ARTIFICIAL INTELLIGENCE IN INSURANCE AND FINANCE

EDITED BY: Glenn Fung, Sou Cheng Choi, Luisa Fernanda Polania Cabrera, Victor Wu and Lawrence Kwan Ho Ma

PUBLISHED IN: Frontiers in Applied Mathematics and Statistics

frontiers Research Topics

ISSN 1664-8714
ISBN 978-2-88971-811-5
DOI 10.3389/978-2-88971-811-5

ARTIFICIAL INTELLIGENCE IN INSURANCE AND FINANCE

Topic Editors:
Glenn Fung, American Family Mutual Insurance Company, Madison, United States
Sou Cheng Choi, Illinois Institute of Technology, United States
Luisa Fernanda Polania Cabrera, Target, United States
Victor Wu, Virginia Commonwealth University, United States
Lawrence Kwan Ho Ma, Hong Kong Blockchain Society, China

Luisa Fernanda Polania Cabrera is an Experienced Professional at Target Corporation (United States). Victor Wu is a Product Manager at GitLab Inc, San Francisco, United States. Sou-Cheng Choi is a Consulting Principle Data Scientist at Allstate Corporation. Lawrence Kwan Ho Ma is the Founder, Director and Chief Scientist of Valigo Limited and Founder, CEO and Chief Scientist of EMALI.IO Limited. Glenn M. Fung is the Chief Research Scientist at American Family Insurance.

Citation: Fung, G., Choi, S. C., Cabrera, L. F. P., Wu, V., Ma, L. K. H., eds. (2021). Artificial Intelligence in Insurance and Finance. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-88971-811-5

Table of Contents

04 ***Editorial: Artificial Intelligence in Insurance and Finance***
Glenn Fung, Luisa F. Polania, Sou-Cheng T. Choi, Victor Wu and Lawrence Ma

07 ***Mileage Extraction From Odometer Pictures for Automating Auto Insurance Processes***
Shailesh Acharya and Glenn Fung

17 ***Task-Optimized Word Embeddings for Text Classification Representations***
Sukrat Gupta, Teja Kanchinadam, Devin Conathan and Glenn Fung

27 ***Deep Learning Methods for Mean Field Control Problems With Delay***
Jean-Pierre Fouque and Zhaoyu Zhang

44 ***Comparing Model Selection Criteria to Distinguish Truncated Operational Risk Models***
Daoping Yu

54 ***Financial Forecasting With α-RNNs: A Time Series Modeling Approach***
Matthew Dixon and Justin London

63 ***An Explainable Bayesian Decision Tree Algorithm***
Giuseppe Nuti, Lluís Antoni Jiménez Rugama and Andreea-Ingrid Cross

72 ***Identifying Actionable Serial Correlations in Financial Markets***
Siew Ann Cheong, Yann Wei Lee, Ying Ying Li, Jia Qing Lim, Jiok Duan Jadie Tan and Xin Ping Joan Teo

86 ***Sentiment-Guided Adversarial Learning for Stock Price Prediction***
Yiwei Zhang, Jinyang Li, Haoran Wang and Sou-Cheng T. Choi

100 ***Auto Defect Detection Using Customer Reviews for Product Recall Insurance Analysis***
Titus Hei Yeung Fong, Shahryar Sarkani and John Fossaceca

114 ***Using Machine Learning to Analyze Merger Activity***
Tiffany Jiang

123 ***AVA: A Financial Service Chatbot Based on Deep Bidirectional Transformers***
Shi Yu, Yuxin Chen and Hussain Zaidi

EDITORIAL
published: 01 December 2021
doi: 10.3389/fams.2021.795207

Editorial: Artificial Intelligence in Insurance and Finance

Glenn Fung[1], *Luisa F. Polania*[2], *Sou-Cheng T. Choi*[3,4]*, *Victor Wu*[5] and *Lawrence Ma*[6]

[1]*American Family Mutual Insurance Company, Madison, WI, United States,* [2]*Palo Alto Networks, Santa Clara, CA, United States,* [3]*Kamakura Corporation, Honolulu, HI, United States,* [4]*Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, United States,* [5]*Ensembl Inc., Chicago, IL, United States,* [6]*Hong Kong Blockchain Society, Hong Kong, China*

Keywords: finance, insurance, risk, algorithm, neural network, transformer, bitcoin, chatbot

Editorial on the Research Topic

Artificial Intelligence in Insurance and Finance

OPEN ACCESS

Edited by:
Young Shin Aaron Kim,
Stony Brook University, United States

Reviewed by:
Matthias Scherer,
Technical University of Munich, Germany

***Correspondence:**
Sou-Cheng T. Choi
schoi32@iit.edu

Specialty section:
This article was submitted to Mathematical Finance, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *14 October 2021*
Accepted: *29 October 2021*
Published: *01 December 2021*

Citation:
Fung G, Polania LF, Choi S-CT, Wu V and Ma L (2021) Editorial: Artificial Intelligence in Insurance and Finance. Front. Appl. Math. Stat. 7:795207. doi: 10.3389/fams.2021.795207

1 INTRODUCTION

Artificial intelligence (AI) has become a phenomenon and caught researchers in almost all domains by surprise with its overwhelming success accompanied by unprecedented accuracies, sometimes even surpassing human experts. The finance and insurance sectors are no exceptions to this AI revolution, especially considering the significant amount of historical data, structured and unstructured, available in most financial and insurance companies. Another motivating factor is the evolving expectation of customers for frictionless and on-demand services, which brings not only challenges but also significant opportunities for applying AI. In the field of insurance, AI promises to reshape claims, underwriting, distribution, and pricing. In the field of finance, AI is having a seismic impact on robo-advisory, fraud prediction, trading strategies, risk assessment, and chatbots, to name a few.

In this context, this research topic brings together 11 papers that have developed new theoretical or applied models employing AI in a variety of financial and insurance problems. We introduce them in the order of the first authors' last names.

2 PAPERS IN THIS RESEARCH TOPIC

Acharya and Fung used state-of-the-art object detection deep learning architectures for extracting vehicle mileage from odometer images taken by mobile devices. Despite availability of commercial solutions for license plates and VIN recognition from images, there are no existing commercial solutions for odometer mileage extraction from images. The authors have also tested empirically the proposed system in unseen odometer images taken in the wild and have achieved satisfactory performance, meeting requirements needed for real-life application in the insurance industry.

Inspired by DNA sequencing and natural language processing, Cheong et al. used historical daily close prices of financial assets and mapped those with positive returns into a sequence, e.g., (AB) represents assets A and B with price increases in a day. By such temporal-spatial sequences from every week, the authors tested against random combinations and dynamically discovered assets whose prices often rise together over consecutive days, thereby informing short-term, likely profitable trading decisions.

In the study of Dixon and London, a novel neural network architecture called the α-recurrent neural networks (α-RNNs) was developed for non-stationary time series forecasting. In a nutshell,

α-RNNs apply exponentially smoothing to the hidden layers of RNNs and are unconditionally stable with potentially infinite memory. Simple statistical tests suffice to configure α-RNNs. Using minute-frequency bitcoin prices and high-frequency futures tick data, the authors demonstrated that α-RNNs achieved similar out-sample accuracies as substantially more complex models including gated recurrent units or long short-term memory models (LSTMs).

Fong et al. developed an innovative insurance application for early, automatic detection of product defects from online customer reviews. The modeling pipeline consists of recurrent neural networks for predicting negative sentiments and the presence of defects; followed by a topic discovery model for clustering negative reviews with defects into similar problems. The application was developed with home products for early intervention and more effective cost management of product defects but is suitable for various artifacts covered by insurance plans.

Fouque and Zhang considered a challenging multiplayer game, in which all players are indistinguishable, and each person seeks to minimize the same objective function of action over time, subject to a stochastic differential equation with delayed effect from an earlier state, given initial conditions. Because there is no closed-form solution, the authors designed two numerical algorithms for estimating an optimal trajectory of controls using neural networks. Last, the authors prove the existence and uniqueness of optimality under certain conditions.

Gupta et al. developed a supervised algorithm that generates task-optimized word embeddings for natural language processing (NLP) applications. Unlike traditional word embeddings that are optimized at the word level, this new algorithm produces embeddings at the sentence-level using a weighted average of an available pre-trained word-level embedding. This allows for more targeted applications of NLP to specific domains and, thus, better performance because the user can inject their weights accordingly. The authors also performed numerical experiments to demonstrate the performance of the algorithm.

Jiang briefly outlined a machine learning approach with LASSO for the challenging problem of forecasting corporate mergers using a large number of annual reports Form10-K filed with the U.S. Securities and Exchange Commission. The unstructured text was preprocessed with NLP techniques and transformed to a high-dimensional term frequency–inverse document frequency matrix. Several potentially fruitful directions are discussed for future research.

Nuti et al. developed a deterministic Bayesian Decision Tree algorithm, applicable to regression and classification problems. It eliminates the need for sampling and pruning. In particular, the algorithm generates a greedy-modal tree (GMT). GMT is salient because models become explainable, an important component and often prerequisite in finance and medicine. The authors tested the new algorithm against various standard benchmarks, demonstrating comparable performance against other existing techniques, showing that accuracy does not have to be sacrificed for explainability.

Bidirectional Encoder Representations from Transformers (BERT) is one of the most advanced AI models for natural languages that have emerged in recent years. Yu et al. leveraged BERT's pre-trained language models to efficiently build a closed-domain chatbot for hierarchical classification of over 380 intents that arise from more than 22,000 questions of financial customers. The article also presents a thorough treatment of out-of-vocabulary words. Finally, model class probabilities are randomly sampled with Monte Carlo methods for computing confidence intervals.

Yu investigated three information criteria (traditional AIC, BIC, and information complexity-based ICOMP) to assess truncated operational risk models. The performances of using the three information criteria to distinguish various fat-tailed distributional models such as Champernowne, Frechet, lognormal, and Weibull distributions were first examined using simulation studies. The author then studied a use case beginning with model fitting and model validation, followed by value-at-risk estimation, and ended up with model selection using various information criteria on the basis of fraud risk data coming from retail banking of Chinese banks.

The article by Zhang et al. focuses on stock price prediction by leveraging sentiment information from tweets as an additional feature. This is achieved by using a conditional generative adversarial network, where the generator is formed by a LSTM network, whereas the discriminator is formed by a multilayer perceptron. The authors showed through experiments that their approach outperformed state-of-the-art methods based on LSTMs and traditional methods, such as linear multiple regression, K-nearest neighbors, and autoregressive integrated moving average.

3 CONCLUSION

In a nutshell, the papers in this research topic illustrate how AI is comprehensively transforming the way financial and insurance businesses operate and interact with their consumers and markets. In particular, deep learning networks are playing an important role in this transformation. For example, many of the articles in this research topic make extensive use of recurrent neural networks to model time series and text data. Similarly, generative adversarial networks and faster region–based convolutional neural networks are also used by some of the articles in this research topic.

It is worth highlighting that the data used in some of the articles go beyond finance and insurance data and extend to social media data, such as tweets. This is motivated by the interplay between social sentiment and financial market movement. It is also worth mentioning that this topic is rich in AI applications to financial and insurance areas, including but not restricted to, financial forecasting, merger activity, financial service chatbots, and risk assessment.

In closing, we would like to express our appreciation to the reviewers for their high-quality feedback and timely responses.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

ACKNOWLEDGMENTS

The authors would like to thank all editors and reviewers for the Special Topic.

Conflict of Interest: GF was employed by American Family Mutual Insurance Company. LP was employed by Palo Alto Networks. S-CC was employed by Kamakura Corporation. VW was employed by Ensembl Inc.

The remaining author declares 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 Applied Mathematics and Statistics

ORIGINAL RESEARCH
published: 10 December 2019
doi: 10.3389/fams.2019.00061

Mileage Extraction From Odometer Pictures for Automating Auto Insurance Processes

Shailesh Acharya and Glenn Fung*

Machine Learning Research and Innovation, American Family Insurance, Madison, WI, United States

For an insurance company is a priority to supply customers with an easy and streamlined way to provide all the information needed when reporting a claim or asking for a quote. A simple and efficient process to do so improves customer experience, reduces human error and accelerates the information collection process. An accurate mileage reading is a key piece of information that is relevant for auto insurance quotes and claims processing. The vehicle mileage can be combined with the License Plate number and the Vehicle Identification Number (VIN) to get a complete overview of the information needed for many insurance processes and workflows. In this paper, we describe a novel solution for extracting vehicle mileage from odometer images taken by mobile devices. There are many available low-cost commercial solutions for both License plate recognition and VIN recognition from images, however, to the best of our knowledge, this is not existing commercial solutions for odometer mileage extraction from images. Our proposed system mainly consists of two parts: (a) identifying the odometer display and, (b) extracting characters inside the display. We leverage existing state-of-the-art object detection deep learning architectures to solve each part and design a post-processing algorithm to identify mileage from the extracted characters. We tested empirically our proposed system in unseen odometer images taken in the wild. We achieve satisfactory performance that meets the requirements needed for real-life applications in the insurance industry.

Keywords: image recognition, information extraction, deep learning, computer vision, optical character recognition

OPEN ACCESS

Edited by:
Qingtang Jiang,
University of Missouri-St. Louis, United States

Reviewed by:
Baobin Li,
University of Chinese Academy of Sciences, China
Lin Li,
Xidian University, China

****Correspondence:***
Shailesh Acharya
sailes437@gmail.com

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *09 August 2019*
Accepted: *15 November 2019*
Published: *10 December 2019*

Citation:
Acharya S and Fung G (2019) Mileage Extraction From Odometer Pictures for Automating Auto Insurance Processes. Front. Appl. Math. Stat. 5:61. doi: 10.3389/fams.2019.00061

1. INTRODUCTION

In a competitive customer-driven auto insurance landscape, businesses are constantly changing the way they interact with customers to improve attraction and retention. Better customer experiences and more efficient interactions with customers lead to satisfaction which is one of the top differentiators that impact customer loyalty. Digitization and process automation allow service providers to unveil timely opportunities to offer effective and time-saving interactions to improve customer experience.

With the incorporation of more sophisticated safety features in modern cars, the increase in claim cost due to the replacement of modern devices is outpacing the decline in claim frequency. Hence, there is pressure on insurance companies to create a more effective way to handle auto claims. Filing a claim is an example of one of the few direct interactions customers have with their insurer and it comes at a time when they are under stress and will most likely appreciate a streamlined process.

However, the typical experience offered today from most insurers when you have an accident involves a process to manage the claim filing that can be slow, expensive (for the insurer) and may involve several insurance representatives. The same idea applies when a potential new customer is inquiring about a new insurance policy.

When asking for a quote for a new policy, potential customers can upload photographs that can be used for retrieving quick information about the car from their phone to a web-based app, which can be analyzed in seconds. This results in a quick and accurate quote. By reducing human error and accelerating the information collection process, we can make processes that involve customer interactions smoother, hence simplifying the policy claims ecosystem for the agent, the customer and the insurer.

Optical character recognition (OCR) is a widely researched problem in computer vision. Text extraction from scanned documents or from pictures taken under controlled lighting has seen significant improvement with the advent of deep learning architectures. However, text extraction from images in the wild is still very challenging. The general purpose OCRs do not work well for images from uncontrolled sources. In this paper, we describe a novel solution for extracting mileage readings from odometer images. In the insurance domain, especially for auto insurance quotes and claims processing, there are three key pieces of information; license plate number, odometer mileage reading and vehicle identification number (VIN). License plate recognition and VIN recognition from images are very popular problems and there exist commercial solutions for both. It is important to note that VIN recognition is a significantly easier problem since for modern cars the VIN number plates are standardized. To the best of our knowledge, few or no work has been done for odometer mileage extraction from images and there are not reliable available commercial solutions for this use-case.

There are several open source and commercial OCRs available in market such as Tesseract [1], and the built-in OCR toolbox in Matlab [2] to name a few. These OCRs systems are designed to read characters from high quality pictures taken by scanners or a camera under good lighting conditions. They use image pre-processing and character segmentation techniques that are very specific to document images. They are trained to recognize printed characters which are different from characters in a odometer display since odometer images contain huge variation in color, intensity, font, and texture. For all these reasons, these OCR systems perform poorly on odometer images. Google cloud vision API [3] is another interesting commercial option that does a better job in extracting text from images in the wild, but its performance on odometer images is nowhere close to our accuracy expectations and does not meet our performance requirements.

We divide the mileage extraction problem into two parts; identifying odometer display and extracting characters inside the display. We leveraged existing object detection architectures to solve each part and finally designed a post processing algorithm to extract mileage number. We tested two different object detection architectures Single Shot Detector (SSD)[4] and Faster RCNN [5]. Our system differs from open source OCR such as tesseract and other commercial OCRs both on the system architecture and the dataset used for training. We used hand labeled odometer pictures to train the character recognition which makes our model much more customized to odometer characters than any other OCRs. We also designed the post processing algorithm to distinguish mileage reading from other characters in odometer display such as tripmeter reading, temperature, etc.

The rest of the paper flows as follows: In section 2 we present relevant related work that uses recent machine learning techniques to extract text from pictures taken in non-restrictive environments and background on FasterRCNN and SSD object detectors. In section 3 we describe the data used to train our system which is described in detail in sections 4 (system workflow). After that, we share results derived from our empirical evaluation of the system in section 5 followed by a description of how the system is being deployed in section 6. We end the paper with conclusions and lessons learned and discuss future work in section 7.

2. PRELIMINARIES

2.1. Related Work

As mentioned before, automatic license plate recognition (ALPR) is a mostly commercially solved problem. Besides traffic monitoring, this technology is used in many applications such as, highway toll collection, border and custom checkpoints, parking access control system and more recently, homeland security. The ALPR problem is similar in some aspects to our problem proposed here since most ALPR system breakdown the problem into similar sub-tasks: number plate detection, character segmentation, character recognition. Deep convolutional networks have been used recently to improve accuracy on ALPR systems [6] and in Bulan et al. [7] they propose the use of synthetically generated images to improve CNN performance while reducing the need for human labeling. A more comprehensive survey view of such system can be seen in Sanap and Narot [8], Sonavane et al. [9], and Du et al. [10].

Faster RCNNs have been successfully used to extract text from pictures taken in the wild, for example in Nagaoka et al. [11], the authors propose an architecture that takes into consideration the characteristics of texts by using multiresolution feature maps to detect texts of various sizes simultaneously. A faster RCNN approach is also used in Rosetta [12], a recently proposed scalable system to extract text from web images.

There are many recent real-world applications to detect text in images where the faster RCNN and the Single Shot Detector (SSD) architectures have been used successfully. A good representative example of such system is presented in Yang et al. [13], where the goal is to extract (detecting and recognizing) text from biomedical literature figures.

However, to best of our knowledge there is few or no work related to extracting mileage readings from odometer pictures.

2.2. Faster RCNN

Early object detectors used pyramidal sliding windows over the input image followed by an image classifier to detect objects at various location and scales. The Fast RCNN architecture

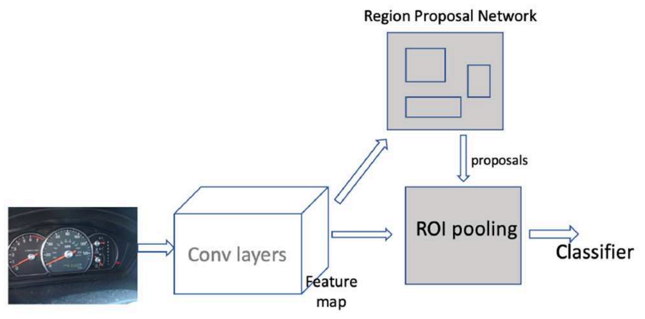


FIGURE 1 | Faster RCNN detector.

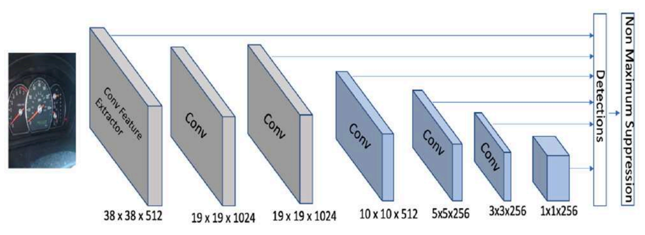


FIGURE 2 | Single Shot Detector extracts detections from feature map at multiple scales.

introduced by Girshick [5] made significant improvement over these architectures by using selective search for region proposals and convolutional feature maps as input. Although, Fast RCNN was significantly faster than the previous architectures, the region proposal technique was still too slow for most real-time applications. Faster RCNN, introduced in Ren et al. [14], solves this problem by using a different region proposal network.

Faster RCNN can be roughly viewed as a combination of two networks: The region proposal network (RPN) and a classifier as shown in **Figure 1**. The RPN takes convolutional feature map inputs and outputs a set of rectangular object proposals and an objectness score for each proposal. But before that, the first step is translating the image to convolutional feature maps by passing the image through a series of convolution layers. In faster RCNN, RPN is modeled with a fully convolutional network [15]. Region proposals are generated by sliding a small sub-network over this convolutional feature map output. The sub-network looks at $n \times n$ spatial windows of input feature maps and projects it into a lower dimensional feature vector. At the end of the sub-network architecture there are two siblings fully connected layers: a box-regression layer and a box-classification layer. The regression layer outputs delta coordinates to adjust the reference anchor coordinates for each spatial window. The box-classification layer predicts the possibility of an anchor box being either background or an object. For the next stage of processing, only the anchors with high scores are retained. The second part of the faster RCNN architecture is a classifier that predicts the class label for the regions proposed by the RPN. The classifier also contains a regression layer that outputs offset coordinates to further tighten the proposed box. The output region from the RPN is passed through a ROI pooling layer to map them to a fixed shape before feeding them to the classifier. The classifier consists of a fully connected layer that outputs softmax scores across all the class labels.

2.3. SSD

The single shot multiBox detector (SSD) was introduced by Liu et al. [4]. The Faster RCNN algorithm produces accurate results but the network is still computationally intensive for use in some real-time applications [4]. The SSD algorithm proposed a series of improvements over the existing object detection architectures for accelerating running time. The main idea behind SSD is predicting category scores and box offsets for a fixed set of default bounding boxes using small convolutional filters applied to feature maps. SSD then generates predictions from different scales of feature maps thereby producing predictions for all of them. Similarly to the faster RCNN algorithm, the input to SSD is a convolutional feature map. In the original paper, the convolutional feature map is generated by passing an image through the Conv5_3 layer of a VGG-16 network. The feature map is downscaled using convolutional filters to get feature maps at multiple scales. **Figure 2** shows original feature maps along with 6 downscaled ones. Each feature map is processed independently using different convolutional models to detect objects at particular scales. There is a set of default boxes associated with each cell of the feature maps. The convolutional model predicts offset coordinates relative to the default boxes and class scores for that box. The offset coordinates move and tighten the default boxes for a better localization of objects. The architecture is trained end-to-end by minimizing the weighted sum of the localization loss and the classification loss.

2.4. Transfer Learning

The success of Deep Learning is contributed mostly by the large datasets available for training the model. However, data acquisition and annotation is costly and time consuming. Both SSD and the Faster RCNN detectors contain deep architecture with large number of parameters. Hence, training them from scratch with small dataset can lead to overfitting.

Transfer learning allows deep networks to be trained on one domain and reused on a different domain. The first few convolution layers of a CNN trained on images learn universal representation of image features. These layers can be reused to build an image classifier with a different dataset. The reused layer can either be fine-tuned on the new network or kept frozen allowing only the newly added layers to be updated. There are several different ways to adopt transfer learning in object detection. **Figures 1**, **2** show that the first step for both the SSD and the faster RCNN detector is transforming the images to convolutional feature maps using a feature extractor. This feature extractor can be constructed from the first few layers of pre-trained image classification architectures such as VGG [16], Inception [17], Resnet [18], etc. trained on a large image classification dataset such as imagenet [19]. When training the object detection model, the layers in the feature extractor can either be kept frozen or updated with a very small learning rate depending on the size of the dataset. Another way of adopting transfer learning in a detection domain is by training a detection model end-to-end using a large object detection dataset such as Pascal VOC [20], MS COCO [21], and fine-tuning it with a new dataset.

3. DATA

Training object detection architectures such as SSD and Faster RCNN requires a large corpus of annotated training samples. Our initial dataset contained only around six thousand (6,000) odometer images. These images were uploaded by customers when filing an auto insurance claim. Before any further processing, we manually filter the dataset to remove images with potential personally identifiable information (PII). We also removed images that do not contain odometers in them. Finally, the gathered dataset has total of 6,209 odometer images. The images came from uncontrolled sources and hence in general the quality of images in the dataset is poor. Most images suffer from non-uniform illumination, insufficient lighting, incorrect orientation and low picture resolution.

3.1. Labeling

The process to label the dataset can be divided in two stages. In the first stage, we aimed to manually segment the odometer display by drawing a bounding box enclosing the display. Here, the term odometer display refers to LCD screens from digital odometers or mechanical meter from analog odometers. In the second stage, our goal was to generate boxes enclosing each individual character inside the odometer display and label the characters with the corresponding digit.

Both of the annotation stages involved labor intensive and repetitive tasks. Hence, we resorted to crowdsourcing as a viable solution for these tasks. There are several commercially available platforms that facilitate crowdsourcing labeling tasks. We used two popular crowdsourcing platforms: Amazon Mechanical Turk (AMT) [22] and Figure Eight (previously known as Crowdflower) [23].

Amazon Mechanical Turk is one of the largest crowdsourcing platforms operating today. At any given time, it has hundreds of active workers ready to work on the given task. It provides flexibility to build customized user interfaces using HTML, CSS and javascript. It also provides some basic customizable templates for annotations tasks like sentiment analysis, image classification, NER, etc.

For our first stage of the annotation process, i.e., manually segmenting the odometer display, we used AMT. For this task, we modified the UI opensourced by Russell et al. [24]. The modified UI allows workers to draw a box over the image, drag it and resize it. We collected 3 boxes from different labelers for each image in order to capture possible annotation errors.

Figure Eight is another crowdsourcing platform that works similar to AMT. In addition to supporting HTML, CSS and Javascript for UI design, it has rich UI templates for labeling different objects in images. It has built-in functionality such as zoom-in, zoom-out, scrolling, etc. that are very relevant for us when drawing character level bounding boxes. The zoom-in functionality facilitates the ability to draw tighter boxes. This platform also monitors the quality of work done by its workers. All workers have to pass tests before they can work on any annotation job. For all these reasons, we found the quality of the annotations on Figure Eight to be better than the ones obtained when using AMT but this comes at an extra price. Hence, we decided to use both platforms for each of our first and second stage of annotation, depending on the trade-off between the cost of labeling vs. the quality of the annotations.

For any sort of annotation task completed through crowdsourcing, it is important that the workers understand the expected outcome of the solicited annotations. It is essential to provide clear and detailed labeling instructions, covering all the corner cases and at the same time being as precise as possible.

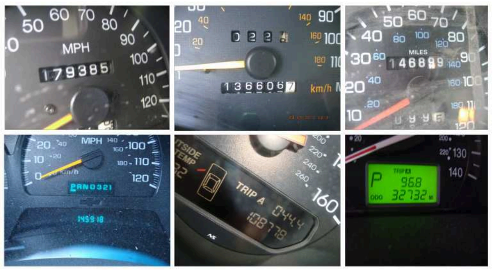


FIGURE 3 | Sample odometer images.

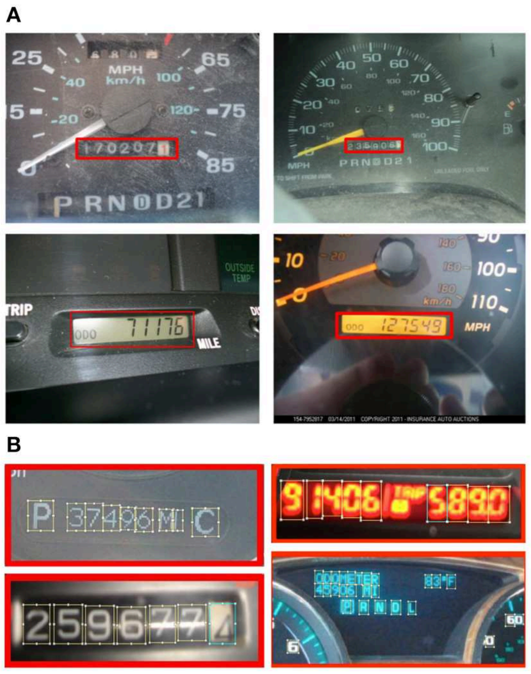


FIGURE 4 | Sample annotations. **(A)** Labeling odometer display. **(B)** Labeling characters.

TABLE 1 | Dataset and distribution.

Total no. of images	6,208
Total no. characters labeled	55,739
Avg. no. char per display	9
Digits	73 %
Alphabets	27 %

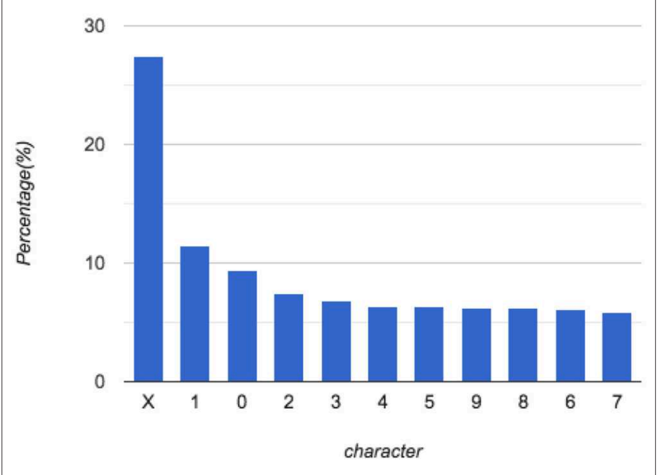


FIGURE 5 | Distribution of characters; X represents non-digit characters.

TABLE 2 | Image quality distribution.

Image Quality	Percentage
Extremely poor	8.1
Poor	12.8
Average	25.3
Good	33.4
Excellent	20.1

We completed the annotation tasks in several batches, we evaluated the annotations quality for each batch and identified the key sources of confusion among the workers. We then changed the instructions accordingly before sending out the next batch. **Figures 3**, **4** show some sample odometer images and the annotation labels.

Table 1 and **Figure 5** show the distribution of the characters in the dataset. 73% of all labeled characters are digits while only 27% of them are letters. With 52 possible alphabet letters (26-lowercase and 26-uppercase), the number of samples for each alphabet class is too small and highly imbalanced. This later inspires us to group all the alphabet characters together in a single class when training the character recognition model.

We also collected additional information from the labelers about the quality of the images in our dataset. During an initial manual inspection, we noticed that a significant portion of the images in the dataset were not of good quality. To confirm this, during annotation, we asked the annotators to rate the image quality of the characters into different categories. **Table 2** shows the distribution of the images in five categories. Note that a significant portion of the images (21%) are marked as being of poor or extremely poor quality.

4. GENERAL WORKFLOW OF THE SYSTEM

The proposed solution consists of two cascaded object detection classifiers followed by a post-processing algorithm (See **Figure 6**). Algorithms for object detection have seen significant improvement over the last few years. In order to leverage the effectiveness of these models, we divide our problem into two sub-problems that can directly be seen as problems in the object detection domain:

- The first is odometer localization where the goal is to locate the odometer display given an input image.
- The second is character recognition where the goal is to locate and recognize characters inside the odometer display.

We next proceed to explain in detail each one of these sub-problems.

4.1. Odometer Localization

The first stage of the pipeline is to isolate and extract the odometer display from the rest of the image. There are commonly two types of odometers: analog and digital. Digital odometers have LCD displays containing a mileage reading and may be accompanied by other information such as temperature, time, fuel status, etc. The analog odometer consists of a mechanical rolling meter. Although, there is large variation in appearance of analog and digital odometers, we do not differentiate these two types for this stage. In order to train the odometer localization model, we trained an object detection model with odometer images where the odometer display box is the object of interest. The position of odometer display is supplied as coordinates (x-center, y-center, height, width) of the odometer display box. Object detection algorithms are usually trained to localize and classify objects in the image. However, for odometer localization there is a single class i.e., odometer display, so the only output we want from the model is the localization coordinates. During inference, the localization model takes an image and output back the coordinates(x-center, y-center, width, height) of the odometer display.

4.2. Character Recognition

The second stage of the pipeline consists of a character recognition model. This an object recognition model trained on images and labels generated in the second stage of annotation. The training images for this stage come from the odometer display labeled in the first stage. We crop the odometer display for each image in the dataset and feed it to the model along with the annotations from the second stage. The second stage produces annotations of position (x-center, y-center, height, width) of each individual character and the corresponding class label. We do make some changes to the class labels before training the classifier. Since we only care about getting the mileage number in the images, it's sufficient to recognize only digits in the images and not the rest of the alphabet characters. Furthermore, if we look at distribution of characters in **Table 1**, we have very few samples per class for the letters in the alphabet. Training a model to recognize individual alphabet characters means we would have very few examples for most class labels and we would risk

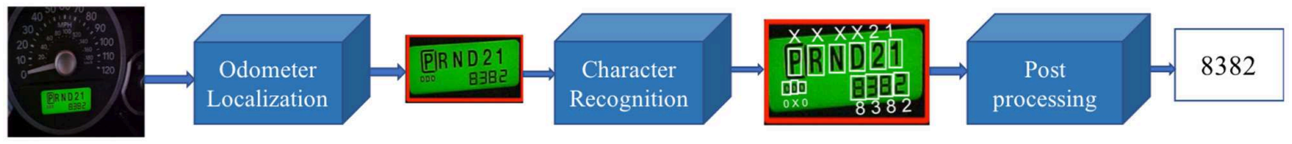


FIGURE 6 | Pipeline of proposed architecture.

Algorithm 1: Post-processing algorithm

```
1: Filter bounding boxes that belong to non-digit character.
2: Augment/extrapolate boxes in horizontal direction by a factor of a quarter of the width ($\frac{width}{4}$) along both sides.
3: Create a graph $G$ where vertices represents bounding box and edges represent overlap between two boxes.
4: Select subgraph with largest number of vertices.
5: Sort boxes in that subgraph horizontally and extract digits in order to form a mileage number.
6: if Mileage has 7 digits then
7:     Discard the last digit
8: end if
9: return Mileage
```

overfitting. Instead, we categorize characters into 11 different class labels, 10 for the digits 0–9 and 1 "non-digit" class for all the alphabets.

4.3. Post-processing

The character recognition stage identifies individual characters inside the odometer display along with their coordinates. In the last part of the pipeline, we want to isolate the digits that are part of the mileage reading. The post-processing step combines nearby characters to form words/numbers and selects the most likely number as the mileage reading. In some digital odometers, we can find additional information being displayed alongside the mileage reading. Some of the most frequently seen additional pieces of information include temperature, time, warning messages, trip meter reading, fuel status, etc. It is essential to distinguish the actual mileage reading from other numbers being displayed on the screen. Similarly, for an analog odometer we observe two variants: most models have six digits while a few older models have 7 digits. Usually the 7th digit changes every 1/10th of a mile and is not considered a significant part of the mileage reading.

In order to deal with special cases like this, we designed a post-processing algorithm that takes care of all these corner cases. The processing algorithm is described in detail below in Algorithm 1.

5. EVALUATION AND EMPIRICAL RESULTS

5.1. Experimental Settings

We randomly selected a small portion of the training set and used it as validation set for all experiments. The hyperparameter selection for all architectures is based on performance in the validation set. We used the object detection API included in tensorflow models [25] to train and evaluate the models. Huang et al. [26] provides in-depth comparison of speed and accuracy of different meta-architectures supported by the API. We used a Amazon Web Services (AWS) Elastic Cloud Compute instance containing 8 GPU with 12 GB memory each for training and testing the models. For both odometer localization task and character recognition task, we train SSD and faster RCNN architectures with several choices of CNN model for Feature extraction such as inception v2 [27], resnet101 [18], inception resnet [28], mobilenet [29], etc. We experimented with both approaches of transfer learning described in the previous section: (a) we fine-tuned a detection model trained on the MS COCO dataset and, (b) we used a classification model trained on the imagenet dataset for feature extraction and trained the remaining layers from scratch. We find that using the detection model trained on the MS COCO dataset gave the best results. Furthermore, SSD got the best performance with inception v2 as features extractor and Faster RCNN got the best results with inception Resnet as the feature extractor. We report the mean average precision for the best performing SSD and faster RCNN for the two stages; Odometer localization and character recognition. We report the final accuracy and error analysis for the faster RCNN architecture which is a winner between the two architectures for both stages.

The best performing faster RCNN model is finetuned version of a faster RCNN detector originally trained on MS COCO dataset. The MS COCO detector was trained with inception resnet architecture [detailed in Szegedy et al. [28]] as feature extractor and 90 different categories in MS COCO dataset as output objects. We finetuned this model by modifying the last layer to detect one class(odometer display) for odometer localization. Similarly, for character recognition we modified the last layer to output 11 classes(0,1,..,9, X). We used a grid anchor generator with scales of $0.25, 0.5, 1.0, and\, 2.0$, aspect ratios of $0.5, 1.0, and\, 2.0$ and strides of 8 for both height and width. This means a total of 12 proposal boxes for each anchor position in the grid. The post processing stage is set to reject all the detections with score $<$ 0.3. The IOU threshold is set to 0.6 for Non maximum suppression. The loss being minimized is the sum of localization loss and classification loss both of which are equally weighted. We used learning rate of 0.0003 and trained the model for $50,000$ steps with a batch size of 8.

