic vectorization representation of the event was obtained by the par2vec [33] method. Then we calculated the cosine similarity between events; if the correlation threshold $n > 0.7$, the

TABLE 2 | Prediction performance of different methods on the two datasets (%).

Dataset	Model	AUC	Prec	Rec	F1
Weibo	SVM	77.11	43.27	70.79	53.71
	DeepInf	82.75	48.86	74.13	58.90
	MvInf	83.75	50.18	75.02	60.13
	HetInf-GCN	**85.10**	67.88	73.60	70.62
	HetInf-GAT	85.02	**68.12**	**74.03**	**70.95**
Digg	SVM	90.65	66.82	78.49	72.19
	DeepInf	88.97	68.80	73.49	71.21
	MIvInf	91.11	**70.35**	78.50	74.20
	HetInf-GCN	90.74	69.94	**79.75**	74.52
	HetInf-GAT	**92.03**	70.12	77.63	**74.68**

two events are semantically strongly correlated, and then we established the relation (edge) between events.

(3) For the extraction of node semantic features (**Figure 2c-2**), we fixed the number of keywords of each node (for example, $n = 20$). If the keyword samples of some nodes are insufficient ($n < 20$), we filled zeros to complete the vector to ensure the consistency of the input of the neural network.

4.3 Baselines

Support Vector Machine (SVM) [34]: A support vector machine (SVM) with linear kernel was used as the classification model. Specifically, the splicing of three features (including semantic features, topology features, and action status) was used as the input vector, and the problem was defined as two classification method.

DeepInf [14]: Our framework was compared with the influence network model based on the graph neural network, which constructs homogeneity subgraph based on user relationship and predicts user node action in the future.

MvInf [15]: Our framework was compared with the state-of-the-art graph neural network model MvInf, which introduces a multi-view structure based on DeepInf and uses the complementarity and consistency between different views to enhance learning performance. The difference is that our proposed model is based on the common influence of events and users.

HetInf and Its Variants: In the heterogeneous multi-attribute hidden layer aggregation module, different graph neural network frameworks were used to distinguish the two methods: HetInf–GCN and HetInf–GAT. Separately, HetInf–GAT uses the GAT [19] method as a method to fuse node features, mainly using attention mechanism to obtain the importance between nodes, while the HetInf–GCN method uses the GCN [27] framework to aggregate the node features and calculate the node influence by learning the node relationship of the subgraph.

4.4 Hyper-Parameter

In our proposed method, we used DeepWalk [24] to embed the node topology features; the restart probability of this method is 0.8, and the output vector length is 64 dimensions. We used the ReLU [35] as the activation function sigma (**Eq. 5**, **6**) and used the Adam [36] optimizer for training, with a learning rate of 0.005, and we set dropout = 0.5. We used 50, 25, and 25% of the instances for training, validation, and testing respectively; the batch size of all datasets was set to 256. In order to accommodate more nodes, we set the total number of nodes in the subgraph to 100 (including two different types of nodes).

TABLE 3 | Relative gain in terms of F1 and Prec. against the best baseline (%).

F1	Weibo	Digg	Prec	Weibo	Digg
MvInf	60.13	74.20	MvInf	50.18	70.35
HetInf	70.95	74.68	HetInf	68.12	70.12
Relative gain	17.9%	0.6%	Relative gain	35.7%	–0.3%

4.5 Result and Analysis

4.5.1 (RQ1) Performance Analysis

How does HetInf perform vs state-of-the-art baselines of influence prediction methods? Will users take actions on the events in the future? What are the advantages of the proposed model compared with baseline? In order to answer question RQ1, we applied four indicators to compare with the previous state-of-the-art model (the same evaluation metrics as MvInf).

It should be noted that there are the following differences from the baseline method: in the semantic feature aggregation module, we used Word2Vec to embed each feature word into a vector with a dimension of 32. Specifically, the number of keywords for each node is 20. The output dimension of Bi-LSTM hidden layer is 128 and was used as GCN input (as shown in **Figure 2c-2**). The final output dimension of the GCN module is 128. In the topological feature aggregation module, the output dimension of DeepWalk is 128 and the state feature dimension was 2 (including action state and ego-state), so the GCN's input dimension is 130, and the output dimension of this module is 128 (similar to the DeepInf method). In the multi-attribute hidden layer aggregation module, for HetInf–GCN, we used two layers of GCN as the aggregation function of the module, in which the input layer of the first layer dimension is 256 and the output dimension of the second layer is 128. For HetInf–GAT, we used the GAT method, the input dimension is 256 and the output dimension is 128. Performance report of all models in **Table 2** and **Table 3** in which the best results are highlighted in bold.

(1) It can be seen from the results that in most cases, our proposed model is better than the baseline, especially in the accuracy and F1 value of microblog dataset (F1: 17.9%, Prec.: 35.7%), which proves that we have obtained the gain of accuracy after introducing heterogeneous networks and establishing event influence relations and verified the effectiveness of the proposed framework.

(2) From the results of the Digg dataset, it can be seen that the heterogeneous graph neural network model with two types of nodes can also bring performance gain (F1: 0.6%, Prec.: 0.3%) (lack of semantic information of heterogeneous nodes), which proves that our proposed model can improve the prediction ability of user behavior only through heterogeneous social networks.

4.5.2 (RQ2) Ablation Analysis

HetInf is a deep learning model combining different modules, which calculates the influence between different nodes and

TABLE 4 | Statistics of the datasets.

Dataset	Model	F1	AUC
Weibo	Only-Topology	62.20	81.17
	Only-Semantic	57.89	74.33
	No-LSTM	54.33	72.65
	No-NN-3	64.48	83.31
	HetInf-GAT	70.95	85.10
	HetInf-GCN	70.62	85.02
Digg	No-NN-3	66.28	85.32
	HetInf-GAT	74.71	92.03
	HetInf-GCN	67.72	90.74

predicts user behavior by aggregating the embedding of different types of node attributes. To answer RQ2, we used Auc and F1 indicators as the standard for evaluating results, we conducted ablation studies to evaluate performances of several model variants which include:

1) No-NN-1 that cancels the LSTM method (NN-1) and then concatenates the vectors to embed the representation of the semantic feature to verify the impact of the semantic feature aggregation module on the results; 2) Only-Topology that uses heterogeneous topology encoding (C-2) to represent each node embedding (cancel C-1 module); 3) Only-Semantic that uses heterogeneous topology encoding (C-1) to represent each node embedding (cancel C-2 module); and 4) No-NN-3 that utilizes a FC layer to combine embeddings of different neighbor representation (replace NN-3). It should be noted that the Digg dataset lacks semantic information, so we only tested the results of 4) to verify the effectiveness of heterogeneous multi-attribute hidden layer aggregation module. The results of predicted AUC and F1 values are shown in **Table 4** and **Figure 3**.

(1) The performance of Only-Topology is better than that of Only-Semantic, indicating that the position of key nodes in the network is more influential (such as opinion leaders).
(2) The performance of Only-Semantic is better than that of No-lstm, indicating that Bi-LSTM–based content encoding is superior to "shallow" encoding like FC for capturing "deep" content feature interactions.
(3) HetInf (including GCN and GAT) is better than No-NN-3 shows that graph convolution network plays a role in capturing node type influence.
(4) HetInf–GAT is superior to HetInF–GCN, indicating that graph attention network can obtain more influential potential relationships than GCN method.

4.5.3 (RQ3) Hyper-Parameter Analysis

To answer question RQ3, we investigated how hyper-parameters affect the predictive performance of the model. We conducted a parameter analysis on the Weibo dataset and used F1 value as an evaluation indicator. Specifically, we tested the impact of three key parameters: 1) semantic attribute embedding dimension; 2) head for multi-head attention; and 3) number of keywords. The experimental results are shown in **Figure 4**.

(1) Semantic Attribute Embedding Dimension: As shown in **Figure 4A**, when the semantic attribute dimension d varies from 16 to 256, the overall evaluation indicator is increasing because more dimensions contain more information. However, when the dimension reaches 128, the performance begins to decline, which is likely due to the result of overfitting.
(2) Head for Multi-head Attention: Like DeepInf, we are concerned about the number of GAT heads in heterogeneous multi-attribute hidden layer. As shown in **Figure 4B**, the increase of heads brings benefits to performance, but after more than 8, the performance remains stable but it has a negative impact on efficiency.
(3) Number of Keywords: The feature words of network nodes represent the semantic bias of nodes, which directly affect the prediction results. As shown in **Figure 4C**, when the numbers changes, it means that the amount of semantic information of network nodes increases and the evaluation improves at the same time. However, when the number of keywords exceeds a certain value, it will bring down performance, which is likely due to the noise caused by sampling too many non-

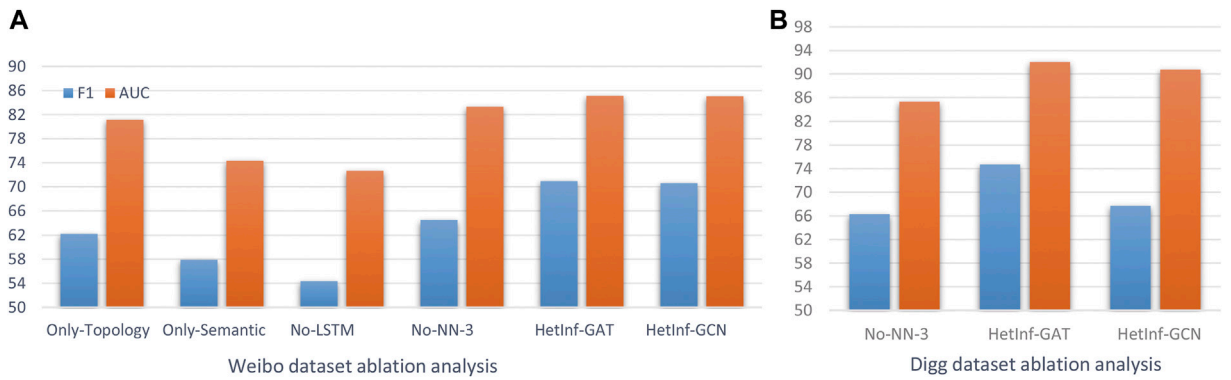


FIGURE 3 | Performances of variant proposed models **(A)** eibo dataset ablation analysis. **(B)** Digg dataset ablation analysis.

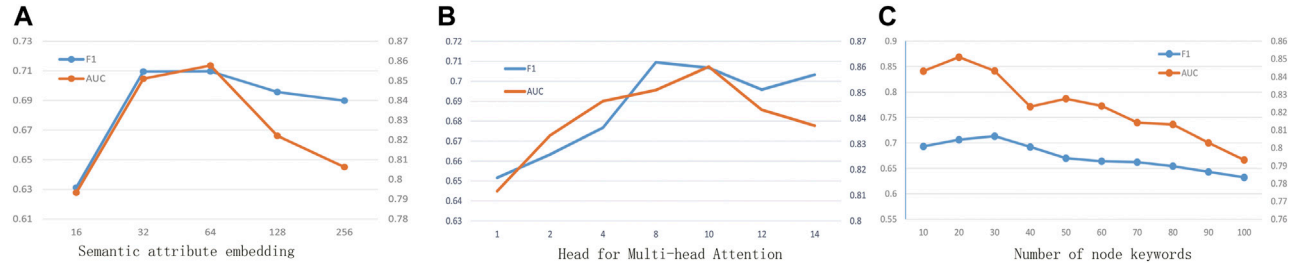


FIGURE 4 | Parameter analysis **(A)** Semantic attribute embedding. **(B)** Head for multi-head attention. **(C)** Number of node Keywords.

significant keywords. As can be seen from the figure, it is best to control the feature words between 20 and 40.

5 RELATED WORK

5.1 Social Influence Prediction

Social influence prediction is a fundamental problem in a social network analysis, which supports downstream tasks. At the micro-perspective [37,38], this problem is mainly modeled by analyzing user relationships. There are many different research directions, such as a user interaction influence analysis [39,40], network structure diversity analysis [41,42], topic influence analysis [10], and influence maximization [43]. Specifically, in the study in reference 44, the existence of social influence was proved by quantitative analysis of mutual influence. The study in reference 31 proposes social local network concepts, using user interaction and network structure to predict user behavior. The study in reference 45 uses topic level influence to model user influence. The study in reference 46 introduces a topic-level influence propagation and aggregation algorithm to derive the indirect influence between nodes. In recent years, with the continuous progress of deep learning, many studies have introduced deep learning into social influence prediction to improve the prediction performance. A popular deep learning method is [14], which provides an end-to-end framework to predict social influence by learning the potential features of users. The study in reference 15 has improved the study in reference 14 to enhance feature representation and result accuracy with a multi-view model. The study in reference 47 proposes a social influence prediction model NNMInf based on neural network multi-label classification. The study in reference 48 introduces a deep neural network framework witch simulate social influence and predict human behavior. Compared with traditional methods, these deep learning models show better learning performance.

5.2 Heterogeneous Graph Neural Network

In recent years, we have identified a huge development of the graph neural network in deep learning technology [27,49], and the state-of-the-art model GAT [19], which represents the method of depth learning-based graphical representation as the graph neural network (GNN), the main idea is as follows: the first step is to calculate the feature representation of neighbor nodes, and the second step is to aggregate neighbors through message passing mechanism to obtain the feature representation of nodes [50].

Recently, the heterogeneous graph neural network has become the main branch of GNN. The main task is to learn the representation of heterogeneous nodes on the graph neural network, so as to adapt to the downstream tasks based on heterogeneous networks. The study in reference 13 realizes node representation of heterogeneous networks by aggregating features of different types of nodes in stages. The study in reference 51 proposes a heterogeneous graph neural network based on hierarchical attention, including node level attention and semantic level attention. Node level attention aims to learn the importance between nodes and their neighbors based on meta-paths, while semantic level attention can learn the importance of different meta-paths. The study in reference 52 proposes a heterogeneous graph neural network method for subgraphs, which trains a classifier to learn the neighbor average features of the random sampling graph of the relational "metagraph." The MAGNN [53] model which contains the node content transformation to encapsulate input node attributes, the intra-meta-path aggregation to incorporate intermediate semantic nodes, and the inter-meta-path aggregation to combine messages from multiple meta-paths. GTN [54], which generates new graph structures by identifying useful connections between unconnected nodes on the original graph, can learn effective node embeddings on the new graphs in an end-to-end fashion. HGNN-AC [55] based on reference 53 proposed a general framework for heterogeneous graph neural network *via* Attribute Completion, including pre-learning of topological embedding and attribute completion with attention mechanism. These heterogeneous graph neural network representation methods enhance the representation ability of nodes and provide a more practical idea for downstream tasks.

6 CONCLUSION

In this study, we studied the problem of influence prediction based on a heterogeneous neural network, introduced a novel model HetInf that combines three neural network modules

models to jointly infer the interaction between events and users in heterogeneous networks, and predicted the future behavior of network users. The local sampling method of heterogeneous networks was improved to capture the law of information dissemination, so as to obtain a more realistic user influence subgraph. Experimental results show that the influence prediction model can benefit from the heterogeneous network as well as joint learning embedding of users and events. In general, the empirical studies verify the effectiveness of our proposed model compared to the baseline methods.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material; further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

LG contributed to the core idea of the experiment design and analysis results under the guidance of BZ. HW assisted in experiment code and experiment analysis. HZ and ZZ analyzed the comparative experiment. BZ supervised the research, provided financial support, and provided the financial support and experimental equipment. BZ is the corresponding author. All authors discussed the results and contributed to the final manuscript.

FUNDING

This work is supported by the National Key Research and Development Program of China (2018YFC0831703) and the National Natural Science Foundation of China (Nos. 61732022, 61732004, 62072131, and 61672020).

REFERENCES

1. Gui H, Liu J, Tao F, Jiang M, Norick B, Kaplan L, et al. Embedding Learning with Events in Heterogeneous Information Networks. *IEEE Trans Knowl Data Eng* (2017) 29:2428–41. doi:10.1109/TKDE.2017.2733530
2. Mehrizi MA, Corò F, Cruciani E, D'Angelo G. Election Control through Social Influence with Unknown Preferences. In: International Computing and Combinatorics Conference. New York, NY: Springer (2020). p. 397–410. doi:10.1007/978-3-030-58150-3_32
3. Dholakia UM, Bagozzi RP, Pearo LK. A Social Influence Model of Consumer Participation in Network- and Small-Group-Based Virtual Communities. *Int J Res marketing* (2004) 21:241–63. doi:10.1016/j.ijresmar.2003.12.004
4. Kim YA, Srivastava J. Impact of Social Influence in E-Commerce Decision Making. vol. 258 of ACM International Conference Proceeding Series. In: ML Gini, RJ Kauffman, D Sarppo, C Dellarocas, F Dignum, editors. Proceedings of the 9th International Conference on Electronic Commerce: The Wireless World of Electronic Commerce, 2007; August 19-22, 2007; Minneapolis, MN, USA. University of MinnesotaACM (2007). p. 293–302. doi:10.1145/1282100.1282157
5. Ye M, Liu X, Lee W. Exploring Social Influence for Recommendation: a Generative Model Approach. In: WR Hersh, J Callan, Y Maarek, M Sanderson, editors. The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12; August 12-16, 2012; Portland, OR, USA. ACM (2012). p. 671–80.
6. Hosni AIE, Li K, Ahmad S. Minimizing Rumor Influence in Multiplex Online Social Networks Based on Human Individual and Social Behaviors. *Inf Sci* (2020) 512:1458–80. doi:10.1016/j.ins.2019.10.063
7. Zhou X, Wu B, Jin Q. User Role Identification Based on Social Behavior and Networking Analysis for Information Dissemination. *Future Generation Comput Syst* (2019) 96:639–48. doi:10.1016/j.future.2017.04.043
8. Li S, Zhao D, Wu X, Tian Z, Li A, Wang Z. Functional Immunization of Networks Based on Message Passing. *Appl Math Comput* (2020) 366:124728. doi:10.1016/j.amc.2019.124728
9. Liu L, Tang J, Han J, Yang S. Learning Influence from Heterogeneous Social Networks. *Data Min Knowl Disc* (2012) 25:511–44. doi:10.1007/s10618-012-0252-3
10. Barbieri N, Bonchi F, Manco G. Topic-Aware Social Influence Propagation Models. *Knowl Inf Syst* (2013) 37:555–84. doi:10.1007/s10115-013-0646-6
11. Zhang Z-K, Liu C, Zhan X-X, Lu X, Zhang C-X, Zhang Y-C. Dynamics of Information Diffusion and its Applications on Complex Networks. *Phys Rep* (2016) 651:1–34. doi:10.1016/j.physrep.2016.07.002
12. Quan Y, Jia Y, Zhou B, Han W, Li S. Repost Prediction Incorporating Time-Sensitive Mutual Influence in Social Networks. *J Comput Sci* (2018) 28:217–27. doi:10.1016/j.jocs.2017.11.015
13. Zhang C, Song D, Huang C, Swami A, Chawla NV. Heterogeneous Graph Neural Network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, August 4–8, 2019 (2019) p. 793–803. doi:10.1145/3292500.3330961
14. Qiu J, Tang J, Ma H, Dong Y, Wang K, Tang J. Deepinf: Social Influence Prediction with Deep Learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Halifax, NS, August 13–17, 2017 (2018). p. 2110–9.
15. Xu H, Jiang B, Ding C. Mvinf: Social Influence Prediction with Multi-View Graph Attention Learning. *Cogn Comput* (2021) 1–7. doi:10.1007/s12559-021-09822-z
16. Calais Guerra PH, Veloso A, Meira W, Jr, Almeida V. From Bias to Opinion: A Transfer-Learning Approach to Real-Time Sentiment Analysis. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, CA, August 21–24, 2011 (2011). p. 150–8.
17. Tong H, Faloutsos C, Pan J. Fast Random Walk with Restart and its Applications. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006); 18-22 December 2006; Hong Kong, China. IEEE Computer Society (2006) p. 613–22. doi:10.1109/icdm.2006.70
18. Goldberg Y, Levy O. *word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method*. CoRR abs/1402.3722 (2014).
19. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. *Graph Attention Networks*. arXiv preprint arXiv:1710.10903 (2017).
20. Niepert M, Ahmed M, Kutzkov K. Learning Convolutional Neural Networks for Graphs. vol. 48 of JMLR Workshop and Conference Proceedings. In: M Balcan KQ Weinberger, editors. Proceedings of the 33nd International Conference on Machine Learning, ICML 2016; June 19-24, 2016; New York City, NY, USA. JMLR.org (2016). p. 2014–23.
21. Duan Y, Chen Z, Wei F, Zhou M, Shum H. Twitter Topic Summarization by Ranking Tweets Using Social Influence and Content Quality. In: M Kay C Boitet, editors. COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers; 8-15 December 2012; Mumbai, India. Indian Institute of Technology Bombay (2012). p. 763–80.
22. Zheng J, Cai F, Chen H. Incorporating Scenario Knowledge into A Unified Fine-Tuning Architecture for Event Representation. In: J Huang, Y Chang, X Cheng, J Kamps, V Murdock, J Wen, et al. editors. Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020; July 25-30, 2020; Virtual Event, China. ACM (2020) p. 249–58. doi:10.1145/3397271.3401173
23. Ramos J. Using Tf-Idf to Determine Word Relevance in Document Queries. *Proc first instructional Conf machine Learn (Citeseer)* (2003) 242:29–48.
24. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online Learning of Social Representations. In: SA Macskassy, C Perlich, J Leskovec, W Wang,

R Ghani, editors. The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14; August 24 - 27, 2014; New York, NY, USA. ACM (2014). p. 701–10.
25. Beamer S, Asanovic K, Patterson D. Direction-Optimizing Breadth-First Search. In: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, November 11–15, 201. IEEE (2012). p. 1–10. doi:10.1109/sc.2012.50
26. Huang Z, Xu W, Yu K. *Bidirectional Lstm-Crf Models for Sequence Tagging*. arXiv preprint arXiv:1508.01991 (2015).
27. Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. Conference Track Proceedings (OpenReview.net). In: 5th International Conference on Learning Representations; April 24-26, 2017; Toulon, France. ICLR 2017 (2017).
28. Ugander J, Backstrom L, Marlow C, Kleinberg J. Structural Diversity in Social Contagion. *Proc Natl Acad Sci* (2012) 109:5962–6. doi:10.1073/pnas.1116502109
29. Lerman K, Ghosh R, Surachawala T. *Social Contagion: An Empirical Study of Information Spread on Digg and Twitter Follower Graphs*. CoRR abs/1202.3162 (2012).
30. Hogg T, Lerman K. Social Dynamics of Digg. *EPJ Data Sci* (2012) 1:5. doi:10.1140/epjds5
31. Zhang J, Tang J, Li J, Liu Y, Xing C. Who Influenced You? Predicting Retweet via Social Influence Locality. *ACM Trans Knowl Discov Data* (2015) 9:1–26. doi:10.1145/2700398
32. Zhang J, Liu B, Tang J, Chen T, Li J. Social Influence Locality for Modeling Retweeting Behaviors. In: F Rossi, editor. IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence; August 3-9, 2013; Beijing, China. IJCAI/AAAI (2013). p. 2761–7.
33. Le QV, Mikolov T. Distributed Representations of Sentences and Documents. vol. 32 of JMLR Workshop and Conference Proceedings. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014; 21-26 June 2014; Beijing, China. JMLR.org (2014). p. 1188–96.
34. Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J. Liblinear: A Library for Large Linear Classification. *J machine Learn Res* (2008) 9:1871–4. doi:10.1145/1390681.1442794
35. Agarap AF. *Deep Learning Using Rectified Linear Units (Relu)*. arXiv preprint arXiv:1803.08375 (2018).
36. Zhang Z. Improved Adam Optimizer for Deep Neural Networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). IEEE (2018). p. 1–2. doi:10.1109/iwqos.2018.8624183
37. Gao L, Liu Y, Zhuang H, Wang H, Zhou B, Li A. Public Opinion Early Warning Agent Model: A Deep Learning cascade Virality Prediction Model Based on Multi-Feature Fusion. *Front Neurorobot* (2021) 15:674322. doi:10.3389/fnbot.2021.674322
38. Li M, Wang X, Gao K, Zhang S. A Survey on Information Diffusion in Online Social Networks: Models and Methods. *Information* (2017) 8:118. doi:10.3390/info8040118
39. Lee RK-W, Hoang T-A, Lim E-P. Discovering Hidden Topical Hubs and Authorities in Online Social Networks. In: M Ester D Pedreschi, editors. Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018; May 3-5, 2018; San Diego, CA, USA. San Diego Marriott Mission ValleySIAM (2018). p. 378–86. doi:10.1137/1.9781611975321.43
40. Cha M, Haddadi H, Benevenuto F, Gummadi PK. In: WW Cohen S Gosling, editors. Measuring user influence in twitter: The million follower fallacy. In Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010; May 23-26, 2010; Washington, DC, USA. The AAAI Press (2010).
41. Qiu J, Chen Q, Dong Y, Zhang J, Yang H, Ding M, et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-training. In: R Gupta, Y Liu, J Tang, BA Prakash, editors. KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; August 23-27, 2020; Virtual Event, CA, USA. ACM (2020). p. 1150–60.
42. Li S, Jiang L, Wu X, Han W, Zhao D, Wang Z. A Weighted Network Community Detection Algorithm Based on Deep Learning. *Appl Maths Comput* (2021) 401:126012. doi:10.1016/j.amc.2021.126012
43. Tian S, Mo S, Wang L, Peng Z. Deep Reinforcement Learning-Based Approach to Tackle Topic-Aware Influence Maximization. *Data Sci Eng* (2020) 5:1–11. doi:10.1007/s41019-020-00117-1
44. Singla P, Richardson M. Yes, There Is a Correlation: - from Social Networks to Personal Behavior on the Web. In: J Huai, R Chen, H Hon, Y Liu, W Ma, A Tomkins, et al. editors. Proceedings of the 17th International Conference on World Wide Web, WWW 2008; April 21-25, 2008; Beijing, China. ACM (2008). p. 655–64.
45. Hu J, Meng K, Chen X, Lin C, Huang J. Analysis of Influence Maximization in Large-Scale Social Networks. *SIGMETRICS Perform Eval Rev* (2014) 41:78–81. doi:10.1145/2627534.2627559
46. Liu L, Tang J, Han J, Jiang M, Yang S. Mining Topic-Level Influence in Heterogeneous Networks. In: J Huang, N Koudas, GJF Jones, X Wu, K Collins-Thompson, A An, editors. Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010; October 26-30, 2010; Toronto, Ontario, Canada. ACM (2010). p. 199–208. doi:10.1145/1871437.1871467
47. Wang X, Guo Z, Wang X, Liu S, Jing W, Liu Y. Nnmlinf: Social Influence Prediction with Neural Network Multi-Label Classification. In: Proceedings of the ACM Turing Celebration Conference-China (2019). p. 1–5.
48. Luceri L, Braun T, Giordano S. Social Influence (Deep) Learning for Human Behavior Prediction. In: S Cornelius, K Coronges, B Gonçalves, R Sinatra, A Vespignani, editors. *International Workshop on Complex Networks*. Cham: Springer (2018). p. 261–9. doi:10.1007/978-3-319-73198-8_22
49. Hamilton WL, Ying Z, Leskovec J. Inductive Representation Learning on Large Graphs. In: I Guyon, U von Luxburg, S Bengio, HM Wallach, R Fergus, SVN Vishwanathan, et al. editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017; December 4-9, 2017; Long Beach, CA, USA (2017). p. 1024–34.
50. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans Neural Netw Learn Syst.* (2021) 32:4–24. doi:10.1109/tnnls.2020.2978386
51. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, et al. Heterogeneous Graph Attention Network. In: L Liu, RW White, A Mantrach, F Silvestri, JJ McAuley, R Baeza-Yates, et al. editors. The World Wide Web Conference, WWW 2019; May 13-17, 2019; San Francisco, CA, USA. ACM (2019). p. 2022–32. doi:10.1145/3308558.3313562
52. Yu L, Shen J, Li J, Lerer A. *Scalable Graph Neural Networks for Heterogeneous Graphs*. CoRR abs/2011.09679 (2020).
53. Fu X, Zhang J, Meng Z, King I. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In: Y Huang, I King, T Liu, M van Steen, editors. WWW '20: The Web Conference 2020; April 20-24, 2020; Taipei, Taiwan. ACM/IW3C2 (2020). p. 2331–41. doi:10.1145/3366423.3380297
54. Yun S, Jeong M, Kim R, Kang J, Kim HJ. Graph Transformer Networks. In: HM Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, EB Fox, R Garnett, editors. Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019; December 8-14, 2019; Vancouver, BC, Canada (2019). p. 11960–70.
55. Jin D, Huo C, Liang C, Yang L. Heterogeneous Graph Neural Network via Attribute Completion. In: J Leskovec, M Grobelnik, M Najork, J Tang, L Zia, editors. WWW '21: The Web Conference 2021; April 19-23, 2021; Virtual Event/Ljubljana, Slovenia. ACM/IW3C2 (2021). p. 391–400. doi:10.1145/3442381.3449914