5.2. Results

A common evaluation technique for object detection models is to measure mean average precision (map) [20] for a certain threshold of the Intersection Over Union (IOU) ratio. A

TABLE 3 | Mean Average Precision of Faster RCNN and SSD architectures for odometer localization and character recognition stage.

map@0.5IOU	SSD	Faster RCNN
Odometer Localization	0.79	0.82
Character Recognition	0.89	0.93

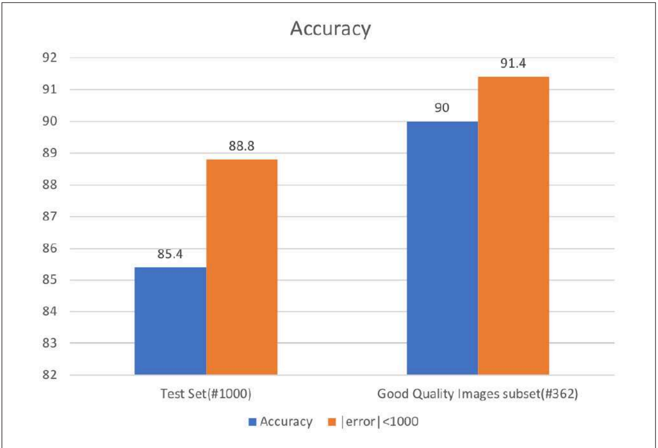


FIGURE 7 | Accuracy results comparisons.

prediction is a true positive if the IOU ratio between the predicted bounding box and the actual box is greater than the IOU threshold. **Table 3** shows the map values (at IOU = 0.5) of the SSD and the faster RCNN models for both the odometer localization the and character recognition task. The results clearly indicate that the faster RCNN algorithm is a winner for both tasks.

Our mileage extraction model contains two object detectors working in conjunction. Rather than detecting an object/character, the objective is to extract the actual mileage reading. To do so, the model has to predict every single digit correctly. For our system, getting those numbers right is more important than getting perfect localization of the odometer display or the individual characters.

In order to measure system performance, We defined a binary measure of end-to-end system accuracy in the following way: the model gets a score equal to 1 if extracted mileage equals the annotated mileage and 0 otherwise. Furthermore, in most business use-cases, it is sufficient to get the mileage within a given error range. For example; if a model predicts the mileage to be 45,607 when the actual mileage is 45,687 then there is an error of 80 miles. For use cases such as insurance quote generation or claims processing a perfectly acceptable margin of error is around a thousand (1,000) miles. Taking this into account, we introduce one more additional end-to-end system evaluation metric in the following way: the model gets score = 1 if absolute(extracted mileage–annotated mileage) < threshold and 0 otherwise (where threshold = 1,000 miles).

Since the overall quality of images in our odometer images dataset is not so good, we performed a further analysis on the effect of the image quality on the performance of the model. We created a subset of the test set comprised of only the good quality images. These images are selected from the test set based on their corresponding annotator rating. This "good-quality images" subset ended up containing 362 images. **Figure 7** shows end-to-end system accuracy for the faster RCNN model for both the original test set and the "good-quality images" subset. For the original test set, we obtain end-to-end accuracy of 85.4% using faster RCNN for both stages. Similarly, we achieve an accuracy of 88.8% within an error boundary of 1,000 miles. For the "good-quality images" subset, we get a general accuracy of 90% and an accuracy of 91.4% within an error bound of 1,000 miles. It is important to note the improvement of 5% in test set accuracy associated with the improvement in image quality. This result presents an opportunity to improve performance by validating the quality of uploaded images in real time and providing immediate feedback and guidance to the customer to generate better quality pictures. Sample results for odometer localization and character recognition are shown in **Figures 8**, **9**.

5.3. Error Analysis

To identify key weakness of the model and opportunities for improvement, we performed a more detailed error analysis. For all the incorrect predictions, we manually assigned the error to one of the three stages in the pipeline. **Figure 10** shows the distribution of the incurred test set errors among the odometer Localization, the character Recognition and the post-processing stage. The localization errors occur when the localization model cannot properly detect the odometer display, either because it did not find the display or because the proposed bounding box is not accurate enough to include all the characters in the display. It is evident from **Figure 10** that a large portion of the errors are coming from the character recognition stage. Errors in this stage include not detecting or recognizing characters inside the odometer display. This error could be minimized by improving the character recognition model. As we mentioned before, image quality is an important factor in improving accuracy and we need to put more effort on ensuring that the uploaded images meet minimum quality standard.

The post processing algorithm constitutes 15% of the total error. This error comprises cases such as failure to group digits together, failure to distinguish mileage from other numbers in the display, identifying the digit after the decimal point as part of the mileage, etc.

6. DEPLOYMENT ARCHITECTURE

The deployment of the odometer mileage detector is a work in progress. However, we are reusing a deployment framework used in the past for similar image recognition models in our company. In this section, we will describe such framework.

Containerized deployment is very popular nowadays. Containers are independent, easily configurable and easily scaled to multiple machines. Microservices running inside containers provide isolation from actual system ingesting the service and provide flexibility to work independently and quickly. We deploy the model as a microservice running in a docker container. Docker allows packaging codes and dependencies into a docker

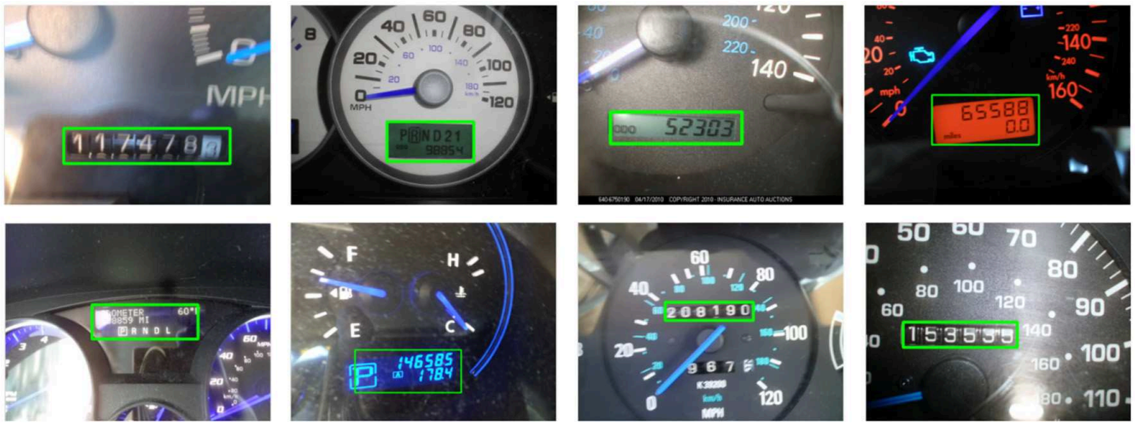


FIGURE 8 | Selected examples of odometer localization. In case of multiple detections, only the most confident box is shown.

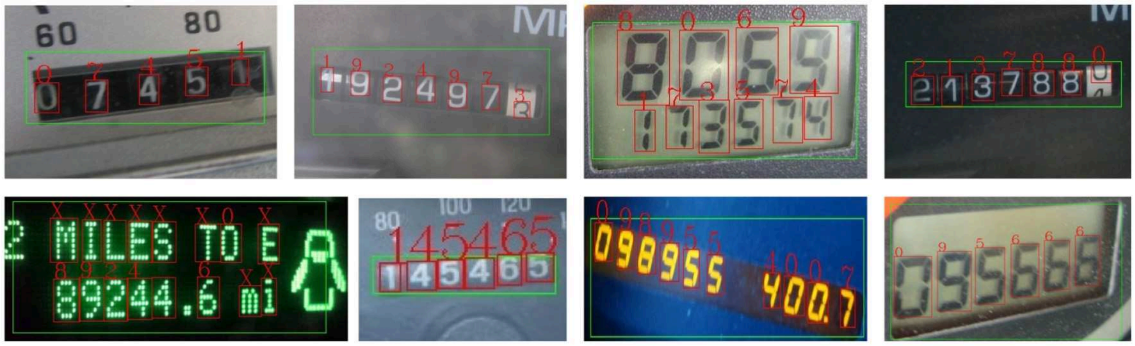


FIGURE 9 | Selected examples of character recognition. Character recognition model scans for characters inside the region(green box) proposed by localization model. Characters in red are predictions from character recognition model. X represents non-digit character.

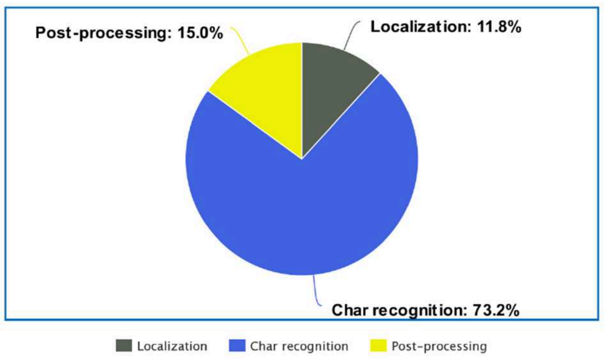


FIGURE 10 | Detailed error analysis by stage.

image that runs inside a docker container. Docker containers are compatible to run on any operating system.

Figure 11 shows the overall architecture used for deployment. We use tools provided by the Amazon Web Service(AWS) ecosystem to launch, scale, orchestrate and run the docker container. Detailed description of each of these tools can be found in the official site [30]. The central component is the docker container hosting the odometer mileage extraction model. We use the Amazon elastic container registry (ECR) to host docker images and Amazon elastic container services (ECS) to run the containers. We use Amazon systems manager parameter store (SMPS) to store runtime parameters and Amazon CodeBuild to build the docker image. Furthermore, Amazon ElasticBeanStalk (EBS) is used to orchestrate the deployment to ECS, as well as to provision and configure other resources such as LoadBalancer, AutoScaling groups, etc. EBS facilitates logging, monitoring and sending notifications to the developers about unexpected service interruptions. We believe that the Continuous Integration/Continuous Delivery(CI/CD) principle [31] is a crucial part of any data science project. We want to be able to train new models or update code base and deploy them into production automatically with minimum effort. This allows data scientist to focus more on improving models rather than spending time on deployment. For CI/CD, we use Jenkins. As soon as we push changes to a git repository, Jenkins builds an image, runs tests and deploys the model to production. Here is a step by step break down of the deployment process:

- Push changes to git repository hosted in bitbucket.

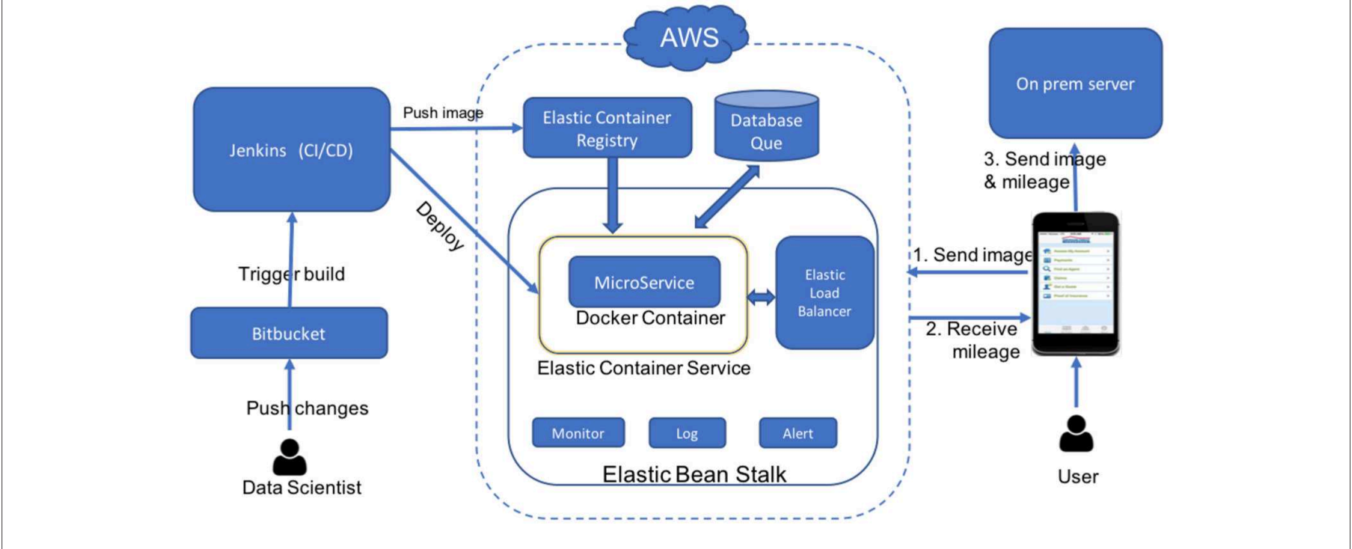


FIGURE 11 | Deployment architecture.

- Jenkins monitors changes in git repository and initiates build process.
- Jenkins builds code, runs test and builds image.
- Jenkins pushes image to ECR and issues deploy to ECS.
- ECS pulls new image from ECR and runs it inside docker container.
- EBS receives a HTTP request with odometer image.
- ELB distributes load across multiple containers and EBS launches additional container instances if necessary.
- Container processes the image and sends mileage back to user app.

The client mobile app makes a HTTP request to odometer server and receives a mileage number in response. It auto-fills the odometer mileage reading into the form. The user will have an option to validate, and correct the mileage reading if necessary, before submitting the form. The odometer picture is uploaded to a on-premises server along with the form during submission.

7. CONCLUSIONS AND FUTURE WORK

In this work we developed a novel solution to the insurance-related problem of extracting mileage readings from odometer images. We leveraged existing object recognition technology and designed a post processing algorithm to identify and extract mileage readings. The developed system was able to get high accuracy in mileage extraction despite having poor quality images. We also have provided a complete implementation design including the tools and technology we are using to deploy, scale and manage the model in production.

Our detailed error analysis provides insights into the shortcomings of the system and unveil opportunities to improve it. We can further improve performance of the model using image guidance and enforcing minimum requirements on image quality. For example, when a user takes a picture of the odometer, the app display could contain a bounding box and the user will be asked to align the odometer display within that bounding box. This technique is commonly used in several applications that read data from credit cards, personal checks, etc. Image guidance could help mitigate the need for having an accurate localization model and hence the errors associated with that model could be minimized significantly. This will also ensure that the images are taken directly facing the odometer display and with a proper orientation.

We are also exploring methods to estimate prediction confidence for the predicted mileage digits. If we are able to estimate prediction confidence, we can automatically accept images when we feel confident that we are predicting the correct mileage reading and ask the user to repeat the process or enter the mileage by hand if we fail to produce a confident enough prediction.

DATA AVAILABILITY STATEMENT

The datasets generated for this study cannot be released publicly due to the privacy concern of the customers. Requests to access these datasets should be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

SA implemented the project, ran experiments, and worked on manuscript. GF initiated the project, managed it, and worked on manuscript.

REFERENCES

1. Smith R. An overview of the tesseract OCR engine. In: *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)* Parana (2007). p. 629–33. doi: 10.1109/ICDAR.2007.4378659
2. Matlab OCR toolbox (2018). Available online at: https://www.mathworks.com/help/vision/ref/ocr.html (accessed February 1, 2019).
3. Hosseini H, Xiao B, Poovendran R. Google's cloud vision API is not robust to noise. *CoRR*. (2017) abs/1704.05051.
4. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, et al. Ssd: single shot multibox detector. In: *European Conference on Computer Vision*. Amsterdam: Springer (2016). p. 21–37.
5. Girshick R. Fast R-CNN. In: *The IEEE International Conference on Computer Vision (ICCV)*. Beijing (2015).
6. Masood SZ, Shu G, Dehghan A, Ortiz EG. License plate detection and recognition using deeply learned convolutional neural networks. *CoRR*. (2017) abs/1703.07330.
7. Bulan O, Kozitsky V, Ramesh P, Shreve M. Segmentation- and annotation-free license plate recognition with deep localization and failure identification. *IEEE Trans Intell Trans Syst*. (2017) **18**:2351–63. doi: 10.1109/TITS.2016.2639020
8. Sanap PR, Narote SP. License plate recognition system-survey. *AIP Conf Proc*. (2010) **1324**:255–60. doi: 10.1063/1.3526208
9. Sonavane K, Soni B, Majhi U. Survey on automatic number plate recognition (ANR). *Int J Comput Appl*. (2015) **125**:1–4. doi: 10.5120/ijca2015905920
10. Du S, Ibrahim M, Shehata MS, Badawy WM. Automatic License Plate Recognition (ALPR): a state-of-the-art review. *IEEE Trans Circ Syst Video Technol*. (2013) **23**:311–25. doi: 10.1109/TCSVT.2012.2203741
11. Nagaoka Y, Miyazaki T, Sugaya Y, Omachi S. Text detection by faster R-CNN with multiple region proposal networks. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 6. Kyoto: IEEE (2017). p. 15–20.
12. Borisyuk F, Gordo A, Sivakumar V. Rosetta: Large scale system for text detection and recognition in images. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. London: ACM (2018). p. 71–9.
13. Yang C, Yin X-C, Yu H, Karatzas D, Cao Y. ICDAR2017 robust reading challenge on text extraction from biomedical literature figures (DeTEXT). In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 1. Kyoto: IEEE (2017). p. 1444–7.
14. Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. In: *Advances in Neural Information Processing Systems*. Montreal, QC: Curran Associates, Inc. (2015). p. 91–9.
15. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (Boston, MA) (2015). p. 3431–40.
16. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. *CoRR*. (2014) abs/1409.1556.
17. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (Boston, MA) (2015). p. 1–9.
18. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (Seattle, WA) (2016). p. 770–8.
19. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. Imagenet large scale visual recognition challenge. *Int J Comput Vision*. (2015) **115**:211–52.
20. Everingham M, Eslami SMA, Van Gool L, Williams CKI, Winn J, Zisserman A. The pascal visual object classes challenge: a retrospective. *Int J Comput Vision*. (2015) **111**:98–136. doi: 10.1007/s11263-014-0733-5
21. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft coco: common objects in context. In: *European Conference on Computer Vision*. Zurich: Springer (2014). p. 740–55.
22. Mechanical Turk (2019). Available online at: https://www.mturk.com/ (accessed February 1, 2019).
23. Figure Eight (2019). Available online at: https://www.figure-eight.com/ (accessed February 1, 2019).
24. Russell BC, Torralba A, Murphy KP, Freeman WT. LabelMe: a database and web-based tool for image annotation. *Int J Comput Vision*. (2008) **77**:157–73. doi: 10.1007/s11263-007-0090-8
25. Github Contributor. *Object Detection API*. GitHub (2019). [commit 947c92bc44df7499baa3da1fefe7d3094a1f4561]. Available online at: https://github.com/tensorflow/models/tree/master/research/object_detection (accessed February 1, 2019).
26. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In: *IEEE CVPR*. Vol. 4. (Honolulu, HI) (2017).
27. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Seattle, WA (2016). p. 2818–26.
28. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, inception-resnet and the impact of residual connections on learning. In: *AAAI* (San Francisco, CA). (2017). p. 12.
29. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. Mobilenets: efficient convolutional neural networks for mobile vision applications. *arXiv preprint*. (2017) arXiv:170404861.
30. Amazon Web Services Ecosystem (2019). Available online at: https://aws.amazon.com/products/ (accessed February 1, 2019).
31. Wikipedia contributors. CI/CD — Wikipedia, The Free Encyclopedia (2019). Available online at: https://en.wikipedia.org/w/index.php?title=CI/CD&oldid=877599340 (accessed February 1, 2019).

Conflict of Interest: SA and GF were employed by the company American Family Insurance.

ORIGINAL RESEARCH
published: 14 January 2020
doi: 10.3389/fams.2019.00067

Task-Optimized Word Embeddings for Text Classification Representations

Sukrat Gupta[†], Teja Kanchinadam[*†], Devin Conathan and Glenn Fung*

Machine Learning Research Group, America Family Insurance, Madison, WI, United States

Word embeddings have introduced a compact and efficient way of representing text for further downstream natural language processing (NLP) tasks. Most word embedding algorithms are optimized at the word level. However, many NLP applications require text representations of groups of words, like sentences or paragraphs. In this paper, we propose a supervised algorithm that produces a task-optimized weighted average of word embeddings for a given task. Our proposed text embedding algorithm combines the compactness and expressiveness of the word-embedding representations with the word-level insights of a BoW-type model, where weights correspond to actual words. Numerical experiments across different domains show the competence of our algorithm.

Keywords: NLP (national language processing), word embedding, text classification, SVM—support vector machine, text representation models

Edited by:
Yiming Ying,
University at Albany, United States

Reviewed by:
Shao-Bo Lin,
Xi'an Jiaotong University (XJTU), China
Sijia Liu,
Mayo Clinic, United States

****Correspondence:***
Teja Kanchinadam
tkanchin@amfam.com

[†]These authors have contributed equally to this work

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *09 August 2019*
Accepted: *16 December 2019*
Published: *14 January 2020*

Citation:
Gupta S, Kanchinadam T, Conathan D and Fung G (2020) Task-Optimized Word Embeddings for Text Classification Representations. Front. Appl. Math. Stat. 5:67. doi: 10.3389/fams.2019.00067

1. INTRODUCTION

Word embeddings, or a learned mapping from a vocabulary to a vector space, are essential tools for state-of-the-art Natural Language Processing (NLP) techniques. Dense word vectors, like Word2Vec [1] and GLoVE [2], are compact representations of a word's semantic meaning, as demonstrated in analogy tasks [3] and part-of-speech tagging [4].

Most downstream tasks, like sentiment analysis and information retrieval (IR), are used to analyze groups of words, like sentences or paragraphs. For this paper, we refer to this more general embedding as a "text embedding."

In this paper we propose a supervised algorithm that produces embeddings at the sentence-level that consist on an weighted average of an available pre-trained word-level embedding. The resulting sentence-level embedding is optimized for the corresponding supervised learning task. The weights that the proposed algorithm produces can be use to estimate the importance of the words with respect to the supervised task. For example, when classifying movie reviews into one of two classes: action movies or romantic movies, words like "action," "romance," "love," and "blood," will get precedence over words, like "movie," "i," and "theater." This leads to the shifting of the text-level vector toward words with larger weights, as can be seen in **Figure 1**.

When we use an unweighted averaged word embedding (**UAEm**) [5] for representing the two reviews, we see that all the words get the same importance, due to which the reviews—**"I like action movies"** and **"I prefer romance flicks"**—end up close to each other in the vector space. Our algorithm, on the other hand, identifies "romance" and "action" as two important words in the vocabulary for the supervised task, and assigns weights with high absolute value to these words. This leads to shifting of the representation of the two reviews toward their respective important words in the vector space, increasing the distance between them. This indicates that, for the task of differentiating an action movie review from a romantic movie review, our algorithm

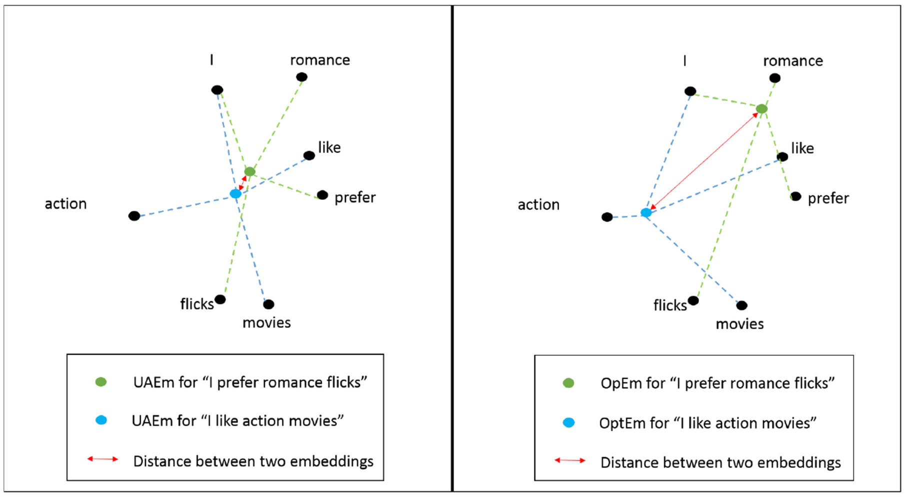

I
romance
like
prefer
action
flicks
movies
UAEm for "I prefer romance flicks"
UAEm for "I like action movies"
Distance between two embeddings
OpEm for "I prefer romance flicks"
OptEm for "I like action movies"
Distance between two embeddings

FIGURE 1 | Unweighted (UAEm) **(left)** and Optimal embeddings (OptEm) **(right)** of two movie reviews in feature space. Distance between the two reviews increases for OptEm representation of the text.

produces a representation at the review level more adequate for discriminating between the two kinds of reviews.

Our algorithm has many advantages over simpler text embedding approaches, like bag-of-words (BoW) and the averaged word-embedding schemes discussed in section 2. In section 4, we show results from experiments on different datasets. In general, we observed that our algorithm is competitive with other methods. Unlike the simpler algorithms, our approach finds a *task-specific* representation. While BoW and some weighting schemes, like tf-idf, rely only on word frequencies to determine word importance, our algorithm computes how important the word is to a specific task. We believe that for some applications, this task-specific representation is important for performance; one would expect the importance of words to be very different whether you are trying to do topic modeling or sentiment analysis.

It is important to note that other deep-learning-based approaches for text classification also implicitly optimize the text-level representation from word-level embedding in the top layers of the neural network. However, in order to train such models large datasets are needed. Our empirical results show that our proposed representation is in general competitive with traditional deep learning based text classification approaches and outperforms them when the training data is relatively small.

Additionally, by generating importance weights to each one of the words in the vocabulary, our algorithm yields a more interpretable result than looking at the weights corresponding to the word-embedding dimensions that have no human-interpretable meaning. Effectively, our text embedding algorithm combines the compactness and expressiveness of the word-embedding representations with the human-interpretability of a BoW-type model.

Furthermore, in contrast with some deep-learning-based approaches, our approach does not impose constraints or require special processing (trimming, padding) with respect to the length of the sentence or text to be classified. In summary, we can summarize the contributions of the paper as follows:

- Our algorithm provides a task optimized text embedding from word level embeddings.
- Our algorithm outperforms other more complex algorithms when training data is relatively small in size.
- Our algorithm can be implemented by leveraging existing libraries in a trivial way as it only requires access to a SVM implementation.
- Our resulting task specific text embedding are as compact as the original word level embedding while providing word level insights similar to a BOW type model.

The rest of the paper is organized as follows: in section 2, we discuss related work. Later, in section 3, we present a detailed explanation and mathematical justification to support our proposed algorithm. In section 4, we present and described our proposed algorithm.

2. RELATED WORK

Various representation techniques for text have been introduced over the course of time. In the recent years, none of these representations have been as popular as the word embeddings, such as Word2Vec [1] and GLoVE [2], that took contextual usage of words into consideration. This has led to very robust word and text representations.

Text embedding has been a more challenging problem over word embeddings due to the variance of phrases, sentences, and text. Le and Mikolov [6] developed a method to generate the embeddings that outperforms the traditional bag-of-words approach [7]. More recently, deeper neural architectures have been developed to generate these embeddings and to perform text classification tasks [8] and some of these architectures involve sequential information of text, such as LSTMs [9], BERT [10], and XLNET [11]. Furthermore, recently developed attention models can also provide insights about word importance, however they require large amounts of training data.

Methods have been developed that use word embeddings to generate text embeddings without having to train on whole texts. These methods are less costly than the ones that train directly on whole text, and can be implemented faster.

Unweighted average word embedding [5] generated text embeddings by computing average of the embeddings of all the words occurring in the text. This is one of the most popular methods of computing text embeddings from trained word embeddings, and, though simple, has been known to outperform the more complex text embedding models especially in out-of-domain scenarios. Arora et al. [12] provided a simpler method to enhance the performance of text embedding generated from simple averaged embedding by the application of PCA.

The unsupervised text embedding methods face the problem of importance-allocation of words while computing the embedding. This is important, as word importance determines how biased the text embedding needs to be toward the more informative words. DeBoom et al. [13] introduced a method that would assign importance to the words based on their tf-idf scores in the text.

Our method generates weights based on the importance of the words perceived through a supervised approach. We use classifiers to determine the weights of the words based on their importance captured through the procedure. The advantage of this method over other methods is that we keep the simplicity of Wieting's algorithm [5], while incorporating the semantically agreeable weights for the words.

3. OPTIMAL WORD EMBEDDINGS

A sentence, paragraph, or document can be represented using a given word-level embedding (**wle**) as follows:

$$A_i = \sum_{j=1}^{k} \delta_{ij}\lambda_j v_j \tag{1}$$

where,

- $A_i \in R^n$ is a vectorial representation or embedding at the sentence, text or document level (we will refer it as **tle** in rest of the paper) of ith sample;
- we will assume that A_i is the ith row of a matrix $A \in R^{m\times n}$ containing a collection of m documents, k is the number of words in the **wle** corpus V;
- $\lambda_j \in R$ is a weighting factor associated with the jth word $v_j \in V$. Note that for the widely used averaged **tle** (text2vec) representation [5], $\lambda_j = 1, \forall j$;
- δ_{ij} is a normalized occurrence count. It is the number of times jth word appears in the document i divided by the total number of words in the document i.

Our proposed algorithm assumes that we have a supervised classification problem for which we want to find an optimal representation at the document (text) level from the word embeddings.

More concretely, we consider the problem of classifying m points in the n-dimensional real space R^n, represented by the $m \times n$ matrix A, according to membership of each point A_i in the classes +1 or -1 as specified by a given $m \times m$ diagonal matrix D with ones or minus ones along its diagonal.

In general, this linear classification problem can formulated as follows:

$$\begin{aligned} \min_{(w,\gamma,y\geq 0)} \quad & cL(y) + R(w) \\ \text{s.t.} \quad & D(Aw - e\gamma) + y \geq e \end{aligned} \tag{2}$$

where,

- $e \in \mathbb{R}^{m\times 1}$ is a column of ones;
- $y \in \mathbb{R}^{m\times 1}$ is a slack vector;
- $(w, \gamma) \in \mathbb{R}^{(n+1)\times 1}$ represents the separating hyperplane.
- L is a loss function that is used to minimize the misclassification error.
- R is a regularization function used to improve generalization.
- c is a constant that controls the trade-off between error and generalization.

Note that, if $L(.)=\|(.)_+\|_2^2$ and $R(.)=\|.\|_2^2$, then Equation (2) corresponds to an SVM formulation [14]. The corresponding unconstrained convex optimization problem is given as:

$$\min_{\omega,\gamma} c\|(e - D(Aw - e\gamma))_+\|_2^2 + \|w\|_2^2 \tag{3}$$

which we will denote by

$$(w, \gamma) = SVM(A, D, c) \tag{4}$$

From (1), we can rewrite A as:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^k \delta_{1j}\lambda_j v_j \\ \sum_{j=1}^k \delta_{2j}\lambda_j v_j \\ \vdots \\ \sum_{j=1}^k \delta_{mj}\lambda_j v_j \end{bmatrix} \tag{5}$$

That is,

$$A = \Delta\Lambda V \tag{6}$$

where $\Delta \in \mathbb{R}^{m\times k}$; is a matrix of occurrences count with δ_{ij} in the (i,j) position. $\Lambda = diag((\lambda_1, \ldots, \lambda_k)) \in \mathbb{R}^{k\times k}$, and $V \in \mathbb{R}^{k\times n}$ is the matrix whose rows are all the word2vec vectors considered in the word2vec corpus or dictionary.

From (3) and (6),

$$\min_{w,\gamma,\boldsymbol{\lambda}} c\|(e - D((\Delta\Lambda V)w - e\gamma))_+\|_2^2 + \|w\|_2^2 \tag{7}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$.

Formulation (7) is a biconvex optimization problem, which can be solved using alternate optimization [15]. By solving this problem, not only do we obtain an SVM-type classifier, but also learn the optimal importance weights for each word in our corpus $(\lambda_1, \ldots, \lambda_k)$ which can be used to interpret classification results for the specific tasks at hand. Though we could have restricted the λ_i to be positive, we choose to leave them unconstrained in order to make our algorithm more scalable and computationally efficient. Another interesting consideration would be to add a relative importance constrained on addition to the non-negativity bounds of the form:

$$\lambda_1 + \lambda_2 + \ldots + \lambda_k = 1 \tag{8}$$

but again, we choose not to for computationally efficiency. We will explore this option in the future.

In (7), if we fix Λ to a constant $\bar{\Lambda}$, we have:

$$\tilde{A} = \Delta \bar{\Lambda} V \tag{9}$$

We can obtain the corresponding optimal solution for (w, γ) by solving $(w*, \gamma*) = SVM(\tilde{A}, D, c)$.

On the other hand, if we fix $(w, \gamma) = (\tilde{w}, \tilde{\gamma})$, we get

$$A\tilde{w} = (\Delta \Lambda V)\tilde{w} = (\Delta \tilde{W})\boldsymbol{\lambda} \tag{10}$$

where $\tilde{W} \in \mathbb{R}^{k\times k} = diag(V\tilde{w})$.

Similarly, from (7) and (10) and making $\tilde{M} = \Delta \tilde{W}$, we have

$$\begin{aligned} &\min_{\gamma,\lambda_i} c\|(e - D(\tilde{M}\lambda - e\gamma))_+\|_2^2 + \|\tilde{w}\|_2^2 \\ \equiv\ &\min_{\gamma,\lambda_i} c\|(e - D(\tilde{M}\lambda - e\gamma))_+\|_2^2 \end{aligned} \tag{11}$$

since $\tilde{w}$ is a constant.

We can obtain an approximate optimal $(\boldsymbol{\lambda}, \gamma)$ by solving $(\boldsymbol{\lambda}^*, \gamma^*) = SVM(\tilde{M}, D, c)$. Note that this solution will consider a regularization term for $\boldsymbol{\lambda}$.

We are ready now to describe our proposed alternate optimization (AO) algorithm to solve formulation (7).

One of the advantages of the algorithm is that it can be easily implemented by using existing open-source SVM libraries, like the ones included in scikit-learn [16] or a more recent GPU-based fast SVM implementation like ThunderSVM [17].

The optimal text embedding algorithm, then, inherits the convergence properties and characteristics of the AO problems [15]. It is important to note that the set of possible solutions to which Algorithm 1 can converge can include certain type of saddle points (i.e., a point that behaves like a local minimizer only when projected along a subset of the variables). However, it is stated in the paper [15] that it is extremely difficult to find examples where converge occurs to a saddle point rather than to a local minimizer.

In order to further reduce the computational complexity of the proposed algorithm, we can consider a simplified loss function $L(.)=\|.\|_2^2$ and $R(.)=\|.\|_2^2$. Then formulation (7) becomes the corresponding unconstrained convex optimization problem:

$$\min_{w,\gamma,\boldsymbol{\lambda}} c\|e - D((\Delta \Lambda V)w - e\gamma)\|_2^2 + \|w\|_2^2 \tag{12}$$

Algorithm 1: Optimal Text Embedding

```
Input  : Training vocabulary matrix ($V$); scaled word
         occurrence matrix ($\Delta$); vector of labels $diag(D)$;
         max number of iterations $maxiter$;
         tolerance $tol$;
         regularization parameters $c_1$ and $c_2$;
Output: optimal word weight vector $\boldsymbol{\lambda}^*$;
         classification hyperplane $(w^*, \gamma^*)$;
Initialize $\forall j\ \lambda_j$=1; $\Lambda_0$=diagonal($\boldsymbol{\lambda}$)
$i = 0$;

while $i \leq maxiter$ or $\|\Lambda_i - \Lambda_{i-1}\| > tol$ do
    iter++;
    Given $\Lambda_{i-1}$, calculate $\tilde{A} = \Delta \bar{\Lambda}_{i-1} V$;
    Solve $(w_i, \gamma) = SVM(\tilde{A}, D, c_1)$;
    Given $(w_i, \gamma)$, calculate $\Delta \tilde{W}_i$ as described in equation
    (10);
    Solve $(\boldsymbol{\lambda}_i, \gamma) = SVM(\tilde{M}, D, c_2)$;
end
$\boldsymbol{\lambda}^* = \boldsymbol{\lambda}_i$;
$(w^*, \gamma^*) = (w_i, \gamma_i)$;
```

Fixing $\Lambda = \tilde{\Lambda}$, from (9) and (12), we have

$$c\|(e - D(\tilde{A}w - e\gamma))\|_2^2 + \|w\|_2^2 \tag{13}$$

This formulation corresponds to a least-squares or Proximal SVM formulation [18, 19], and its solution can be obtained by solving a simple system of linear equations. We will denote formulation (13) by

$$(w, \gamma) = LSSVM(\tilde{A}, D, c) \tag{14}$$

If $\bar{A} = \left[\tilde{A}\ \vdots - e\right]$ then the solution to (13) is given by

$$(w, \gamma) = (\bar{A}^T\bar{A} + \frac{1}{c}I)^{-1}\bar{A}^T De \tag{15}$$

On the other hand, fixing $(w, \gamma) = (\tilde{w}, \tilde{\gamma})$, we have

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} c\|e - D((\Delta \tilde{W}\boldsymbol{\lambda}) - e\gamma)\|_2^2 + \|\tilde{w}\|_2^2 \\ \equiv\ &\min_{\boldsymbol{\lambda}} c\|e - D((\Delta \tilde{W}\boldsymbol{\lambda}) - e\gamma)\|_2^2 \end{aligned} \tag{16}$$

since $\tilde{w}$ is a constant. Hence,

$$\boldsymbol{\lambda} = ((\Delta \tilde{W})^T(\Delta \tilde{W}))^{-1}(\Delta \tilde{W})^T De(1 - \gamma) \tag{17}$$

Furthermore,

$$(\Delta \tilde{W})^T(\Delta \tilde{W}) = \tilde{W}^T \Delta^T \Delta \tilde{W} \tag{18}$$

From (17) and (18),

$$\begin{aligned} \boldsymbol{\lambda} &= (\tilde{W}^T \Delta^T \Delta \tilde{W})^{-1}(\Delta \tilde{W})^T De(1 - \gamma) \\ &= (\Delta^T \Delta \tilde{W})^{-1}(\tilde{W})^{-1}\tilde{W}\Delta^T De(1 - \gamma) \\ &= \mathbf{diag}(\tfrac{1}{V\tilde{w}})(\Delta^T \Delta)^{-1}\Delta^T De(1 - \gamma) \end{aligned} \tag{19}$$

For some problems, $\Delta^T\Delta$ can be ill-conditioned, which may lead to incorrect values for $\boldsymbol{\lambda}$. In order to improve conditioning we add a Tikhonov regularization perturbation [20]. (19) becomes

$$\boldsymbol{\lambda} = \mathbf{diag}(\frac{1}{V\tilde{w}})(\Delta^T\Delta + \epsilon I)^{-1}\Delta^T De(1-\gamma) \tag{20}$$

where ϵ is a very small value.

Note that $(\Delta^T\Delta + \epsilon I)^{-1}$ involves calculating the inverse of a $k \times k$ matrix, where k is the number of words in the word2vec dictionary. In some cases, k can be much larger than m, the number of training set examples. If this is the case, we can use the Sherman-Morrison-Woodbury formula [21]:

$$(Z + uv^T)^{-1} = Z^{-1} - Z^{-1}u(I + v^T Z^{-1}u)^{-1}v^T Z^{-1} \tag{21}$$

with $Z = \epsilon I$, $u = v = \Delta^T$. Then $(\Delta^T\Delta + \epsilon I)^{-1}$ becomes

$$(\Delta^T\Delta + \epsilon I)^{-1} = \frac{1}{\epsilon}(I - \Delta^T(\Delta\Delta^T + \epsilon I)^{-1}\Delta) \tag{22}$$

which involves inverting an $m \times m$ matrix with $m << k$.

The $\boldsymbol{\lambda}$ we obtained is a vector of weights of the words that would be used in (1) to calculate text2vec of a given sample.

Algorithm 1 can be modified to consider formulation (3) instead of (13) by making two simple changes:

1. Substitute line 6 of Algorithm 1 by: Solve Equation (15) to obtain (w_i, γ);
2. Substitute line 8 of Algorithm 1 by: Solve Equation (19) to obtain $\boldsymbol{\lambda}_i$;

4. EXPERIMENTS

We used binary classification as the task for evaluating our algorithm performance by comparing it to the following methods:

1. **UAEm**: Unweighted average of the word vectors that comprise the sentence or document [5].
2. **WAEm**: Weighted averaged text representations. We computed **WAEm** using tf-idf coefficients as the weights as described in De Boom et al. [13].
3. **FastText** [22], an open-source, free, library that allows users to learn text representations and text classifiers. The classifiers are based in a simple shallow model instead of deep one which allows the framework to train models in a fast manner.
4. **AdvCNN** [8] is a CNN based deep network which comprises of parallel convolutional layers with varying filter widths and it achieves state-of-the-art performance on sentiment analysis and question classification.
5. **VanillaCNN** is a custom CNN architecture we designed and is similar to Kim [8] except that in this case there is only one convolutional layer instead of parallel layers.

Note that in both the CNN experiments we have initialized the embedding layer with pre-trained word2vec models and these vectors are kept static.

We implemented two versions of our Algorithm 1: SVM-based (**SVM-OptEm**) (Formulation 3) and least square SVM-based (**LSSVM-OptEm**) (Formulation 12).

In SVM-OptEm, we used a support vector machine (SVM) [23] as the classifier. We used a scikit-learn [24] implementation of SVM for the experiments.

In LSSVM-OptEm, we used a least square support vector machine (LS-SVM) [23] as the classifier.

4.1. Datasets

To showcase the performance of our model, we chose fifteen different binary classification tasks over the subsets of different datasets. Twelve public datasets are briefly described in **Table 1**.

We also performed experiments on three datasets belonging to the insurance domain.

- **BI-1** and **BI-2**: These datasets consist of the claim notes with binary classes based on topic of phone conversation. These notes were taken by call representative of the company after the phone call was completed. For BI-1, we classified the call notes into two categories based on claim complexity: simple and complex. For BI-2, we wanted to identify notes that documented a failed attempt made by the call representative to get in touch with the customer. It is important to note that the corpus is same for these two datasets but the classification task is different.
- **TRANSCRIPTS**: These datasets consist of the phone transcripts with two classes: pay-by-phone calls and others. These transcripts were generated inside the company for the calls received at the call center. Each call would be assigned a class based on the purpose of the call.

4.2. Word Embeddings

We chose to work on different word2vec-based word embeddings. These word embeddings have either been pre-trained models or in-house trained models. These embeddings were used on the datasets based on their contextual relevance.

- **wikipedia** [38]: The skip-gram model was trained on English articles in Wikipedia by FastText [39].
- **google-news** [40]: The model was trained on Google News Data, and is available on the Google Code website [41].
- **amzn**: The skip-gram model was trained in-house on amazon reviews [27, 28]. Gensim [42] was used to train the model.
- **yelp**: The skip-gram model was trained in-house on yelp reviews [37]. Gensim was used to train the model.
- **transcript**: The continuous bag-of-words model was trained in-house on the transcripts generated in the of the calls from call centers. Gensim was used to train the model over approximately 3 million transcripts.
- **claim-notes**: The continuous bag-of-words model was trained in-house on the notes taken by call representatives after the call was completed. C-based code from Google Word2vec website [41] was used to train the model over approximately 100 million notes.

We used different word2vec models to verify that our models works well independently of the underlying embedding

TABLE 1 | Brief description of the public datasets used for our experiments.

Dataset	Description	Positive class	References
20NEWSGRP-SCI	20 Newsgroup documents	Science-related documents	[25]
AMZN-EX	Amazon reviews	Electronics review	[26]
AMZNBK-SENT	Amazon book reviews	Positive review	[27, 28]
BBC	BBC news articles	Sports article	[29]
BLOG-GENDER	Blog articles	Male Writer	[30]
DBPEDIA	Wikipedia articles	Artist article	[31–33]
IMDB	IMDB movie reviews	Positive review	[34]
SCIPAP	Sentences from scientific papers	Owner-written sentence	[35]
SST	Movie reviews	Positive sentiment	[36]
YAHOO-ANS	Questions from Yahoo's question-answer dataset	Health-related question	[33]
YELP-REST	Yelp Restaurant Reviews	Restaurant-related review	[37]
YELP-STAR	Yelp Reviews	Positive review	[37]

representation. Moreover, it also gives better contextual representation of words for these datasets.

4.3. Text Processing

The method of processing employed on text was similar to the one done for training the word2vec models. This ensured the consistency of word-occurrence in the dataset in lieu to the model that would be used for mapping the words.

Different word2vec models had different processing procedures, such as substitutions based on regular expressions, removal of non-alphabetical words, and lowercasing the text. Accordingly, text-processing was done for the training data.

4.4. Results

To compare performance of the algorithms tested, we decided to use area under curve (AUC) for evaluation. This metric was chosen in order to remove the possibility of unbalanced datasets affecting the efficacy of the accuracy of the models.

The performance of our models for the experiments can be seen in **Table 2**.

Our algorithm provides better or comparable performance against **UAEm** and **WAEm**. This performance is achieved over multiple iterations, as seen in **Figure 2**. The number of iterations required to reach the best performance for our model varies with the dataset and training size.

It is important to note that our proposed algorithm tends to achieve better AUC performance when the training data is small which it is the case for many scenarios in the insurance domain where labels are difficult and expensive to obtain. This fact make the algorithm a good choice for active learning frameworks where labels are scarce specially at early iterations of such approaches.

In general, our algorithm approached an "equilibrium" stage for the vector $\boldsymbol{\lambda}$, as seen in **Figure 3**. In other words, as the algorithm iterate, the norm of the difference between the current weights and the weights from the previous iteration of the words approaches zero. This behavior is seen consistently for all the experimental cases. This shows that our algorithm exhibits good convergence behavior as expected.

4.5. Text Representation

One of the advantages our model holds over **UAEm** and **WAEm** is that our model can be used to extract the most important words in the training set. As our model reconfigures the weights of the words at each iteration, it also indirectly reassigns the degree of importance to these words. We can obtain these words by taking the absolute values of the weights assigned to these words at the end of the iteration. This information can be used for improving different algorithms, such as visual representation of text and topic-discovery, and as features for other models.

Figure 4 shows weights of top 15 words for three of our datasets. Weights assigned to the words are based on the role they play in helping the classifier determine the class of any given sample.

Table 3 shows the top 10 words for three of our datasets. The words are determined by taking the absolute value of the weights i.e., λ^* learned from the algorithm and rank them in descending order. For a human eye, these words clearly makes sense with respect to the given classification task. For example,

1. *AMZN-EX* classification task is to predict items belonging to eletronics category based on reviews.
2. *YAHOO-ANS* classification task is to predict health related questions.
3. *DBPEDIA* classification task is to predict artistic articles.

We also found that words that are least informative about the given task have weights(λ^*) close to zero.

Following our results presented in **Table 2**, we want to highlight the following observations:

- Our method is competitive with more sophisticated models. As a matter of fact, we are winning on 7 out of 15 text classification tasks from various domains.
- Our method seems to significantly outperform other approaches when the dataset size is relatively small in size. This might be very relevant in situations where labeling data is expensive to obtain which is often the case in many industrial applications.

TABLE 2 | Binary text classification AUC and accuracy results for test data for: **UAEm**, **WAEm**, **VanillaCNN**, **AdvCNN**, the SVM-based implementation **(SVM-OptEm)** and the least square SVM-based implementation **(LSSVM-OptEm)**.

Dataset	**Word2Vec model**	**Avg length (words)**	**Data size**		**Area under curve**						
			Train	**Test**	**UAEm**	**WAEm**	**SVM-OptEm**	**LSSVM-OptEm**	**FastText**	**VanillaCNN**	**AdvCNN**
20NEWSGRP-SCI	Google-news	86	3000	2000	0.904	0.9053	**0.9427**	0.9040	0.9081	0.9150	0.9139
AMZN-EX	Wikipedia	100	10,000	10,000	0.9914	0.9897	0.9887	0.9822	0.9838	0.9921	**0.9924**
AMZNBK-SENT	Amzn	5	10,000	10,000	0.9294	0.9218	0.9344	0.9269	0.9294	0.9273	**0.9378**
BBC	Google-news	458	1,850	500	0.9978	0.9973	0.9959	**0.9978**	0.9946	0.9921	0.9948
BLOG-GENDER	Wikipedia	422	2,000	1,000	0.7668	0.7536	**0.7992**	0.7813	0.7580	0.7428	0.7569
DBPEDIA	Wikipedia	48	10,000	10,000	0.9921	0.9870	0.9930	0.9935	0.9934	0.9974	**0.9976**
IMDB	Wikipedia	237	5,000	2,500	0.9116	0.8935	**0.9321**	0.9209	0.9206	0.8981	0.9102
SCIPAP	Wikipedia	26	1,500	750	0.8630	0.8515	0.9208	**0.9220**	0.9205	0.8973	0.9105
SST	Google-news	11	10,000	10,000	0.9016	0.8990	0.9040	0.8967	0.8722	**0.9203**	0.9168
YAHOO-ANS	Wikipedia	12	20,000	10,000	0.9316	0.9293	0.9287	0.9248	0.8819	**0.9334**	0.9280
YELP-REST	Yelp	117	40,000	40,000	0.9733	0.9709	0.9696	0.9627	0.9342	0.9773	**0.9779**
YELP-STAR	Yelp	125	20,000	10,000	0.9707	0.9652	0.9707	0.9665	0.9567	0.9747	**0.9778**
BI-1	Claim-notes	128	1,508	561	0.8850	0.8270	0.8852	**0.9114**	0.9014	0.7023	0.7907
BI-2	Claim-notes	137	1,081	238	0.7653	0.666	0.8007	**0.8338**	0.5403	0.4875	0.5640
TRANSCRIPTS	Transcript	828	5,000	3,000	0.9616	0.9604	0.9638	0.9620	0.9617	**0.9745**	0.9736

The highest score for each evaluation metric is in boldface.

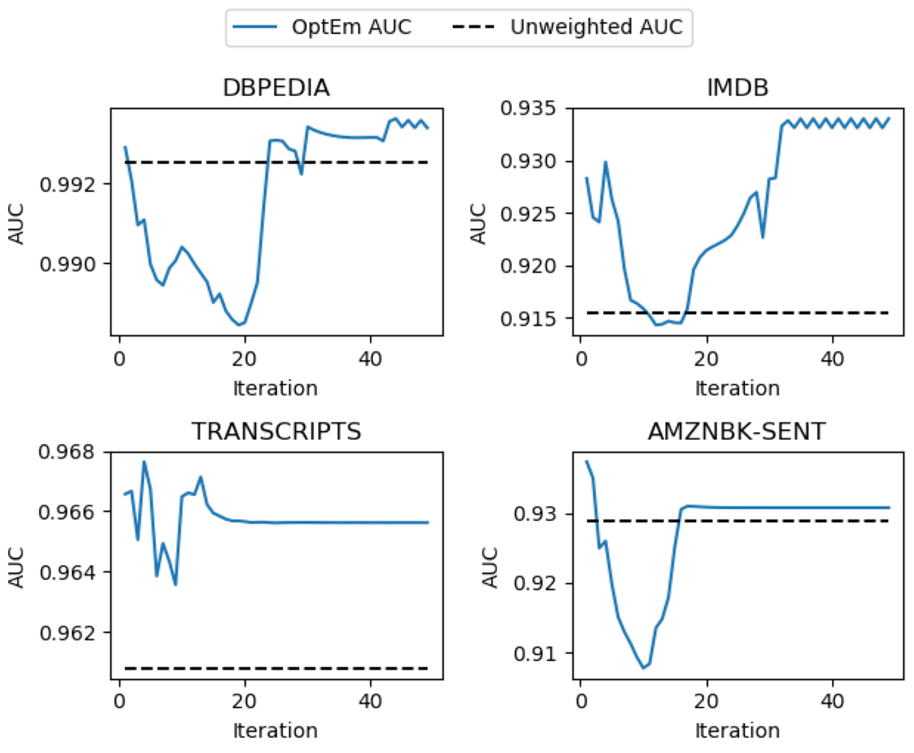


FIGURE 2 | AUC scores of test data over iterations.

5. CONCLUSIONS AND FUTURE WORK

Our paper provides an alternative way of sentence/document-level representation for supervised text classification, based on optimization of the weights of words in the corresponding text to be classified. This approach takes labels into consideration when generating optimal word's weights for these words. Numerical experiments show that our proposed algorithm is competitive with respect with other state-of-the-art techniques and outperformed CNNs when the training data was small and we even show that this approach is not sensitive to document lengths.

Our model also brings additional benefits to the table. It provides a ranking of the relevance of the words with respect to the text classification problem at hand. This ranking

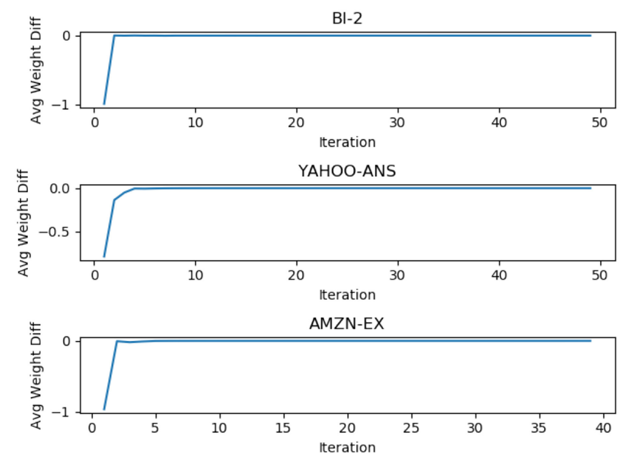


FIGURE 3 | Average of weight difference over iteration for three datasets. This difference approaches zero over iterations.

TABLE 3 | Top 10 Words with highest absolute weights for AMZN-EX, YAHOO-ANS, AND DBPEDIA.

AMZN-EX	YAHOO-ANS	DBPEDIA
Book	Period	Born
Sound	Profile	Author
Product	Mushrooms	Singer
Player	Medicare	Directed
Use	Daily	Album
Unit	Youngest	Artist
Price	Longest	Writer
Quality	Anger	Known
Lens	Aerobics	Musician
Radio	Confirm	Novelist

For a human eye, most of these words makes sense given the classification task.

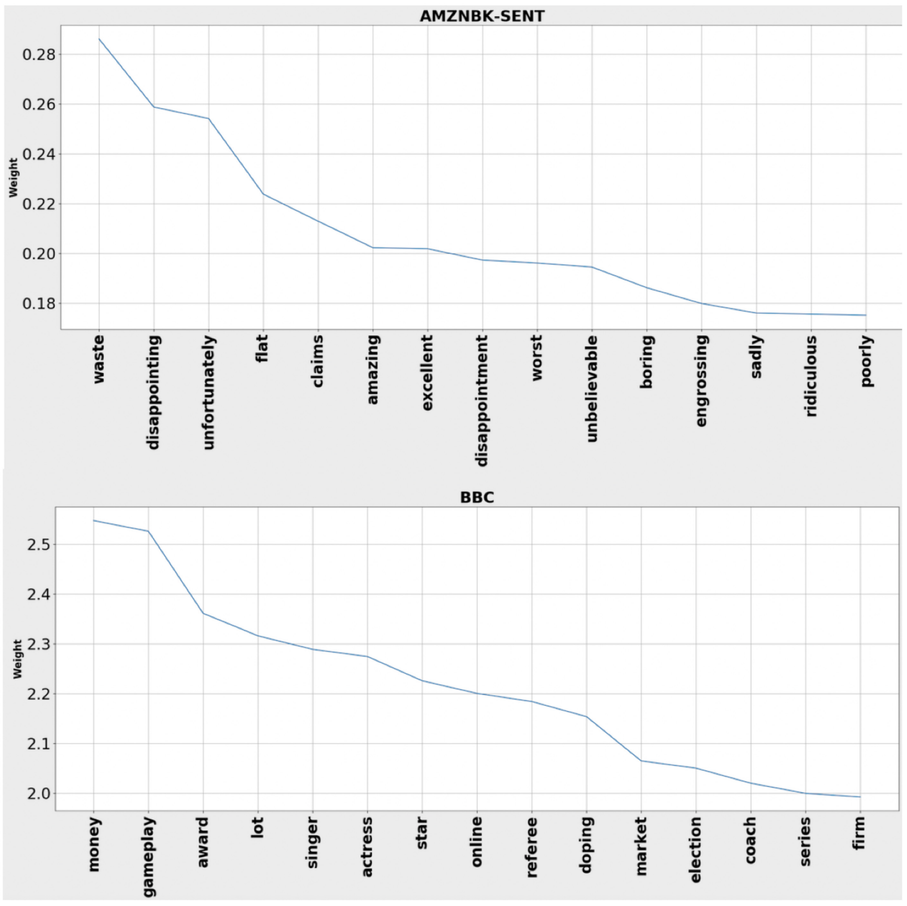


FIGURE 4 | Weights of top 15 words identified by OptEm for two of the datasets used in our experiments. The words appear to be very informative; some can be easily associated to corresponding class.

of words by importance can be used for different NLP applications related to the same task, such as extraction-based summarization, context-matching, and text cleaning. By learning the optimal weights of the words, our model also tends to remove or ignore less informative words, thus performing its own version of feature selection. Our text embedding algorithm combines the compactness and expressiveness of the word-embedding representations with the human-interpretability of a BoW-type model.

We intend to extend this work to make the proposed algorithm more scalable in order to incorporate larger, more complex classification models and tasks, such as multi-label, multi-class classification and summarization.

We want to explore using other normalizations and constraints to the weight vector. One possibility is to explore 1-norm regulation for the weight vector to make it more sparse and have a more aggressive feature (word) selection. Another interesting direction is to consider group regularization similar [43], where the groups of words are suggested by a graph naturally defined by the distances between the words provided by the word embedding. In this way, semantically similar words would be weighted similarly and the result of the algorithm would be a clustering of terms by semantic meaning or topics that are relevant to the classification problem at hand.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

GF was the technical advisor and the central figure for driving this project to completion. SG was responsible for running all the initial set of experiments and dataset preparation. TK was responsible for finishing all of the remaining experiments and manuscript writing. DC was part of research discussions and brainstorming.

REFERENCES

1. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: *Proceedings of ICLR Workshop*. Scottsdale, AZ (2013).
2. Pennington J, Socher R, Manning CD. GloVe: global vectors for word representation. In: *Empirical Methods in Natural Language Processing (EMNLP)* (2014). p. 1532–43. Available online at: http://www.aclweb.org/anthology/D14-1162
3. Levy O, Goldberg Y. Linguistic regularities in sparse and explicit word representations. In: *CoNLL*. Ann Arbor, MI (2014).
4. Lin C, Ammar W, Dyer C, Levin LS. Unsupervised POS induction with word embeddings. *CoRR.* (2015) abs/1503.06760. Available online at: http://arxiv.org/abs/1503.06760.
5. Wieting J, Bansal M, Gimpel K, Livescu K. Towards universal paraphrastic sentence embeddings. CoRR. *abs/1511.08198* (2015).
6. Le Q, Mikolov T. Distributed representations of sentences and documents. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. ICML'14.* (2014). p.II–1188–II–1196. Available online at: http://dl.acm.org/citation.cfm?id=3044805.3045025
7. Harris Z. Distributional structure. *Word*. (1954) **10**:146–62.
8. Kim Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:14085882*. (2014). doi: 10.3115/v1/D14-1181
9. Palangi H, Deng L, Shen Y, Gao J, He X, Chen J, et al. Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval. *IEEE/ACM Trans Audio Speech Lang Proc.* (2016) **24**:694–707. doi: 10.1109/TASLP.2016.2520371
10. Devlin J, Chang MW, Lee K, Toutanova K. Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:181004805* (2018). doi: 10.18653/v1/N19-1423
11. Yang Z, Dai Z, Yang Y, Carbonell J, SalakhutdinovRR, Le QV. Xlnet: generalized autoregressive pretraining for language understanding. In: Wallach H, Larochelle H, Beygelzimer A, d'Alch é-Buc F, Fox E, Garnett R, editors. *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. (2019). p. 5754–64. Available online at: http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf
12. Arora S, Liang Y, Ma T. A simple but tough-to-beat baseline for sentence embeddings. In: *5th International Conference on Learning Representations, ICLR 2017* (Toulon). Available online at: https://openreview.net/forum?id=SyK00v5xx
13. De Boom C, Van Canneyt S, Demeester T, Dhoedt B. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recogn Lett.* (2016) **80**:150–6. doi: 10.1016/j.patrec.2016.06.012
14. Lee YJ, Mangasarian OL. SSVM: a smooth support vector machine for classification. *Comput Optim Appl.* (2001) **20**:5–22. doi: 10.1023/A:1011215321374
15. Bezdek JC, Hathaway RJ. Convergence of alternating optimization. *Neural Parallel Sci Comput.* (2003) **11**:351–68.
16. scikit-learn. *Support Vector Machines* (2017). Available online at: http://scikit-learn.org/stable/modules/svm.html
17. Wen Z, Shi J, Li Q, He B, Chen J. ThunderSVM: a fast SVMlibrary on GPUs and CPUs. *J Mach Learn Res.* (2018) 19:797–801.
18. Fung G, Mangasarian OL. Proximal support vector machine classifiers. In: Provost F, Srikant R, editors. *Proceedings KDD-2001: Knowledge Discovery and Data Mining*. San Francisco, CA; New York, NY: Asscociation for Computing Machinery (2001). p. 77–86. Available online at: Ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps
19. Suykens JAK, Vandewalle J. Least squares support vector machine classifiers. *Neural Process Lett.* (1999) **9**:293–300. doi: 10.1023/A:1018628609742
20. Neumaier A. Solving ill-conditioned and singular linear systems: a tutorial on regularization. *SIAM Rev.* (1998) **40**:636–66. doi: 10.1137/S0036144597321909
21. Golub GH, Van Loan CF. *Matrix Computations, 3rd Edn*. Baltimore, MD: Johns Hopkins University Press (1996).
22. Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers.* Valencia: Association for Computational Linguistics (2017). p. 427–31.
23. Cortes C, Vapnik V. Support-vector networks. *Mach Learn.* (1995) 20:273–97. doi: 10.1023/A:1022627411411
24. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. *J Mach Learn Res.* (2011) **12**:2825–30.
25. Lang K. NewsWeeder: learning to filter netnews. In: *Proceedings of the 12th International Machine Learning Conference*. Tahoe City, CA (1995). p. 331–9.
26. Blitzer J, Dredze M, Pereira F. Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In: *Proceedings of the Association for Computational Linguistics* (ACL) (2007).
27. McAuley J, Targett C, Shi Q, van den Hengel A. Image-based recommendations on styles and substitutes. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in*

Information Retrieval, SIGIR '15. New York, NY: ACM (2015). p. 43–52. Available online at: http://doi.acm.org/10.1145/2766462.27\penalty-\@M67755

28. He R, McAuley J. Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: *Proceedings of the 25th International Conference on World Wide Web. WWW '16*. Republic and Canton of Geneva: International World Wide Web Conferences Steering Committee (2016). p. 507–17. Available online at: https://doi.org/10.1145/2872427.2883037
29. Greene D, Cunningham P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In: *Proc. 23rd International Conference on Machine learning (ICML'06)*. Pittsburgh, PA: ACM Press (2006). p. 377–84.
30. Mukherjee A, Liu B. Improving gender classification of blog authors. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10*. Stroudsburg, PA: Association for Computational Linguistics (2010). p. 207–17. Available online at: http://dl.acm.org/citation.cfm?id=1870658.1870679
31. DBPedia. *DBPedia* (2018). Available online at: http://wiki.dbpedia.org/
32. Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, et al. DBpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semant Web J*. (2015) **6**:167–95. doi: 10.3233/SW-140134
33. Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15*. Cambridge, MA: MIT Press (2015). p. 649–57. Available online at: http://dl.acm.org/citation.cfm?id=2969239.2969312
34. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C. Learning word vectors for sentiment analysis. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, OR: Association for Computational Linguistics (2011). p. 142–50. Available online at: http://www.aclweb.org/anthology/P11-1015
35. Lichman M. *UCI* Machine Learning Repository. Irvine, CA (2013). Available online at: http://archive.ics.uci.edu/ml
36. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, et al. Parsing With compositional vector grammars. In: *EMNLP* (2013).
37. Yelp. *Yelp Dataset Challenge* (2018). Available online at: https://www.yelp.com/dataset/challenge
38. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. *arXiv preprint arXiv:160704606*. (2016). doi: 10.1162/tacl_a_00051
39. Facebook. *FastText* (2018). Available online at: https://fasttext.cc/
40. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS'13*. Curran Associates Inc. (2013). p. 3111–9. Available online at: http://dl.acm.org/citation.cfm?id=2999792.2999959
41. Google. *Word2Vec* (2018). Available online at: https://code.google.com/archive/p/word2vec/
42. Řehůřek R, Sojka P. Software framework for topic modelling with large corpora. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta : ELRA (2010). p. 45–50. Available online at: http://is.muni.cz/publication/884893/en
43. Fung G, Stoeckel J. SVM feature selection for classification of SPECT images of Alzheimer's disease using spatial information. *Knowl Inform Syst*. (2007) **11**:243–58. doi: 10.1007/s10115-006-0043-5

Conflict of Interest: Authors were employed by the company American Family Insurance. The authors declare that this study received funding from American Family Insurance. The funder was not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

ORIGINAL RESEARCH
published: 12 May 2020
doi: 10.3389/fams.2020.00011

Deep Learning Methods for Mean Field Control Problems With Delay

Jean-Pierre Fouque and Zhaoyu Zhang*

Department of Statistics and Applied Probability, University of California Santa Barbara, Santa Barbara, CA, United States

We consider a general class of mean field control problems described by stochastic delayed differential equations of McKean–Vlasov type. Two numerical algorithms are provided based on deep learning techniques, one is to directly parameterize the optimal control using neural networks, the other is based on numerically solving the McKean–Vlasov forward anticipated backward stochastic differential equation (MV-FABSDE) system. In addition, we establish the necessary and sufficient stochastic maximum principle of this class of mean field control problems with delay based on the differential calculus on function of measures, and the existence and uniqueness results are proved for the associated MV-FABSDE system under suitable conditions.

Mathematical Subject Classification (2000): 93E20, 60G99, 68-04

Keywords: deep learning, mean field control, delay, McKean-Vlasov, stochastic maximum principal

OPEN ACCESS

Edited by:
Sou Cheng Choi,
Illinois Institute of Technology,
United States

Reviewed by:
Tomas Tichy,
VSB-Technical University of Ostrava,
Czechia
Samy Wu Fung,
University of California, Los Angeles,
United States

Correspondence:
Jean-Pierre Fouque
fouque@pstat.ucsb.edu

Specialty section:
This article was submitted to Mathematical Finance, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *02 November 2019*
Accepted: *07 April 2020*
Published: *12 May 2020*

Citation:
Fouque J-P and Zhang Z (2020) Deep Learning Methods for Mean Field Control Problems With Delay. Front. Appl. Math. Stat. 6:11. doi: 10.3389/fams.2020.00011

1. INTRODUCTION

Stochastic games were introduced to study the optimal behaviors of agents interacting with each other. They are used to study the topic of systemic risk in the context of finance. For example, in Carmona et al. [1], the authors proposed a linear quadratic inter-bank borrowing and lending model, and solved explicitly for the Nash equilibrium with a finite number of players. Later, this model was extended in Carmona et al. [2] by considering delay in the control in the state dynamic to account for the debt repayment. The authors analyzed the problem via a probabilistic approach which relies on stochastic maximum principle, as well as via an analytic approach which is built on top of an infinite dimensional dynamic programming principle.

Both mean field control and mean field games are used to characterize the asymptotic behavior of a stochastic game as the number of players grows to infinity under the assumption that all the agents behave similarly, but with different notion of equilibrium. The mean field games consist of solving a standard control problem, where the flow of measures is fixed, and solving a fixed point problem such that this flow of measures matches the distribution of the dynamic of a representative agent. Whereas, the mean field control problem is a non-standard control problem in the sense that the law of state is present in the McKean–Vlasov dynamic, and optimization is performed while imposing the constraint of distribution of the state. More details can be found in Carmona and Delarue [3] and Bensoussan et al. [4].

In this paper, we considered a general class of mean field control problem with delay effect in the McKean–Vlasov dynamic. We derived the adjoint process associated with the McKean–Vlasov stochastic delayed differential equation, which is an anticipated backward stochastic differential equation of McKean–Vlasov type due to the fact that the conditional expectation of the future of adjoint process as well as the distribution of the state dynamic are involved. This type of anticipated backward stochastic differential equations (BSDE) was introduced in Peng and Yang [5], and for the general theory of BSDE, we refer Zhang [6]. The necessary and sufficient part of stochastic maximum principle for control problem with delay in state and control can be found in Chen and Wu [7]. Here, we also establish a necessary and sufficient stochastic maximum principle based on differential calculus on functions of measures as we consider the delay in the distribution. In the meantime, we also prove the existence and uniqueness of the system of McKean–Vlasov forward anticipated backward stochastic differential equations (MV-FABSDE) under some suitable conditions using the method of continuation, which can be found in Zhang [6], Peng and Wu [8], Bensoussan et al. [9], and Carmona et al. [2]. For a comprehensive study of FBSDE theory, we refer to Ma and Yong [10].

When there was no delay effect in the dynamic, Ma et al. [11] proved the relation between the solution to the FBSDE and quasi-linear partial differential equation (PDE) via "Four Step Scheme." E et al. [12] and Raissi [13] explored the use of deep learning for solving these PDEs in high dimensions. However, the class of fully coupled MV-FABSDE considered in our paper has no explicit solution. Here, we present one algorithm to tackle the above problem by means of deep learning techniques. Due to the non-Markovian nature of the state dynamic, we apply the long short-term memory (LSTM) network, which is able to capture the arbitrary long-term dependencies in the data sequence. It also partially solves the vanishing gradient problem in vanilla recurrent neural networks (RNNs), as was shown in Hochreiter and Schmidhuber [14]. The idea of our algorithm is to approximate the solution to the adjoint process and the conditional expectation of the adjoint process. The optimal control is readily obtained after the MV-FABSDE being solved. We may also emphasize that the way that we present here for numerically computing conditional expectation may have a wide range of applications, and it is simple to implement. We also present another algorithm solving the mean field control problem by directly parameterizing the optimal control. Similar idea can be found in the policy gradient method in the regime of reinforcement learning [15] as well as in Han and E [16]. Numerically, the two algorithms that we propose in this paper yield the same results. Besides, our approaches are benchmarked to the case with no delay for which we have explicit solutions.

The paper is organized as follows. We start with an N-player game with delay, and let number of players goes to infinity to introduce a mean field control problem in section 2. Next, in section 3, we mathematically formulate the feedforward neural networks and LSTM networks, and we propose two algorithms to numerically solve the mean field control problem with delay using deep learning techniques. This is illustrated on a simple linear-quadratic toy model, however with delay in the control. One algorithm is based on directly parameterizing the control, and the other depends on numerically solving the MV-FABSDE system. In addition, we also provide an example of solving a linear quadratic mean field control problem with no delay both analytically, and numerically. The adjoint process associated with the delayed dynamic is derived, as well as the stochastic maximum principle is proved in section 4. Finally, the uniqueness and existence solution for this class of MV-FABSDE are proved under suitable assumptions via continuation method in section 5.

2. FORMULATION OF THE PROBLEM

We consider an N-player game with delay in both state and control. The dynamic $(X_t^i)_{0\le t\le T}$ for player $i \in \{1,\dots,N\}$ is given by a stochastic delayed differential equation (SDDE),

$$\begin{aligned} dX_t^i &= b^i(t,\mathbf{X}_t,\mathbf{X}_{t-\tau},\alpha_t^i,\alpha_{t-\tau}^i)dt + \sigma^i(t,\mathbf{X}_t,\mathbf{X}_{t-\tau},\alpha_t^i,\alpha_{t-\tau}^i)dW_t^i, \\ &\quad t\in(0,T], \\ X_0^i &= x_0^i, \\ X_t^i &= \alpha_t^i = 0; \quad t\in[-\tau,0), \end{aligned} \tag{2.1}$$

for $T>0$, $\tau>0$ given constants, where $\mathbf{X}_t = (X_t^1,\cdots,X_t^N)$, and where $((W_t^i)_{t\in[0,T]})_{i=1,\cdots,N}$ are N independent Brownian motions defined on the space $(\Omega,\mathcal{F},\mathbb{P})$, $(\mathcal{F}_t)_{0\le t\le T}$ being the natural filtration of Brownian motions.

$$(b^i,\sigma^i):[0,T]\times\times\mathbb{R}^N\times\mathbb{R}^N\times A\times A\to\mathbb{R}\times\mathbb{R},$$

are progressively measurable functions with values in $\mathbb{R}$. We denote A a closed convex subset of $\mathbb{R}$, the set of actions that player i can take, and denote $\mathbb{A}$ the set of admissible control processes. For each $i\in\{1,\dots,N\}$, A-valued measurable processes $(\alpha_t^i)_{0\le t\le T}$ satisfy an integrability condition such that $\mathbb{E}\left[\int_{-\tau}^T |\alpha_t^i|^2dt\right] < +\infty$.