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Physics

ORIGINAL RESEARCH
published: 26 January 2022
doi: 10.3389/fphy.2021.763904

Node Classification in Attributed Multiplex Networks Using Random Walk and Graph Convolutional Networks

Beibei Han, Yingmei Wei, Lai Kang, Qingyong Wang and Yuxuan Yang*

College of Systems Engineering, National University of Defense Technology, Changsha, China

Node classification, as a central task in the graph data analysis, has been studied extensively with network embedding technique for single-layer graph network. However, there are some obstacles when extending the single-layer network embedding technique to the attributed multiplex network. The classification of a given node in the attributed multiplex network must consider the network structure in different dimensions, as well as rich node attributes, and correlations among the different dimensions. Moreover, the distance node context information of a given node in each dimension will also affect the classification of the given node. In this study, a novel network embedding approach for the node classification of attributed multiplex networks using random walk and graph convolutional networks (AMRG) is proposed. A random walk network embedding technique was used to extract distant node information and the results are considered as pre-trained node features to be concatenated with the original node features inputted into the graph convolutional networks (GCNs) to learn node representations for each dimension. Besides, the consensus regularization is introduced to capture the similarities among different dimensions, and the learnable neural network parameters of GCNs for different dimensions are also constrained by the regularization mechanism to improve the correlations. As well as an attention mechanism is explored to infer the importance for a given node in different dimensions. Extensive experiments demonstrated that our proposed technique outperforms many competitive baselines on several real-world multiplex network datasets.

Keywords: attributed multiplex network, node classification, random walk, graph convolutional networks, network embedding

OPEN ACCESS

Edited by:
Shudong Li,
Guangzhou University, China

Reviewed by:
Chengyi Xia,
Tianjin University of Technology, China
Jin Huang,
South China Normal University, China

****Correspondence:***
Yingmei Wei
weiyingmei@nudt.edu.cn

Specialty section:
This article was submitted to Social Physics, a section of the journal Frontiers in Physics

Received: *24 August 2021*
Accepted: *21 December 2021*
Published: *26 January 2022*

Citation:
Han B, Wei Y, Kang L, Wang Q and Yang Y (2022) Node Classification in Attributed Multiplex Networks Using Random Walk and Graph Convolutional Networks. Front. Phys. 9:763904. doi: 10.3389/fphy.2021.763904

1 INTRODUCTION

Node classification [1,2] is a basic and central task in the graph data analysis, such as the user division in social networks [3], the paper classification in citation network [4]. Network embedding techniques (or network representation learning or graph embedding) utilize a dense low-dimensional vector to represent nodes [5–7]. This provides an efficient way to solve various graph analytic problems, including node classification [5–7], recommendation [8,9], link prediction [10,11]. Most existing network embedding techniques for node classification are designed for standard single-layer graph networks [1,2,5,12–14], such as DeepWalk [13], node2vec [10], LINE [12], and classical graph neural networks (GNNs)

such as graph convolutional networks (GCNs) [5], GAT Veličković and Cucurull [15], and GraphSAGE [14]. However, most real-world complex interacting systems [16] are modeled as multilayer graph networks, including social networks [3], citation-collaboration networks [4], which are formed by several layers describing interactions of various types. For example, two users could be connected to each other across multiple social networks (e.g., Twitter, Facebook, and LinkedIn). Using multilayer graph networks can provide more comprehensive and accurate description about these two users. When the same set of nodes are connected in the way of multiple link types or relationship types, the resulting multilayer graph network is also called multiplex graph network or multiplex network [17,18] [1] If the nodes in multiplex graph network contain attributes, such network is called attributed multiplex graph network. The attributes can provide useful guidance to perform node classification graph data analysis. For example, if two users in a social network share hobbies or interests, these two users may belong to the same cluster.

Several studies have been conducted on multiplex network representation learning. However, some issues remain that require further consideration. For instance, previous techniques such as PMNE [19], MELL [20], MVE [21] and MNE [11] have learned to integrate node embedding information from different dimensions in multiplex network representations. However, these techniques have mostly overlooked node attributes. Other models that consider node attributes (e.g., mGCN [22], MGCN [23], DMGI [24], and HAN [25]) have either failed to consider interactions among diverse dimensions (mGCN) or focus on multiplex graphs with explicit adjacency links among different dimensions (MGCN [23]). DMGI and HAN consider heterogeneous graphs constructed based on the meta-path between different node, which differs from the multiplex network. MGAT [26] introduces a constrained regularization term in GAT [15] to learn the interactions between different dimensions, that is learnable parameters constraint of GAT. However, MGAT fails to consider the node embedding matrices similarity [27] among different relationships. Furthermore, while GAT can be calculated in parallel with multi-head attention, the memory complexity for parameter storage is higher than that of the GCN model [5]. In addition, MGAT utilizes a two-layer GAT to integrate node information from its neighbors, which only captures information from 2-hop nodes. Including more than two layers in a GNN often results in over-smoothing [7]. However, the node2vec network embedding [10], which is based on random walk technique, can be used to search for 10-hop contextual information [10] and to capture the structural equivalence (i.e. two nodes are far apart each other but have the same structural roles). This suggests that information from a larger receptive domain helps to capture more comprehensive node representations, which can be used to improve the results of node classification.

Recently, the Graph Neural Network (GNN) deep nonlinear network embedding framework represented by GCN [5], which can encode node attributes and network structure simultaneously, has achieved great success on the node classification graph data analysis task for attributed single-layer network. The direct method about node classification of the attributed multiplex network is to extend the GCN [5] to multiplex network. However, some obstacles are existed. First, different dimensions of an given attributed multiplex network share the same node set and node attributes. Hence, different dimensions are typically similarities or may have some characteristics in common [28]. For instance, citation networks represent citations between papers. Similarly, paper similarity networks represent the commonality among papers as articles that cite each other typically share a common research topic. Therefore, the citation and paper similarity dimensional networks exhibit a certain degree of overlap. Second, different dimensions of multiplex networks are related [17]. For example, in social networks, friend relationship dimensional network can determine the topology of a message forwarding dimensional network. Besides, the degree of importance differs in dimensional networks as the significance of a given node may vary in different dimensions. Third, two nodes may share similar structure roles but are far apart each other (i.e. structural equivalence [10]). These two distance nodes may belong to the same cluster. Therefore, the primary challenge for node classification of multiplex network is then designing a model to extract the node information, oriented by the downstream node classification task, capable of generating a comprehensive embedding (consensus) that considers node attributes, their interaction and similarities among different dimensions, the corresponding degree of importance in diverse dimension networks, and the distance node context information.

In this paper, we propose a novel graph embedding framework for the node classification of attributed multiplex graph to solve the above mentioned problems. At first, a random walk network embedding technique was included to address the over-smoothing problem [7] that occurs in GCNs, which is unable to capture distance node context information (more than 2-hop). We use the node2vec random walk network embedding to learn distance node context information for each dimension, and the obtained node embeddings are considered as pre-trained node features, which learn the distance node neighborhood to capture the structural equivalence. Then, pre-trained node features are concatenated with the original node attributes to form new node attributes inputted into the two-layer GCNs [5] to learn each dimensional graph network respectively. At the same time, a regularized consistency constraint was then introduced to node embeddings from different dimension to learn similarities [28] between nodes and their counterparts in others dimension. And the learnable weight parameters of different GCNs for different dimensional graph networks were then constrained using a regularization term [26]. Finally, the attention mechanism is

[1]In this paper, we use the terminology *graph network*, *network*, *graph* interchangeably.

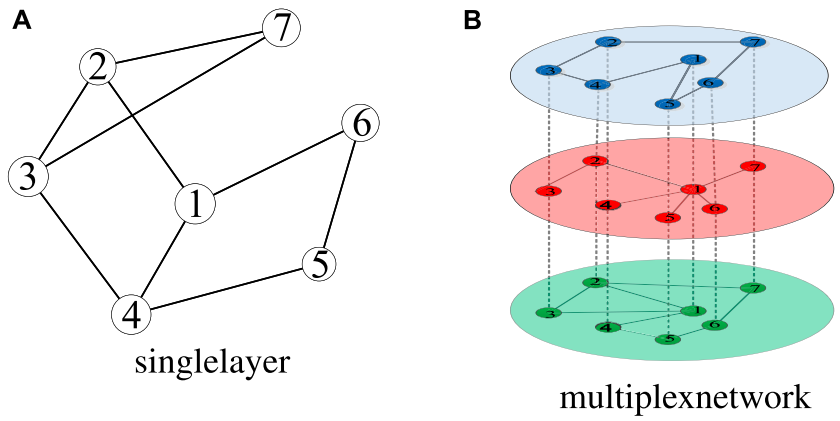


FIGURE 1 | Single-layer and multiplex networks with the same set of nodes but different link types. **(A)** The single-layer network. **(B)** Different nodes are connected to each other by a continuous line in each relationship. A single node is connected to its counterparts by a dashed line.

used to adaptively learn the importance weights of a given node in different dimensional, prior to integrating the node embedding results from different dimensions to generate global consensus node representations. This model can be trained end-to-end oriented by the downstream node classification task. In this way, a more comprehensive, informative, and high-quality node representation for the node classification can be achieved using these strategies discussed above. The primary contributions of the proposed technique can be summarized as follows.

- We provide a novel node classification methodology for attributed multiplex networks using random walk network embedding and graph convolutional networks (AMRG), which can fuse the node attributes and capture distant node context information.
- We use regularized constraints to learn cross dimensional similarities and correlations among different dimensions. Then, the integration of node embeddings from different dimensional is performed based on an attention mechanism.
- Extensive experiments were conducted to evaluate the effectiveness and efficiency of this approach, by comparing it with some competitive baselines on real-world attributed multiplex networks.

The remainder of this paper is organized as follows. First, we summarize related work in **Section 2** and then introduce our approach in **Section 3**. **Section 4** provides experimental conditions and results. Finally, conclusions are discussed in **Section 5**.

2 RELATED WORK

This section summarizes related studies on network embedding for single-layer networks (**Figure 1A**) and multiplex networks (**Figure 1B**). The acquired node embeddings can be used to perform node classification tasks.

2.1 Single-Layer Network Embedding

Network embedding methods [1,5,10,12,13] are used to learn low-dimensional and dense vector representations for nodes in real graph networks, while preserving network structure and facilitating further analysis of graph networks. Various network embedding techniques have been proposed based on deep learning, inspired by word2vec models such as Skip-Gram [29], including DeepWalk [13] and node2vec [10]. DeepWalk [13] first performs a random walk on a network to generate an unbiased random sequence composed of nodes. The neural network (Skip-Gram) is subsequently used to train network node representations by treating nodes as words and node sequences as sentences. Node2vec [10] extends DeepWalk by introducing two parameters (p and q) used to improve the random walk strategy (i.e. BFS and DFS) exploring a more comprehensive graph structure (a biased random walk). Other network embedding models have focused on mining and analysis for specific network structures. For example, LINE [12] is a classic approach that learns node embedding information by preserving both first-order and second-order proximities in the graph. Similarly, SDNE [30] utilizes a semi-supervised deep autoencoder model to capture first-order and second-order proximities. NetMF [31] unifies DeepWalk, LINE, and node2vec into a single matrix factorization framework.

The techniques discussed above focus on mining graph structure, without considering node attribute information. However, nodes in real-world networks often contain rich attribute data, such as abstract text in a publication network and user profiles in social networks (called attributed networks). Considering node attribute information in the learning process has been shown to improve the quality of network representation learning and provide a more comprehensive node embedding strategy to facilitate downstream tasks [5,32–34]. TADW [32] has been used to demonstrate the equivalence between DeepWalk and matrix factorization in attributed network representation learning. It was also the first algorithm used to jointly learn node attributes (textural features) and network structure, which are achieved via matrix factorization. However, TADW only

considers second-order and higher-order proximity, leaving out first-order proximity considerations (i.e., homophily properties). HSCA [33] jointly learns homophily data, structural content, and node attributes to develop an effective node representation. MIRand [34] is an unsupervised algorithm for attributed single-layer graph network embedding based on random walk. This approach first establishes a two-layer graph network, one of which depicts structural information from the input graph while the other describes node attributes or content. MIRand performs the random walk according to node informativeness, intelligently traversing between structure and attribute layers.

Inspired by the success of CNNs in computer vision, graph neural networks (GNNs) [35] generalize 2D convolutions from Euclidean images to non-Euclidean graph data, which provides a powerful end-to-end method for learning node representations while addressing graph-related tasks. Graph convolutions are performed by aggregating neighborhood node information, which naturally considers node attributes. Representative work concerning convolution operators applied to graph data is found in Graph Convolutional Network (GCN) [5]. Michael Schlichtkrull et al. first applied a GCN framework to model relational data, focusing on knowledge graph datasets, which is called Relational Graph Convolutional Networks (R-GCNs) [36]. In contrast, GAT [15] models specify different weights for varying neighborhood nodes when performing convolution operations. GAT assumes that different neighboring nodes exhibit different importance levels for the objective node while aggregating neighboring nodes. GraphSAGE [14] is an extension of the GCN framework that uses inductive node embedding. For a complete overview of network embedding techniques, readers are referred to recent studies on the topic [35,37].

2.2 Multiplex Network Embedding

Although these techniques have proven to be effective and efficient in various scenarios, they each attempt to process standard single-layer graphs, which implies the graph only consists of one type of relationship as shown in **Figure 1A**. However, in practical applications, most networks exhibit multiple relationships between nodes. For example, in social networks, the relationship between two users could be friendship, co-worker, or simply advice [38]. Although these diverse relationships can independently form different networks to be analyzed separately, specific interactions and associations exist among them [17,28].

PMNE [19] uses three methods to learn global embedding information for analyzing multiplex networks, including network aggregation, result aggregation, and layer co-analysis which considers interaction information and thus achieves the best overall performance than network aggregation and result aggregation. Ryuta Matsuno et al. proposed MELL [20] for multiplex networks. It first requires embedding vectors, for the same nodes in diverse relationships, to be close to each other in order to share all layer structures. It then introduces a layer vector that can capture each layer's connectivity for use in differentiating edge probabilities in each relationship. As such, MELL focuses specifically on link prediction tasks. MVE [21] is a novel collaboration framework for multiplex network embedding, which promotes the collaboration of different views and introduces an attention mechanise to learning the weights of different views. Such can obtain robust node embedding results. However, MVE only consideres the network structure (i.e. attributes of nodes are ignored). MNE [11] uses one high-dimensional common embedding and a lower-dimensional additional embedding for each type of relationship, each of which can be learned jointly based on a unified framework. MANE [39] jointly models both connections in each relationship and network interactions from different relationships in a unified framework. Essentially, MANE focuses on processing heterogeneous graphs with different types of nodes and edges. However, node attributes are not considered by the models discussed above.

HAN [25] uses a novel heterogeneous graph neural network with node-level and semantic-level attentions for attributed multiplex networks, generating node embeddings *via* aggregating features from meta-path based neighbors. This approach attempts to describe heterogeneous graphs generated from meta-paths considering the semantics between nodes. Similarly, mGCN [22] utilizes GCN to learn node representations for each relationship. In order to jointly learn cross-layer interactions, the authors used a weighted average over relation-specific representations to produce generalized descriptions in which weights were calculated based on projection matrices from different networks. Unlike in mGCN, our proposed approach adds regularized consensus constraints and trainable weight parameters constraints among GCN for different dimensional graph networks on the objective function. In fact, the mGCN only learns node embeddings for each dimension, using a weighted average of the embedding results to generate overall node representations without considering the interactions between different dimensional networks. Masha Ghorbani et al. extended the GCN model to form a multi-layer graph embedding called MGCN [23]. This approach utilizes GCN models to learn node representations within relationships. However, MGCN focuses on multi-layer graphs with explicit adjacency links between nodes with different relationship types. The types of nodes found in these relationship networks can vary widely. For instance, one layer could denote an airport network, while another describes a power grid. MGAT [26], an extension of the GAT model, introduces regularization terms for model parameters on the objective function to optimize multiplex network embedding. The primary difference between our method and MGAT is that we extend the GCN model while requiring less memory than GAT for parameter storage [15]. Furthermore, our technique is advantageous because it uses a random walk network embedding technique to learn 10-hop node information [10] as the pre-trained node feature which concatenated with the original node features, and resulting new node features were input into the GCN model, while MGAT only learns 2-hop data. MGAT also fails to consider the similarities of node embeddings between different dimensional graph network.

The difference between single-layer and multiplex graph network embedding: With the advent of the big data where many different links of interconnected objects, it is difficult to

model these interacting objects as single-layer graph networks but can naturally model using multiplex graph networks to describe different links. For example, two users could be connected to each other across multiple social platform (e.g. Twitter, Facebook, and LinkedIn). Therefore, each social relationship can be modeled as a graph network. The traditional graph network analysis methods with single-layer network embedding technique can be utilized to analyze these three graph networks separately, which may result in incorrect analysis results. As the single-layer graph network can only describe some or even biased information between nodes. Another way is to transform these three graph network (e.g. Twitter, Facebook, and LinkedIn) as a weighted or an unweighted single-layer graph network. The weight represents the number of link types (i.e. 0, 1, 2, 3) between connected nodes. Then the single-layer network embedding methods can be used on the transformed single-layer graph network. Although it is easy to perform the graph network analysis in this way, the interactive information among different relationships through the same nodes in different links are ignored. The multiplex graph network can model more comprehensively characterize of the complex systems than single-layer graph network. Besides, most of the multiplex graph network embedding methods capture the interactive information and common information among different relationships, and the differences between different relationships. Therefore, the graph analysis results of the multiplex graph network embedding are more accurate than single-layer network embedding.

3 PROPOSED MODEL

3.1 Problem Statement and Framework

Attributed Multiplex Graph Network. Given a single-layer attributed network formally denoted as $G = \{V, E, X\}$, the vertices V represent n nodes in the graph. The term E is a set of edges representing the presence of a connection or relationship between two nodes, where $e_{ij} = (v_i, v_j) \in E$ describes the relationship between node v_i and node v_j. In addition, $|V|$ and $|E|$ denote the size of the vertice and edge sets respectively, and $X \in \mathrm{R}^{n \times F}$ is a matrix that represents attributes for the n nodes, F represents the dimension of node features, A is an adjacency matrix for the graph G (with a size of $|V| \times |V|$), and $A_{ij} \in \{0, 1\}$ denotes connections for unweighted network graphs. The condition $A_{ij} = 1$ represents a link between v_i and v_j, otherwise $A_{ij} = 0$. In practical applications, A is typically sparse and high-dimensional, especially for very large-scale networks. Attributed multiplex networks with $|M|$ different relationship types can be represented as $G = \{V, E^{(1)}, E^{(2)}, \ldots, E^{(M)}, X\}$, where $G^{(r)} = \{V, E^{(r)}, X\}$ is a graph of the relation type (or dimension) r and $A = \{A^{(1)}, A^{(2)}, \ldots, A^{(M)}\}$ is a set of adjacency matrices for the graph G. Multiplex network embedding attempts to learn global consensus node representations for each node $v_i \in V$ with a d dimensional dense vector, through better collaboration among different dimensions. This suggests the correlations among diverse relation types should be considered for a comprehensive and informative node representation. Low dimensional and consensus vectors can be represented as $z_i \in Z \in R^{n \times d}$ for each node $v_i \in V$, where $d \ll |V|$. These notations are summarized in **Table 1**.

An Overview of the Framework. The overall framework for AMRG is illustrated in **Figure 2**. It is primarily composed of four components, including 1) a random walk network embedding model used to capture distance neighboring node information as pre-trained node features, 2) dimension specific node embeddings with the GCN model, 3) cross dimension learning, and 4) an attention-based mechanism used to learn node importance in different dimensions for fusing different dimensions adequately.

We use the node2vec random walk network embedding technique to capture distance node feature of each dimensional graph network, and then the averaged node embedding of $|M|$ node embeddings are concatenated with the original node feature. The resulting is considered as new node features, which are inputted into the GCN model. Then the GCN model can be utilized for the relation-type specific network G^r, to learn a set of node representations H_r. However, unlike the conventional GCN method [5], a weight was added to the self-connections which is down in the same way in [24]. Large weights ($w > 1$) indicate the node itself plays a more important role in generating its embedding than its neighboring nodes in the process of aggregating neighbor information. Furthermore, learnable weight parameters of $|M|$ GCNs are constrained using a regularization term [26]. In addition, we introduce a regularized consistency constraint for each network embedding $H_{r \in M}$ to capture node similarities from their counterparts. These two constraints from different dimensional graph network can be beneficial [17,19,24] for the downstream node classification, as this can capture more comprehensive information of the multiplex network. Finally, a global consensus node embedding was generated by weighted average different dimensional network embeddings based on the attention mechanism. This obtained embedding is a comprehensive, higher-quality, informative node representation, which can be used for classification and visualization tasks.

3.2 Capturing Distance Neighboring Node Information

The advantage of GCN is that it not only considers the network structure, but also fuses the node attributes. GCN is typically using two convolution layers [5], which means that it can only captures 2-hop node neighboring information. However, more than two convolution layers will result in the over-smoothing problem. Node information from a larger receptive domain (10-hop) with node2vec random walk network embedding technique [10] can help to capture richer node features. However, node2vec leaves out of consideration the node attributes. We thus combine the node2vec random walk network embedding technique and GCN to learn the node embeddings of multiplex graph network. The resulting node embeddings can not only fuse node's attributes, but also make up for GCN's inability to learn distance node information to capture the structural equivalence.

Given the rth dimensional network, we can utilize the node2vec random walk technique to learn 10-hop neighboring

TABLE 1 | Notations.

Notation	Description
$G = \{G^{(1)}, G^{(2)}, \ldots, G^{(M)}\}$	The multiplex network
G^r	The network for dimension r
V	Set of vertices
$E = \{E^{(1)}, E^{(2)}, \ldots, E^{(M)}\}$	Set of edges
$\|V\|$	Number of vertices
$\|E\|$	Number of edges
$\|M\|$	Number of relationship types
$A = \{A^{(1)}, A^{(2)}, \ldots, A^{(M)}\}$	Adjacency matrices for G
F	Dimension of node features
d	Dimension of learned node representations
d_{rw}	Dimension of learned node representations for distance node
v_i	The node i
$A^r \in R^{n\times n}$	Adjacency matrix for G^r
$H_r \in R^{n\times d}$	Node representations matrix for G^r
$X \in R^{n\times F}$	The node feature matrix
$z_i \in R^{1\times d}$	The global consensus node embedding for node i
$Z \in R^{n\times d}$	The global consensus node embedding matrix

node information. The resulting network embedding can be represented as $X_{rw}^r \in R^{n\times d_{rw}}$ $(d_{rw} \ll n)$. For $|M|$ different dimensional networks of multiplex graph network, we can use the same method to acquire $X_{rw}^1, X_{rw}^2, \ldots, X_{rw}^r, \ldots, X_{rw}^M$. Then, we average $|M|$ node embeddings as:

$$X_{rw} = \frac{1}{|M|} \cdot \left(X_{rw}^1 + X_{rw}^2 + \cdots + X_{rw}^r + \cdots + X_{rw}^M\right) \tag{1}$$

The X_{rw} is considered as the pre-trained node feature which contains the distance neighboring node information. Then the X_{rw} is concatenated with the original node attributes X (i.e., $X_{new} = X + X_{rw}, X_{new} \in R^{n\times(F+d_{rw})}$). X_{new}, considered as the new node feature, is inputted into the GCN model.