Given an initial condition $\mathbf{x}_0 = (x_0^1,\cdots,x_0^N)\in\mathbb{R}^N$, each player would like to minimize his objective functional:

$$J^i(\boldsymbol{\alpha}) = \mathbb{E}\left[\int_0^T f^i(t,\mathbf{X}_t,\mathbf{X}_{t-\tau},\alpha_t^i)dt + g^i(\mathbf{X}_T)\right], \tag{2.2}$$

for some Borel measurable functions $f^i:[0,T]\times\mathbb{R}^N\times\mathbb{R}^N\times A\to\mathbb{R}$, and $g^i:\mathbb{R}^N\to\mathbb{R}$.

In order to study the mean-field limit of $(\mathbf{X}_t)_{t\in[0,T]}$, we assume that the system (2.1) satisfy a symmetric property, that is to say, for each player i, the other players are indistinguishable. Therefore, drift b^i and volatility σ^i in (2.1) take the form of

$$(b^i,\sigma^i)(t,\mathbf{X}_t,\mathbf{X}_{t-\tau},\alpha_t^i,\alpha_{t-\tau}^i) = (b^i,\sigma^i)(t,X_t^i,\mu_t^N,X_{t-\tau}^i,\mu_{t-\tau}^N,\alpha_t^i,\alpha_{t-\tau}^i),$$

and the running cost f^i and terminal cost g^i are of the form

$$\begin{aligned} f^i(t,\mathbf{X}_t,\mathbf{X}_{t-\tau},\alpha_t^i) &= f^i(t,X_t^i,\mu_t^N,X_{t-\tau}^i,\mu_{t-\tau}^N,\alpha_t^i) \text{ and } g^i(\mathbf{X}_T) \\ &= g^i(X_T^i,\mu_T^N), \end{aligned}$$

where we use the notation μ_t^N for the empirical distribution of $\mathbf{X} = (X^1,\cdots,X^N)$ at time t, which is defined as

$$\mu_t^N = \frac{1}{N}\sum_{j=1}^N \delta_{X_t^j}.$$

Next, we let the number of players N go to $+\infty$ before we perform the optimization. According to symmetry property and the theory of propagation of chaos, the joint distribution of the N dimensional process $(\mathbf{X}_t)_{0\le t\le T} = (X_t^1,\dots,X_t^N)_{0\le t\le T}$ converges to a product distribution, and the distribution of each single marginal process converges to the distribution of $(X_t)_{0\le t\le T}$ of the following McKean–Vlasov stochastic delayed differential equation (MV-SDDE). For more detail on the argument without delay, we refer to Carmona and Delarue [3] and Carmona et al. [17].

$$\begin{aligned} dX_t &= b(t, X_t, \mu_t, X_{t-\tau}, \mu_{t-\tau}, \alpha_t, \alpha_{t-\tau})dt \\ &\quad + \sigma(t, X_t, \mu_t, X_{t-\tau}, \mu_{t-\tau}, \alpha_t, \alpha_{t-\tau})dW_t, \quad t \in (0, T], \\ X_0 &= x_0, \\ X_t &= \alpha_t = 0, \quad t \in [-\tau, 0). \end{aligned} \tag{2.3}$$

We then optimize after taking the limit. The objective for each player of (2.2) now becomes

$$J(\alpha) = \mathbb{E}\left[\int_0^T f(X_t, \mu_t, X_{t-\tau}, \mu_{t-\tau}, \alpha_t)dt + g(X_T, \mu_T)\right], \tag{2.4}$$

where we denote $\mu_t := \mathcal{L}(X_t)$ the law of X_t.

3. SOLVING MEAN-FIELD CONTROL PROBLEMS USING DEEP LEARNING TECHNIQUES

Due to the non-Markovian structure, the above mean-field optimal control problem (2.3 and 2.4) is difficult to solve either analytically or numerically. Here we propose two algorithms together with four approaches to tackle the above problem based on deep learning techniques. We would like to use two types of neural networks, one is called the feedforward neural network, and the other one is called Long Short-Term Memory (LSTM) network.

For a feedforward neural network, we first define the set of layers $\mathbb{M}^{\rho}_{d,h}$, for $x \in \mathbb{R}^d$, as

$$\mathbb{M}^{\rho}_{d,h} := \{M : \mathbb{R}^d \to \mathbb{R}^h | M(x) = \rho(Ax + b), A \in \mathbb{R}^{h\times d}, b \in \mathbb{R}^h\}. \tag{3.1}$$

d is called input dimension, h is known as the number of hidden neurons, $A \in \mathbb{R}^{h\times d}$ is the weight matrix, $b \in \mathbb{R}^h$ is the bias vector, and ρ is called the activation function. The following activation functions will be used in this paper, for some $x \in \mathbb{R}$,

$$\rho_{ReLU}(x) := x^+ = \max(0, x); \quad \rho_s(x) := \tfrac{1}{1+e^{-x}};$$
$$\rho_{\tanh}(x) := \tanh(x); \quad \rho_{Id}(x) := x.$$

Then feedforward neural network is defined as a composition of layers, so that the set of feedforward neural networks with l hidden layers we use in this paper is defined as

$$\begin{aligned} \mathbb{NN}^l_{d_1,d_2} &= \{\tilde{M} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} | \tilde{M} = M_l \circ \cdots \circ M_1 \circ M_0, \\ M_0 &\in \mathbb{M}^{\rho_{ReLU}}_{d_1,h_1}, M_l \in \mathbb{M}^{\rho_{Id}}_{h_l,d_2}, M_i \in \mathbb{M}^{\rho_{ReLU}}_{h_i,h_{i+1}}, h_\cdot \in \mathbb{Z}^+, i = 1, \dots, l-1\}. \end{aligned} \tag{3.2}$$

The LSTM network is one of RNN architectures, which are powerful for capturing long-range dependence of the data. It is proposed in Hochreiter and Schmidhuber [14], and it is designed to solve the shrinking gradient effects which basic RNN often suffers from. The LSTM network is a chain of cells. Each LSTM cell is composed of a cell state, which contains information, and three gates, which regulate the flow of information. Mathematically, the rule inside tth cell follows,

$$\begin{aligned} \Gamma_{f_t} &= \rho_s(A_f x_t + U_f a_{t-1} + b_f), \\ \Gamma_{i_t} &= \rho_s(A_i x_t + U_i a_{t-1} + b_i), \\ \Gamma_{o_t} &= \rho_s(A_o x_t + U_o a_{t-1} + b_o), \\ c_t &= \Gamma_{f_t} \odot c_{t-1} + \Gamma_{i_t} \odot \rho_{\tanh}(A_c x_t + U_c a_{t-1} + b_c), \\ a_t &= \Gamma_{o_t} \odot \rho_{\tanh}(c_t), \end{aligned} \tag{3.3}$$

where the operator $\odot$ denotes the Hadamard product. $(\Gamma_{f_t}, \Gamma_{i_t}, \Gamma_{o_t}) \in \mathbb{R}^h \times \mathbb{R}^h \times \mathbb{R}^h$ represents forget gate, input gate and output gate, respectively, h refers the number of hidden neurons. $x_t \in \mathbb{R}^d$ is the input vector with d features. $a_t \in \mathbb{R}^h$ is known as the output vector with initial value $a_0 = 0$, and $c_t \in \mathbb{R}^h$ is known as the cell state with initial value $c_0 = 0$. $A_\cdot \in \mathbb{R}^{h\times d}$ are the weight matrices connecting input and hidden layers, $U_\cdot \in \mathbb{R}^{h\times h}$ are the weight matrix connecting hidden and output layers, and $b \in \mathbb{R}^h$ represents bias vector. The weight matrices and bias vectors are shared through all time steps, and are going to be learned during training process by back-propagation through time (BPTT), which can be implemented in Tensorflow platform. Here we define the set of LSTM network up to time t as

$$\begin{aligned} \mathbb{LSTM}_{d,h,t} = \Big\{ & M : (\mathbb{R}^d)^t \times \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}^h \times \mathbb{R}^h \mid M(x_0, \dots, x_t, \\ & a_0, c_0) = (a_t, c_t), c_t = \Gamma_{f_t} \odot c_{t-1} + \Gamma_{i_t} \odot \rho_{\tanh}(A_c x_t + U_c a_{t-1} + b_c), \\ & a_t = \Gamma_{o_t} \odot \rho_{\tanh}(c_t), a_0 = c_0 = 0 \Big\}, \end{aligned} \tag{3.4}$$

where $\Gamma_{f_\cdot}, \Gamma_{i_\cdot}, \Gamma_{o_\cdot}$ are defined in (3.3).

In particular, we specify the model in a linear-quadratic form, which is inspired by Carmona et al. [2] and Fouque and Zhang [18]. The objective function is defined as

$$J(\alpha) = \mathbb{E}\left[\int_0^T \left(\frac{1}{2}\alpha_t^2 + \frac{c_f}{2}(X_t - m_t)^2\right) dt + \frac{c_t}{2}(X_T - m_T)^2\right], \tag{3.5}$$

subject to

$$\begin{aligned} dX_t &= (\alpha_t - \alpha_{t-\tau})dt + \sigma dW_t, \quad t \in [0, T], \\ X_0 &= x_0, \\ X_t = \alpha_t &= 0, \quad t \in [-\tau, 0), \end{aligned} \tag{3.6}$$

where $\sigma, c_f, c_t > 0$ are given constants, and $m_t := \int_{\mathbb{R}} x d\mu_t(x)$ denotes the mean of X at time t, and $\mu_t := \mathcal{L}(X_t)$. In the following subsections, we solve the above problem numerically using two algorithms together with four approaches. The first two approaches are to directly approximate the control by either a LSTM network or a feedforward neural network, and minimize the objective (3.5) using stochastic gradient descent algorithm. The third and fourth approaches are to introduce the adjoint process associated with (3.6), and approximate the adjoint process and the conditional expectation of adjoint process using neural networks.

3.1. Approximating the Optimal Control Using Neural Networks

We first set $\Delta t = T/N = \tau/D$ for some positive integer N. The time discretization becomes

$$-\tau = t_{-D} \leq t_{-D+1} \leq \cdots \leq t_0 = 0 = t_0 \leq t_1 \leq \cdots \leq t_N \leq T,$$

for $t_i - t_{i-1} = \Delta t$, $i \in \{-D+1, \cdots, 0, \cdots, N-1, N\}$. The discretized SDDE (3.6) according to Euler-Maruyama scheme now reads

$$X_{t_{i+1}} = X_{t_i} + (\alpha_{t_i} - \alpha_{t_{i-D}})\Delta t + \sigma\sqrt{\Delta t}\Delta W_{t_i}, \text{ for } i \in \{0, \cdots, N-1\}, \quad (3.7)$$

where $(\Delta W_{t_i})_{0\leq i\leq N-1}$ are independent, normal distributed sequence of random variables with mean 0 and variance 1.

First, from the definition of open loop control, and due to non-Markovian feature of (3.6), the open-loop optimal control is a function of the path of the Brownian motions up to time t, i.e., $\alpha(t, (W_s)_{0\leq s\leq t})$. We are able to describe this dependency by a LSTM network by parametrizing the control as a function of current time and the discretized increments of Brownian motion path, i.e.,

$$\begin{aligned}(a_{t_i}, c_{t_i}) &= \varphi^1(t_i, (\Delta W_s)_{t_0\leq s\leq t_i}|\Phi^1) \text{ for } \varphi^1 \in \mathbb{LSTM}_{2,h_1,t} \\ \text{and } \Phi^1 &= (A_f, A_i, A_o, A_c, U_f, U_i, U_o, U_c, b_f, b_i, b_o, b_c), \\ \alpha(t_i, (W_s)_{t_0\leq s\leq t_i}) &\approx \psi^1(a_{t_i}|\Psi^1) \text{ for } \psi^1 \in \mathbb{M}^{Id}_{h_1,1} \text{ and } \Psi^1 = (A, b),\end{aligned} \quad (3.8)$$

for some $h_1 \in \mathbb{Z}^+$. We remark that the last dense layer is used to match the desired output dimension.

The second approach is again directly approximate the control but with a feedforward neural network. Due to the special structure of our model, where the mean of dynamic in (3.6) is constant, the mean field control problem coincides with the mean field game problem. In Fouque and Zhang [18], authors solved the associated mean field game problem using infinite dimensional PDE approach, and found that the optimal control is a function of current state and the past of control. Therefore, the feedforward neural network with l layers, which we use to approximate the optimal control, is defined as

$$\begin{aligned}\alpha_{t_i}(X_{t_i}, (\alpha_s)_{t_{i-D}\leq s<t_i}) &\approx \psi^2(X_{t_i}, (\alpha_s)_{t_{i-D}\leq s\leq t_i}|\Psi^2) \\ &\text{for } \psi^2 \in \mathbb{NN}^l_{D+1,1}, \ \Psi^2 = (A_0, b_0, \ldots, A_l, b_l).\end{aligned} \quad (3.9)$$

From Monte Carlo algorithm, and trapezoidal rule, the objective function (3.5) now becomes

$$J = \frac{1}{M}\sum_{j=1}^{M}\Bigg[\Bigg(\frac{1}{2}(\alpha^{(j)}_{t_0})^2 + \frac{c_f}{2}(X^{(j)}_{t_0} - \bar{X}_{t_0})^2 + \sum_{i=1}^{N-1}\left((\alpha^{(j)}_{t_i})^2 + c_f(X^{(j)}_{t_i} - \bar{X}_{t_i})^2\right) + \frac{1}{2}(\alpha^{(j)}_{t_N})^2 + \frac{c_f}{2}(X^{(j)}_{t_N} - \bar{X}_{t_N})^2\Bigg)\frac{\Delta t}{2} + \frac{c_t}{2}(X^{(j)}_{t_N} - \bar{X}_{t_N})^2\Bigg], \quad (3.10)$$

where M denotes the number of realizations and $\bar{X} := \frac{1}{M}\sum_{j=1}^{M} X^{(j)}$ denotes the sample mean. After plugging in the neural network either given by (3.8) or (3.9), the optimization problem becomes to find the best set of parameters either (Φ^1, Ψ^1) or Ψ^2 such that the objective $J(\Phi^1, \Psi^1)$ or $J(\Psi^2)$ is minimized with respect to those parameters.

The algorithm works as follows:

Algorithm 1: Algorithms for solving mean field control problem with delay by directly approximating the optimal control using neural networks

```
Initialization of parameters $\Theta_1 = (\Phi^1, \Psi^1)$ for approach 1 (3.8) or $\Theta_1 = (\Psi^2)$ for approach 2 (3.9);
for each epoch $e = 1, 2, \ldots$ do
    • Generate $\Delta W \in \mathbb{R}^{M\times N}$ for $\Delta W^{(j)}_{t_i} := W_{ji} \sim N(0,1)$, $j \in \{1, \ldots, M\}$ and $i \in \{1, \ldots, N\}$ ;
    • $\alpha^{(j)}_{t_i} = 0$ for $i = \{-D, \ldots, -1\}, \forall j$;
    • $X^{(j)}_0 = \bar{X}_0 = x_0, \alpha^{(j)}_0 \approx \varphi^{(j)}_0(\Theta_e), \forall j$, for some network $\varphi$ given by (3.8) or by (3.9) at $t_0$ with proper inputs;
    • $J = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{2}(\varphi^{(j)}_0)^2\frac{\Delta t}{2}$ ;
    for $(i = 0, \ldots, N-1)$ do
        • $X^{(j)}_{t_{i+1}} = X^{(j)}_{t_i} + (\alpha^{(j)}_{t_i} - \alpha^{(j)}_{t_{i-D}})\Delta t + \sigma\sqrt{\Delta t}\Delta W^{(j)}_{t_i}, \forall j$;
        • $\bar{X}_{t_{i+1}} = \frac{1}{M}\sum_{j=1}^{M} X^{(j)}_{t_{i+1}}$ ;
        • $\alpha^{(j)}_{t_{i+1}} \approx \varphi^{(j)}_{t_{i+1}}(\Theta_e), \forall j$, is given by either (3.8) or (3.9) at $t_{i+1}$ ;
        if $(i = N-1)$ then
            • $J = J + \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{2}(\varphi^{(j)}_{t_{i+1}})^2 + \frac{c_f}{2}(X^{(j)}_{t_{i+1}} - \bar{X}_{t_{i+1}})^2\right)\frac{\Delta t}{2}$;
        else
            • $J = J + \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{2}(\varphi^{(j)}_{t_{i+1}})^2 + \frac{c_f}{2}(X^{(j)}_{t_{i+1}} - \bar{X}_{t_{i+1}})^2\right)\Delta t$;
        end
    end
    • $J = J + \frac{1}{M}\sum_{j=1}^{M}\frac{c_t}{2}(X^{(j)}_{t_N} - \bar{X}_{t_N})^2, \forall j$ ;
    • Compute the gradient $\nabla J(\Theta_e)$ by backpropagation through time ;
    • Stop if $J(\Theta_e)$ converges, or $|\nabla J(\Theta_e)| < \delta$ for some threshold $\delta$, and return $\Theta_e$;
    • Otherwise, update $\Theta_{e+1} = \Theta_e - \eta\nabla J(\Theta_e)$, according to stochastic gradient descent algorithm, for some learning rate $\eta > 0$ small ;
end
```

In the following graphics, we choose $x_0 = 0, c_f = c_t = 1, \sigma = 1, T = 10, \tau = 4, \Delta t = 0.1, M = 4{,}000$. For approach 1, the neural network $\varphi \in \mathbb{NN}$, which is defined in (3.2), is composed of 3 hidden layers with $d_1 = 42, h_1 = 64, h_2 = 128, h_3 = 64, d_2 = 1$. For approach 2, the LSTM network $\varphi \in \mathbb{LSTM}$, which is defined in (3.4), consists of 128 hidden neurons. For a specific representative path, the underlying Brownian motion paths are approximately the same for different approaches. **Figure 1** compares one representative optimal trajectory of the state dynamic and the control, and they coincide. **Figure 2** plots the sample average of optimal trajectory of the state dynamic and the control, which are trajectories of approximately 0, and this is the same as the theoretical mean.

3.2. Approximating the Adjoint Process Using Neural Networks

The third and fourth approaches are based on numerically solving the MV-FABSDE system using LSTM network and feedforward neural networks. From section 4, we derive the adjoint process, and prove the sufficient and necessary parts of stochastic maximum principle. From (4.7), we are able to write the backward stochastic differential equation associated to (3.6) as,

$$dY_t = -c_f(X_t - m_t)dt + Z_t dW_t,\ t \in [0,T], \tag{3.11}$$

with terminal condition $Y_T = c_t(X_T - m_T)$, and $Y_s = 0$ for $s \in (T, T+\tau]$. The optimal control $(\hat{\alpha}_t)_{0\le t\le T}$ can be obtained in terms of the adjoint process Y_t from the maximum principle, and it is given by

$$\hat{\alpha}_t = -Y_t + \mathbb{E}[Y_{t+\tau}|\mathcal{F}_t].$$

From the Euler-Maruyama scheme, the discretized version of (3.6) and (3.11) now reads, for $i = \{0,\dots,N-1\}$,

$$X_{t_{i+1}} = X_{t_i} + (\hat{\alpha}_{t_i} - \hat{\alpha}_{t_{i-D}})\Delta t + \sigma\sqrt{\Delta t}\Delta W_{t_i}, \text{where } \Delta W_{t_i} \sim N(0,1). \tag{3.12}$$

$$Y_{t_{i+1}} = Y_{t_i} - c_f(X_{t_i} - \bar{X}_{t_i})\Delta t + Z_{t_i}\sqrt{\Delta t}\Delta W_{t_i}, \tag{3.13}$$

where we use the sample average $\bar{X}_t = \frac{1}{M}\sum_{j=1}^M X_t^{(j)}$ to approximate the expectation of X_t. In order to solve the above MV-FABSDE system, we need to approximate $(Y_{t_i}, \mathbb{E}[Y_{t_{i+D}}|\mathcal{F}_{t_i}], Z_{t_i})_{0\le t_i\le t_N}$.

The third approach consists of approximating $(Y_{t_i}, \mathbb{E}[Y_{t_{i+D}}|\mathcal{F}_{t_i}], Z_{t_i})_{0\le t_i\le t_N}$ using three LSTM networks as functions of current time and the discretized path of Brownian motions, respectively, i.e.,

$$\begin{aligned}
(a^Y_{t_i}, c^Y_{t_i}) &= \varphi^Y(t_i, (\Delta W_s)_{t_0\le s\le t_i}|\Phi^Y)\\
&\quad \text{for } \varphi^Y \in \mathbb{LSTM}_{2,h_Y,t_i} \text{ and}\\
&\quad \Phi^Y = (A^Y_f, A^Y_i, A^Y_o, A^Y_c, U^Y_f, U^Y_i, U^Y_o, U^Y_c, b^Y_f, b^Y_i, b^Y_o, b^Y_c),\\
Y_{t_i} &\approx \psi^Y(a^Y_{t_i}|\Psi^Y) \text{ for } \psi^Y \in \mathbb{M}^{Id}_{h_Y,1} \text{and} \Psi^Y = (A^Y, b^Y),
\end{aligned}$$

$$\begin{aligned}
(a^{EY}_{t_i}, c^{EY}_{t_i}) &= \varphi^{EY}(t_i, (\Delta W_s)_{t_0\le s\le t_i}|\Phi^{EY})\\
&\quad \text{for } \varphi^{EY} \in \mathbb{LSTM}_{2,h_{EY},t_i} \text{ and}\\
&\quad \Phi^{EY} = (A^{EY}_f, A^{EY}_i, A^{EY}_o, A^{EY}_c, U^{EY}_f, U^{EY}_i, U^{EY}_o, U^{EY}_c, b^{EY}_f, b^{EY}_i, b^{EY}_o, b^{EY}_c),\\
E[Y_{t_{i+D}}|\mathcal{F}_{t_i}] &\approx \psi^{EY}(a^{EY}_{t_i}|\Psi^{EY}) \text{ for } \psi^{EY} \in \mathbb{M}^{Id}_{h_{EY},1} \text{and} \Psi^{EY} = (A^{EY}, b^{EY}),
\end{aligned}$$

$$\begin{aligned}
(a^Z_{t_i}, c^Z_{t_i}) &= \varphi^Z(t_i, (\Delta W_s)_{t_0\le s\le t_i}|\Phi^Z)\\
&\quad \text{for } \varphi^Z \in \mathbb{LSTM}_{2,h_Z,t_i}\\
&\quad \text{and } \Phi^Z = (A^Z_f, A^Z_i, A^Z_o, A^Z_c, U^Z_f, U^Z_i, U^Z_o, U^Z_c, b^Z_f, b^Z_i, b^Z_o, b^Z_c),\\
Z_{t_i} &\approx \psi^Z(a^Z_{t_i}|\Psi^Z) \text{ for } \psi^Z \in \mathbb{M}^{Id}_{h_Z,1} \text{and} \Psi^Z = (A^Z, b^Z),
\end{aligned} \tag{3.14}$$

for some $h_Y, h_{EY}, h_Z \in \mathbb{Z}^+$. Again, the last dense layers are used to match the desired output dimension.

Since approach 3 consists of three neural networks with large number of parameters, which is hard to train in general, we would like to make the following simplification in approach 4 for approximating $(Y_{t_i}, \mathbb{E}[Y_{t_{i+D}}|\mathcal{F}_{t_i}], Z_{t_i})_{t_0\le t_i\le t_N}$ via combination of one LSTM network and three feedforward neural networks. Specifically,

$$\begin{aligned}
(a_{t_i}, c_{t_i}) &= \varphi(t_i, (\Delta W_s)_{t_0\le s\le t_i}|\Phi)\\
&\quad \text{for } \varphi \in \mathbb{LSTM}_{2,h,t_i} \text{ and}\\
&\quad \Phi = (A_f, A_i, A_o, A_c, U_f, U_i, U_o, U_c, b_f, b_i, b_o, b_c),\\
Y_{t_i} &\approx \psi^Y(a_{t_i}|\Psi^Y) \text{ for } \psi^Y \in \mathbb{NN}^l_{h,1} \text{ and}\\
&\quad \Psi^Y = (A^Y_0, b^Y_0, \dots, A^Y_l, b^Y_l), E[Y_{t_{i+D}}|\mathcal{F}_{t_i}]\\
&\approx \psi^{EY}(a_{t_i}|\Psi^{EY}) \text{ for } \psi^{EY} \in \mathbb{NN}^l_{h,1} \text{ and}\\
&\quad \Psi^{EY} = (A^{EY}_0, b^{EY}_0, \dots, A^{EY}_l, b^{EY}_l),\\
Z_{t_i} &\approx \psi^Z(a_{t_i}|\Psi^Z) \text{ for } \psi^Z \in \mathbb{NN}^l_{h,1} \text{ and}\\
&\quad \Psi^Z = (A^Z_0, b^Z_0, \dots, A^Z_l, b^Z_l).
\end{aligned} \tag{3.15}$$

In words, the algorithm works as follows. We first initialize the parameters $(\Theta^Y, \Theta^{EY}, \Theta^Z) = ((\Phi^Y, \Psi^Y), (\Phi^{EY}, \Psi^{EY}), (\Phi^Z, \Psi^Z))$ either in (3.14) or $(\Theta^Y, \Theta^{EY}, \Theta^Z) = ((\Phi, \Psi^Y), \Psi^{EY}), \Psi^Z)$ in (3.15). At time 0, $X_0 = x_0$, $(Y_0, \mathbb{E}[Y_{t_D}|\mathcal{F}_0], Z_0) \approx (\varphi^Y_0(\Theta^Y), \varphi^{EY}_0(\Theta^{EY}), \varphi^Z_0(\Theta^Z))$ for some network $(\varphi^Y, \varphi^{EY}, \varphi^Z)$ given by either (3.14) or (3.15), and $\alpha_0 = -Y_{t_0} + \mathbb{E}[Y_{t_D}|\mathcal{F}_{t_0}]$. Next, we update $X_{t_{i+1}}$ and $Y_{t_{i+1}}$ according to (3.12), and the solution to the backward equation at t_{i+1} is denoted by $\tilde{Y}_{t_{i+1}}$. In the meantime, $Y_{t_{i+1}}$ is also approximated by a neural network. In such case, we refer to $\tilde{Y}_\cdot$ as the label, and $Y_\cdot$ given by the neural network as the prediction. We would like to minimize the mean square error between these two. At time T, Y_{t_N} is also supposed to match $c_t(X_{t_N} - \bar{X}_{t_N})$, from the terminal condition of (3.11). In addition, the conditional expectation $\mathbb{E}[Y_{t_{i+D}}|\mathcal{F}_{t_i}]$ given by a neural network should be the best predictor of $\tilde{Y}_{t_{i+D}}$, which implies that we would like to find the set of parameters Θ^{EY} such that $\mathbb{E}[(\tilde{Y}_{t_{i+D}} - \varphi^{EY}_{t_i}(\Theta^{EY}))^2]$ is minimized for all $t_i \in \{t_0, \dots, t_{N-D}\}$. Therefore, for M samples, we would like to minimize two objective functions L_1 and L_2 defined as

$$\begin{aligned}
L_1(\Theta^Y, \Theta^Z) &= \frac{1}{M}\Bigg[\sum_{j=1}^M\sum_{i=0}^N(\varphi^{Y,(j)}_{t_i} - \tilde{Y}^{(j)}_{t_i})^2\\
&\quad + \sum_{j=1}^M\Big(\varphi^{Y,(j)}_{t_N} - c_t(X^{(j)}_{t_N} - \bar{X}_{t_N})\Big)^2\Bigg],
\end{aligned} \tag{3.16}$$

$$L_2(\Theta^{EY}) = \frac{1}{M}\sum_{j=1}^M\sum_{i=0}^{N-D}\Big(\varphi^{EY,(j)}_{t_i} - \tilde{Y}^{(j)}_{t_{i+D}}\Big)^2.$$

The algorithm works as follows:

Algorithm 2: Algorithms for solving mean field control problem with delay according to MV-FABSDE

```
Initialization of parameters $(\Theta_1^Y, \Theta_1^{EY}, \Theta_1^Z)$ for approach 3 as in (3.14) or approach 4 as in (3.15);
for each epoch $e = 1, 2, \dots$ do
    • Generate $\Delta W \in \mathbb{R}^{M\times N}$ for $\Delta W_{t_i}^{(j)} := \Delta W_{ji} \sim N(0,1)$, $j \in \{1,\dots,M\}$ and $i \in \{1,\dots,N\}$ ;
    • $\alpha_{t_i}^{(j)} = 0$ for $i \in \{-D,\dots,-1\}, \forall j$ ;
    • $X_0^{(j)} = \bar{X}_0 = x_0, \forall j$;  $\left(Y_0^{(j)}, \mathbb{E}[Y_{t_D}^{(j)}|\mathcal{F}_0], Z_0^{(j)}\right) \approx \left(\varphi_0^{Y,(j)}(\Theta_e^Y), \varphi_0^{EY,(j)}(\Theta_e^{EY}), \varphi_0^{Z,(j)}(\Theta_e^Z)\right)$ given by either (3.14) or by (3.15); $\alpha_0^{(j)} \approx -\varphi_0^{Y,(j)} + \varphi_0^{EY,(j)}$ at $t_0$ ;
    • $L_1(\Theta_e^Y, \Theta_e^Z) = 0,\ L_2(\Theta_e^{EY}) = 0$ ;
    for $(i = 0, \dots, N-1)$ do
        • $X_{t_{i+1}}^{(j)} = X_{t_i}^{(j)} + (\alpha_{t_i}^{(j)} - \alpha_{t_{i-D}}^{(j)})\Delta t + \sigma\sqrt{\Delta t}\Delta W_{t_i}^{(j)}, \forall j$;
        • $\bar{X}_{t_{i+1}} = \frac{1}{M}\sum_{j=1}^M X_{t_{i+1}}^{(j)}$ ;
        • $\tilde{Y}_{t_{i+1}}^{(j)} = Y_{t_i}^{(j)} - c_f(X_{t_i}^{(j)} - \bar{X}_{t_i})\Delta t + Z_{t_i}\sqrt{\Delta t}\Delta W_{t_i}^{(j)}, \forall j$ ;
        if $(i \le N - D)$ then
            • $\left(Y_{t_{i+1}}^{(j)}, \mathbb{E}[Y_{t_{i+1+D}}^{(j)}|\mathcal{F}_{t_{i+1}}], Z_{t_{i+1}}^{(j)}\right) \approx \left(\varphi_{t_{i+1}}^{Y,(j)}(\Theta_e^Y), \varphi_{t_{i+1}}^{EY,(j)}(\Theta_e^{EY}), \varphi_{t_{i+1}}^{Z,(j)}(\Theta_e^Z)\right), \forall j$ given by (3.14) or by (3.15) at $t_{i+1}$ ;
            • $L_1 += \frac{1}{M}\sum_{j=1}^M \left(\varphi_{t_{i+1}}^{Y,(j)} - \tilde{Y}_{t_{i+1}}^{(j)}\right)^2$;
        else
            • $\left(Y_{t_{i+1}}^{(j)}, \mathbb{E}[Y_{t_{i+1+D}}^{(j)}|\mathcal{F}_{t_{i+1}}], Z_{t_{i+1}}^{(j)}\right) \approx \left(\varphi_{t_{i+1}}^{Y,(j)}(\Theta_e^Y), 0, \varphi_{t_{i+1}}^{Z,(j)}(\Theta_e^Z)\right), \forall j$ given by (3.14) or by (3.15) at $t_{i+1}$ ;
            • $L_1 += \frac{1}{M}\sum_{j=1}^M \left(\varphi_{t_{i+1}}^{Y,(j)} - \tilde{Y}_{t_{i+1}}^{(j)}\right)^2$;
        end
    end
    for $(i = 0, 1, \dots, N-D)$ do
        • $L_2 += \frac{1}{M}\sum_{j=1}^M \left(\varphi_{t_i}^{EY,(j)} - \tilde{Y}_{t_{i+D}}^{(j)}\right)^2$;
    end
    • $L_1 += \frac{1}{M}\sum_{j=1}^M \left(\varphi_{t_N}^{Y,(j)} - c_t(X_{t_N}^{(j)} - \bar{X}_{t_N})\right)^2$ ;
    • Compute the gradient $\nabla L_1(\Theta^Y, \Theta^Z)$ and $\nabla L_2(\Theta^{EY})$ by backpropagation through time;
    • Stop if $L_1(\Theta^Y, \Theta^Z)$ are close to 0, and $L_2(\Theta^{EY})$ converges, return $(\Theta^Y, \Theta^{EY}, \Theta^Z)$;
    • Otherwise, update $\Theta_{e+1}^Y, \Theta_{e+1}^Z$ and $\Theta_{e+1}^{EY}$ according to SGD algorithm;
end
```

Again, in the following graphics, we choose $x_0 = 0, c_f = c_t = 1, \sigma = 1, T = 10, \tau = 4, \Delta t = 0.1, M = 4,000$. In approach 3, each of the three LSTM networks approximating $Y_{t_i}, \mathbb{E}[Y_{t_i+D}|\mathcal{F}_{t_i}]$ and Z_{t_i} consists of 128 hidden neurons, respectively. In approach 4, the LSTM consists of 128 hidden neurons, and each of the feedforward neural networks has parameters $d_1 = 128, h_1 = 64, h_2 = 128, h_3 = 64, d_2 = 1$. For a specific representative path, the underlying Brownian motion paths are the same for different approaches. **Figure 3** compares one representative optimal trajectory of the state dynamic and the control via two approaches, and they coincide. **Figure 4** plots the sample average of optimal trajectory of the dynamic and the control, which are trajectories of 0, which is the same as the theoretical mean. Comparing to **Figures 1**, **2**, as well as based on numerous experiments, we find that given a path of Brownian motion, the two algorithms would yield similar optimal trajectory of state dynamic and similar path for the optimal control. From **Figure 6**, the loss L_1 as defined in (3.16) becomes approximately 0.02 in 1,000 epochs for both approach 3 and approach 4. This can also be observed from **Figure 5**, since the red dash line and the blue solid line coincide for both left and right graphs. In addition, from the righthand side of **Figure 6**, we observe the loss L_2 as defined in (3.16) converges to 50 after 400 epochs. This is due to the fact that the conditional expectation can be understood as an orthogonal projection. **Figure 7** plots 64 sample paths of the process $(Z_{t_i})_{t_0\le t_i\le t_N}$, which seems to be a deterministic function since σ is constant in this example. Finally, **Figure 8** shows the convergence of the value function as number of epochs increases. Both algorithms arrive approximately at the same optimal value which is around 6 after 400 epochs. This confirms that the out control problem has a unique solution. In section 5, we show that the MV-FASBDE system is uniquely solvable. It is also observable that the first algorithm converges faster than the second one, since it directly paramerizes the control using one neural network, instead of solving the MV-FABSDE system, which uses three neural networks.

3.3. Numerically Solving the Optimal Control Problem With No Delay

Since the algorithms we proposed embrace the case with no delay, we illustrate the comparison between numerical results and the analytical results. By letting $\tau > T$ we obtain $\alpha_{t-\tau} = 0$ in (3.6), and we aim at solving the following linear-quadratic mean-field control problem by minimizing

$$J(\alpha) = \mathbb{E}\left[\int_0^T \left(\frac{1}{2}\alpha_t^2 + \frac{c_f}{2}(X_t - m_t)^2\right)dt + \frac{c_t}{2}(X_T - m_T)^2\right], \tag{3.17}$$

subject to

$$\begin{aligned} dX_t &= \alpha_t dt + \sigma dW_t, \quad t \in [0,T], \\ X_0 &= x_0. \end{aligned} \tag{3.18}$$

Again, from section 4, we find the optimal control

$$\hat{\alpha}_t = -Y_t,$$

where (Y_t, Z_t) is the solution of the following adjoint process,

$$dY_t = -c_f(X_t - m_t)dt + Z_t dW_t. \tag{3.19}$$

Next, we make the ansatz

$$Y_t = \phi_t(X_t - m_t), \tag{3.20}$$

for some deterministic function ϕ_t, satisfying the terminal condition $\phi_T = c_t$. Differentiating the ansatz, the backward equation should satisfy

$$dY_t = (\dot{\phi}_t - \phi_t^2)(X_t - m_t)dt + \phi_t\sigma dW_t, \tag{3.21}$$

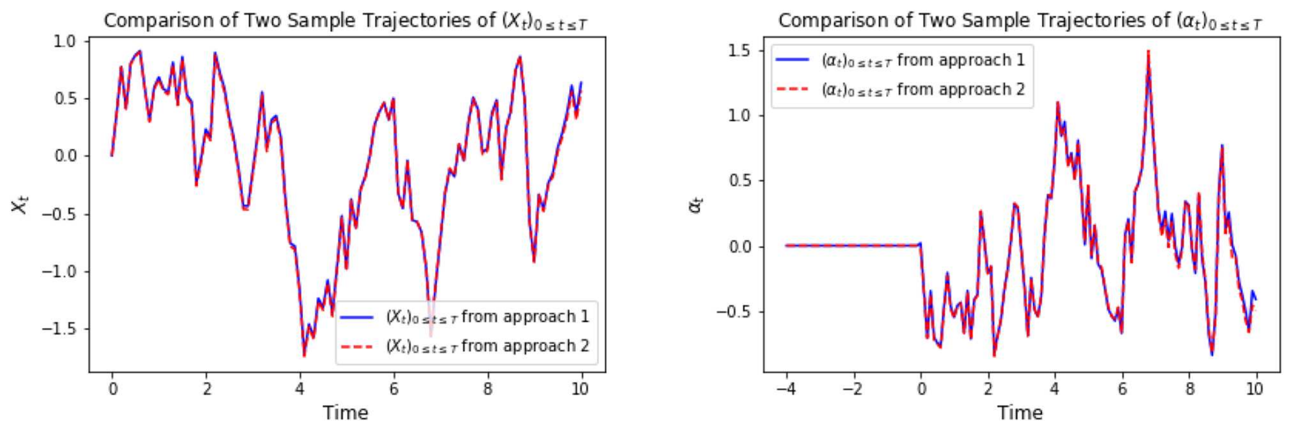


FIGURE 1 | On the left, we compare one representative optimal trajectory of $(X_{t_i})_{t_0\le t_i\le t_N}$. The plot on the right show the comparison of one representative optimal trajectory of $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ between approach 1 and approach 2.

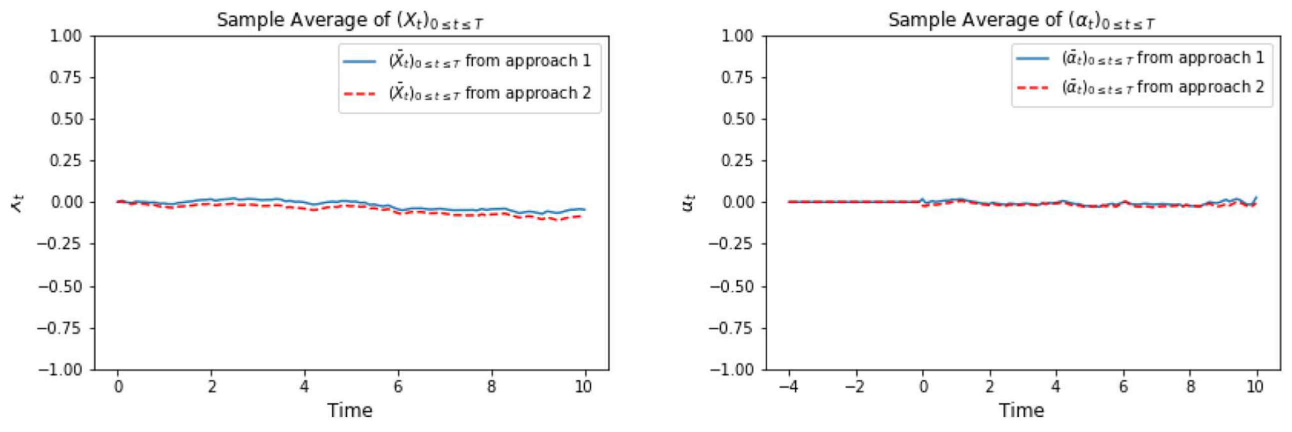


FIGURE 2 | On the left, we compare the sample mean of optimal trajectories of $(X_{t_i})_{t_0\le t_i\le t_N}$. The plot on the right show the comparison of sample mean of trajectories of optimal control $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ between approach 1 and approach 2.

where $\dot{\phi}_t$ denotes the time derivative of ϕ_t. Comparing with (3.19), and identifying the drift and volatility term, ϕ_t must satisfy the scalar Riccati equation,

$$\begin{cases}\dot{\phi}_t = \phi_t^2 - c_f,\\ \phi_T = c_t,\end{cases} \tag{3.22}$$

and the process Z_t should satisfy

$$Z_t = \phi_t \sigma, \tag{3.23}$$

which is deterministic. If we choose $x_0 = 0,\ c_f = c_t = 1, T = 10$, $\phi_t = 1$ solves the Riccati equation (3.22), so that $Z_t = 1,\ \forall t \in [0, T]$, and from (3.20), the optimal control satisfies

$$\hat{\alpha}_t = -(X_t - m_t). \tag{3.24}$$

Numerically, we apply the two deep learning algorithms proposed in the previous section. The first algorithm directly approximates the control. According to the open loop formulation, we set

$$\begin{aligned}(a_{t_i}, c_{t_i}) &= \varphi(t_i, (\Delta W_s)_{t_0\le s\le t_i}|\Phi)\\ &\quad \text{for } \varphi \in \mathbb{LSTM}_{2,h,t} \text{ and}\\ &\quad \Phi = (A_f, A_i, A_o, A_c, U_f, U_i, U_o, U_c, b_f, b_i, b_o, b_c),\\ \alpha(t_i, (W_s)_{t_0\le s\le t_i}) &\approx \psi(a_{t_i}|\Psi) \text{ for } \psi \in \mathbb{M}^{Id}_{h,1} \text{ and } \Psi = (A, b),\end{aligned} \tag{3.25}$$

for some $h \in \mathbb{Z}^+$. We remark that the last dense layer is used to match the desired output dimension. The second algorithm numerically solves the forward backward system as in (3.18) and (3.19). From the ansatz (3.20) and the Markovian feature, we approximate $(Y_t, Z_t)_{0\le t\le T}$ using two feedforward neural networks, i.e.,

$$\begin{aligned}Y_{t_i} &\approx \psi^1(t_i, X_{t_i}|\Psi^1) \text{ for } \psi^1 \in \mathbb{NN}^l_{2,1},\ \Psi^1 = (A^1_0, b^1_0, \dots, A^1_l, b^1_l);\\ Z_{t_i} &\approx \psi^2(t_i, X_{t_i}|\Psi^2) \text{ for } \psi^2 \in \mathbb{NN}^l_{2,1},\ \Psi^2 = (A^2_0, b^2_0, \dots, A^2_l, b^2_l).\end{aligned} \tag{3.26}$$

Figure 9 shows the representative optimal trajectory of $(X_{t_i} - \bar{X}_{t_i})_{t_0\le t_i\le t_N}$ and $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ from both algorithms, which are

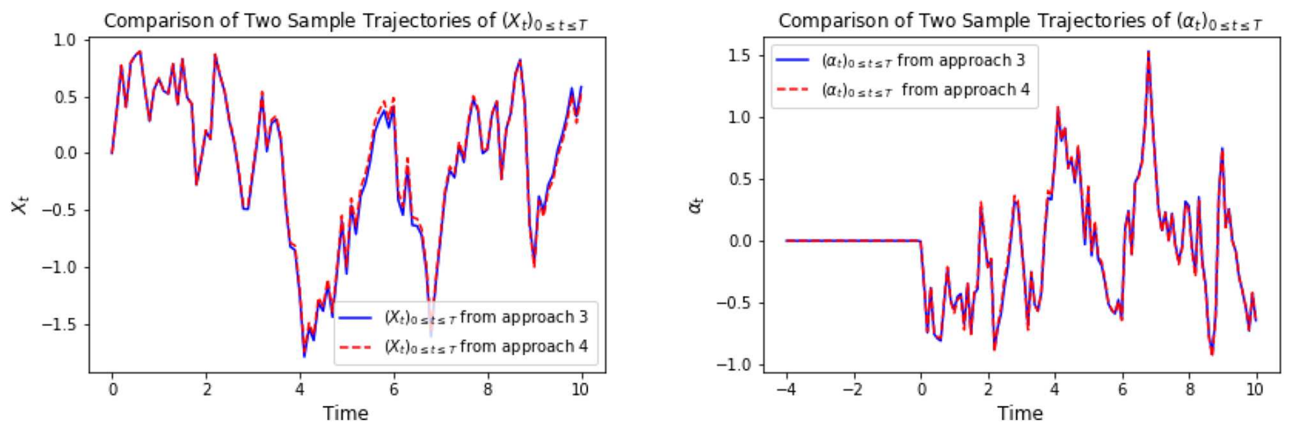


FIGURE 3 | On the left, we compare one representative optimal trajectory of $(X_{t_i})_{t_0\le t_i\le t_N}$. The plot on the right shows the comparison of one representative optimal trajectory of $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ between approach 3 and approach 4.

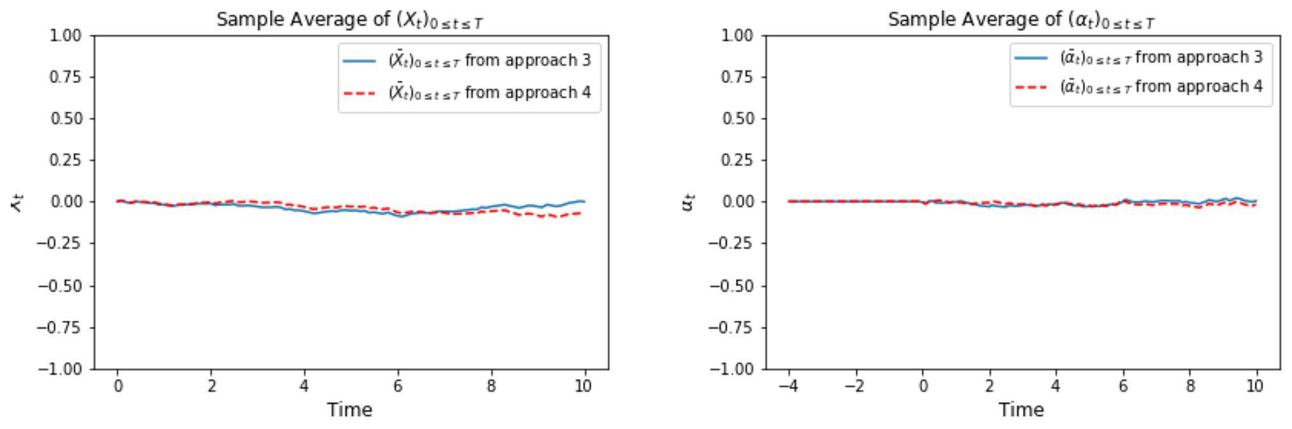


FIGURE 4 | On the left, we compare the sample mean of optimal trajectories of $(X_{t_i})_{t_0\le t_i\le t_N}$. The plot on the right shows the comparison of sample mean of trajectories of optimal control $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ between approach 3 and approach 4.

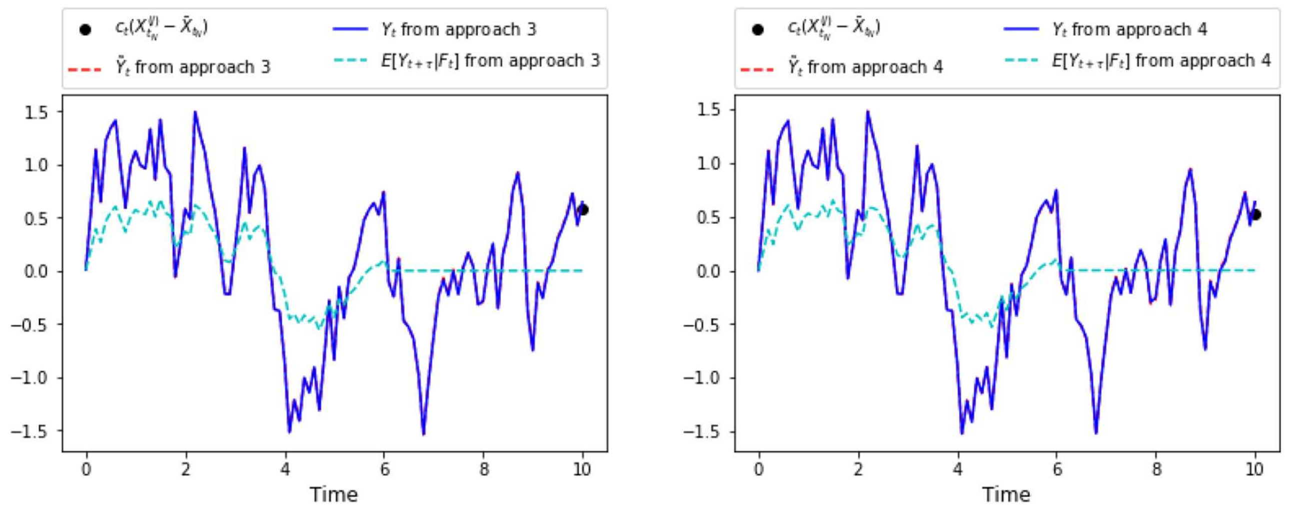


FIGURE 5 | Plots of representative trajectories of $[(Y_{t_i})_{t_0\le t_i\le t_N}, (\tilde{Y}_{t_i})_{t_0\le t_i\le t_N}, (\mathbb{E}[Y_{t_{i+D}}|\mathcal{F}_{t_i}])_{t_0\le t_i\le t_N}]$, from approach 3 (one the left) and from approach 4 (on the right).

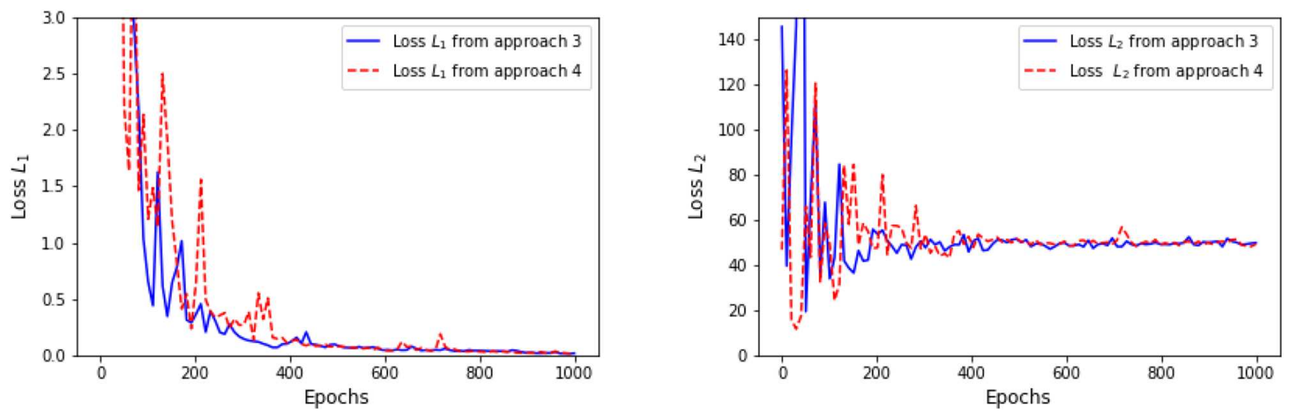


FIGURE 6 | Convergence of loss L_1 (on the left) and convergence of loss L_2 (on the right) as defined in (3.16) from approach 3 and approach 4.

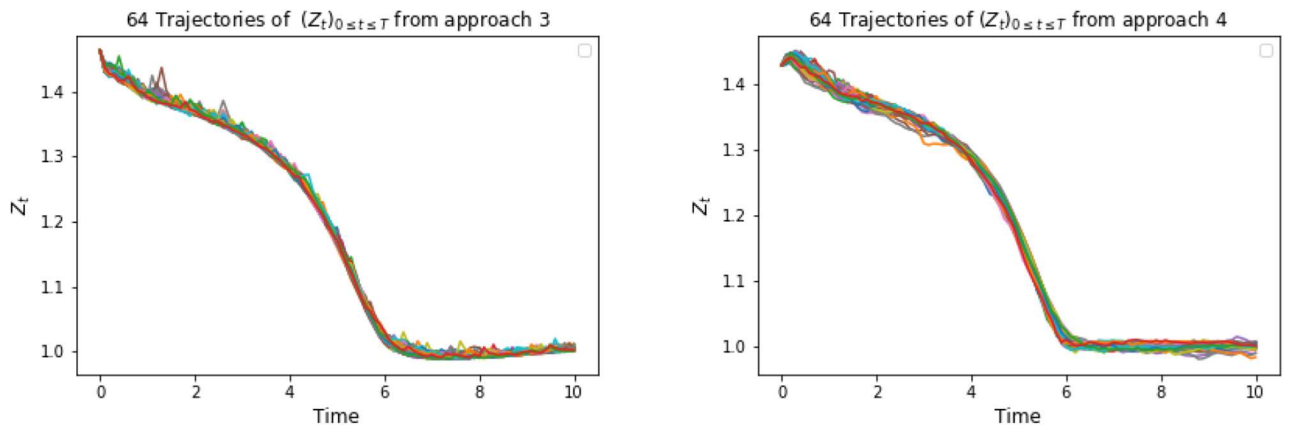


FIGURE 7 | 64 trajectories of $(Z_{t_i})_{t_0\le t_i\le t_N}$ based on approach 3 (on the left) and approach 4 (on the right).

exactly the same. The symmetry feature can be seen from the computation (3.24). On the left of **Figure 10** confirms the mean of the processes $(X_{t_i} - \bar{X}_{t_i})_{t_0\le t_i\le t_N}$ and $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ are 0 from both algorithms. The right picture of **Figure 10** plots the data points of x against α, and we can observe that the optimal control α is linear in x as a result of (3.24), and the slope tends to be -1, since $\phi = 1$ solves the scalar Riccati equation (3.22). Finally, **Figure 11** plots representative optimal trajectories of the solution to the adjoint equations (Y_t, Z_t). On the left, we observe that the adjoint process $(Y_t)_{0\le t\le T}$ matches the terminal condition, and on the right, $(Z_t)_{0\le t\le T}$ appears to be a deterministic process of value 1, and this matches the result we compute previously.

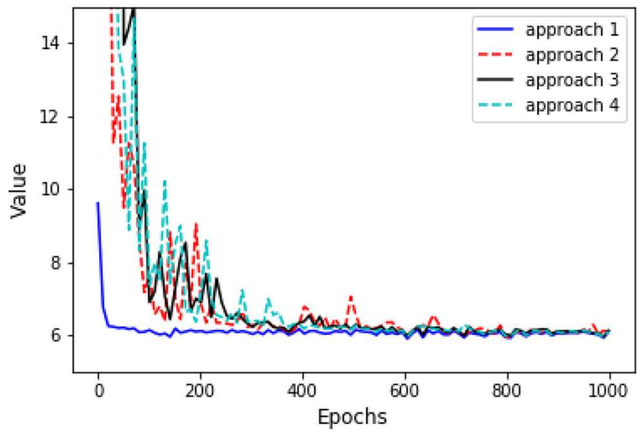


FIGURE 8 | Comparison convergence of objective values as in (3.10) among four approaches.

4. STOCHASTIC MAXIMUM PRINCIPLE FOR OPTIMALITY

In this section, we derive the adjoint equation associated to our mean field stochastic control problem (2.3) and (2.4). The necessary and sufficient parts of stochastic maximum principle

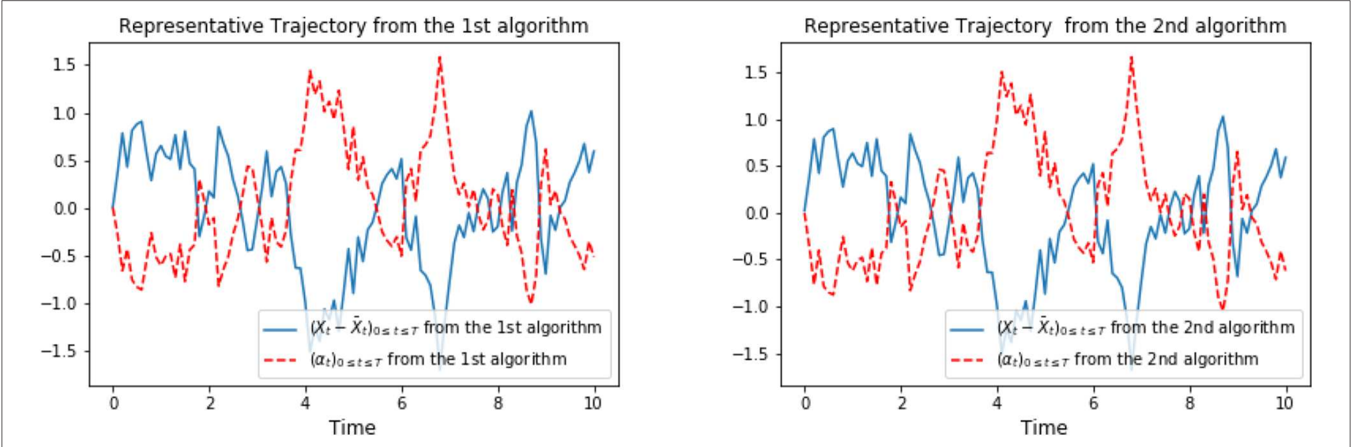


FIGURE 9 | Representative optimal trajectory of $(X_{t_i} - \bar{X}_{t_i})_{t_0\le t_i\le t_N}$ and $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ from algorithm 1 (on the left) and from algorithm 2 (on the right).

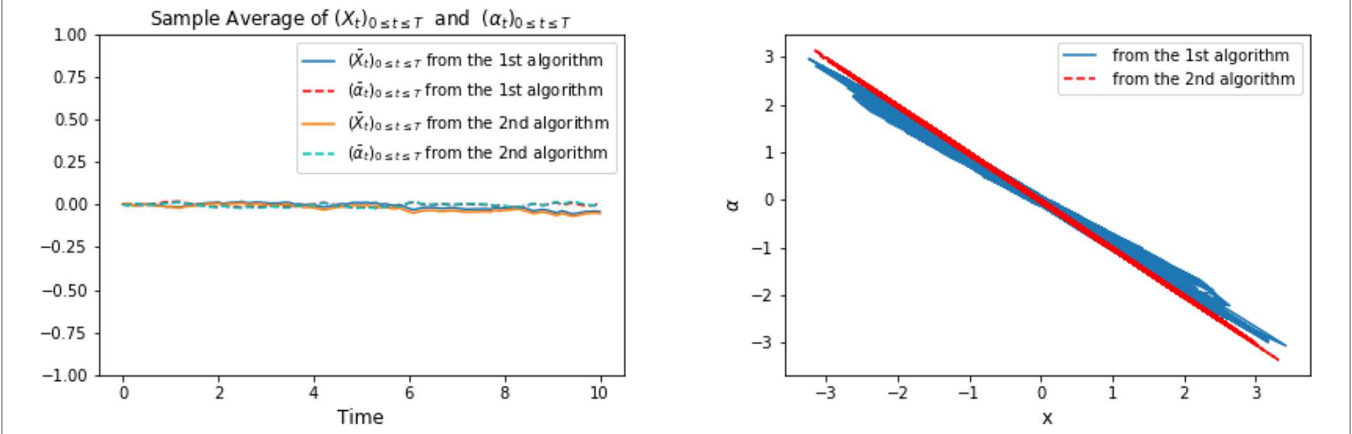


FIGURE 10 | The picture on the left plots the sample averages of $(X_{t_i})_{t_0\le t_i\le t_N}$ and $(\alpha_{t_i})_{t_0\le t_i\le t_N}$ for both algorithm 1 and 2; The plot on the right shows the points of x against α for both algorithms.

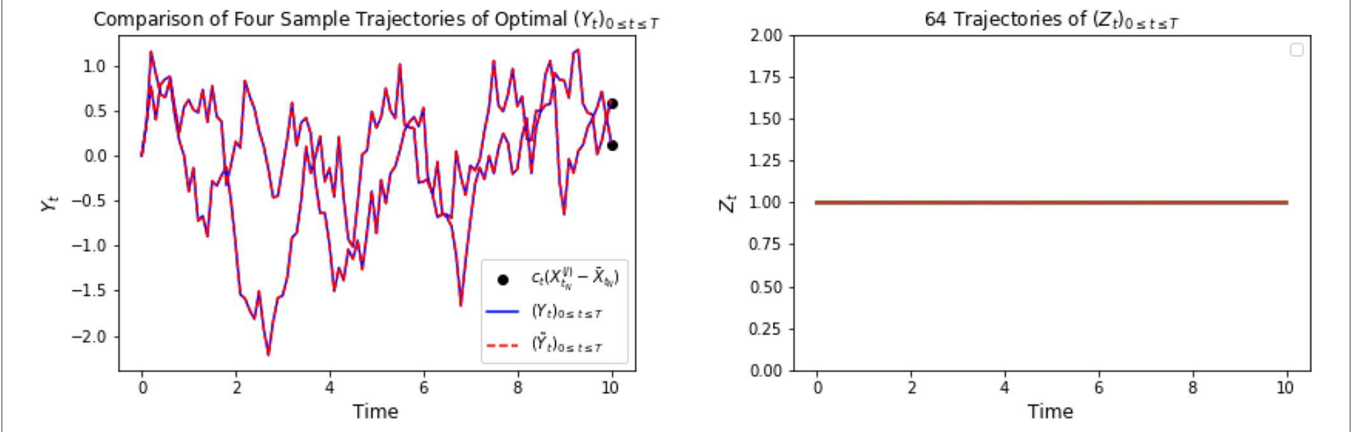


FIGURE 11 | The plot on the left shows representative trajectories of $[(Y_{t_i})_{t_0\le t_i\le t_N}, (\tilde{Y}_{t_i})_{t_0\le t_i\le t_N}]$. The picture on the right plots 64 trajectories of $(Z_{t_i})_{t_0\le t_i\le t_N}$.

have been proved for optimality. We assume

(H4.1) b, σ are differentiable with respect to $(X_t, \mu_t, X_{t-\tau}, \mu_{t-\tau}, \alpha_t, \alpha_{t-\tau})$; f is differentiable with respect to $(X_t, \mu_t, X_{t-\tau}, \mu_{t-\tau}, \alpha)$; g is differentiable with respect to (X_T, μ_T). Their derivatives are bounded.

In order to simplify our notations, let $\theta_t = (X_t, \mu_t, \alpha_t)$. For $0 < \epsilon < 1$, we denote α^ϵ the admissible control defined by

$$\alpha_t^\epsilon := \alpha_t + \epsilon(\beta_t - \alpha_t) := \alpha_t + \epsilon \Delta\alpha_t,$$

for any $(\alpha)_{0\le t\le T}$ and $(\beta)_{0\le t\le T} \in \mathbb{A}$. $X_t^\epsilon := X_t^{\alpha^\epsilon}$ is the corresponding controlled process. We define

$$\nabla X_t := \lim_{\epsilon\to 0} \frac{X_t^\epsilon - X_t^\alpha}{\epsilon}$$

to be the variation process, which should follow the following dynamic for $t \in (0, T]$,

$$\begin{aligned} d\nabla X_t =& \Big[\partial_x b(t,\theta_t,\theta_{t-\tau})\nabla X_t + \partial_{x_\tau} b(t,\theta_t,\theta_{t-\tau})\nabla X_{t-\tau} \\ &+ \tilde{\mathbb{E}}[\partial_\mu b(t,\theta_t,\theta_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] + \tilde{\mathbb{E}}[\partial_{\mu_\tau} b(t,\theta_t,\theta_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] \\ &+ \partial_\alpha b(t,\theta_t,\theta_{t-\tau})\Delta\alpha_t + \partial_{\alpha_\tau} b(t,\theta_t,\theta_{t-\tau})\Delta\alpha_{t-\tau}\Big]dt \\ &+ \Big[\partial_x \sigma(t,\theta_t,\theta_{t-\tau})\nabla X_t + \partial_{x_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\nabla X_{t-\tau} \\ &+ \tilde{\mathbb{E}}[\partial_\mu \sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] + \tilde{\mathbb{E}}[\partial_{\mu_\tau}\sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] \\ &+ \partial_\alpha\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_t + \partial_{\alpha_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_{t-\tau}\Big]dW_t \end{aligned} \quad (4.1)$$

with initial condition $\nabla X_t = \Delta\alpha_t = 0$, $t \in [-\tau, 0)$. $(\tilde{X}_t, \nabla\tilde{X}_t)$ is a copy of $(X_t, \nabla X_t)$ defined on $(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{P}})$, where we apply differential calculus on functions of measure, see Carmona and Delarue [3] for detail. $\partial_x b, \partial_{x_\tau} b, \partial_\mu b, \partial_{\mu_\tau} b, \partial_\alpha b, \partial_{\alpha_\tau} b$ are derivatives of b with respect to $(X_t, X_{t-\tau}, \mu_t, \mu_{t-\tau}, \alpha_t, \alpha_{t-\tau})$, respectively, and we use the same notation for $\partial_\cdot\sigma$.

In the meantime, the Gateaux derivative of functional $\alpha \to J(\alpha)$ is given by

$$\begin{aligned} &\lim_{\epsilon\to 0}\frac{J(\alpha^\epsilon) - J(\alpha)}{\epsilon} \\ =&\mathbb{E}\left[\partial_x g(X_T,\mu_T)\nabla X_T + \tilde{\mathbb{E}}[\partial_\mu g(X_T,\mu_T)(\tilde{X}_T)\nabla\tilde{X}_T]\right] \\ &+ \mathbb{E}\int_0^T \Big[\partial_x f(\theta_t, X_{t-\tau},\mu_{t-\tau})\nabla X_t + \tilde{\mathbb{E}}[\partial_\mu f(\theta_t, X_{t-\tau},\mu_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] \\ &+ \partial_{x_\tau} f(\theta_t, X_{t-\tau},\mu_{t-\tau})\nabla X_{t-\tau} + \tilde{\mathbb{E}}[\partial_{\mu_\tau} f(\theta_t, X_{t-\tau},\mu_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] \\ &+ \partial_\alpha f(\theta_t, X_{t-\tau},\mu_{t-\tau})(\Delta\alpha_t)\Big]dt \end{aligned} \quad (4.2)$$

In order to determine the adjoint backward equation of $(Y_t, Z_t)_{0\le t\le T}$ associated to (2.3), we assume it is of the following form:

$$\begin{aligned} dY_t &= -\varphi_t dt + Z_t dW_t, \quad t \in [0,T], \\ Y_T &= \partial_x g(X_T,\mu_T) + \tilde{\mathbb{E}}[\partial_\mu g(X_T,\mu_T)(\tilde{X}_T)], \\ Y_t &= Z_t = 0, \quad t \in (T, T+\tau] \end{aligned} \quad (4.3)$$

Next, we apply integration by part to ∇X_t and Y_t. It yields

$$\begin{aligned} d(\nabla X_t Y_t) = & Y_t\Big[\partial_x b(t,\theta_t,\theta_{t-\tau})\nabla X_t + \partial_{x_\tau} b(t,\theta_t,\theta_{t-\tau})\nabla X_{t-\tau} \\ &+ \tilde{\mathbb{E}}[\partial_\mu b(t,\theta_t,\theta_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] + \tilde{\mathbb{E}}[\partial_{\mu_\tau} b(t,\theta_t,\theta_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] \\ &+ \partial_\alpha b(t,\theta_t,\theta_{t-\tau})\Delta\alpha_t + \partial_{\alpha_\tau} b(t,\theta_t,\theta_{t-\tau})\Delta\alpha_{t-\tau}\Big]dt \\ &+ Y_t\Big[\partial_x\sigma(t,\theta_t,\theta_{t-\tau}) + \partial_{x_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\nabla X_{t-\tau} \\ &+ \tilde{\mathbb{E}}[\partial_\mu\sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] + \tilde{\mathbb{E}}[\partial_{\mu_\tau}\sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] \\ &+ \partial_\alpha\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_t + \partial_{\alpha_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_{t-\tau}\Big]dW_t - \varphi_t\nabla X_t dt \\ &+ \nabla X_t Z_t dW_t + Z_t\Big[\partial_x\sigma(t,\theta_t,\theta_{t-\tau})\nabla X_t \\ &+ \partial_{x_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\nabla X_{t-\tau} + \tilde{\mathbb{E}}[\partial_\mu\sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] \\ &+ \tilde{\mathbb{E}}[\partial_{\mu_\tau}\sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] + \partial_\alpha\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_t \\ &+ \partial_{\alpha_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_{t-\tau}\Big]dt \end{aligned}$$

We integrate from 0 to T, and take expectation to get

$$\begin{aligned} &\mathbb{E}[\nabla X_T Y_T] \\ &= \mathbb{E}\int_0^T Y_t\Big[\partial_x b(t,\theta_t,\theta_{t-\tau})\nabla X_t + \partial_{x_\tau} b(t,\theta_t,\theta_{t-\tau})\nabla X_{t-\tau} \\ &+ \tilde{\mathbb{E}}[\partial_\mu b(t,\theta_t,\theta_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] + \tilde{\mathbb{E}}[\partial_{\mu_\tau} b(t,\theta_t,\theta_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] \\ &+ \partial_\alpha b(t,\theta_t,\theta_{t-\tau})\Delta\alpha_\tau + \partial_{\alpha_\tau} b(t,\theta_t,\theta_{t-\tau})\Delta\alpha_{t-\tau}\Big]dt \\ &- \mathbb{E}\int_0^T \varphi_t\nabla X_t dt + \mathbb{E}\int_0^T Z_t\Big[\partial_x\sigma(t,\theta_t,\theta_{t-\tau})\nabla X_t \\ &+ \partial_{x_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\nabla X_{t-\tau} + \tilde{\mathbb{E}}[\partial_\mu\sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_t)\nabla\tilde{X}_t] \\ &+ \tilde{\mathbb{E}}[\partial_{\mu_\tau}\sigma(t,\theta_t,\theta_{t-\tau})(\tilde{X}_{t-\tau})\nabla\tilde{X}_{t-\tau}] + \partial_\alpha\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_t \\ &+ \partial_{\alpha_\tau}\sigma(t,\theta_t,\theta_{t-\tau})\Delta\alpha_{t-\tau}\Big]dt \end{aligned} \quad (4.4)$$

Using the fact that $Y_t = Z_t = 0$ for $t \in (T, T+\tau]$, we are able to make a change of time, and by Fubini's theorem, so that (4.4) becomes

$$\begin{aligned} \mathbb{E}[\nabla X_T Y_T] =& \mathbb{E}\int_0^T \Big(Y_t\partial_x b(t,\theta_t,\theta_{t-\tau}) + Y_{t+\tau}\partial_{x_\tau} b(t+\tau,\theta_{t+\tau},\theta_t) \\ &+ \tilde{\mathbb{E}}[\partial_\mu b(t,\tilde{\theta}_t,\tilde{\theta}_{t-\tau})(X_t)] \\ &+ \tilde{\mathbb{E}}[\partial_{\mu_\tau} b(t+\tau,\tilde{\theta}_{t+\tau},\tilde{\theta}_t)(X_t)\Big)\nabla X_t dt \\ &+ \mathbb{E}\int_0^T\Big(\partial_\alpha b(t,\theta_t,\theta_{t-\tau}) + \partial_{\alpha_\tau} b(t+\tau,\theta_{t+\tau},\theta_t)\Big)\Delta\alpha_t dt \\ &- \mathbb{E}\Big[\int_0^T \varphi_t\nabla X_t\Big]dt + \mathbb{E}\int_0^T\Big(Z_t\partial_x\sigma(t,\theta_t,\theta_{t-\tau}) \\ &+ Z_{t+\tau}\partial_{x_\tau}\sigma(t+\tau,\theta_{t+\tau},\theta_t) + \tilde{\mathbb{E}}[\partial_\mu\sigma(t,\tilde{\theta}_t,\tilde{\theta}_{t-\tau})(X_t)] \\ &+ \tilde{\mathbb{E}}[\partial_{\mu_\tau}\sigma(t+\tau,\tilde{\theta}_{t+\tau},\tilde{\theta}_t)(X_t)\Big)\nabla X_t dt \\ &+ \mathbb{E}\int_0^T\Big(\partial_\alpha\sigma(t,\theta_t,\theta_{t-\tau}) + \partial_{\alpha_\tau}\sigma(t+\tau,\theta_{t+\tau},\theta_t)\Big)\Delta\alpha_t dt \end{aligned} \quad (4.5)$$

Now we define the Hamiltonian H for $(t,x,\mu,x_\tau,\mu_\tau,y,z,\alpha,\alpha_\tau) \in [0,T]\times\mathbb{R}\times\mathcal{P}_2(\mathbb{R})\times\mathbb{R}\times\mathcal{P}_2(\mathbb{R})\times\mathbb{R}\times\mathbb{R}\times A\times A$ as

$$\begin{aligned}&H(t,x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau,y,z)\\ &\quad= b(t,x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau)y+\sigma(t,x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau)z\\ &\qquad+f(t,x,\mu,x_\tau,\mu_\tau,\alpha)\end{aligned} \tag{4.6}$$

Using the terminal condition of Y_T, and plugging (4.5) into (4.2), and setting the integrand containing ∇X_t to zero, we are able to obtain the adjoint equation is of the following form

$$\begin{aligned}dY_t=&-\Big\{\partial_x H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\alpha_t,\alpha_{t-\tau},Y_t,Z_t)\\ &+\tilde{\mathbb{E}}[\partial_\mu H(t,\tilde{X}_t,\mu_t,\tilde{X}_{t-\tau},\mu_{t-\tau},\tilde{\alpha}_t,\tilde{\alpha}_{t-\tau},\tilde{Y}_t,\tilde{Z}_t)(X_t)]\\ &+\mathbb{E}[\partial_{x_\tau}H(t+\tau,X_{t+\tau},\mu_{t+\tau},X_t,\mu_t,\alpha_{t+\tau},\alpha_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\\ &+\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}H(t+\tau,\tilde{X}_{t+\tau},\mu_{t+\tau},\tilde{X}_t,\mu_t,\tilde{\alpha}_t,\tilde{\alpha}_{t-\tau},\tilde{Y}_{t+\tau},\tilde{Z}_{t+\tau})\\ &(X_{t+\tau})]|\mathcal{F}_t]\Big\}dt+Z_t dW_t\\ Y_T=&\partial_x g(X_T,\mu_T)+\tilde{\mathbb{E}}[\partial_\mu g(\tilde{X}_T,\mu_T)(X_T)].\end{aligned} \tag{4.7}$$

Theorem 4.1. *Let $(\alpha_t)_{0\le t\le T}\in\mathbb{A}$ be optimal, $(X_t)_{0\le t\le T}$ be the associated controlled state, and $(Y_t,Z_t)_{0\le t\le T}$ be the associated adjoint processes defined in (4.7). For any $\beta\in A$, and $t\in[0,T]$,*

$$\begin{aligned}&\Big(\partial_\alpha H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\alpha_t,\alpha_{t-\tau},Y_t,Z_t)\\ &+\mathbb{E}[\partial_{\alpha_\tau}H(t+\tau,X_{t+\tau},\mu_{t+\tau},X_t,\mu_t,\alpha_{t+\tau},\alpha_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\Big)\\ &\qquad(\beta-\alpha_t)\ge 0 \text{ a.e.}\end{aligned} \tag{4.8}$$

Proof: Given any $(\beta_t)_{0\le t\le T}\in\mathbb{A}$, we perturbate α_t by $\epsilon(\beta_t-\alpha_t)$ and we define $\alpha_t^\epsilon:=\alpha_t+\epsilon(\beta_t-\alpha_t)$ for $0\le\epsilon\le 1$. Using the adjoint process (4.7), and apply integration by parts formula to $(\nabla X_t Y_t)$. Then plug the result into (4.2), and the Hamiltonian H is defined in (4.6). Also, since α is optimal, we have

$$\begin{aligned}0&\le\lim_{\epsilon\to 0}\frac{J(\alpha^\epsilon)-J(\alpha)}{\epsilon}\\ &=\mathbb{E}\int_0^T\Big([\partial_\alpha H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)+\\ &\quad\mathbb{E}[\partial_{\alpha_\tau}H(t+\tau,\theta_{t+\tau},\theta_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\Big)(\beta_t-\alpha_t)dt\end{aligned} \tag{4.9}$$

Now, let $C\in\mathcal{F}_t$ be an arbitrary progressively measurable set, and denote C' the complement of C. We choose β_t to be $\beta_t:=\beta\mathbb{1}_C+\alpha_t\mathbb{1}_{C'}$ for any given $\beta\in A$. Then,

$$\begin{aligned}&\mathbb{E}\int_0^T\Big([\partial_\alpha H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)\\ &+\mathbb{E}[\partial_{\alpha_\tau}H(t+\tau,\theta_{t+\tau},\theta_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\Big)(\beta_t-\alpha_t)\mathbb{1}_C dt\ge 0,\end{aligned} \tag{4.10}$$

which implies,

$$\begin{aligned}&(\partial_\alpha H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)\\ &+\mathbb{E}[\partial_{\alpha_\tau}H(t+\tau,\theta_{t+\tau},\theta_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t])(\beta-\alpha_t)\ge 0.\text{ a.e.}\end{aligned} \tag{4.11}$$

□

Remark 4.2. When we further assume that H is convex in $(\alpha_t,\alpha_{t-\tau})$, then for any $\beta,\beta_\tau\in A$ in Theorem 4.1, we have

$$\begin{aligned}&H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\alpha_t,\alpha_{t-\tau},Y_t,Z_t)\\ &\le H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\beta,\beta_\tau,Y_t,Z_t),\text{ a.e.}\end{aligned}$$

as a direct consequence of (4.8).