3.3 Dimension Specific Node Embedding With GCN

Graph convolutional neural networks provide a powerful solution for generating node representations for a given graph [5], which naturally incorporate node attributes. In this section, we utilize multi-layer GCN to learn the dimension specific node embedding. For a given input graph (A, X), the layer-wise propagation rule can be expressed as [5]:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

where $\tilde{A} = A + I_n$ is an adjacency matrix with added self-connections, I_n is the identity matrix, $X \in R^{n\times F}$ is a feature matrix for the input graph, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix for $\tilde{A}$, $W^{(l)}$ is the trainable weight parameters matrix for the lth layer in the GCN, $\sigma(\cdot)$ denotes an activation function (i.e., $\mathrm{Re}LU(\bullet) = \max(0, \bullet)$), and $H^{(l)} \in R^{n\times D}$ is the activation matrix for the lth layer, initialized as $H^{(0)} = X$. In this way, the resulting node embeddings that capture node attributes X and the graph structure A simultaneously [5].

For a dimension specific network G^r, the representation learning model can be denoted as:

$$H_r^{(l+1)} = \sigma\left(\tilde{D}_r^{-\frac{1}{2}}\tilde{A}_r\tilde{D}_r^{-\frac{1}{2}}H_r^{(l)}W_r^{(l)}\right) \tag{3}$$

where $A^r \in R^{|V|\times|V|}$ is an adjacency matrix for the graph G^r and D^r is the corresponding diagonal degree matrix. Unlike in conventional GCN, we modify $\tilde{A}_r$ and define it as $\tilde{A}_r = A_r + wI_n$ Park and Kim [24]. In this expression, $w \in R$ is the weight of self-connections used to measure the relative importance between objective nodes and its neighboring nodes in generating objective node embeddings. A value of $w > 1$ implies the objective node itself is more important than its neighboring nodes, with increasing values of w representing higher importance. The term $\tilde{D}_{ii}^r = \sum_j \tilde{A}_{ij}^r$ represents the degree matrix for $\tilde{A}_r$. In this paper, a two-layer GCN was used to learn dimension specific node embeddings, and the node feature of the input graph is X_{new}, i.e. $H^{(0)} = X_{new}$. The last layer output embedding matrix was denoted H_r, which describes a dimension specific node embedding for the graph G^r. The resulting node embeddings H_r that capture node original attributes X, distance node neighboring information X_{rw} and the graph structure A simultaneously [5].

3.4 Cross Dimension Modeling

For a given dimension $r \in M$, we can obtain the node representation $H_{r\in M} \in R^{n\times d}$, which provides distance

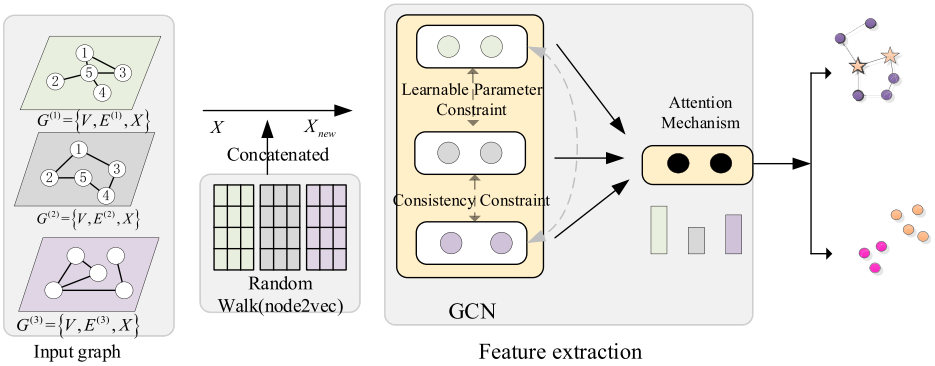


FIGURE 2 | The framework for our proposed method of attributed multiplex network embedding.

TABLE 2 | Dataset statistics.

Dataset	Nodes	Links	Features	Relationships
Citeseer	3,312	21,462	3,703	2
Cora	2,708	19,023	1,433	2
Lazega	71	2,571	71	3
ACM	3,025	2,240 042	1,870	2
DBLP	4,057	11,783 886	334	3
IMDB	4,780	80,216	2,000	2
Amazon	7,621	1,384 799	2,000	3

information for G^r. Each $H_{r\in M}$ is acquired independently by training a two-layer GCN model, as described by **Eq. 3**. However, these embedding matrices fail to take advantage of interactions and similarities between diverse dimensions. This inspired us to devise a way of jointly learning embedding information from diverse dimensional networks to develop a more comprehensive node representation. This task was accomplished by adding two regularization constraint mechanisms to the objective function representing the consistency constraint among the $|M|$ network embedding matrices $H_{r\in M}$, and the trainable weight parameter constraint among $|M|$ GCNs. These two constraints are discussed below.

Regularized consistency constraints: We first applied the normalization to each node embedding matrix to obtain the normalized matrices (i.e. transforming the H_r to H_{r-nor}). The normalized matrices were then exploited to collect similarity information between each pair of counterpart nodes by $H_{r-nor} \cdot H_{r-nor}^T$, and the resulting is the similarity of n nodes. The regularized consistency constraint can be defined as follows:

$$L_{CC} = \|H_{r-nor} \cdot H_{r-nor}^T - H_{r'-nor} \cdot H_{r'-nor}^T\|_2^2, r \in M, r' \in M, r \neq r' \tag{4}$$

where $\cdot^T$ represents the transpose. This constraint can adaptively capture node similarity information among diverse dimensional graph networks when training the proposed model oriented by the node classification task.

Regularized trainable weight parameter constraints among $|M|$ GCNs: As in Xie et al. [26], we utilize regularized constraints for trainable weight parameters among $|M|$ GCNs models. This constraint can be defined as follows:

$$L_{WP} = \sum_{r=1}^{M} \sum_{r'=1, r' \neq r}^{M} \|W_r - W_{r'}\|_2^2 \tag{5}$$

where W_r and $W_{r'}$ are trainable GCN weight matrices for the relation type r and r'.

3.5 Attention Mechanisms for Fusing Different Dimensions

Now attention mechanisms were used to learn corresponding importance weights from different dimensional graph network during each step of model iteration. When optimization of the proposed model ceases, the learned weights represent the importance of diverse dimensional graph networks. The learned dimensional attention matrices for the n nodes were used to embed $H_{r\in M}$ into the final global consensus node representation as follows:

$$Z_{global} = \sum_{r=1}^{M} \alpha_r H_r, \quad \alpha_r \in R^{n\times n}, H_r \in R^{n\times d} \tag{6}$$

where α_r is the learned importance of n nodes for the r dimension network.

Now, the node v_i can be used as an example to illustrate how importance values can be acquired for the node v_i and how attention matrices α_r were acquired for the relation type r. For each $r \in M$, the embedding of node v_i in H_r is given by the row vector $h_r^i \in R^{1\times d}$. We first transform h_r^i *via* a nonlinear transformation (i.e. the $f(x)$ function in **Eq. 7**) and then apply a shared attention vector $p \in R^{h\prime\times 1}$, designed to determine the weight ε_r^i:

$$\varepsilon_r^i = p^T \cdot f\left(W \cdot \left(h_r^i\right)^T + b\right) \tag{7}$$

Here, $W \in R^{h\prime\times d}$ is a weight matrix, $b \in R^{h\prime\times 1}$ is a bias vector, and $f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$ is the tanh function (an activation function). The *Softmax* function can then be used to normalize different attention values in diverse dimensional graph networks. The final weight for node v_i can be calculated as:

$$a_r^i = \frac{\exp\left(\varepsilon_r^i\right)}{\sum_{r=1}^{M} \exp\left(\varepsilon_r^i\right)} \tag{8}$$

Larger values of a_r^i imply that corresponding embeddings are more important. For n nodes in the r dimension network, we can first obtain a learned weight column vector $a_r = [a_r^i], a_r \in R^{n\times 1}$ and then transform the column vector a_r into a $\alpha_r = diag(a_r), \alpha_r \in R^{n\times n}$ diagonal matrix.

3.6 Optimization Objective

Node Classification: The output embedding matrix in **Eq. 6** was used for node classification tasks in combination with a linear transformation and a *Softmax* function. Prediction results were then calculated as follows:

$$\hat{Y} = softmax\left(W_{nc} \cdot Z_{global} + b_{nc}\right) \tag{9}$$

where $W_{nc} \in R^{n\times n}$ is a weight matrix, $b_{nc} \in R^{n\times C}$ is a bias vector,$\hat{Y} = [\hat{y}_{ic}] \in R^{n\times C}$ is the predicted result for n nodes, and $\hat{y}_{ic}$ represents the predicted probability of node v_i belonging to class c. The $softmax(x) = \frac{\exp(x)}{\sum_{c=1}^{C} \exp(x_c)}$ function is a normalizer across all classes. The cross-entropy loss was then minimized over all training nodes using a loss function defined as follows:

$$L_{nc} = \sum_{v_i \in S} \sum_{i=1}^{C} \left[-y_i \ln \hat{y}_i\right] \tag{10}$$

where S represents the training set, y_i is the real label for node v_i, and $\hat{y}_i$ is the predicted label.

Overall Objective Function: The consistency constraint in **Eq. 4** was jointly optimized, along with the trainable weight parameter constraints in **Eq. 5** and the node classification function in **Eq. 10**. An overall objective function was given by:

$$\min_{\theta} L = \min_{\theta} \left(L_{nc} + \alpha L_{CC} + \beta L_{WP} \right) \tag{11}$$

Here, α and β are hyper-parameters used to control the importance of consistency constraints and trainable weight parameter constraint terms. Labeled data were then used to guide the learnable parameters $\theta = \{W_r, \alpha_r, W, b, W_{nc}, b_{nc}\}, r = 1, 2, \ldots, M$, which were automatically learned and updated via gradient descent and back-propagation algorithms. Convergence of the overall objective function L could then be used to obtain a global node representation Z_{global}. This process is summarized in Algorithm 1.

Algorithm 1. The proposed technique.

Require: The input graph $G = \{V, E^{(1)}, E^{(2)}, \cdots, E^{(M)}, X_{new}\}$, the embedding dimension d ,parameters α, β, w.
Ensure: The global embedding matrix Z_{global}
1: Random initialize the learnable parameters: $\theta = \{W_r, \alpha_r, W, b, W_{nc}, b_{nc}\}, r = 1, 2, \cdots, M$.
2: **while** the loss function (L) in Eq.(11) has not converged, **do**
3: Compute each dimension node embedding matrix $H_{r \in M}$ using Eq.(3).
4: Corss dimension modeling using Eqs.(4) and (5).
5: Compute the attention coefficient $\alpha_r = diag(a_r), \alpha_r \in R^{n \times n}$ using Eq.(7) and Eq.(8).
6: Fusing $|M|$ node embeddings matrix for all nodes using Eq.(6) according to the attention mechanism, obtaining the Z_{global}.
7: Calculate the partial derivative $\frac{\partial L}{\partial \theta}$ and optimize the parameter θ using a back-propagation algorithm.
8: **end while**

For a given multiplex graph network $G = \{V, E^{(1)}, E^{(2)}, \ldots, E^{(M)}, X\}$ with M relationships, we first use the node2vec random walk technique to capture the 10-hop neighnoring node information, and then **Eq. 1** is utilized to obtain the pre-trained node feature X_{rw} concatenating with the original node attributes X to get X_{new}. The proposed multiplex graph convolutional network is used to encode all nodes of the multiplex graph network into vectors. We first set the hyper-parameters including the embedding dimension d, parameters α, β, w. When training the proposed model oriented with the node classification task, the learnable parameters are first random initialized, and optimized with the back-propagation algorithm. While the loss function in **Eq. 11** has not converged, the proposed model will computer each dimension (relationship) node embedding with **Eq. 3**, cross dimension interactions with **Eq 4** and **Eq. 5** and attention coefficient using **Eq. 7** and **Eq. 8**. The global node embedding Z_{global} for all nodes from the multiplex relationships is generated according to the obtained attention coefficient.

3.7 Time Complexity

Our proposed technique is primarily composed of four components: 1) the pre-training of node features using a random walk network embedding method, 2) dimension specific node embedding using a GCN model, 3) cross dimension modeling terms, and 4) an attention-based mechanism used to generate global node representations by integrating embeddings of different dimensions. The time complexity of random walk can be expressed as $O(|V|)$. Dimension specific embeddings $O(L|E|d' + L|V|(F + d_{rw})d')$ could then be learned using a GCN model, where $F + d_{rw}$ is the dimension of input node features and d' is the output dimension of one convolution layer. The term L represents the number of GCN layers (2 in this study). The time complexity required for learning global node representations is given by $O(|V|d'|M|)$, where $|M|$ is the number of relation types. Updating attention weights for diverse dimensions in the process of model training, the time complexity can be expressed as $O(|S|d'|M|)$, where $|S|$ is the number of training data. In practice, this quantity of training data is typically small, such that $|S| \ll |E|$. In most practical networks, $|V| \ll |E|$. As such, the total time complexity of node classification tasks can be simplified as $O(|V|(F + d_{rw})d' + |E|d')$.

4 EXPERIMENTS

4.1 Experimental Setup

We construct our experiment on the popular Pytorch framework (https://pytorch.org). All the experiments are performed on a computer with 2.6 GHz 4-core Intel Core i9 processor and the GPU is RTX2080.

Datasets: The proposed method was evaluated on several real-world datasets, as described in **Table 2**. Lazega is a dense network, while the other datasets are sparse networks.

- Citeseer [5]:Citeseer is a citation network consisting of 3,312 research papers, where nodes are publications divided into six different research areas [5]. Node features are bag-of-words representations for individual papers. We can construct the multiplex graph network including two dimensional: a citation dimensional network (where edges represent citation links between papers) and a paper similarity dimensional network. It is a k-nearest neighbor (kNN) graph constructed by calculating the cosine similarity based on the node features and edges representing the top 10 similar papers (i.e., k is 10).
- Cora [5]: Cora is a citation network containing 2,708 machine learning papers divided into seven classes [5]. Node features are bag-of-words representations of individual papers. We can utilize the same approach as for the Citeseer dataset to construct a multiplex network with two dimensions (i.e., citation and node similarity dimensions).
- Lazega [38]: Lazega is a multiplex social network with three relationship types (i.e., strong coworker, advice, and friendship networks) among 71 attorneys (partners and associates) at a law firm. The law school was selected for node label classification.
- ACM [24,28]: It is a multiplex network about the paper-paper relationships consisting of two views which are the two papers are written by same author and two papers contain same subjects respectively. The features of nodes are the elements of a bag-of-words represented of keywords. The nodes are divided into three classes.
- DBLP [24,28]: The dataset is made up of three views about the authors-authors re-pationships, which is another multiplex network from the DBLP. The three views are the two authors have worked together on papers, two authors have published papers with the same terms, and two papers have published papers with the same terms. The classes of the nodes represent the DM(KDD,WSDM,

ICDM), AI(ICML, AAAI,IJCAI), CV(CVPR),NLP (ACL,NAACL, EMNLP), which are the authors' research areas.

- IMDB [24,28]: This is a movie network from the IMDB dataset. In this paper, the IMDB dataset is made up of two relationships (i.e. movies are acted by the same actor and movies are directed by the same director). The features of nodes are the bag-of-words represented of plots. The nodes are divided according to the movies' genre.
- Amazon [24,28]: This dataset is a multiplex network including three views (i.e. also-viewed, also-bought, and bought-together) between items. The items are divided into four different categories (i.e. Beauty, Automotive, Patio Lawn and Garden). The features of the items are the description of items.

Baseline: AMRG was compared with the following competitive baselines.

- DeepWalk [13]: DeepWalk is designed for standard single-layer network embedding without considering node attributes [13]. This approach first utilizes random walk in the networks and then applies a skip-gram algorithm to learn node representations.
- node2vec [10]: On the top of the DeepWalk, node2vec adds two parameters to control the random walk process, forming a biased random walk [10].
- NetMF [31]: It is a general framework that unifies DeepWalk, LINE, node2vec, and PTE by converting a negative sampling into a matrix factorization method for learning network representations.
- MGAT [26]: MGAT is a multiplex network embedding with the Graph Attention Networks (GAT) model.

MGAT was implemented using Pytorch, though the source code is not provided here. The source code published by the authors was utilized for all other baselines.

Parameter Settings: The output node embedding dimension for all datasets was set to 32 to provide a fair comparison. We carefully turn parameters of our proposed model to get optimal performance. For our method, a two-layer GCN was trained with hidden layer dimensions of 64, 128, 256,768, 512, 1024 and output dimensions of 32. The objective function in **Eq. 11** was minimized for the training set using a learning rate of 0.000 95–0.005 and the Adam optimizer. A dropout rate of 0.5 was used in addition to weight decay values of 0.000 three for Cora, 0.002 for Citeseer, 0.000 seven for Lazega, 0.000 five for DBLP and ACM, 0.000 nine for Amazon and 0.03 for IMDB. Consistency constraint coefficients and trainable weight parameters (α and β) were searched in the intervals $\{0.001, 0.05, 0.2, 0.9, 1.0, 0.1\}$ and $\{0.04, 0.05, 0.1, 0.5, 0.6, 0.7, 1.0\}$, respectively. The self-connection weight was set to 2.0 for Citeseer and Amazon datasets, 3.0 for ACM and IMDB datasets and to 1.0 for Cora, Lazega and DBLP datasets. The Lazega dataset contains node relationship types but not node features. As such, a unit diagonal matrix was used as the feature matrix. The node2vec random walk node embedding dimension (d_{rw}) for learning distance node information was set to 8. These values of parameters are summarized in **Table 3**.

TABLE 3 | The values of parameters.

Parameters	Values
d	32
learning rate	0.000 95–0.005
hidden layer dimensions	{64,128,256,768,512,1024}
dropout rate	0.5
weight decay (Cora)	0.000 3
weight decay (Citeseer)	0.002
weight decay (Lazega)	0.000 7
weight decay (DBLP,ACM)	0.000 5
weight decay (Amazon)	0.000 9
weight decay (IMDB)	0.03
α	{0.001,0.05,0.2,0.9,1.0,0.1}
β	{0.04,0.05,0.1,0.5,0.6,0.7,1.0}
w (Citeseer, Amazon)	2.0
w (ACM,IMDB)	3.0
w (Cora, Lazega,DBLP)	1.0
d_{rw}	8
p, q	{1,2,0.5}

For baselines, DeepWalk is a special case of node2vec with $p = q = 1$. In the conventional node2vec algorithm, hyperparameters were set to $p = 2$ and $q = 0.5$, with a window size of 10 and five for negative samples. Other baseline hyperparameters were set as in the original papers.

4.2 Node Classification

Accuracy (ACC) was used to evaluate node classification performance for the all datasets. We randomly selected 10% of the nodes to establish a training set, 10% to form the validation set, and the remaining 80% formed the test set. A total of 120 epochs were used for each of the three datasets. The proposed approach was compared with its variants and state-of-the-art baselines to monitor node classification performance for the following scenarios.

- AMRG/rw: A random walk strategy was not used to learn distance neighboring node information.
- AMRG/att: Our proposed method without the attention mechanism.
- AMRG/cc: Our proposed method without the regularized consistency constraint.
- AMRG/wp: Our proposed method without the regularized trainable weight parameters constraints among $|M|$ GCNs.

The baselines of Node2vec and NetMF were designed for standard single-layer networks. As such, we first acquired node embeddings for each dimension separately and averaged the resulting embeddings together to produce the overall node embeddings. This process was repeated 5 times to produce averaged results for baselines. For our proposed method, as the node2vec random walk technique to capture the 10-hop distance node information and the GCN model will produce a slightly different outcome each time, so the performance of the

TABLE 4 | Node classification Accuracy (%) (bold: Best).

Dataset	Lazega	Cora	Citeseer	ACM	DBLP	IMDB	Amazon
DeepWalk	91.02	66.12	57.25	73.16	52.01	53.52	63.33
Node2vec	90.25	67.92	58.01	73.10	53.91	52.23	65.12
NetMF	92.05	72.95	57.15	74.65	54.98	57.64	-
MGAT	96.32	85.49	72.92	92.26	84.75	66.87	51.30
AMRG	96.49	86.25	74.31	94.05	91.44	70.28	78.12
AMRG/rw	96.33	85.56	74.27	93.18	88.79	69.05	76.72
AMRG/att	94.74	85.69	73.93	93.72	93.04	70.00	76.14
AMRG/cc	92.23	85.74	73.89	93.43	91.37	69.86	78.09
AMRG/wp	92.98	85.60	72.58	93.06	87.06	68.87	77.58

node classification for our proposed model will be evaluated 5 times, and average 5 times results.

Node classification accuracy (ACC) values are reported in **Table 4**, where the bolded number denotes the best result. The following are evident from the table:

- Compared with all other baselines, our proposed method consistently achieved the best performance for all datasets. These results demonstrate the effectiveness of our model for attributed multiplex network embedding, oriented by node classification tasks. From **Table 4**, we observe the random walk technique improved the accuracy of node classification by learning 10-hop node information. This suggests that node information from larger receptive fields is important.
- AMRG/att generates node embeddings using a two-layer GCN for each dimensional network and simply averages the embeddings without considering the attention mechanism. **Table 4** suggests that attention-based weighting in each dimension can boost overall performance. This result is consistent with our assumption that node importance differs in each dimension. However, DBLP can obtain better performance on AMRG/att, it implies that three dimensional graph networks are almost equal degree of importance.
- The coefficients (α and β) for AMRG/cc and AMRG/wp were set to zero respectively in **Eq. 11**. The results suggest these two methods are inferior to AMRG. This indicates that consistency and weight parameters constraints are important for improving the overall performance, demonstrating the need to capture similarities and interactions among diverse dimensions.
- In addition, our technique improved on the classification performance of MGAT (the best performing baseline method except for Amazon dataset, the MGAT is not suitable for Amazon) by as much as 6.69% for the DBLP dataset. This performance was only slightly better than MGAT for the Lazega data. This is likely because Lazega is a small, dense multiplex network with only 71 nodes. As such, node information spreads quickly in graphs through the two-layer GCN. The accuracy of node classification for the Cora all dataset was worse than that of other variants without pre-trained node features, which again indicates the importance of node information from larger receptive fields. However, as NetMF method uses the matrix factorization, the model does not convergence for Amazon, which explains the shortcomings of the NetMF.

TABLE 5 | Node classification F1-score (%) (bold: Best).

Dataset	Lazega	Cora	Citeseer	ACM	DBLP	IMDB	Amazon
DeepWalk	89.32	64.78	56.52	72.67	51.99	51.54	61.79
Node2vec	89.89	66.01	57.56	72.76	53.35	51.33	64.31
NetMF	92.01	75.98	61.41	74.65	54.11	55.21	-
MGAT	96.31	84.06	69.43	91.01	84.09	65.67	51.02
AMRG	96.32	84.08	71.32	92.90	90.51	70.24	72.85

- The difference between our method and MGAT is that we used consistency constraints to embed each dimension network with pre-trained 10-hop node information. **Table 4** indicates the overall performance of our method is superior to MGAT, which suggests that consistency constraints and pre-trained methods with random walk are important mechanisms for learning high-quality node embeddings for multiplex network. Furthermore, compared with MGAT, our approach achieved superior node embeddings through two-layer general and simple GCNs with fewer model parameters than MGAT, which is based on the GAT model with multiple attention heads that lead to rapid growth in the number of parameters Veličković and Cucurull [15].

Similarly, node classification F1-score values also are presented in **Table 5**, where the bolded number denotes the best result. These results demonstrate that our method is stable and competitive.

4.3 Analysis of Attention Mechanisms

Now we use the Lazega,a dense network, and three sparse networks (Cora, IMDB, and DBLP) as examples to were used to analyze changing trends in attention values for node classification tasks. The results are shown in **Figure 3**, where the x-axis denotes different numbers of epochs during the model training process and the y-axis is the corresponding attention value. As seen in **Figure 3A**, the attention values for advice, friendship, and strong coworker dimensional networks are nearly the same for the Lazega dataset in epoch 0. However, this attention value varies with increasing training epochs. The attention value for the advice dimensional network increases in epoch 20 (compared to epoch 0), while the attention value for the strong coworkers dimensional network decreases.

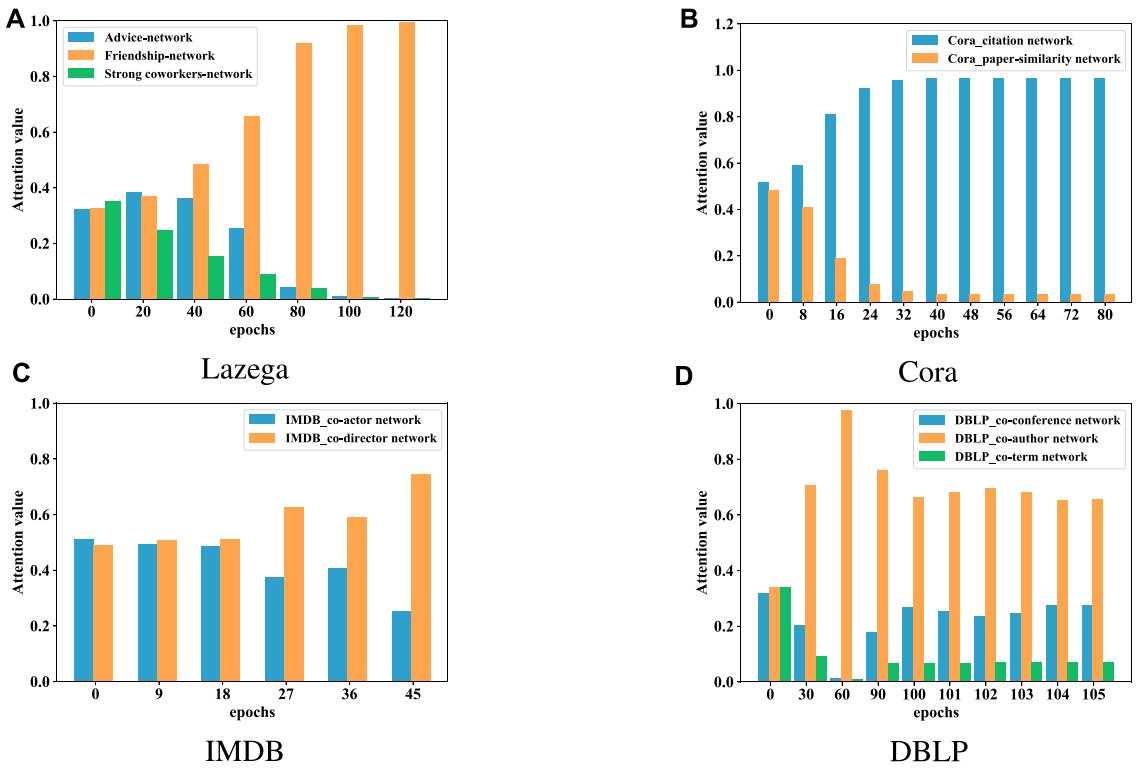


FIGURE 3 | (A) Lazega. **(B)** Cora. **(C)** IMDB. **(D)** DBLP.