Theorem 4.3. *Let $(\alpha_t)_{0\le t\le T}\in\mathbb{A}$ be an admissible control. Let $(X_t)_{0\le t\le T}$ be the controlled state, and $(Y_t,Z_t)_{0\le t\le T}$ be the corresponding adjoint processes. We further assume that for each t, given Y_t and Z_t, the function $(x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau)\to H(t,x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau,Y_t,Z_t)$, and the function $(x,\mu)\to g(x,\mu)$ are convex. If*

$$\begin{aligned}&H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\alpha_t,\alpha_{t-\tau},Y_t,Z_t)\\ &=\inf_{\alpha'\in\mathbb{A}}H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\alpha'_t,\alpha'_{t-\tau},Y_t,Z_t),\end{aligned} \tag{4.12}$$

for all t, then $(\alpha_t)_{0\le t\le T}$ is an optimal control.

Proof: Let $(\alpha'_t)_{0\le t\le T}\in\mathbb{A}$ be a admissible control, and let $(X'_t)_{0\le t\le T}=(X_t^{\alpha'})_{0\le t\le T}$ be the corresponding controlled state. From the definition of the objective function as in (2.4), we first use convexity of g, and the terminal condition of the adjoint process Y_t in (4.7), then use the fact that H is convex, and because of (4.12), we have the following

$$\begin{aligned}&J(\alpha)-J(\alpha')=\mathbb{E}[g(X_T,\mu_T)-g(X'_T,\mu'_T)]\\ &\quad+\mathbb{E}\int_0^T[f(t,\theta_t,X_{t-\tau},\alpha_{t-\tau})-f(t,\theta'_t,X'_{t-\tau},\alpha'_{t-\tau})]dt\\ \le\ &\mathbb{E}[\partial_x g(X_T,\mu_T)(X_T-X'_T)+\tilde{\mathbb{E}}[\partial_\mu g(X_T,\mu_T)(\tilde{X}_T)(\tilde{X}_T-\tilde{X}'_T)]]\\ +\ &\mathbb{E}\int_0^T[f(t,\theta_t,X_{t-\tau},\alpha_{t-\tau})-f(t,\theta'_t,X'_{t-\tau},\alpha'_{t-\tau})]dt\\ =\ &\mathbb{E}[(\partial_x g(X_T,\mu_T)+\tilde{E}[\partial_\mu g(\tilde{X}_T,\mu_T)(X_T)])(X_T-X'_T)]\\ +\ &\mathbb{E}\int_0^T[f(t,\theta_t,X_{t-\tau},\alpha_{t-\tau})-f(t,\theta'_t,X'_{t-\tau},\alpha'_{t-\tau})]dt\\ =\ &\mathbb{E}[Y_T(X_T-X'_T)]+\mathbb{E}\int_0^T[f(t,\theta_t,X_{t-\tau},\alpha_{t-\tau})\\ -\ &f(t,\theta'_t,X'_{t-\tau},\alpha'_{t-\tau})]dt=\mathbb{E}\int_0^T\Big[\big(b(t,\theta_t,\theta_{t-\tau})-b(t,\theta'_t,\theta'_{t-\tau})\big)Y_t\\ +\ &\big(\sigma(t,\theta_t,\theta_{t-\tau})-\sigma(t,\theta'_t,\theta'_{t-\tau})\big)Z_t\Big]dt-\mathbb{E}\int_0^T\Big[\big(\partial_x H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)\\ &\quad+\tilde{\mathbb{E}}[\partial_\mu H(t,\tilde{\theta}_t,\tilde{\theta}_{t-\tau},\tilde{Y}_t,\tilde{Z}_t)(X_t)]\big)(X_t-X'_t)\Big]dt\\ -\ &\mathbb{E}\int_0^T\Big[\Big(\mathbb{E}[\partial_{x_\tau}H(t+\tau,\theta_{t+\tau},\theta_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\\ +\ &\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}H(t+\tau,\tilde{\theta}_{t+\tau},\tilde{\theta}_t,\tilde{Y}_{t+\tau},\tilde{Z}_{t+\tau})(X_t)]|\mathcal{F}_t]\Big)(X_t-X'_t)\Big]dt\\ +\ &\mathbb{E}\int_0^T\Big[H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)-H(t,\theta'_t,\theta'_{t-\tau},Y_t,Z_t)\Big]dt\end{aligned}$$

$$
\begin{aligned}
&+\ \mathbb{E}\int_0^T\Big[\big(b(t,\theta_t,\theta_{t-\tau})-b(t,\theta_t,\theta_{t-\tau})\big)Y_t+\big(\sigma(t,\theta_t,\theta_{t-\tau})\\
&\quad-\sigma(t,\theta_t,\theta_{t-\tau})\big)Z_t\Big]dt\le-\mathbb{E}\int_0^T\Big[\partial_x H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)(X_t-X'_t)\\
&\quad+\tilde{\mathbb{E}}[\partial_\mu H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)(\tilde{X}_t)(\tilde{X}_t-\tilde{X}_t')\Big]dt\\
&-\ \mathbb{E}\int_0^T\Big[\partial_{x_\tau}H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)(X_{t-\tau}-X'_{t-\tau})\\
&+\ \tilde{\mathbb{E}}[\partial_{\mu_\tau}H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)(\tilde{X}_{t-\tau})(\tilde{X}_{t-\tau}-\tilde{X}'_{t-\tau})]\Big]dt\\
&+\ \mathbb{E}\int_0^T\Big[H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)-H(t,\theta'_t,\theta'_{t-\tau},Y_t,Z_t)\Big]dt\\
&\le\ \mathbb{E}\int_0^T\Big[\partial_\alpha H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)(\alpha_t-\alpha'_t)+\partial_{\alpha_\tau}H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)\\
&\quad(\alpha_{t-\tau}-\alpha'_{t-\tau})\Big]dt\le\mathbb{E}\int_0^T\Big(\partial_\alpha H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)\\
&+\ \mathbb{E}[\partial_{\alpha_\tau}H(t+\tau,\theta_{t+\tau},\theta_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\Big)(\alpha_t-\alpha'_t)dt\le 0.
\end{aligned}
\tag{4.13}
$$

□

5. EXISTENCE AND UNIQUENESS RESULT

Given the necessary and sufficient conditions proven in section 4, we use the optimal control $(\hat{\alpha}_t)_{0\le t\le T}$ defined by

$$
\begin{aligned}
&\hat{\alpha}(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},Y_t,Z_t,\mathbb{E}[Y_{t+\tau}|\mathcal{F}_t],\mathbb{E}[Z_{t+\tau}|\mathcal{F}_t])\\
&=\arg\min_{\alpha\in\mathbb{A}}H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\alpha_t,\alpha_{t-\tau},Y_t,Z_t),
\end{aligned}
\tag{5.1}
$$

to establish the solvability result of the McKean–Vlasov FABSDE (2.3) and (4.7) for $t\in[0,T]$:

$$
\begin{aligned}
dX_t&=b(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\hat{\alpha}_t,\hat{\alpha}_{t-\tau})dt\\
&\quad+\sigma(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\hat{\alpha}_t,\hat{\alpha}_{t-\tau})dW_t,\\
dY_t&=-\Big\{\partial_x H(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},\hat{\alpha}_t,\hat{\alpha}_{t-\tau},Y_t,Z_t)\\
&\quad+\tilde{\mathbb{E}}[\partial_\mu H(t,\tilde{X}_t,\mu_t,\tilde{X}_{t-\tau},\mu_{t-\tau},\tilde{\hat{\alpha}}_t,\tilde{\hat{\alpha}}_{t-\tau},\tilde{Y}_t,\tilde{Z}_t)(X_t)]\\
&\quad+\mathbb{E}[\partial_{x_\tau}H(t+\tau,X_{t+\tau},\mu_{t+\tau},X_t,\mu_t,\hat{\alpha}_{t+\tau},\hat{\alpha}_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\\
&\quad+\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}H(t+\tau,\tilde{X}_{t+\tau},\mu_{t+\tau},\tilde{X}_t,\mu_t,\tilde{\hat{\alpha}}_{t+\tau},\tilde{\hat{\alpha}}_t,\tilde{Y}_{t+\tau},\tilde{Z}_{t+\tau})\\
&\quad(X_t)]|\mathcal{F}_t]\Big\}dt+Z_t dW_t
\end{aligned}
\tag{5.2}
$$

with initial condition $X_0=x_0$; $X_t=\hat{\alpha}_t=0$ for $t\in[-\tau,0)$ and terminal condition $Y_T=\partial_x g(X_T,\mu_T)+\tilde{\mathbb{E}}[\partial_\mu g(\tilde{X}_T,\mu_T)(X_T)]$. In addition to assumption (H 4.1), we further assume

(H5.1) The drift and volatility functions b and σ are linear in $x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau$. For all $(t,x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau)\in[0,T]\times\mathbb{R}\times\mathcal{P}_2(\mathbb{R})\times\mathcal{P}_2(\mathbb{R})\times A\times A$, we assume that

$$
\begin{aligned}
b(t,x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau)&=b_0(t)+b_1(t)x+\bar{b}_1(t)m\\
&\quad+b_2(t)x_\tau+\bar{b}_2(t)m_\tau+b_3(t)\alpha+b_4(t)\alpha_\tau,\\
\sigma(t,x,\mu,x_\tau,\mu_\tau,\alpha,\alpha_\tau)&=\sigma_0(t)+\sigma_1(t)x+\bar{\sigma}_1(t)m\\
&\quad+\sigma_2(t)x_\tau+\bar{\sigma}_2(t)m_\tau+\sigma_3(t)\alpha+\sigma_4(t)\alpha_\tau,
\end{aligned}
\tag{5.3}
$$

for some measurable deterministic functions $b_0,b_1,\bar{b}_1,b_2,\bar{b}_2,b_3,b_4,\sigma_0,\sigma_1,\bar{\sigma}_1,\sigma_2,\bar{\sigma}_2,\sigma_3,\sigma_4$ with values in $\mathbb{R}$ bounded by R, and we have used the notation $m=\int xd\mu(x)$ and $m_\tau=\int xd\mu_\tau(x)$ for the mean of measures μ and μ_τ, respectively.

(H5.2) The derivatives of f and g with respect to $(x,x_\tau,\mu,\mu_\tau,\alpha)$ and (x,μ) are Lipschitz continuous with Lipschitz constant L.

(H5.3) The function f is strongly L-convex, which means that for any $t\in[0,T]$, any $x,x',x_\tau,x'_\tau\in\mathbb{R}$, any $\alpha,\alpha'\in A$, any $\mu,\mu',\mu_\tau,\mu'_\tau\in\mathcal{P}_2(\mathbb{R})$, any random variables X and X' having μ and μ' as distribution, and any random variables X_τ and X'_τ having μ_τ and μ'_τ as distribution, then

$$
\begin{aligned}
&f(t,x',\mu',x'_\tau,\mu'_\tau,\alpha')-f(t,x,\mu,x_\tau,\mu_\tau,\alpha)\\
&-\ \partial_x f(t,x,\mu,x_\tau,\mu_\tau,\alpha)(x'-x)\\
&-\ \partial_{x_\tau}f(t,x,\mu,x_\tau,\mu_\tau,\alpha)(x'_\tau-x_\tau)\\
&-\ \mathbb{E}[\partial_\mu f(t,x,\mu,x_\tau,\mu_\tau,\alpha)(X)\cdot(X'-X)]\\
&-\ \mathbb{E}[\partial_{\mu_\tau}f(t,x,\mu,x_\tau,\mu_\tau,\alpha)(X_\tau)\cdot(X'_\tau-X_\tau)]\\
&-\ \partial_\alpha f(t,x,\mu,x_\tau,\mu_\tau,\alpha)(\alpha'-\alpha)\ge\kappa|\alpha'-\alpha|^2.
\end{aligned}
\tag{5.4}
$$

The function g is also assumed to be L-convex in (x,μ).

Theorem 5.1. *Under assumptions (H5.1-H5.3), the McKean–Vlasov FABSDE (5.2) is uniquely solvable.*

The proof is based on continuation methods. Let $\lambda\in[0,1]$, consider the following class of McKean–Vlasov FABSDEs, denoted by MV-FABSDE(λ), for $t\in[0,T]$:

$$
\begin{aligned}
dX_t&=(\lambda b(t,\theta_t,\theta_{t-\tau})+I_t^b)dt+(\lambda\sigma(t,\theta_t,\theta_{t-\tau})+I_t^\sigma)dW_t,\\
dY_t&=-\Big\{\lambda\Big(\partial_x H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t)+\tilde{\mathbb{E}}[\partial_\mu H(t,\tilde{\theta}_t,\tilde{\theta}_{t-\tau},\tilde{Y}_t,\tilde{Z}_t)(X_t)]\\
&\quad+\mathbb{E}[\partial_{x_\tau}H(t+\tau,\theta_{t+\tau},\theta_t,Y_{t+\tau},Z_{t+\tau})|\mathcal{F}_t]\\
&\quad+\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}H(t+\tau,\tilde{\theta}_{t+\tau},\tilde{\theta}_t,\tilde{Y}_{t+\tau},\tilde{Z}_{t+\tau})(X_t)]|\mathcal{F}_t]\Big)\\
&\quad+I_t^f\Big\}dt+Z_t dW_t,
\end{aligned}
\tag{5.5}
$$

where we denote $\theta_t=(X_t,\mu_t,\alpha_t)$, with optimality condition

$$
\begin{aligned}
\alpha_t=\hat{\alpha}(t,X_t,\mu_t,X_{t-\tau},\mu_{t-\tau},Y_t,Z_t,\mathbb{E}[Y_{t+\tau}|\mathcal{F}_t],\\
\mathbb{E}[Z_{t+\tau}|\mathcal{F}_t]),\ t\in[0,T],
\end{aligned}
$$

and with initial condition $X_0=x_0$; $X_t=\alpha_t=0$ for $t\in[-\tau,0)$ and terminal condition

$$
Y_T=\lambda\Big\{\partial_x g(X_T,\mu_T)+\tilde{\mathbb{E}}[\partial_\mu g(\tilde{X}_T,\mu_T)(X_T)\Big\}+I_T^g,
$$

and $Y_t=0$ for $t\in(T,T+\tau]$, where $(I_t^b,I_t^\sigma,I_t^f)_{0\le t\le T}$ are some square-integrable progressively measurable processes with values in $\mathbb{R}$, and $I_T^g\in L^2(\Omega,\mathcal{F}_T,\mathbb{P})$ is a square integrable $\mathcal{F}_T$-measurable random variable with value in $\mathbb{R}$.

Observe that when $\lambda=0$, system (5.5) becomes decoupled standard SDE and BSDE, which has an unique solution. When setting $\lambda = 1$, $I_t^b = I_t^\sigma = I_t^f = 0$ for $0 \le t \le T$, and $I_T^g = 0$, we are able to recover the system of (5.2).

Lemma 5.2. *Given $\lambda_0 \in [0,1)$, for any square-integrable progressively measurable processes $(I_t^b, I_t^\sigma, I_t^f)_{0\le t\le T}$, and $I_T^g \in L^2(\Omega, \mathcal{F}_T, \mathbb{P})$, such that system FABSDE(λ_0) admits a unique solution, then there exists $\delta_0 \in (0,1)$, which is independent on λ_0, such that the system MV-FABSDE(λ) admits a unique solution for any $\lambda \in [\lambda_0, \lambda_0 + \delta_0]$.*

Proof: Assuming that $(\check{X}, \check{Y}, \check{Z}, \check{\alpha})$ are given as an input, for any $\lambda \in [\lambda_0, \lambda_0 + \delta_0]$, where $\delta_0 > 0$ to be determined, denoting $\delta := \lambda - \lambda_0 \le \delta_0$, we take

$$\begin{aligned}
I_t^b \leftarrow& \delta[b(t, \check{\theta}_t, \check{\theta}_{t-\tau})] + I_t^b,\\
I_t^\sigma \leftarrow& \delta[\sigma(t, \check{\theta}_t, \check{\theta}_{t-\tau})] + I_t^\sigma,\\
I_t^f \leftarrow& \delta\Big[\partial_x H(t, \check{\theta}_t, \check{\theta}_{t-\tau}, Y_t, Z_t) + \tilde{\mathbb{E}}[\partial_\mu H(t, \tilde{\check{\theta}}_t, \tilde{\check{\theta}}_{t-\tau}, \tilde{\check{Y}}_t, \tilde{\check{Z}}_t)(X_t)]\\
&+ \mathbb{E}[\partial_{x_\tau} H(t+\tau, \check{\theta}_{t+\tau}, \check{\theta}_t, \check{Y}_{t+\tau}, \check{Z}_{t+\tau})|\mathcal{F}_t]\\
&+ \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} H(t+\tau, \tilde{\check{\theta}}_{t+\tau}, \tilde{\check{\theta}}_t, \tilde{\check{Y}}_{t+\tau}, \tilde{\check{Z}}_{t+\tau})(\check{X}_t)]|\mathcal{F}_t]\Big] + I_t^f,\\
I_T^g \leftarrow& \delta\Big[\partial_x g(\check{X}_T, \mu_T) + \tilde{\mathbb{E}}[\partial_\mu g(\tilde{\check{X}}_T, \mu_T)(\check{X}_T)\Big] + I_T^g.
\end{aligned} \tag{5.6}$$

According to the assumption, let (X, Y, Z) be the solutions of MV-FABSDE(λ_0) corresponding to inputs $(\check{X}, \check{Y}, \check{Z})$, i.e., for $t \in [0, T]$

$$\begin{aligned}
dX_t =& (\lambda_0 b_t + \delta \check{b}_t + I_t^b)dt + (\lambda_0 \sigma_t + \delta \check{\sigma}_t + I_t^\sigma)dW_t,\\
dY_t =& -\Big\{\lambda_0(\partial_x H_t + \tilde{\mathbb{E}}[\partial_\mu \tilde{H}_t(X_t)]\\
&+ \mathbb{E}[\partial_{x_\tau} H_{t+\tau}|\mathcal{F}_t] + \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} \tilde{H}_{t+\tau}(X_t)]|\mathcal{F}_t])\\
&+ \delta(\partial_x \check{H}_t + \tilde{\mathbb{E}}[\partial_\mu \tilde{\check{H}}_t(\check{X}_t)] + \mathbb{E}[\partial_{x_\tau} \check{H}_{t+\tau}|\mathcal{F}_t]\\
&+ \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} \tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t]) + I_t^f\Big\}dt + Z_t dW_t,
\end{aligned} \tag{5.7}$$

with initial condition, $X_0 = x_0$, $X_s = \alpha_s = 0$ for $s \in [-\tau, 0)$, and terminal condition

$$Y_T = \lambda_0\Big(\partial_x g_T + \tilde{\mathbb{E}}[\partial_\mu \tilde{g}_T(X_T)]\Big) + \delta\Big(\partial_x \check{g}_T + \tilde{\mathbb{E}}[\partial_\mu \tilde{\check{g}}_T(\check{X}_T)]\Big) + I_T^g, \tag{5.8}$$

and $Y_t = Z_t = 0$ for $t \in (T, T+\tau]$, where we have used simplified notations,

$$\begin{aligned}
b_t &:= b(t, \theta_t, \theta_{t-\tau}); \quad \check{b}_t := b(t, \check{\theta}_t, \check{\theta}_{t-\tau});\\
\sigma_t &:= \sigma(t, \theta_t, \theta_{t-\tau}); \quad \check{\sigma}_t := \sigma(t, \check{\theta}_t, \check{\theta}_{t-\tau});\\
\partial_x H_t &:= \partial_x H(t, \theta_t, \theta_{t-\tau}, Y_t, Z_t);\\
\tilde{\mathbb{E}}[\partial_\mu \tilde{H}_t(X_t)] &:= \tilde{\mathbb{E}}[\partial_\mu H(t, \tilde{\theta}_t, \tilde{\theta}_{t-\tau}, \tilde{Y}_t, \tilde{Z}_t)(X_t)]\\
\mathbb{E}[\partial_{x_\tau} H_{t+\tau}|\mathcal{F}_t] &:= \mathbb{E}[\partial_{x_\tau} H(t+\tau, \theta_{t+\tau}, \theta_t, Y_{t+\tau}, Z_{t+\tau})|\mathcal{F}_t];\\
\mathbb{E}[\tilde{\mathbb{E}}[\partial_\mu \tilde{H}_t(X_t)]|\mathcal{F}_t] &:= \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} H(t+\tau, \tilde{\theta}_{t+\tau}, \tilde{\theta}_t, \tilde{Y}_{t+\tau}, \tilde{Z}_{t+\tau})\\
&\qquad (X_t)]|\mathcal{F}_t];\\
\partial_x g_T &:= \partial_x g(X_T, \mu_T); \quad \tilde{\mathbb{E}}[\partial_\mu \tilde{g}_T(X_T)]\\
&:= \tilde{\mathbb{E}}[\partial_\mu g(\tilde{X}_T, \mu_T)(X_T)]\\
&\text{similar notation for } \partial_x \check{H}_t,\ \tilde{\mathbb{E}}[\partial_\mu \tilde{\check{H}}_t(\check{X}_t)],\ \mathbb{E}[\partial_{x_\tau} \check{H}_{t+\tau}|\mathcal{F}_t],\\
&\text{and } \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} \tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t].
\end{aligned} \tag{5.9}$$

We would like to show that the map $\Phi : (\check{X}, \check{Y}, \check{Z}, \check{\alpha}) \to \Phi(\check{X}, \check{Y}, \check{Z}, \check{\alpha}) = (X, Y, Z, \alpha)$ is a contraction. Consider $(\Delta X, \Delta Y, \Delta Z, \Delta\alpha) = (X - X', Y - Y', Z - Z', \alpha - \alpha')$, where $(X', Y', Z', \alpha') = \Phi(\check{X}', \check{Y}', \check{Z}', \check{\alpha}')$. In addition, for the following computation, we have used simplified notation:

$$\begin{aligned}
\Delta b_t &:= b(t, \theta_t, \theta_{t-\tau}) - b(t, \theta'_t, \theta'_{t-\tau});\\
\Delta \check{b}_t &:= b(t, \check{\theta}_t, \check{\theta}_{t-\tau}) - b(t, \check{\theta}'_t, \check{\theta}'_{t-\tau});\\
\Delta \sigma_t &:= \sigma(t, \theta_t, \theta_{t-\tau}) - \sigma(t, \theta'_t, \theta'_{t-\tau});\\
\Delta \check{\sigma}_t &:= \sigma(t, \check{\theta}_t, \check{\theta}_{t-\tau}) - \sigma(t, \check{\theta}'_t, \check{\theta}'_{t-\tau})\\
\partial_x g_T &:= \partial_x g(X_T, \mu_T) - \partial_x g(X'_T, \mu_T);\\
\Delta\tilde{\mathbb{E}}[\partial_\mu \tilde{g}_T(X_T)] &:= \tilde{\mathbb{E}}[\partial_\mu g(\tilde{X}_T, \mu_T)(X_T)]\\
&\quad - \tilde{\mathbb{E}}[\partial_\mu g(\tilde{\check{X}}'_T, \mu_T)(X'_T)]\\
\Delta \partial_x H_t &:= \partial_x H(t, \theta_t, \theta_{t-\tau}, Y_t, Z_t)\\
&\quad - \partial_x H(t, \theta'_t, \theta'_{t-\tau}, Y_t, Z_t)\\
\Delta\tilde{\mathbb{E}}[\partial_\mu \tilde{H}_t(X_t)] &:= \tilde{\mathbb{E}}[\partial_\mu H(t, \tilde{\theta}_t, \tilde{\theta}_{t-\tau}, \tilde{Y}_t, \tilde{Z}_t)(X_t)]\\
&\quad - \tilde{\mathbb{E}}[\partial_\mu H(t, \tilde{\theta}'_t, \tilde{\theta}'_{t-\tau}, \tilde{Y}_t, \tilde{Z}_t)(X'_t)]\\
\Delta\mathbb{E}[\partial_{x_\tau} H_{t+\tau}|\mathcal{F}_t] &:= \mathbb{E}[\partial_{x_\tau} H(t+\tau, \theta_{t+\tau}, \theta_t, Y_{t+\tau}, Z_{t+\tau})|\mathcal{F}_t]\\
&\quad - \mathbb{E}[\partial_{x_\tau} H(t+\tau, \theta'_{t+\tau}, \theta'_t, Y_{t+\tau}, Z_{t+\tau})|\mathcal{F}_t]\\
\Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_\mu \tilde{H}_t(X_t)]|\mathcal{F}_t] &:= \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} H(t+\tau, \tilde{\theta}_{t+\tau}, \tilde{\theta}_t, \tilde{Y}_{t+\tau},\\
&\quad \tilde{Z}_{t+\tau})(X_t)]|\mathcal{F}_t] - \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} H(t+\tau, \tilde{\theta}'_{t+\tau}, \tilde{\theta}'_t,\\
&\quad \tilde{Y}_{t+\tau}, \tilde{Z}_{t+\tau})(X'_t)]|\mathcal{F}_t]\\
&\text{similar notation for } \Delta\partial_x \check{H}_t,\ \Delta\tilde{\mathbb{E}}[\partial_\mu \tilde{\check{H}}_t(\check{X}_t)],\ \Delta\mathbb{E}[\partial_{x_\tau} \check{H}_{t+\tau}|\mathcal{F}_t],\\
&\text{and } \Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} \tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t].
\end{aligned} \tag{5.10}$$

Applying integration by parts to $\Delta X_t Y_t$, we have

$$\begin{aligned}
&d(\Delta X_t Y_t)\\
&= Y_t\Big\{[\lambda_0 \Delta b_t + \delta \Delta \check{b}_t]dt + [\lambda_0 \Delta\sigma_t + \delta\Delta\check{\sigma}_t]dW_t\Big\}\\
&\quad - \Delta X_t\Big\{\lambda_0(\partial_x H_t + \tilde{\mathbb{E}}[\partial_\mu \tilde{H}_t(X_t)] + \mathbb{E}[\partial_{x_\tau} H_{t+\tau}|\mathcal{F}_t]\\
&\quad + \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} \tilde{H}_{t+\tau}(X_t)]|\mathcal{F}_t]) + \delta(\partial_x \check{H}_t + \tilde{\mathbb{E}}[\partial_\mu \tilde{\check{H}}_t(\check{X}_t)]\\
&\quad + \mathbb{E}[\partial_{x_\tau} \check{H}_{t+\tau}|\mathcal{F}_t] + \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau} \tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t])\Big\}dt\\
&\quad + \Delta X_t Z_t dW_t + (\lambda_0 \Delta\sigma_t + \delta\Delta\check{\sigma}_t)Z_t dt.
\end{aligned} \tag{5.11}$$

After integrating from 0 to T, and taking expectation on both sides, we obtain

$$\begin{aligned}
&\mathbb{E}[\Delta X_T Y_T] \\
=&\lambda_0 \mathbb{E}\int_0^T \Big(\Delta b_t Y_t + \Delta\sigma_t Z_t - \Delta X_t(\partial_x H_t + \tilde{\mathbb{E}}[\partial_\mu \tilde{H}_t(X_t)] \\
&+ \mathbb{E}[\partial_{x_\tau} H_{t+\tau}|\mathcal{F}_t] + \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{H}_{t+\tau}(X_t)]|\mathcal{F}_t])\Big)dt \\
&+ \delta\mathbb{E}\int_0^T \Big(\Delta\check{b}_t Y_t + \Delta\check{\sigma} Z_t - \Delta X_t(\partial_x \check{H}_t + \tilde{\mathbb{E}}[\partial_\mu \tilde{\check{H}}_t(\check{X}_t)] \\
&+ \mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t] + \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t])\Big)dt
\end{aligned} \tag{5.12}$$

In the meantime, from the terminal condition of Y_T given in (5.8), and since g is convex, we also have

$$\begin{aligned}
&\mathbb{E}[\Delta X_T Y_T] \\
=&\mathbb{E}\Big[\Delta X_T\Big(\lambda_0(\partial_x g_T + \tilde{\mathbb{E}}[\partial_\mu \tilde{g}_T(X_T)]) + \delta(\partial_x \check{g}_T \\
&+\tilde{\mathbb{E}}[\partial_\mu \tilde{\check{g}}_T(\check{X}_T)]) + I_T^g\Big)\Big] \geq \lambda_0\mathbb{E}[g(X_T,\mu_T) - g(X_T' - \mu_T')] \\
&+ \delta\Delta X_T(\partial_x\check{g}_T + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]) + \Delta X_T I_T^g
\end{aligned} \tag{5.13}$$

Following the proof of sufficient part of maximum principle and using (5.12), and (5.13), we find

$$\begin{aligned}
\lambda_0(J(\alpha) - J(\alpha'))
&= \lambda_0\mathbb{E}[g(X_T,\mu_T) - g(X_T',\mu_T')] \\
&+ \lambda_0\mathbb{E}\int_0^T [f(t,\theta_t,X_{t-\tau},\mu_{t-\tau}) \\
&- f(t,\theta_t',X_{t-\tau}',\mu_{t-\tau}')]dt \\
&\leq \mathbb{E}[\Delta X_T Y_T] - \delta\Delta X_T(\partial_x\check{g}_T + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]) - \Delta X_T I_T^g \\
&+ \lambda_0\mathbb{E}\int_0^T [f(t,\theta_t,X_{t-\tau},\mu_{t-\tau}) - f(t,\theta_t',X_{t-\tau}',\mu_{t-\tau}')]dt \\
&= \lambda_0\mathbb{E}\int_0^T \Big[\Delta b_t Y_t + \Delta\sigma_t Z_t \\
&- \Delta X_t(\partial_x H_t + \tilde{\mathbb{E}}[\partial_\mu\tilde{H}_t(X_t)] + \mathbb{E}[\partial_{x_\tau}H_{t+\tau}|\mathcal{F}_t] \\
&+ \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{H}_{t+\tau}(X_t)]|\mathcal{F}_t])\Big]dt \\
&+ \delta\mathbb{E}\int_0^T \Big[\Delta\check{b}_t Y_t + \Delta\check{\sigma}_t Z_t - \Delta X_t(\partial_x\check{H}_t + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t(\check{X}_t)] \\
&+ \mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t] + \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(X_t)]|\mathcal{F}_t])\Big]dt \\
&+ \lambda_0\mathbb{E}\int_0^T [H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t) - H(t,\theta_t',\theta_{t-\tau}',Y_t,Z_t)]dt \\
&- \lambda_0\mathbb{E}\int_0^T (\Delta b_t Y_t + \Delta_t\sigma Z_t)dt - \delta\Delta X_T(\partial_x\check{g}_T \\
&+ \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]) - \Delta X_T I_T^g \\
&= \lambda_0\mathbb{E}\int_0^T \Big[H(t,\theta_t,\theta_{t-\tau},Y_t,Z_t) - H(t,\theta_t',\theta_{t-\tau}',Y_t,Z_t) \\
&- \Delta X_t(\partial_x H_t + \tilde{\mathbb{E}}[\partial_\mu\tilde{H}_t(X_t)] + \mathbb{E}[\partial_{x_\tau}H_{t+\tau}|\mathcal{F}_t] \\
&+ \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{H}_{t+\tau}(X_t)]|\mathcal{F}_t])\Big]dt + \delta\mathbb{E}\int_0^T\Big[\Delta\check{b}_t Y_t \\
&+ \Delta\check{\sigma} Z_t - \Delta X_t(\partial_x\check{H}_t + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t(\check{X}_t)] + \mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t] \\
&+ \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t])\Big]dt \\
&- \delta\Delta X_T(\partial_x\check{g}_T + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]) - \Delta X_T I_T^g \\
&\leq -\mathbb{E}\int_0^T \lambda_0\kappa|\Delta\alpha_t|^2 dt + \delta\mathbb{E}\int_0^T\Big[\Delta\check{b}_t Y_t + \Delta\check{\sigma}_t Z_t \\
&- \Delta X_t(\partial_x\check{H}_t + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t(\check{X}_t)] + \mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t] \\
&+\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t])\Big]dt - \delta\Delta X_T(\partial_x\check{g}_T \\
&+ \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]) - \Delta X_T I_T^g
\end{aligned} \tag{5.14}$$

Reverse the role of α and α', we also have

$$\begin{aligned}
&\lambda_0(J(\alpha') - J(\alpha)) \\
\leq& -\mathbb{E}\int_0^T \lambda_0\kappa|\Delta\alpha_t'|^2 dt + \delta\mathbb{E}\int_0^T\Big[\Delta\check{b}_t' Y_t' + \Delta\check{\sigma}_t' Z_t' \\
&- \Delta X_t'(\partial_x\check{H}_t' + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t'(\check{X}_t')] + \mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}'|\mathcal{F}_t] \\
&+ \mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}'(\check{X}_t')]|\mathcal{F}_t])\Big]dt - \delta\Delta X_T'(\partial_x\check{g}_T \\
&+ \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T')]) - \Delta X_T' I_T^g
\end{aligned} \tag{5.15}$$