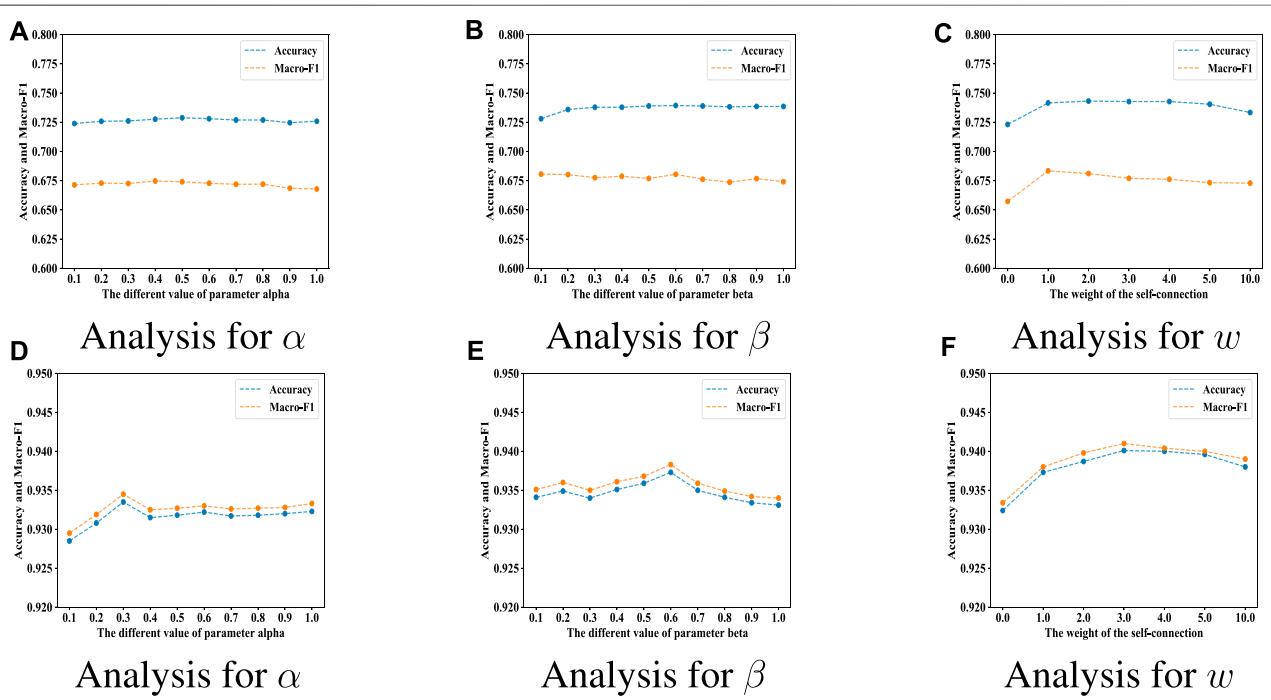


FIGURE 4 | The performance of our method for parameters α, β, w on the Citeseer **(A–C)** and ACM **(D–F)** datasets for node classification task.

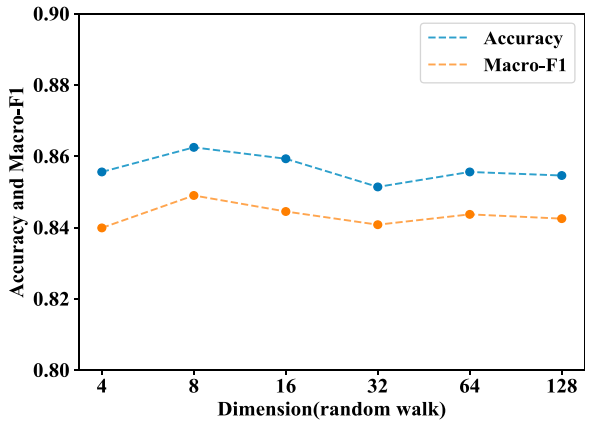


FIGURE 5 | The performance of our method for parameters d_{rw} on the Cora dataset for node classification task.

Similarly, the attention values for the advice and strong coworker dimensional networks gradually decrease until the model converges. However, the attention value continues increasing for the friendship dimensional network during the training process. This illustrates that our approach can adaptively learn the importance weights of diverse relation type network embeddings during individual steps. For example, information provided by the friendship dimension network was more important than that of the advice and strong coworker dimension for the Lazega dataset. Similar trends were observed for the Cora, IMDB and DBLP datasets. The citation dimension network also proved to be more important than the paper similarity dimension network in the overall system, as shown in **Figure 3B**.

For the IMDB dataset, when the epoch is 18, the accuracy of node classification is the best (i.e. 70.28). At this moment, the attention value for co-actor network and co-director network is almost equal. From **Table 4**, we can see that when the attention value for co-actor network and co-director network is equal (i.e.AMRG/att, attention values are 0.5 and 0.5 respectively), the accuracy is 70.00, which is in close proximity to 70.28. This demonstrate that the attention mechanism is important. For DBLP, when the epoch is 103, the accuracy of node classification is the best.

4.4 Analysis of Variants

In this section, we analyze the effectiveness and sensitivity of pre-trained node vector dimensions, two regularization constraints, and the weight of self-connections. Specifically, we evaluate the performance (i.e., accuracy and macro-F1) of our method for node classification tasks with respect to α,β,w and d_{rw}. Since these datasets produced similar results, we use the Citeseer, ACM and Cora datasets as examples respectively.

Analysis for α: In this test, the value of β was set to zero, which eliminates L_{WP} in **Eq. 11**. The parameter α was varied from 0.1 to 1.0, as shown in **Figure 4A**. The accuracy and macro-F1 score for these node classification tasks remained relatively stable with increasing α for Citeseer. And when the α is 0.3, the accuracy and macro-F1 score is the best for ACM in **Figure 4D**.

Analysis for β: Here, the value of α was set to zero, which eliminates the consistency constraint L_{CC} in **Eq. 11**. **Figure 4B** demonstrates how different values of the coefficient β affected node classification performance. The value of β was increased from 0.1 to 1.0, as the accuracy slowly increased and decreased (the maximum was in 0.7), and the macro-F1 score remained nearly constant. Similarity trends can be found in **Figure 4E** and the maximum was β in 0.6.

Analysis for w: The impact of the weight on these self-connections was observed by varying w from 0.0 to 10.0, as shown in **Figure 4C** for Citeseer. A value of $w = 0.0$ implies the node itself only considers neighboring nodes in generating embeddings, producing low accuracy and macro-F1 scores. Larger values ($w > 1$) indicate the node itself is more important than its neighboring nodes. The accuracy first increases and then remains relatively stable, before dropping quickly for $w = 10.0$. The maximum accuracy and macro-F1 scores occurred for $w = 2$ and $w = 1$, respectively, with macro-F1

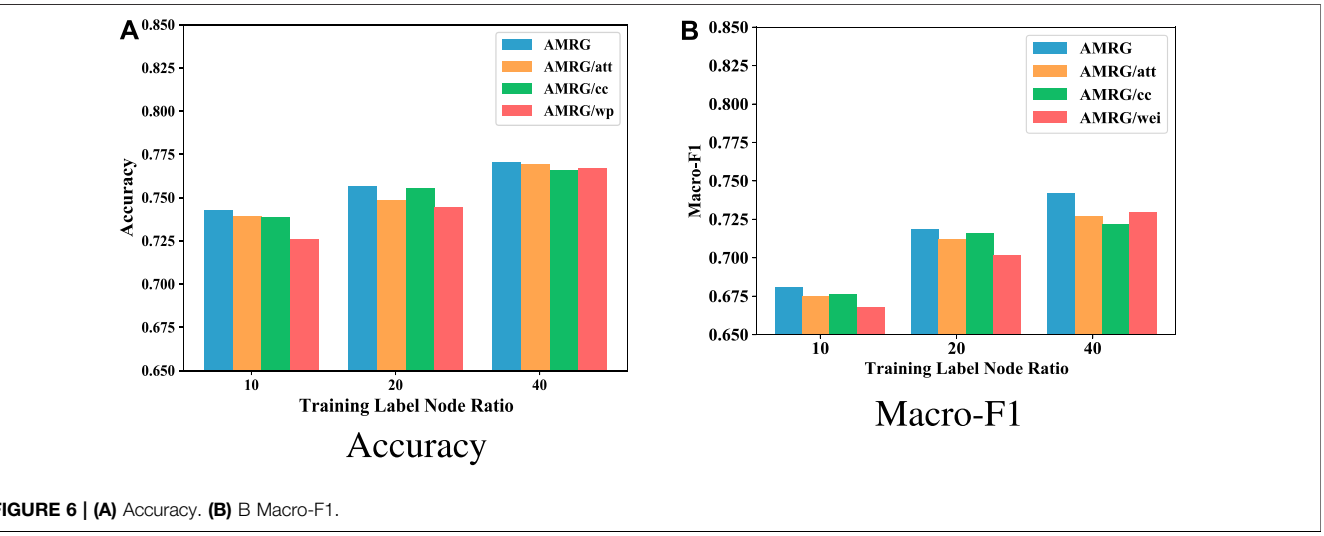


FIGURE 6 | (A) Accuracy. **(B)** B Macro-F1.

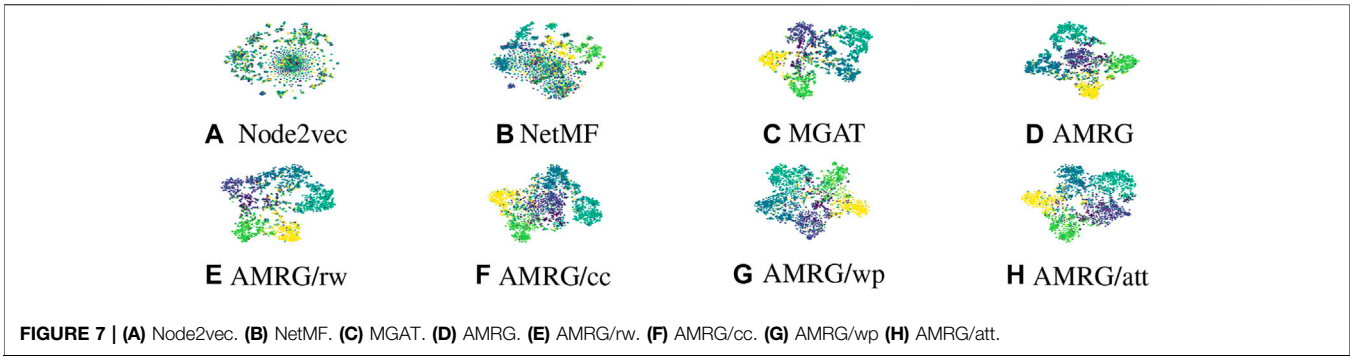


FIGURE 7 | (A) Node2vec. **(B)** NetMF. **(C)** MGAT. **(D)** AMRG. **(E)** AMRG/rw. **(F)** AMRG/cc. **(G)** AMRG/wp **(H)** AMRG/att.

decreasing slowly as w increased. For ACM in **Figure 4F**, the best accuracy and macro-F1 occurred for $w = 3$.

Analysis of random walk dimension d_{rw}: **Figure 5** suggests the accuracy of node classification was maximized for d_{rw} in 8, decreasing for higher dimensions. This demonstrates that pre-trained node features, which contains distance node neighboring information, can improve model performance. When the embedding dimension d_{rw} is small, the obtained pre-trained node features are not enough to describe the distance neighborhood information. On the contrary, if the value of embedding dimension d_{rw} is large, it will introduce some noise on the obtained pre-trained node features.

Analysis of different training nodes: We tested the performance of our method and its variants using training node fractions of 10, 20, and 40% for the attributed multiplex Citeeer network, as shown in **Figure 6**. The proposed method consistently outperformed its variants, indicating the importance and effectiveness of attention mechanisms, consistency constraints, and trainable weight parameter constraints of $|M|$ GCN for each dimensional graph network for boosting overall performance. The influence of attention mechanisms and these two constraints on node classification performance varied with different training label node ratios.

4.5 Analysis of Key Factor

In the previous section, we analyse the effect of the different single variant on the performance of the proposed model. At present, we take the Citeseer, ACM and Amazon as examples to evaluate the key factor of regularized consistency constraints, weight constraints and attention mechanism. That is to say, which one of three factors is the most significant to improve the AMRG. The evaluation indicator is the node classification accuracy.

- AMRG(wp): The AMRG only with the weight constraints.
- AMRG(cc): The AMRG only with the regularized consistency constraints.
- AMRG(att): The AMRG only with the attention mechanism.

Table 6 shows that these three strategies (i.e. regularized consistency constraints, weight constraints and attention mechanism) have different importance on the three datasets. For Citeseer, it shows that the weight constraints is the key factor, as the performance of AMRG (wp) is better than AMRG (att) and AMRG (cc). From the results of ACM and Amazon, we can see that the regularized consistency constraints is the key factor for ACM and the attention mechanism is the key factor for Amazon. As the performance of AMRG (cc) is better and AMRG (att) is better than the other variants.

TABLE 6 | Node classification accuracy (%) (Bold: Best; underline: Runner-up).

	Citeseer	ACM	Amazon
AMRG	74.31	94.05	78.12
AMRG (att)	71.90	93.26	77.23
AMRG (cc)	72.92	93.72	76.40
AMRG (wp)	73.70	93.22	75.53

4.6 Visualization

Task visualization will be performed to provide a more intuitive comparison and to further show the effectiveness of our proposed method. As the results on different datasets will exhibit similar trends, we take the Citeseer dataset as an example to evaluate our method. Output embeddings in the last layer, prior to the *Softmax* operation, were utilized for node classification tasks and to plot the resulting node embeddings using t-SNE Maaten and van der Hinton [40]. Results for the Citeseer dataset are shown in **Figure 7** and are colored using real labels.

It is evident from the figure that the results of the node2vec and NetMF baselines are not satisfactory because nodes with different classes are mixed together. In contrast, MGAT and our method consider node features, producing better results than node2vec and NetMF. This further demonstrates the importance of node features for mining hidden graph information. The three variants are inferior to our method, which is consistent with the results of **Table 4**. Our approach has the clearest distinct boundaries among the diverse classes and the same classes are grouped together. This demonstrates the importance of pre-trained mechanisms with random walk network embedding, consistency constraints, trainable weight parameter constraints among $|M|$ GCN, and attention mechanisms.

5 CONCLUSION

In this paper, a new approach for node classification in attributed multiplex networks was developed using random

walk network embedding and GCN. Random walk network embedding was first used to capture 10-hop node information as pre-trained node features. Then they were concatenated with the original node attributes, the resulting as the new node features inputted into the GCN model. A two-layer GCN was then utilized to learn each dimensional network. In order to achieve more comprehensive, informative, higher-quality and global consensus node representations, we introduced regularized consistency constraints to capture the similarities among different dimensional network embeddings. Besides, trainable weight parameter constraints were used to learn the interactive information from diverse dimensions. Furthermore, an attention mechanism was utilized to learn weights for diverse dimensional networks, and then to fuse them based on the weights. Extensive experiments were conducted using real-world networks applied to node classification, visualization, analysis of attention mechanisms, and parameter sensitivity. The results demonstrated that these strategies of our approach can boost overall performance of node classification for multiplex graph networks, which outperformed many competitive baselines. In the future, we plan to extend this framework to larger, more complex, and time-varying graphs. Another promising direction involves learning node representations combining edge features.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

Conceptualization, BH, YW, and LK; Data curation, BH, QW, and YY; Funding acquisition, YW and LK; Methodology, BH, YW, LK, and QW; Validation, Visualization, BH and YW; Supervision, YW and LK; Writing—Original draft preparation, BH and YW; Writing—Reviewing and Editing, BH, YW, LK, QW, and YY; All authors have read and agreed to the published version of the manuscript.

FUNDING

This paper is support by the National Natural Science Foundation of China (NSFC) under grant number 61 873 274 and Postgraduate Scientific Research Innovation Project of Hunan Province under grant number CX20200075.

REFERENCES

1. Wang Z, Wang J, Guo Y, Gong Z. Zero-shot Node Classification with Decomposed Graph Prototype Network. In: KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. New York, NY, USA: Association for Computing Machinery (2021). p. 1769–79. doi:10.1145/3447548.3467230
2. Zhu Y, Xu Y, Yu F, Liu Q, Wu S, Wang L. Graph Contrastive Learning with Adaptive Augmentation. In: WWW '21: Proceedings of the Web Conference 2021. New York, NY, USA: Association for Computing Machinery (2021). p. 2069–80. doi:10.1145/3442381.3449802
3. Tajeuna EG, Bouguessa M, Wang S. Modeling and Predicting Community Structure Changes in Time-Evolving Social Networks. *IEEE Trans Knowl Data Eng* (2019) 31:1166–80. doi:10.1109/TKDE.2018.2851586
4. Sun Y, Yu Y, Han J. Ranking-based Clustering of Heterogeneous Information Networks with star Network Schema. In: KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: Association for Computing Machinery (2009). p. 797–806. doi:10.1145/1557019.1557107
5. Kipf TN, Welling M. Semi-supervised Classification with Graph Convolutional Networks. In: International Conference on Learning Representations (ICLR) (2017).
6. Li ZWX-M, Han Q. *Deeper Insights into Graph Convolutional Networks for Semi-supervised Learning*. Menlo Park, California, USA: AAAI (2018).
7. Li G, Muller M, Thabet A, Ghanem B. Deepgcns: Can Gcns Go as Deep as Cnns?. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, South Korea, October 27–November 3, 2019 (2019). p. 9266–75. doi:10.1109/ICCV.2019.00936
8. Wang M, Lin Y, Lin G, Yang K, Wu X-m. M2grl: A Multi-Task Multi-View Graph Representation Learning Framework for Web-Scale Recommender Systems. In: KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: Association for Computing Machinery (2020). p. 2349–58. doi:10.1145/3394486.3403284
9. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, August 19 - 23, 2018. New York, NY, USA: Association for Computing Machinery (2018). 974–83. doi:10.1145/3219819.3219890
10. Grover A, Leskovec J.. node2vec: Scalable Feature Learning for Networks. *KDD* (2016) 2016:855–64. doi:10.1145/2939672.2939754
11. Zhang H, Qiu L, Yi L, Song Y. Scalable Multiplex Network Embedding. In: IJCAI'18: Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, July 13-19, 2018. AAAI Press (2018). p. 3082–8. doi:10.24963/ijcai.2018/428
12. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. Line. In: Proceedings of the 24th International Conference on World Wide Web (WWW'15), Montréal, Québec, Canada, April 11 - 15, 2016 (2015). p. 1067–77. doi:10.1145/2736277.2741093
13. Perozzi B, Al-Rfou R, Skiena S. DeepWalk. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, August 24 - 27, 2014. New York, NY: Association for Computing Machinery (2014). p. 701–10. doi:10.1145/2623330.2623732
14. Hamilton WL, Ying R, Leskovec J. *Inductive Representation Learning on Large Graphs*. Granada, Spain: NIPS (2017).
15. Veličković GCARALPBY, Cucurull P. *Graph Attention Networks*. Vienna: ICLR (2018).
16. Li C, Wang L, Sun S, Xia C. Identification of Influential Spreaders Based on Classified Neighbors in Real-World Complex Networks. *Appl Maths Comput* (2018) 320:512–23. doi:10.1016/j.amc.2017.10.001
17. Nicosia V, Latora V. Measuring and Modeling Correlations in Multiplex Networks. *Phys Rev E* (2015) 92:032805. doi:10.1103/PhysRevE.92.032805
18. Wang Z, Guo Q, Sun S, Xia C. The Impact of Awareness Diffusion on Sir-like Epidemics in Multiplex Networks. *Appl Maths Comput* (2019) 349:134–47. doi:10.1016/j.amc.2018.12.045
19. Liu W, Chen P-Y, Yeung S, Suzumura T, Chen L. Principled Multilayer Network Embedding. *Principled multilayer Netw embedding* (2017) 2017: 134–41. doi:10.1109/ICDMW.2017.23
20. Murata R, Matsuno T. Mell: Effective Embedding Method for Multiplex Networks. In: WWW '18: Companion Proceedings of the The Web Conference 2018, Lyon, France, April 23–27, 2018. New York, NY: Association for Computing Machinery (2018). p. 1261–8. doi:10.1145/3184558.3191565

21. Qu M, Tang J, Shang J, Ren X, Zhang M, Han J. An Attention-Based Collaboration Framework for Multi-View Network Representation Learning. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, November 6–10, 2017. New York, NY: Association for Computing Machinery (2017). p. 1767–76. doi:10.1145/3132847.3133021
22. Ma SACYDTJ, Wang Y. *Multi-dimensional Graph Convolutional Networks* (2018).
23. Ghorbani M, Baghshah MS, Rabiee HR. MGCN: Semi-supervised Classification in Multi-Layer Graphs with Graph Convolutional Networks. In: ASONAM '19: International Conference on Advances in Social Networks Analysis and Mining (2019). p. 208–11. doi:10.1145/3341161.3342942
24. Park DHJYH, Kim C. *Unsupervised Attributed Multiplex Network Embedding*. Menlo Park, California, USA: AAAI (2020).
25. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, et al. Heterogeneous Graph Attention Network. In: The World Wide Web Conference (WWW '19). New York, NY: Association for Computing Machinery (2019). 2022–2032.
26. Xie Y, Gong Y, Tang MZ, Han C. Mgat: Multi-View Graph Attention Networks. *Neural Networks* (2020) 132:180–9. doi:10.1016/j.neunet.2020.08.021
27. Gong M, Liu W, Xie Y, Tang Z, Xu M. Heuristic 3d Interactive Walk for Multilayer Network Embedding. *IEEE Trans Knowl Data Eng* (2020) 2020:1. doi:10.1109/TKDE.2020.3021393
28. Fan S, Wang X, Shi C, Lu E, Lin K, Wang B.. One2multi Graph Autoencoder for Multi-View Graph Clustering. In: Proceedings of The Web Conference 2020 (WWW '20), Taipei, Taiwan, April 20–24, 2020. New York, NY: Association for Computing Machinery (2020). 3070–76. doi:10.1145/3366423.3380079
29. Mikolov ICKCGJ, Sutskever T. *Distributed Representations of Words and Phrases and Their Compositionality*. Granada, Spain: NIPS (2013). p. 3111–9.
30. Wang D, Cui P, Zhu W. Structural Deep Network Embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2016, San Francisco, CA, August 13–17, 2016. New York, NY: Association for Computing Machinery (2016). p. 1225–34. doi:10.1145/2939672.2939753
31. Qiu J, Dong Y, Ma H, Li J, Wang K, Tang J. Network Embedding as Matrix Factorization: Unifying Deepwalk, Line, Pte, and Node2vec. In: WSDM '18: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, February 5–9, 2018. New York, NY: Association for Computing Machinery (2018). p. 459–67. doi:10.1145/3159652.3159706
32. Yang C, Liu Z, Zhao D, Sun M, Chang EY. Network Representation Learning with Rich Text Information. In Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15). Buenos Aires Argentina (2015). July 25 - 31, 2015. AAAI Press, 2111–2117.
33. Zhang D, Yin J, Zhu X, Zhang C. Homophily, Structure, and Content Augmented Network Representation Learning. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, December 12–15, 2016. IEEE (2016). p. 609–18. doi:10.1109/ICDM.2016.0072
34. Bandyopadhyay AKHMM, Biswas S. *A Multilayered Informative Random Walk for Attributed Social Network Embedding*. Santiago de Compostela: ECAI (2020).
35. Zhou GZZYCLZWLLCSM, Cui J. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434 [Cs, Stat] 2019* (2019).
36. Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M. Modeling Relational Data with Graph Convolutional Networks. In: European Semantic Web Conference. Heraklion, Greece: Springer, Cham (2018). p. 593–607. doi:10.1007/978-3-319-93417-4_38
37. Goyal P, Ferrara E. Graph Embedding Techniques, Applications, and Performance: A Survey. *Knowledge-Based Syst* (2018) 151:78–94. doi:10.1016/j.knosys.2018.03.022
38. Grossetti M, Lazega E. The Collegial Phenomenon. The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership. *Revue Française de Sociologie* (2003) 44:186. doi:10.2307/3323128
39. Li J, Chen C, Tong H, Liu H. Multi-layered Network Embedding. In: Proceedings of the 2018 SIAM International Conference on Data Mining (SDM). San Diego: Society for Industrial and Applied Mathematics (2018) p. 684–92. doi:10.1137/1.9781611975321.77
40. Maaten G, van der Hinton LJP. Visualizing High-Dimensional Data Using T-Sne. *J Machine Learn Res* (2008) 9:2579–605.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Advantages of publishing in Frontiers

OPEN ACCESS
Articles are free to read for greatest visibility and readership

FAST PUBLICATION
Around 90 days from submission to decision

HIGH QUALITY PEER-REVIEW
Rigorous, collaborative, and constructive peer-review

TRANSPARENT PEER-REVIEW
Editors and reviewers acknowledged by name on published articles

Frontiers
Avenue du Tribunal-Fédéral 34
1005 Lausanne | Switzerland

Visit us: www.frontiersin.org
Contact us: frontiersin.org/about/contact

REPRODUCIBILITY OF RESEARCH
Support open data and methods to enhance research reproducibility

DIGITAL PUBLISHING
Articles designed for optimal readership across devices

FOLLOW US
@frontiersin

IMPACT METRICS
Advanced article metrics track visibility across digital media

EXTENSIVE PROMOTION
Marketing and promotion of impactful research

LOOP RESEARCH NETWORK
Our network increases your article's readership