Summing (5.14) and (5.15), using the fact that b and σ have the linear form, using change of time and Lipschitz assumption, it yields

$$\begin{aligned}
&2\lambda_0\kappa\mathbb{E}\int_0^T |\Delta\alpha_t|^2 dt \\
\leq& \delta\mathbb{E}\int_0^T\Big[\Delta\check{b}_t\Delta Y_t + \Delta\check{\sigma}\Delta Z_t - \Delta X_t(\Delta\partial_x\check{H}_t \\
&+ \Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t(X_t)] + \Delta\mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t] \\
&+ \Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(X_t)]|\mathcal{F}_t])\Big]dt + \delta\Delta X_T(\partial_{x'}\check{g}_T' \\
&- \partial_x\check{g}_T + \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T'(\check{X}_T)] - \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]) \\
\leq& \frac{1}{2}\mathbb{E}\int_0^T\Big[\epsilon(|\Delta X_t|^2 + |\Delta Y_t|^2 + |\Delta Z_t|^2) \\
&+ \frac{1}{\epsilon}\delta^2\Big(|\Delta\check{b}_t|^2 + |\Delta\check{\sigma}|^2 + |\Delta\partial_x\check{H}_t + \Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t(X_t)] \\
&+ \Delta\mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t] + \Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(X_t)]|\mathcal{F}_t]|^2\Big)\Big]dt
\end{aligned}$$

$$+\ \frac{1}{2}\left(\epsilon|\Delta X_T|^2+\frac{1}{\epsilon}\delta^2\left|\partial_{x'}\check{g}'_T-\partial_x\check{g}_T+\tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}'_T(\check{X}_T)]\right.\right.$$
$$\left.\left.-\ \tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]\right|^2\right)\le\frac{1}{2}\epsilon\mathbb{E}\left[\int_0^T\epsilon(|\Delta X_t|^2+|\Delta Y_t|^2\right.$$
$$\left.+\ |\Delta Z_t|^2+|\Delta\alpha_t|^2)dt+|\Delta X_T|^2\right]$$
$$+\ \frac{1}{2}\delta\frac{C}{\epsilon}\mathbb{E}\left[\int_0^T(|\Delta\check{X}_t|^2+|\Delta\check{Y}_t|^2+|\Delta\check{Z}_t|^2\right.$$
$$\left.+\ |\Delta\check{\alpha}_t|^2)]dt+|\Delta\check{X}_T|^2\right],\tag{5.16}$$

Next, we apply Ito's formula to ΔX_t^2,

$$\begin{aligned}&d\Delta X_t^2\\&=2\Delta X_t dX_t+d\langle X,X\rangle_t\\&=2\Delta X_t(\lambda_0\Delta b_t+\delta\Delta\check{b}_t)dt+2\Delta X_t(\lambda_0\Delta\sigma_t\\&\quad+\delta\Delta\check{\sigma}_t)dW_t+\left(\lambda_0\Delta\sigma_t+\delta\Delta\check{\sigma}_t\right)^2dt\end{aligned}\tag{5.17}$$

Then integrate from 0 to T, and take expectation,

$$\begin{aligned}&\mathbb{E}[|\Delta X_t|^2]\\&=2\lambda_0\mathbb{E}\int_0^t|\Delta X_s\Delta b_s|ds+2\delta\mathbb{E}\int_0^t|\Delta X_s\Delta\check{b}_s|ds\\&\quad+\mathbb{E}\int_0^t|\lambda_0\Delta\sigma_s+\delta\Delta\check{\sigma}_s|^2ds\\&\le\lambda_0\mathbb{E}\int_0^t(|\Delta X_s|^2+|\Delta b_s|^2)ds+\mathbb{E}\int_0^t(|\Delta X_s|^2+\delta^2|\Delta\check{b}_s|^2)ds\\&\quad+\mathbb{E}\int_0^t(2\lambda_0^2|\Delta\sigma_s|^2+2\delta^2|\Delta\check{\sigma}_s|^2)ds\\&\le C\mathbb{E}\int_0^{t+\tau}(|\Delta X_s|^2+|\Delta\alpha_s|^2)ds+\delta C\mathbb{E}\int_0^{t+\tau}(|\Delta\check{X}_s|^2+|\Delta\check{\alpha}_s|^2)ds\end{aligned}\tag{5.18}$$

From Gronwall's inequality, we can obtain

$$\sup_{0\le t\le T}\mathbb{E}[|X_t|^2]\le C\mathbb{E}\int_0^T|\Delta\alpha_t|^2dt+\delta C\mathbb{E}\int_0^T(|\Delta\check{X}_t|^2+|\Delta\check{\alpha}_t|^2)dt\tag{5.19}$$

Similarly, applying Ito's formula to $|\Delta Y_t|^2$, and taking expectation, we have

$$\begin{aligned}&\mathbb{E}\left[|\Delta Y_t|^2+\int_t^T|\Delta Z_s|^2ds\right]\\&=2\lambda_0\mathbb{E}\int_t^T\left|\Delta Y_t\left(\Delta\partial_x H_t+\Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{H}_t(X_t)]\right.\right.\\&\quad\left.\left.+\Delta\mathbb{E}[\partial_{x_\tau}H_{t+\tau}|\mathcal{F}_t]+\Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{H}_{t+\tau}(X_t)]|\mathcal{F}_t]\right)\right|\\&\quad+2\delta\mathbb{E}\int_t^T\left|\Delta Y_t\left(\Delta\partial_x\check{H}_t+\Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t(\check{X}_t)]+\Delta\mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t]\right.\right.\\&\quad\left.\left.+\Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t]\right)\right|+\mathbb{E}|\Delta Y_T|^2\end{aligned}$$

$$\begin{aligned}&\le\mathbb{E}\int_t^T\left(\frac{1}{\epsilon}|\Delta Y_t|^2+\epsilon\lambda_0^2\left|\Delta\partial_x H_t+\Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{H}_t(X_t)]\right.\right.\\&\quad\left.\left.+\Delta\mathbb{E}[\partial_{x_\tau}H_{t+\tau}|\mathcal{F}_t]+\Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{H}_{t+\tau}(X_t)]|\mathcal{F}_t]\right|^2\right)dt\\&\quad+\mathbb{E}\int_t^T\left(|\Delta Y_t|^2+\delta^2\left|\Delta\partial_x\check{H}_t+\Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{\check{H}}_t(\check{X}_t)]\right.\right.\\&\quad\left.\left.+\Delta\mathbb{E}[\partial_{x_\tau}\check{H}_{t+\tau}|\mathcal{F}_t]+\Delta\mathbb{E}[\tilde{\mathbb{E}}[\partial_{\mu_\tau}\tilde{\check{H}}_{t+\tau}(\check{X}_t)]|\mathcal{F}_t]\right|^2\right)dt\\&\quad+\mathbb{E}\left|\lambda_0\left(\Delta\partial_x g_T+\Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{g}_T(X_T)]\right)\right.\\&\quad\left.+\delta\left(\Delta\partial_x\check{g}_T+\Delta\tilde{\mathbb{E}}[\partial_\mu\tilde{\check{g}}_T(\check{X}_T)]\right)\right|^2\end{aligned}\tag{5.20}$$

Choose $\epsilon = 96\max\{R^2,L\}$, and from assumption (H5.1 - H5.2) and Gronwall's inequality, we obtain a bound for $\sup_{0\le t\le T}\mathbb{E}|\Delta Y_t|^2$; and then substitute the it back to the same inequality, we are able to obtain the bound for $\int_0^T E|Z_t|^2dt$. By combining these two bounds, we deduce that

$$\begin{aligned}&\mathbb{E}\left[\sup_{0\le t\le T}|Y_t|^2+\int_0^T|Z_t|^2dt\right]\\&\le C\mathbb{E}\left(\sup_{0\le t\le T}|\Delta X_t|^2+\int_0^T|\Delta\alpha_t|^2dt\right)\\&\quad+\delta C\mathbb{E}\left[\sup_{0\le t\le T}\left(|\Delta\check{X}_t|^2+|\Delta\check{Y}_t|^2\right)\right.\\&\quad\left.+\int_0^T\left(|\Delta\check{Z}_t|^2+|\Delta\check{\alpha}_t|^2\right)dt\right]\end{aligned}\tag{5.21}$$

Finally, combining (5.19) and (5.21), and (5.16), we deduce

$$\begin{aligned}&\mathbb{E}\left[\sup_{0\le t\le T}|\Delta X_t|^2+\sup_{0\le t\le T}|\Delta Y_t|^2+\int_0^T\left(|\Delta Z_t^2|+|\Delta\alpha_t|^2\right)dt\right]\\&\le\delta C\mathbb{E}\left[\sup_{0\le t\le T}|\Delta\check{X}_t|^2+\sup_{0\le t\le T}|\Delta\check{Y}_t|^2+\int_0^T\left(|\Delta\check{Z}_t|^2+|\Delta\check{\alpha}_t|^2\right)dt\right]\end{aligned}\tag{5.22}$$

Let $\delta_0=\frac{1}{2C}$, it is clear that the mapping Φ is a contraction for all $\delta\in(0,\delta_0)$. It follows that there is a unique fixed point which is the solution of MV-FABSDE(λ) for $\lambda=\lambda_0+\delta,\ \delta\in(0,\delta_0)$. □

Proof of Theorem 5.1: For $\lambda = 0$, FABSDE(0) has a unique solution. Using Lemma 5.2, there exists a $\delta_0 > 0$ such that FBSDE(δ) has a unique solution for $\delta\in[0,\delta_0]$, assuming $(n-1)\delta_0<1\le n\delta_0$. Following by a induction argument, we repeat Lemma 5.2 for n times, which gives us the existence of the unique solution of FABSDE(1). □

6. CONCLUSION

Overall, we presented a comprehensive study of a general class of mean-field control problems with delay effect. The state dynamics is described by a McKean–Vlasov stochastic delayed differential equation. We derive the adjoint process associated to the dynamics, which is in the form of an anticipated backward stochastic differential equation of McKean–Vlasov type. We also prove a version of stochastic maximum principle, which gives necessary and sufficient conditions for the optimal control. Furthermore, we prove the existence and uniqueness of this class of forward anticipated backward stochastic differential equations under suitable conditions.

However, due to the lack of explicit solutions, numerical methods are needed. The non-linear nature of the problem due to the McKean–Vlasov aspect combined with non-Markovianity due to delay rule out classical numerical methods. Our study show that deep learning methods can deal with these obstacles. we proposed two algorithms based on machine learning to numerically solve the control problem. One is to directly approximate the control using a neural network, while the loss is given by the objective function in the control problem. The other algorithm is based on the system of forward and backward stochastic differential equations. We approximate the adjoint processes $(Y_\cdot, Z_\cdot)$ and the conditional expectation of the adjoint process $\mathbb{E}[Y_{\cdot+\tau}|\mathcal{F}_\cdot]$ using neural networks. In this case, there are two loss functions that we need to minimize as shown in (3.16). The first loss is associated with the adjoint process $Y_\cdot$, and the other one is related to $\mathbb{E}[Y_{\cdot+\tau}|\mathcal{F}_\cdot]$. After minimizing the losses, the optimal control can be readily obtained from the adjoint processes. In addition, **Figure 8** illustrates that the optimal value to the control problem, that is computed from different methods, converges. Moreover, our method also works when the control problem has no delay. As a sanity check, we provide an example of control problem without delay effect, and we show that the solution obtained from the algorithm matches the explicit solution.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

FUNDING

This work supported by NSF grant DMS-1814091.

REFERENCES

1. Carmona R, Fouque JP, Sun LH. Mean field games and systemic risk. *Commun Math Sci.* (2015) **13**:911–33. doi: 10.4310/CMS.2015.v13.n4.a4
2. Carmona R, Fouque JP, Mousavi SM, Sun LH. Systemic risk and stochastic games with delay. *J Optim Appl.* (2018) **179**:366–99. doi: 10.1007/s10957-018-1267-8
3. Carmona R, Delarue F. *Probabilistic Theory of Mean Field Games with Applications I & II*. Springer International Publishing (2018).
4. Bensoussan A, Frehse J, Yam SCP. *Mean Field Games and Mean Field Type Control Theory*. New York, NY: Springer-Verlag (2013).
5. Peng S, Yang Z. Anticipated backward stochastic differential equations. *Ann Probab.* (2009) **37**:877–902. doi: 10.1214/08-AOP423
6. Zhang J. *Backward Stochastic Differential Equations*. New York, NY: Springer-Verlag (2017).
7. Chen L, Wu Z. Maximum principle for the stochastic optimal control problem with delay and application. *Automatica.* (2010) **46**:1074–80. doi: 10.1016/j.automatica.2010.03.005
8. Peng S, Wu Z. Fully coupled forward-backward stochastic differential equations and applications to optimal control. *SIAM J Control Optim.* (1999) **37**:825–43. doi: 10.1137/S0363012996313549
9. Bensoussan A, Yam SCP, Zhang Z. Well-posedness of mean-field type forward-backward stochastic differential equations. *Stochastic Process Appl.* (2015) **125**:3327–54. doi: 10.1016/j.spa.2015.04.006
10. Ma J, Yong J. *Forward-Backward Stochastic Differential Equations and their Applications*. Berlin; Heidelberg: Springer-Verlag (2007).
11. Ma J, Protter P, Yong J. Solving forward-backward stochastic differential equations explicitly–a four step scheme. *Probab Theory Relat Fields.* (1994) **98**:339–59.
12. Weinan E, Han J, Jentzen A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun Math Stat.* (2017) **5**:349–80. doi: 10.1007/s40304-017-0117-6
13. Raissi M. Forward-backward stochastic neural networks: deep learning of high-dimensional partial differential equations. (2018). *arxiv[Preprint].arxiv:1804.07010.*
14. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput.* (1997) 9:1735–80.
15. Sutton RS, McAllester DA, Singh SP, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. In: *Advances in Neural Information Processing Systems (NIPS)*. (1999). p. 1057–63. Available online at: https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation
16. Han J, Weinan E. *Deep Learning Approximation for Stochastic Control Problems, Deep Reinforcement Learning Workshop*. Conference on Neural Information Processing Systems (NIPS) (2016).
17. Carmona R, Delarue F, Lachapelle A. Control of McKean-Vlasov dynamics versus mean field games. *Math Fin Econ.* (2013) **7**:131–66. doi: 10.1007/s11579-012-0089-y
18. Fouque JP, Zhang Z. Mean field game with delay: a toy model. *Risks.* (2018) **6**:1–17. doi: 10.3390/risks6030090

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.

ORIGINAL RESEARCH
published: 04 August 2020
doi: 10.3389/fams.2020.00028

Comparing Model Selection Criteria to Distinguish Truncated Operational Risk Models

*Daoping Yu**

School of Computer Science and Mathematics, University of Central Missouri, Warrensburg, MO, United States

In this paper three information criteria are employed to assess the truncated operational risk models. The performances of the three information criteria on distinguishing the models are compared. The competing models are constructed using Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull distributions, respectively. Simulation studies are conducted before a case study. In the case study, certain distributional models conform to the external fraud type of risk data in retail banking of Chinese banks. However, those models are difficult to distinguish using standard information criteria such as Akaike Information Criterion and Bayesian Information Criterion. We have found no single information criterion is absolutely more effective than others in the simulation studies. But the information complexity based ICOMP criterion says a little bit more if AIC and/or BIC cannot kick the Lognormal model out of the pool of competing models.

Keywords: information criteria, model selection, operational risk, truncated models, Value at Risk (VaR)

OPEN ACCESS

Edited by:
Lawrence Kwan Ho Ma,
Hong Kong Blockchain Society,
Hong Kong

Reviewed by:
Dimitrios D. Thomakos,
University of Peloponnese, Greece
Simon Grima,
University of Malta, Malta

****Correspondence:***
Daoping Yu
dyu@ucmo.edu

Specialty section:
This article was submitted to Mathematical Finance, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *01 May 2020*
Accepted: *01 July 2020*
Published: *04 August 2020*

Citation:
Yu D (2020) Comparing Model Selection Criteria to Distinguish Truncated Operational Risk Models. Front. Appl. Math. Stat. 6:28. doi: 10.3389/fams.2020.00028

1. INTRODUCTION

This paper mainly applies model selection information criteria to operational risk models subject to data truncation. In the practice of collecting operational loss data of a bank, certain losses below a threshold value are not recorded. Thus, the data at hand can be viewed as truncated from below. The truncated models, compared to the shifted models and naive models, have been determined to be appropriate to model loss data for operational risk [1].

Model selection differs from model validation. Perhaps there are a few models passing the validation process. Model selection criteria are further used to separate a most suitable model from the rest. Traditional information criteria such as Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) have been documented [2] for a banking organization to employ when comparing alternative models. The recent research work of Isaksson [3] and Svensson [4] have explored model selection for operational risk models. Using AIC and/or BIC, they have determined the overall best distributional model(s) for internal data or external data for operational risk in financial institutions.

The purpose of this paper is to assess the effectiveness of information complexity based ICOMP criterion, compared with AIC and/or BIC, to distinguish the fat tailed distributional models for operational risk.

The structure of the remaining of this paper is as follows. In section 2, six candidate distributions for construction of the truncated models of operational risk are reviewed: they are Champernowne distribution, Frechet distribution, Lognormal distribution, Lomax distribution, Paralogistic distribution, and Weibull distribution. In section 3, we give an introduction to such standard information criteria as AIC and BIC for model selection. After reviewing standard information criteria, another information criterion based on information complexity, known as ICOMP, is presented. In section 4, we conduct simulation studies for the six distributional models. In section 5, we walk the readers through the model fitting, model validation, VaR estimation, and model selection procedure, using a real data set for external fraud type of losses in retail banking across commercial banks in China. The practical performance of all information criteria presented in this paper is compared. Concluding remarks are provided in section 6.

2. PARAMETRIC DISTRIBUTIONS FOR OPERATIONAL RISK

The real data of operational losses usually exhibit fat tail properties. The main body of the data are of low severity and high frequency. In other words, operational losses of small sizes occur on a frequent basis. The losses of significantly larger sizes would occur less frequently, but cannot be ignored, since a few extremely large magnitude of losses could be very influential to the financial health and security of a financial institution. A few appropriate distributions for modeling the skewed type of operational loss data have been studied in the literature: for instance, the Champernowne distribution (see for example, [5]), the Lognormal distribution (see for example, [6]), and the Lomax distribution (also known as two-parameter Pareto distribution) as a special case of the Generalized Pareto Distribution (see for example, [7]), etc. There are a few other distributions that are suitable to model fat tailed risks, such as the Frechet distribution, the Paralogistic distribution, and the Weibull distribution (see for instance, [8]).

2.1. Champernowne Distribution

The Champernowne distribution is originally proposed in the study of income distribution, and it is a generalization of the logistic distribution, firstly introduced by an econometrician D. G. Champernowne [9, 10], in the development of distributions to describe the logarithm of income. The Champernowne distribution has probability density function (pdf)

$$f_{\mathrm{CHAMP}(\alpha,M)}(x) = \frac{\alpha M^{\alpha} x^{\alpha-1}}{(x^{\alpha}+M^{\alpha})^{2}}, \quad x>0, \tag{1}$$

and cumulative distribution function (cdf)

$$F_{\mathrm{CHAMP}(\alpha,M)}(x) = \frac{x^{\alpha}}{x^{\alpha}+M^{\alpha}}, \quad x>0, \tag{2}$$

where $\alpha > 0$ is the shape parameter and $M > 0$ is another parameter that represents the median of the distribution. The Champernowne distribution looks more like a Lognormal distribution near x value of 0 when $\alpha > 1$, while converging to a Lomax distribution in the tail. In addition, by inverting the cdf given in (2) one obtains the quantile function (qf) as

$$F^{-1}_{\mathrm{CHAMP}(\alpha,M)}(\beta) = M\left(\frac{\beta}{1-\beta}\right)^{1/\alpha}, \quad 0<\beta<1. \tag{3}$$

For more details about the application of Champernowne distribution to operational risk modeling, readers are referred to a monograph (see [5]).

2.2. Frechet Distribution

The Frechet distribution is also known as the inverse Weibull distribution. The pdf of the two-parameter Frechet distribution is given by

$$f_{\mathrm{Frechet}(\alpha,\theta)}(x) = \frac{\alpha\,(x/\theta)^{-\alpha}\,e^{-(x/\theta)^{-\alpha}}}{x}, \quad x>0, \tag{4}$$

and the cdf is given by

$$F_{\mathrm{Frechet}(\alpha,\theta)}(x) = e^{-(x/\theta)^{-\alpha}}, \quad x>0. \tag{5}$$

The qf is found by inverting (5) and given by

$$F^{-1}_{\mathrm{Frechet}(\alpha,\theta)}(\beta) = \theta\left(-\log\beta\right)^{-1/\alpha}, \quad 0<\beta<1. \tag{6}$$

For the Frechet distribution, $\alpha > 0$ is the shape parameter, and $\theta > 0$ is the scale parameter.

2.3. Lognormal Distribution

The Lognormal distribution with parameters μ and σ is defined as the distribution of a random variable X whose logarithm is normally distributed with mean μ and variance σ^2. The two-parameter Lognormal distribution has pdf

$$f_{\mathrm{LN}(\mu,\sigma)}(x) = \begin{cases} \dfrac{1}{\sqrt{2\pi}\sigma x}\, e^{-\dfrac{\left(\log x-\mu\right)^2}{2\sigma^2}}, & \text{if } x>0, \\ 0, & \text{if } x\le 0, \end{cases} \tag{7}$$

where $-\infty < \mu < \infty$ is the location parameter and $\sigma > 0$ is the scale parameter.

The two-parameter Lognormal distribution has cdf

$$F_{\mathrm{LN}(\mu,\sigma)}(x) = \Phi\left(\frac{\log x-\mu}{\sigma}\right), \quad x>0, \tag{8}$$

and by inverting (8), the quantile function

$$F^{-1}_{\mathrm{LN}(\mu,\sigma)}(\beta) = e^{\mu+\sigma\Phi^{-1}(\beta)}, \quad 0<\beta<1 \tag{9}$$

is obtained. Here Φ and Φ^{-1} denote the cdf and qf of the standard normal distribution, respectively.

2.4. Lomax Distribution

The pdf of the two-parameter Pareto distribution, also called Lomax distribution, is given by

$$f_{\mathrm{LOMAX}(\alpha,\theta)}(x) = \alpha\theta^{\alpha}\,(x+\theta)^{-\alpha-1}, \quad x>0, \tag{10}$$

and the cdf is given by

$$F_{\mathrm{LOMAX}(\alpha,\theta)}(x) = 1-\left(\theta/(x+\theta)\right)^{\alpha}, \quad x>0, \tag{11}$$

and the qf is found by inverting (11) and given by

$$F^{-1}_{\mathrm{LOMAX}(\alpha,\theta)}(\beta) = \theta\left((1-\beta)^{-1/\alpha} - 1\right), \quad 0 < \beta < 1. \tag{12}$$

We refer readers to Arnold [11] for more details of Pareto distributions, and see Klugman et al. [8] for the applications of Pareto distributions to insurance loss modeling.

2.5. Paralogistic Distribution

The pdf of the two-parameter Paralogistic distribution is given by

$$f_{\mathrm{PL}(\alpha,\theta)}(x) = \frac{\alpha^2\,(x/\theta)^\alpha}{x\,(1+(x/\theta)^\alpha)^{\alpha+1}}, \quad x > 0, \tag{13}$$

and the cdf is given by

$$F_{\mathrm{PL}(\alpha,\theta)}(x) = 1 - \left(1 + (x/\theta)^\alpha\right)^{-\alpha}, \quad x > 0, \tag{14}$$

and the qf is found by inverting (14) and given by

$$F^{-1}_{\mathrm{PL}(\alpha,\theta)}(\beta) = \theta\left((1-\beta)^{-1/\alpha} - 1\right)^{1/\alpha}, \quad 0 < \beta < 1. \tag{15}$$

The Paralogistic distribution is characterized by the shape parameter $\alpha > 0$ and the scale parameter $\theta > 0$.

2.6. Weibull Distribution

The pdf of the two-parameter Weibull distribution is given by

$$f_{\mathrm{Weibull}(\alpha,\theta)}(x) = \frac{\alpha\,(x/\theta)^\alpha\,e^{-(x/\theta)^\alpha}}{x}, \quad x > 0, \tag{16}$$

and the cdf is given by

$$F_{\mathrm{Weibull}(\alpha,\theta)}(x) = 1 - e^{-(x/\theta)^\alpha}, \quad x > 0, \tag{17}$$

and the qf is found by inverting (17) and given by

$$F^{-1}_{\mathrm{Weibull}(\alpha,\theta)}(\beta) = \theta\left(-\log(1-\beta)\right)^{1/\alpha}, \quad 0 < \beta < 1. \tag{18}$$

The parameter $\alpha > 0$ characterizes the shape of the Weibull distribution, and the parameter $\theta > 0$ characterizes the scale.

3. INFORMATION CRITERIA

Given a data set, $X_1, \ldots, X_n$, the amount of objective information contained in the data is fixed. However, different models may be fitted to the same data set in order to extract the information. We want to select the model that best approximates the distribution of the data. There are various information criteria for model selection, such as the Akaike Information Criterion [12], Bayesian Information Criterion [13], and Information Complexity (ICOMP) [14, 15]. The defining formulas of all these three contain negative double log-likelihood.

Let $\mathcal{L}\left(\theta_1, \ldots, \theta_k \,\middle|\, X_1, \ldots, X_n\right)$ be the likelihood function of a model with k parameters based on a sample of size n, and let $\widehat{\theta}_1$, …, $\widehat{\theta}_k$ denote the corresponding estimators of those parameters using the method of Maximum Likelihood Estimation (MLE).

The AIC is defined as:

$$\mathrm{AIC} = -2\log\mathcal{L}\left(\widehat{\theta}_1, \ldots, \widehat{\theta}_k \,\middle|\, X_1, \ldots, X_n\right) + 2k. \tag{19}$$

The BIC is defined as:

$$\mathrm{BIC} = -2\log\mathcal{L}\left(\widehat{\theta}_1, \ldots, \widehat{\theta}_k \,\middle|\, X_1, \ldots, X_n\right) + k\log n. \tag{20}$$

The ICOMP is defined as:

$$\mathrm{ICOMP} = -2\log\mathcal{L}\left(\widehat{\theta}_1, \ldots, \widehat{\theta}_k \,\middle|\, X_1, \ldots, X_n\right) + 2C_1\left(\mathbf{I}^{-1}(\widehat{\theta}_1, \ldots, \widehat{\theta}_k)\right), \tag{21}$$

where

$$C_1\left(\mathbf{I}^{-1}\right) = \frac{s}{2}\log\left(\frac{\mathrm{tr}\left(\mathbf{I}^{-1}\right)}{s}\right) - \frac{1}{2}\log\left(\det\left(\mathbf{I}^{-1}\right)\right),$$

with s, tr, and det denoting the rank, trace, and determinant of $\mathbf{I}^{-1}$ (the inverse of Fisher information matrix), respectively.

Using those information criteria, the preferred model is the one that minimizes AIC, BIC, or ICOMP. In general, it is known that when the number of parameters increases, the model likelihood increases as well. We see that there is a competition between the increase in the log-likelihood value and the increase in the number of model parameters in the AIC and BIC formulas. If the increase in the log-likelihood value is not sufficient to compensate the increase in the number of parameters, then it is not worthwhile to have the additional parameters. Note also that the BIC criterion penalizes the model dimensionality more than AIC for $\log n > 2$.

ICOMP penalizes the interdependencies among parameter estimators of MLE, i.e., the complexity of variance-covariance structure of model's maximum likelihood estimators via the inverse Fisher information matrix. It is a generalization of the maximal information measure proposed by van Emden [16]. Let $\mathbf{I}_n = n\cdot\mathbf{I}$ denote the Fisher information matrix based on a sample of size n. The ICOMP penalty term is a function of the matrix $\mathbf{I}$ rank, trace and determinant. The minimum value of the penalty term is zero, which is reached when the variances of parameter estimators are equal and the covariances are zeros. The ICOMP criterion is very effective for regression-type models.

4. SIMULATION STUDIES

To get an idea of how each of AIC, BIC, and ICOMP performs in selecting distributional models, let us first of all conduct some simulation studies. The purpose of the simulation is to see which criterion can capture the true underlying distribution from which the data is generated from.

We are going to simulate data from each of Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull distributions, with specified parameter values. For each distribution, the simulation is done twice, the first time with a relatively small sample size of 100, and the second time with a relatively large sample size of 1,000. Thus, there are in total 12 data sets being simulated. Then we fit all six distributions to each of the 12 simulated data sets.

In **Tables 1–12**, we will present the results of the simulation studies, about the performance of all three information criteria.

In **Table 1**, AIC values of 2,142 and 2,141 are both regarded as the smallest AIC values since the difference in a magnitude of one is too small and that might be due to round up of decimals. That tiny difference is insignificant. The same logic applies to the lowest BIC values of 2,147 and 2,146. When the sample size is 100, AIC and BIC cannot separate the true Champernowne

TABLE 1 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	**2,142**	2,191	2,148	2,144	2,141	2,155
BIC	2,147	2,196	2,153	2,149	2,146	2,160
ICOMP	2,155	2,205	2,144	2,160	2,155	2,171

Data generated from Champernowne (shape = 1.5, scale = 12,000), sample size 100.

TABLE 2 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	21,823	22,085	21,859	21,911	21,833	22,181
BIC	21,833	22,095	21,869	21,921	21,843	22,190
ICOMP	21,836	22,099	21,855	21,928	21,848	22,197

Data generated from Champernowne (shape = 1.5, scale = 12,000), sample size 1,000.

TABLE 3 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	2,348	2,339	2,353	2,364	2,353	2,399
BIC	2,353	2,344	2,358	2,370	2,358	2,404
ICOMP	2,362	2,354	2,349	2,374	2,370	2,408

Data generated from Frechet (shape = 1, scale = 20,000), sample size 100.

TABLE 4 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	24,051	23,913	24,138	24,197	24,101	24,784
BIC	24,061	23,923	24,147	24,207	24,111	24,794
ICOMP	24,066	23,928	24,134	24,213	24,135	24,803

Data generated from Frechet (shape = 1, scale = 20,000), sample size 1,000.

TABLE 5 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	1,821	1,833	1,822	1,833	1,822	1,850
BIC	1,826	1,838	1,827	1,839	1,827	1,855
ICOMP	1,833	1,843	1,818	1,840	1,832	1,867

Data generated from Lognormal (location = 7, scale = 3), sample size 100.

TABLE 6 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	18,900	19,029	18,876	19,034	18,921	19,027
BIC	18,910	19,039	18,886	19,044	18,931	19,037
ICOMP	18,912	19,039	18,872	19,043	18,932	19,044

Data generated from Lognormal (location = 7, scale = 3), sample size 1,000.

TABLE 7 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	2,045	2,098	2,051	2,044	2,045	2,052
BIC	2,050	2,103	2,056	2,049	2,050	2,058
ICOMP	2,057	2,111	2,047	2,058	2,058	2,068

Data generated from Lomax (shape = 1.2, scale = 8,000), sample size 100.

TABLE 8 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	21,075	21,400	21,104	21,075	21,075	21,353
BIC	21,085	21,410	21,114	21,085	21,085	21,363
ICOMP	21,089	21,415	21,100	21,090	21,089	21,370

Data generated from Lomax (shape = 1.2, scale = 8,000), sample size 1,000.

TABLE 9 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	2,085	2,148	2,093	2,098	2,080	2,083
BIC	2,091	2,153	2,098	2,104	2,086	2,088
ICOMP	2,097	2,162	2,089	2,107	2,094	2,096

Data generated from Paralogistic (shape = 2, scale = 20,000), sample size 100.

TABLE 10 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	20,973	21,385	20,996	21,254	20,955	21,081
BIC	20,983	21,395	21,005	21,264	20,965	21,091
ICOMP	20,984	21,398	20,992	21,262	20,968	21095

Data generated from Paralogistic (shape = 2, scale = 20,000), sample size 1,000.

TABLE 11 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	2,096	2,146	2,101	2,146	2,107	2,087
BIC	2,101	2,152	2,106	2,151	2,112	2,092
ICOMP	2,111	2,160	2,097	2,156	2,120	2,105

Data generated from Weibull (shape = 0.3, scale = 5,000), sample size 100.

TABLE 12 | Simulation: Information criteria for Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	19,332	–	19,418	19,937	19,469	19,194
BIC	19,342	–	19,428	19,947	19,478	19,203
ICOMP	19,346	–	19,414	19,945	19,480	19,211

Data generated from Weibull (shape = 0.3, scale = 5,000), sample size 1,000.

model from the Paralogistic model. ICOMP tends to favor the Lognormal more than all other models, even though the true model is Champernowne instead of Lognormal. This bias can be corrected by increasing the sample size.

In **Table 2**, across each row of the three information criteria, the lowest value has been highlighted. All the three information criteria are able to distinguish the true Champernowne model from the other models, if the data is generated with a large sample size of 1,000.

From **Table 3**, AIC and BIC can determine the best model that is consistent with the underlying true model that generates the data, since the true Frechet model produces the lowest AIC and BIC values among all models. ICOMP says the Lognormal model has the smallest information complexity value, while the true Frechet model turns out to have the second lowest complexity value. This error may be due to the relatively small sample size of 100, and it can be eliminated by increasing the sample size to 1,000, which can be seen in the following table.

When the sample size is 1,000, all AIC, BIC, and ICOMP criteria are successful in identifying the Frechet model as the true model from which the data is generated. This is supported by the lowest values of all information criteria for the Frechet model in **Table 4**.

In **Table 5**, we have highlighted a few lowest values of AIC and/or BIC, ignoring the slight difference of one in the values, perhaps due to round up error. It seems that neither AIC nor BIC can separate the true Lognormal model from the other two competing models: the Champernowne model and the Paralogistic model, because all three models produce the lowest AIC and BIC values. ICOMP successfully distinguishes the true Lognormal model from all other models. This simulation study of fitting various models to Lognormal data of sample size 100 suggests that ICOMP is indeed more effective than AIC and BIC to identify the true Lognormal model when the sample size is as small as 100. But this advantage of ICOMP may disappear if we increase the sample size to 1,000, as shown in the next table.

The Lognormal model produces the lowest AIC value, the lowest BIC value, and the lowest ICOMP value, as indicated in **Table 6**. Thus, all three information criteria successfully select the true underlying model that generates the data, when the true model is Lognormal and the sample size is 1,000.

From **Table 7**, we can tell that AIC and BIC fail to distinguish among the Champernowne model, the true Lomax model, and the Paralogistic model. We ignore the difference of one in the values that may be due to round up error, and those highlighted values are treated as the smallest ones in that particular row. ICOMP has a biased favor toward the Lognormal model, suggested by the lowest ICOMP value of 2,047 produced by the Lognormal model. However, the true model is Lomax instead of Lognormal. This type of error will go away if the sample size is bigger as indicated in the coming table.

When the true model that generates the data is Lomax, even if the sample size is as large as 1,000, none of AIC, BIC, or ICOMP seems to be able to distinguish the competing three different models: they are the Champernowne model, the true Lomax model, and the Paralogistic model, since they all produce the lowest information criteria values across each row in **Table 8**. The magnitude of one in the difference of the highlighted values may be subject to round up issues, and such tiny differences do not say much about distinguishing the models.

When the sample size is 100, AIC and BIC can weakly identify the true Paralogistic model that generates the data, since the Paralogistic model produces the smallest AIC and BIC values in **Table 9**. The reason we say weakly identifying the true model is because the AIC and BIC values of a competing Weibull model are close to those of the true model. However, ICOMP makes a wrong decision to select the Lognormal model (with the lowest ICOMP value of 2,089), even though the ICOMP value (2,094) of the true Paralogistic model comes as the next lowest. ICOMP will make a right decision when the sample size is increased to 1,000 as in the subsequent table.

When the sample size is as large as 1,000, all AIC, BIC, and ICOMP can correctly identify that the true underlying process that generates the data is the Paralogistic model, which has the lowest information criteria values as highlighted in **Table 10**.

From **Table 11**, you can tell that AIC and BIC are able to identify the true Weibull model that has generated the data of a relatively small sample size 100, since the Weibull model produces the lowest AIC and BIC values. But ICOMP fails to do so, and ICOMP erroneously picks the Lognormal model (with the lowest ICOMP value of 2,097) and puts the true Weibull model (with the next lowest ICOMP value of 2,105) in the second place.

Fortunately, when the sample size increases from 100 to 1000, ICOMP seems to correct its mistake as of wrongly picking the Lognormal model in the previous table, and now ICOMP also identifies the true Weibull model that has generated the data. AIC and BIC are still working well. All three information criteria end up to have the smallest value for the Weibull model. By the way, in this **Table 12**, there are no values produced for the Frechet model, since the numerical procedure in maximizing the log-likelihood does not converge. Obviously, when the sample size is 1,000, the Frechet model is definitely not a good fit for the data generated from the Weibull model.

Even if we have only presented 12 tables as above, more simulation studies can be done across a wide range of the parameter values for each distribution, and ICOMP still exhibits a tendency to favor the Lognormal model when the sample size is small. That sort of tendency can disappear when the sample size gets large enough.

Let us make a remark here. When the true data generating process is a Lognormal model, ICOMP beats AIC and BIC if the sample size is small. When the data of small sample size is simulated from another distribution, ICOMP might lose the competition to AIC and/or BIC.

5. CASE STUDY: EXTERNAL FRAUD RISK

In this section, we illustrate the model performance on the real data. We walk through the entire process of modeling, beginning with model fitting and model validation, then Value-at-Risk (VaR) estimation, and ends up with model selection using various information criteria.

We have collected data for operational losses of external fraud type in retail banking in branches of major commercial banks of China in 2009–2015. We have recorded the amount of principal indemnification involved in each event. If a savings account holder or a debit card holder loses their money of deposit due to external fraud, a certain proportion of the original loss amount of principal deposit may be indemnified or reimbursed by the bank to mitigate the loss of bank customers. That proportion is determined by the court in accord with how much responsibility the bank is supposed to assume during the events of external fraud. Usual proportion numbers could be 100, 90, 80, 70, 50, or 0%. There are 181 data points that represent the cost of debit card or savings account external fraud events for branches of major commercial banks in China. The cost measure is in Chinese Yuan (RMB) currency. Even though the data have been collected over a span of a few years, the loss sizes are not scaled for inflation. Also the legal costs, involved in the law sue process, are not included. To illustrate the impact of data modeling threshold on the considered models, we split the data set into two portions: losses that are at least 26,000 RMB, which will be regarded as observed and used for model building and VaR estimation; and losses that are below 26,000 RMB, which will be treated as unrecorded by the bank or unobserved. The modeling threshold is a different concept from the reporting threshold: the former is the threshold chosen by the model builders, and the latter is the threshold chosen by each individual bank. We use a modeling threshold of 26,000 RMB just for demonstration purposes. Such a choice of modeling threshold results in 103 observed losses. A quick summary statistics of the 103 observed data shows that it is right-skewed and tentatively heavy-tailed, with the first quartile 40,085, median 72,000, and the third quartile 220,879; its mean is 342,838, standard deviation 944,314, and skewness 5.731.

Such a real dataset is used to conduct the case study, and we consider three distributional models; they are Champernowne, Lomax, and Lognormal models. MLE estimators are solved numerically, when explicit formulas can not be obtained for maximization of the log-likelihood functions. In **Table 13**, we report the resulting MLE parameter estimates. Further, we assess all models using the visual inspection in **Figures 1–6** and formal goodness-of-fit test statistics. The KS and AD goodness-of-fit test statistics are given in **Table 14**. The VaR estimates are provided in **Table 15**. Finally the information criteria values are summarized in **Table 16**.

5.1. Model Fitting

Using the 103 observations of the aforementioned construction of dataset, exceeding (inclusively) the modeling threshold 26, 000 Chinese RMB, we fit the Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull distributions to the data by the truncated approach. The values of parameter estimators are obtained using numerical procedure, when there are no closed form formulas for the parameter estimators. The results are reported in the following table.

For the truncated Champernowne model, the shape parameter estimate is 0.84, and the scale parameter estimate 11,654 is an estimate of the median of the fitted Champernowne distribution. For the truncated Frechet model, the shape parameter estimate is 0.812, and the scale parameter estimate is 16,440. For the truncated Lognormal model, the location parameter estimate of the Lognormal distribution is 6.66, and the scale parameter estimate is 2.952. For the truncated Lomax model, the shape parameter estimate is 0.82, and the scale parameter estimate is

TABLE 13 | Parameter MLEs using truncated approach of the Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

Parameter	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
Shape	0.84	0.812	–	0.82	0.912	0.285
Scale	11,654	16,440	2.952	12,812	12,199	5,500
Location	–	–	6.66	–	–	–

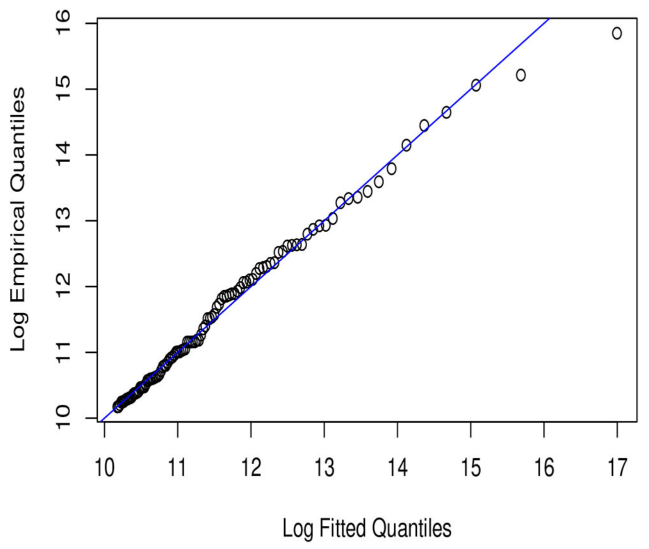


FIGURE 1 | Champernowne QQ-plot.

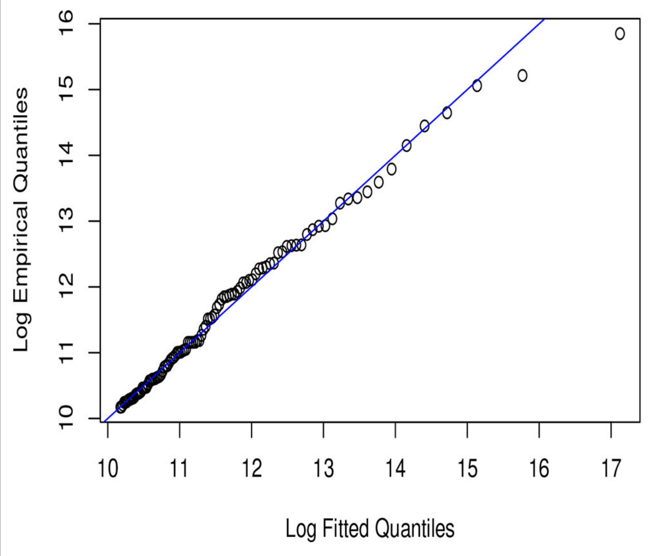


FIGURE 2 | Frechet QQ-plot.

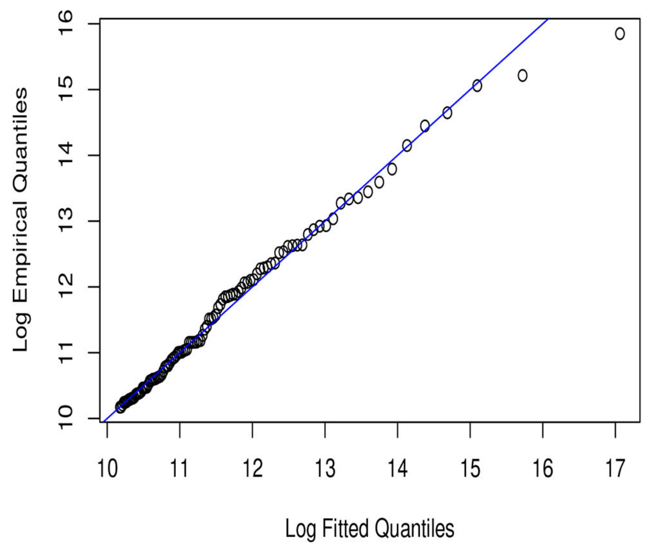


FIGURE 4 | Lomax QQ-plot.

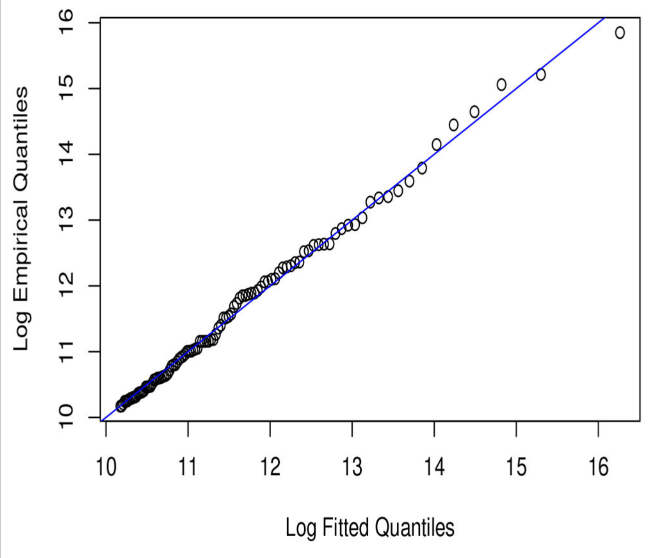


FIGURE 3 | Lognormal QQ-plot.

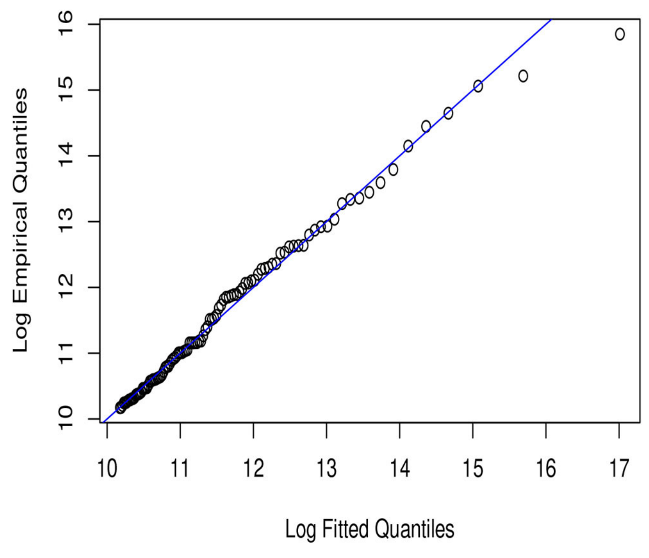


FIGURE 5 | Paralogistic QQ-plot.

12,812, which in general is not the median estimate unless the shape parameter were exactly equal to one. The shape parameter estimate of the Champernowne model 0.84 seems pretty close to the shape parameter estimate 0.82 of the Lomax model. If we compare the cdf of the Champernowne distribution with that of the Lomax distribution, we could find that they would coincide when the shape parameter α of both were equal to one. For the truncated Paralogistic model, the shape parameter estimate is 0.912, and the scale parameter estimate is 12,199. For the truncated Weibull model, the shape parameter estimate is 0.285, and the scale parameter estimate is 5,500.

5.2. Model Validation

To validate the fitted models we employ visual inspection tools like quantile-quantile plots (QQ-plots) and furthermore two goodness-of-fit test statistics, Kolmogorov-Smirnov (KS) test statistic and Anderson-Darling (AD) test statistic.

In **Figures 1–6**, we present plots of the fitted-versus-empirical quantiles for the six truncated distributional models. In all the plots both fitted and empirical quantiles have been taken the logarithmic transformation. It seems hard to distinguish the truncated Champernowne (**Figure 1**), Frechet (**Figure 2**), Lomax (**Figure 4**), and Paralogistic (**Figure 5**) models, as indicated by the graphical inspection of QQ-plots. The truncated

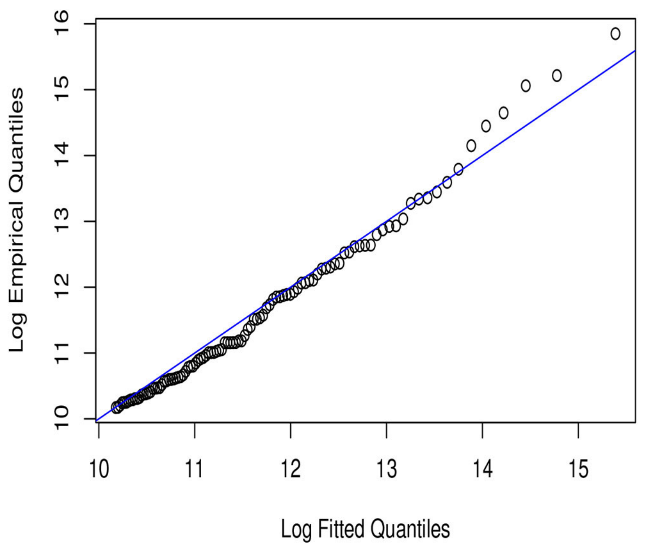


FIGURE 6 | Weibull QQ-plot.

TABLE 14 | The KS and AD statistics for the fitted models, using truncated approach.

Test statistics	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
KS	0.063	0.066	0.054	0.068	0.064	0.107
AD	0.309	0.309	0.257	0.319	0.320	1.285

TABLE 15 | External Fraud Risk: VaR(β) estimates, measured in millions and based on the fitted models, using truncated approach.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
VaR(0.99)	2.759	4.737	0.746	3.406	3.064	1.173
VaR(0.995)	5.092	11.154	1.292	7.913	7.074	1.920
VaR(0.999)	18.905	81.098	3.804	55.834	49.06	4.874

TABLE 16 | External Fraud Risk: Information criteria for truncated Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models.

	Champernowne	Frechet	Lognormal	Lomax	Paralogistic	Weibull
AIC	2,676	2,676	2,675	2,676	2,676	2,678
BIC	2,681	2,682	2,680	2,682	2,682	2,683
ICOMP	2,695	2,694	2,676	2,695	2,694	2,697

Lognormal (**Figure 3**) model looks good overall by examining the QQ plot visually. The truncated Weibull (**Figure 6**) model has the QQ plot a little bit further away from the identity line compared to the other five models.

Looking at **Figure 1**, all points except for the two most extremely large observations lie nearly on or very close to the identity line. Compared with **Figure 1** fitting the truncated Champernowne model, we see that **Figure 3** indicates a slightly better fit of the truncated Lognormal model in the most extreme right tail (for the largest two observations) because those two points in **Figure 3** get closer to the identity line compared to in **Figure 1**. However, such improvement comes at the cost of a slightly worse fit of the next few observations just below the largest two.

Visual inspection of **Figures 1**, **4** tells us that they look very similar to each other. This is consistent with what we have observed in the closeness of parameter estimates in **Table 13** between the truncated Champernowne model and the truncated Lomax model.

The KS goodness-of-fit test statistic measures the maximum absolute distance between the fitted cdf and the empirical cdf. The AD goodness-of-fit test statistic measures the cumulative weighted quadratic distance (placing more weight on tails) between the fitted cdf and the empirical cdf. The KS and AD statistics have been evaluated using the MLE parameter estimates from **Table 13**. The following is the table of the KS statistics and the AD statistics for each of the fitted models.

From **Table 14**, it looks like that all fitted models have low KS and AD values except that the Weibull model produces relatively higher values of KS and AD statistics than the other models. This is consistent with the visual inspection of the QQ-plots.

5.3. VaR Estimates

The final product of operational risk modeling is VaR estimation, built upon the parameter estimates of the fitted models. The concept of VaR is useful in both finance and insurance, and let's define VaR as follows: Let $0 < \beta < 1$. VaR is defined as

$$VaR(\beta) = F^{-1}(\beta), \tag{22}$$

where $F^{-1}(\beta)$ denotes a quantile evaluated at level β of the cumulative distribution function F, and F^{-1} is a notation used for the generalized inverse of F.

Having gone through the model fitting and model validation, we now are ready to compute the estimates of VaR(β) for all three models. We summarize the results in **Table 15**.

From **Table 15** we see that the considered six models, five of which exhibited nearly as good fit as one another to the studied data, produce substantially different VaR estimates, especially at the very extreme right tail. For example, the Frechet model produces the most conservative VaR estimates at all three levels. The Frechet distribution comes from the family of distributions to model extreme values. The fitted Frechet model turns out to be more heavy-tailed than the other fitted models. On the other hand, the fitted Lognormal model ends up to be less heavy-tailed than the others, and Lognormal VaR values are smaller than the VaR values of other models. For instance, the 99.9% VaR of the truncated Lognormal model is 3.8 million RMB; while it is 18.9 million RMB for the truncated Champernowne model and 55.8 million RMB for the truncated Lomax model. The 99% VaR and 99.5% VaR of the six models maintain the same order of ranking, but the magnitude of gap is not as much as in the highest level of 99.9% VaR. Despite producing very different VaR estimates, we will see in the following section that the six models are very close in terms of the values of information criteria, with an exception for ICOMP of the Lognormal model.

5.4. Information Criteria Results

Finally, in **Table 16** we present the values of all information criteria considered in this paper for the truncated Champernowne, Frechet, Lognormal, Lomax, Paralogistic, and Weibull models. The log-likelihood function has been obtained using the truncated distributional approach. Likewise, the Fisher information matrix $\mathbf{I}(\theta_1, \ldots, \theta_k)$ has been derived using the truncated likelihood for the respective parametric distributions.

As we can see from **Table 16**, the six models are indistinguishable using traditional information criteria such as AIC and BIC, since the values of each criterion are very close for all models. In particular, a slight difference in the AIC values with a magnitude of 1 out of more than 2,000 provides little evidence. It is very similar if we look at the difference of 1 or 2 in the BIC values. The use of the more refined information measure ICOMP does not help among the truncated Champernowne, Frechet, Lomax, Paralogistic, and Weibull models, but has the ability to distinguish the truncated Lognormal model from the other five models. The ICOMP favors the truncated Lognormal model since it produces significantly lower value of ICOMP than the other five models. We can see there is a difference of no <18 out of roughly twenty six hundred, which contrasts that tiny difference in the values of AIC and/or BIC.

6. CONCLUDING REMARKS

In this paper, we have studied the problem of model selection in operational risk modeling, which arises due to that various truncated models are all validated but cannot be distinguished. Through the numerical illustrations of simulation studies and a case study in the previous two sections, the performances of the traditional well-known model selection criteria such as Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) and another information criterion based on Information Complexity (ICOMP) have been assessed and compared.

We summarize our conclusions as follows. Is ICOMP really credible when choosing a Lognormal model? It depends on the size of the sample, and also relies on whether or not the Lognormal model gets kicked out of the pool of candidate models by AIC and/or BIC. Simulation studies have shown that when the true underlying model is Lognormal and the sample size is 100, ICOMP can successfully identify the Lognormal as the true model, while AIC and/or BIC cannot distinguish the Lognormal model from some of the other models (for example, the Champernowne model, and the Paralogistic model). However, when the sample size in the simulation increases to 1,000, AIC and/or BIC can also successfully separate the true Lognormal model from other models. In this way, when the sample size is 1,000, ICOMP loses its competitive advantage. On the other hand, when the underlying true model is Champernowne, Frechet, Lomax, Paralogistic, or Webull, instead of Lognormal, if the sample size is set to be 100, ICOMP will still erroneously choose Lognormal as the winning model, while AIC and/or BIC will not make such a mistake. Fortunately, this disadvantage of ICOMP will disappear when the sample size increases to 1,000. The reason why ICOMP prefers the Lognormal model so much is probably because the information complexity of the Lognormal variance covariance matrix is simpler than other models.

In order to make full use of the advantages of all the three information criteria and avoid the disadvantages of them, a large sample size is desirable. Otherwise, caution has to be used when the sample size of data at hand is small, for which we propose two stages of model selection. In the first stage, we use AIC and/or BIC to select the best model. If it turns out that AIC and/or BIC are successful, then we use ICOMP to strengthen the selection of the best model. Otherwise, if AIC and/or BIC fails, we will enter the second stage. Due to the failure of AIC and/or BIC, there must be several indistinguishable competing models. We consider two different situations. One situation is that AIC and/or BIC have kicked the Lognormal model out from the candidate pool, if so then we cannot continue to use ICOMP to avoid ICOMP choosing the Lognormal by mistake. The other situation is that after using AIC and/or BIC, the Lognormal is still surviving in the pool of candidate models, then ICOMP may be used further.

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/s.

AUTHOR CONTRIBUTIONS

The author confirms being the sole contributor of this work and has approved it for publication.

ACKNOWLEDGMENTS

The author thanks the referees for their review and comments which have helped the author significantly improve the paper.

REFERENCES

1. Yu D, Brazauskas V. Model uncertainty in operational risk modeling due to data truncation: a single risk case. *Risks.* (2017) **5**:49. doi: 10.3390/risks5030049
2. Basel Coordination Committee. Supervisory guidance for data, modeling, and model risk management under the operational risk advanced measurement approaches. *Basel Coord Committ Bull.* (2014) **14**:1–17. Available online at: https://www.federalreserve.gov/bankinforeg/basel/files/bcc1401.pdf.
3. Isaksson D. *Modelling IT risk in banking industry: A study on how to calculate the aggregate loss distribution of IT risk* (Thesis). Jönköping University, Jönköping, Sweden (2017).
4. Svensson KP. *A Bayesian approach to modeling operational risk when data is scarce* (Thesis). Lund University, Lund, Sweden (2015).
5. Bolance C, Guillen M, Gustafsson J, Nielsen JP. *Quantitative Operational Risk Models*. Boca Raton, FL: CRC Press (2012).

6. Cavallo A, Rosenthal B, Wang X, Yan J. Treatment of the data collection threshold in operational risk: a case study with the Lognormal distribution. *J Oper Risk.* (2012) **7**:3–38. doi: 10.21314/JOP.2012.101
7. Horbenko N, Ruckdeschel P, Bae T. Robust estimation of operational risk. *J Oper Risk.* (2011) **6**:3–30.
8. Klugman SA, Panjer HH, Willmot GE. *Loss Models: From Data to Decisions.* 4th Edn. Hoboken, NJ: Wiley (2012).
9. Champernowne DG. The Oxford meeting, September 25–29. *Econometrica.* (1936) 5:361–83.
10. Champernowne DG. The graduation of income distributions. *Econometrica.* (1952) **20**:591–615.
11. Arnold BC. *Pareto Distributions.* 2nd Edn. London: Chapman & Hall (2015).
12. Akaike H. Information theory and an extension of the maximum likelihood principle. In: *Second International Symposium on Information Theory*. Petrox BN, Csaki F, Editors. Budapest: Academiai Kiado (1973). p. 267–81.
13. Schwarz G. Estimating the dimension of a model. *Ann Stat.* (1978) **6**:461–4.
14. Bozdogan H. ICOMP: A new model selection criteria. In: Bock H, Editor. *Classification and Related Methods of Data Analysis.* Amsterdam: Elsevier Science Publishers B.V. (1988) 599–608.
15. Bozdogan H. On the information-based measure of covariance complexity and its application to the evaluation of multivariate linear models. *Commun Stat Theor Methods.* (1990) **19**:221–78.
16. van Emden MH. An analysis of complexity. In: *Mathematical Centre Tracts, Vol. 35*. Mathematisch Centrum, Amsterdam: Elsevier (1971).

Conflict of Interest: The author declares 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 Applied Mathematics and Statistics

ORIGINAL RESEARCH
published: 11 February 2021
doi: 10.3389/fams.2020.551138

Financial Forecasting With α-RNNs: A Time Series Modeling Approach

Matthew Dixon[1,2]* *and Justin London*[2]

[1]*Department of Applied Math, Illinois Institute of Technology, Chicago, IL, United States,* [2]*Stuart School of Business, Illinois Institute of Technology, Chicago, IL, United States*

The era of modern financial data modeling seeks machine learning techniques which are suitable for noisy and non-stationary big data. We demonstrate how a general class of exponential smoothed recurrent neural networks (α-RNNs) are well suited to modeling dynamical systems arising in big data applications such as high frequency and algorithmic trading. Application of exponentially smoothed RNNs to minute level Bitcoin prices and CME futures tick data, highlight the efficacy of exponential smoothing for multi-step time series forecasting. Our α-RNNs are also compared with more complex, "black-box", architectures such as GRUs and LSTMs and shown to provide comparable performance, but with far fewer model parameters and network complexity.

Keywords: recurrent neural networks, exponential smoothing, bitcoin, time series modeling, high frequency trading

OPEN ACCESS

Edited by:
Glenn Fung,
Independent Researcher, Madison,
United States

Reviewed by:
Alex Jung,
Aalto University, Finland
Abhishake Rastogi,
University of Potsdam, Germany

****Correspondence:***
Matthew Dixon
matthew.dixon@iit.edu

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *12 April 2020*
Accepted: *13 October 2020*
Published: *11 February 2021*

Citation:
Dixon M and London J (2021) Financial Forecasting With α-RNNs: A Time Series Modeling Approach. Front. Appl. Math. Stat. 6:551138. doi: 10.3389/fams.2020.551138

1. INTRODUCTION

Recurrent neural networks (RNNs) are the building blocks of modern sequential learning. RNNs use recurrent layers to capture non-linear temporal dependencies with a relatively small number of parameters (Graves, 2013). They learn temporal dynamics by mapping an input sequence to a hidden state sequence and outputs, via a recurrent layer and a feedforward layer.

There have been exhaustive empirical studies on the application of recurrent neural networks to prediction from financial time series data such as historical limit order book and price history (Borovykh et al., 2017; Dixon, 2018; Borovkova and Tsiamas, 2019; Chen and Ge, 2019; Mäkinen et al., 2019; Sirignano and Cont, 2019). Sirignano and Cont (2019) find evidence that stacking networks leads to superior performance on intra-day stock data combined with technical indicators, whereas (Bao et al., 2017) combine wavelet transforms and stacked autoencoders with LSTMs on OHLC bars and technical indicators. Borovykh et al. (2017) find evidence that dilated convolutional networks out-perform LSTMs on various indices. Dixon (2018) demonstrate that RNNs outperform feed-forward networks with lagged features on limit order book data.

There appears to be a chasm between the statistical modeling literature (see, e.g., Box and Jenkins 1976; Kirchgässner and Wolters 2007; Hamilton 1994) and the machine learning literature (see. e.g., Hochreiter and Schmidhuber 1997; Pascanu et al. 2012; Bayer 2015). One of the main contributions of this paper is to demonstrate how RNNs, and specifically a class of novel exponentially smoothed RNNs (α-RNNs), proposed in (Dixon, 2021), can be used in a financial time series modeling framework. In this framework, we rely on statistical diagnostics in combination with cross-validation to identify the best choice of architecture. These statistical tests characterize stationarity and memory cut-off length and provide insight into whether the data is suitable for longer-term forecasting and whether the model must be non-stationary.

In contrast to state-of-the-art RNNs such as LSTMs and Gated Recurrent Units (GRUs) (Chung et al., 2014), which were designed primarily for speech transcription, the proposed class of α-RNNs is designed for times series forecasting using numeric data. α-RNNs not only alleviate the gradient problem but are designed to i) require fewer parameters and numbers of recurrent units and considerably fewer samples to attain the same prediction accuracy[1]; ii) support both stationary and non-stationary times series[2]; and iii) be mathematically accessible and characterized in terms of well known concepts in classical time series modeling, rather than appealing to logic and circuit diagrams.

As a result, through simple analysis of the time series properties of α-RNNs, we show how the value of the smoothing parameter, α, directly characterizes its dynamical behavior and provides a model which is both more intuitive for time series modeling than GRUs and LSTMs while performing comparably. We argue that for time series modeling problems in finance, some of the more complicated components, such as reset gates and cell memory present in GRUs and LSTMs but absent in α-RNNs, may be redundant for our data. We exploit these properties in two ways i) first, we using a statistical test for stationarity to determine whether to deploy a static or dynamic α-RNN model; and ii) we are able to reduce the training time, memory requirements for storing the model, and in general expect α-RNN to be more accurate for shorter time series as they require less training data and are less prone to over-fitting. The latter is a point of practicality as many applications in finance are not necessarily big data problems, and the restrictive amount of data favors an architecture with fewer parameters to avoid over-fitting.

The remainder of this paper is outlined as follows. **Section 2** introduces the static α-RNN. **Section 3** bridges the time series modeling approach with RNNs to provide insight on the network properties. **Section 4** introduces a dynamic version of the model and illustrates the dynamical behavior of α. Details of the training, implementation and experiments using financial data together with the results are presented in **Section 5**. Finally, **Section 6** concludes with directions for future research.

2. α-RNNS

Given auto-correlated observations of covariates or predictors, $\boldsymbol{x}_t$, and continuous responses $\boldsymbol{y}_t$ at times $t = 1, \ldots, N$, in the time series data $\mathcal{D} := \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^N$, our goal is to construct an m-step ($m > 0$) ahead times series predictor, $\hat{\boldsymbol{y}}_{t+m} = F(\underline{\boldsymbol{x}}_t)$, of an observed target, $\boldsymbol{y}_{t+m} \in \mathbb{R}^n$, from a p length input sequence $\underline{\boldsymbol{x}}_t$

$$\boldsymbol{y}_{t+m} := F(\underline{\boldsymbol{x}}_t) + u_t, \qquad \text{where} \qquad \underline{\boldsymbol{x}}_t := \left\{\boldsymbol{x}_{t-p+1}, \ldots, \boldsymbol{x}_t\right\},$$

$\boldsymbol{x}_{t-j} =: L^j[\boldsymbol{x}_t]$ is the j^{th} lagged observation of $\mathbf{x}_t \in \mathbb{R}^d$, for $j = 0, \ldots, p-1$ and u_t is the homoscedastic model error at time t. We introduce the α-RNN model (as shown in **Figure 1**):

$$\hat{\boldsymbol{y}}_{t+m} = F_{W,b,\alpha}(\underline{\boldsymbol{x}}_t) \tag{1}$$

where $F_{W,b,\alpha}(\underline{\boldsymbol{x}}_t)$ is an $\alpha \in [0,1]$ smoothed RNN with weight matrices $W := (W_h, U_h, W_y)$, where the input weight matrix $W_h \in \mathbb{R}^{H\times d}$, the recurrence weight matrix $U_h \in \mathbb{R}^{H\times H}$, the output weight matrix $W_y \in \mathbb{R}^{n\times H}$, and H is the number of hidden units. The hidden and output bias vectors are given by $\boldsymbol{b} := (\boldsymbol{b}_h, \boldsymbol{b}_y)$.

For each index in a sequence, s = t-p+2, . . . ,t, forward passes repeatedly update a hidden *internal* state $\hat{h}_s \in \mathbb{R}^H$, using the following model:

$$\begin{aligned} \text{(output)} \quad & \hat{\boldsymbol{y}}_{t+m} = W_y \hat{\boldsymbol{h}}_t + \boldsymbol{b}_y, \\ \text{(hidden state update)} \quad & \hat{\boldsymbol{h}}_s = \sigma\left(U_h \tilde{\boldsymbol{h}}_{s-1} + W_h \boldsymbol{x}_s + \boldsymbol{b}_h\right), \\ & s = t-p+2, \ldots, t \\ \text{(smoothing)} \quad & \tilde{\boldsymbol{h}}_s = \alpha \hat{\boldsymbol{h}}_s + (1-\alpha)\tilde{\boldsymbol{h}}_{s-1}, \end{aligned}$$

where $\sigma() := tanh()$ is the activation function and $\tilde{\boldsymbol{h}}_s \in \mathbb{R}^H$ is an exponentially smoothed version of the hidden state $\hat{\boldsymbol{h}}_s$, with the starting condition in each sequence, $\hat{\boldsymbol{h}}_{t-p+1} = \sigma(W_h \boldsymbol{x}_{t-p+1})$.

3. UNIVARIATE TIMES SERIES MODELING WITH ENDOGENOUS FEATURES

This section bridges the time series modeling literature (Box and Jenkins, 1976; Kirchgässner and Wolters, 2007; Li and Zhu, 2020) and the machine learning literature. More precisely, we show the conditions under which plain RNNs are identical to autoregressive time series models and thus how RNNs generalize autoregressive models. Then we build on this result by applying time series analysis to characterize the behavior of static α-RNNs.

We shall assume here for ease of exposition that the time series data is univariate and the predictor is endogenous[3], so that the data is $\mathcal{D} := \{y_t\}_{t=1}^N$.

We find it instructive to show that plain RNNs are non-linear AR(p) models. For ease of exposition, consider the simplest case of a RNN with one hidden unit, $H = 1$. Without loss of generality, we set $U_h = W_h = \phi$, $W_y = 1$, $b_h = 0$ and $b_y = \mu$. Under backward substitution, a plain-RNN, $F_{W,b}(\underline{\boldsymbol{x}}_t)$, with sequence length p, is a non-linear auto-regressive, $NAR(p)$, model of order p: :

$$\begin{aligned} \hat{h}_{t-p+1} &= \sigma\left(\phi y_{t-p+1}\right) \\ \hat{h}_{t-p+2} &= \sigma\left(\phi \hat{h}_{t-p+1} + \phi y_{t-p+2}\right) \\ \ldots &= \ldots \\ \hat{h}_t &= \sigma\left(\phi \hat{h}_{t-1} + \phi y_t\right) \\ \hat{y}_{t+m} &= \hat{h}_t + \mu \end{aligned}$$

then

$$\hat{y}_{t+m} = \mu + \sigma(\phi(1 + \sigma(\phi(L + \sigma(\phi(L^2 + \cdots + \sigma(\phi L^{p-1}) \cdots)[y_t]. \tag{2}$$

[1]Sample complexity bounds for RNNs have recently been derived by (Akpinar et al., 2019). Theorem 3.1 shows that for a recurrent units, inputs of length at most b, and a single real-valued output unit, the network requires only $\mathcal{O}(a^4 b/\epsilon^2)$ samples in order to attain a population prediction error of ε. Thus the more recurrent units required, the larger the amount of training data needed.

[2]By contrast, plain RNNs model stationary time series, and GRUs/LSTMs model non-stationary, but no hybrid exists which provides the modeler with the control to deploy either.

[3]The sequence of features is from the same time series as the predictor hence $n = d = 1$.

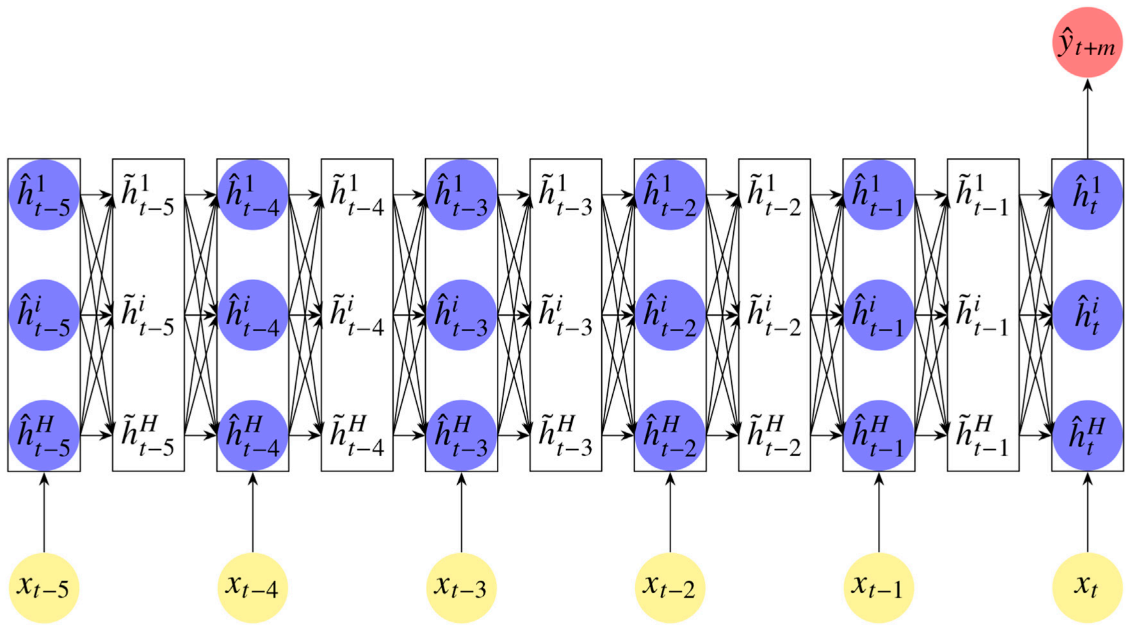

$\hat{y}_{t+m}$
$\hat{h}^1_{t-5}$ $\tilde{h}^1_{t-5}$ $\hat{h}^1_{t-4}$ $\tilde{h}^1_{t-4}$ $\hat{h}^1_{t-3}$ $\tilde{h}^1_{t-3}$ $\hat{h}^1_{t-2}$ $\tilde{h}^1_{t-2}$ $\hat{h}^1_{t-1}$ $\tilde{h}^1_{t-1}$ $\hat{h}^1_{t}$
$\hat{h}^i_{t-5}$ $\tilde{h}^i_{t-5}$ $\hat{h}^i_{t-4}$ $\tilde{h}^i_{t-4}$ $\hat{h}^i_{t-3}$ $\tilde{h}^i_{t-3}$ $\hat{h}^i_{t-2}$ $\tilde{h}^i_{t-2}$ $\hat{h}^i_{t-1}$ $\tilde{h}^i_{t-1}$ $\hat{h}^i_{t}$
$\hat{h}^H_{t-5}$ $\tilde{h}^H_{t-5}$ $\hat{h}^H_{t-4}$ $\tilde{h}^H_{t-4}$ $\hat{h}^H_{t-3}$ $\tilde{h}^H_{t-3}$ $\hat{h}^H_{t-2}$ $\tilde{h}^H_{t-2}$ $\hat{h}^H_{t-1}$ $\tilde{h}^H_{t-1}$ $\hat{h}^H_{t}$
x_{t-5} x_{t-4} x_{t-3} x_{t-2} x_{t-1} x_t

FIGURE 1 | An illustrative example of an α-RNN with an alternating hidden recurrent layer (with blue nodes) and a smoothing layer (white block), "unfolded" over a sequence of six time steps. Each lagged feature $\boldsymbol{x}_{t-i}$ in the sequence $\underline{\boldsymbol{x}}_t$ is denoted by the yellow nodes. The hidden recurrent layer contains H units (blue nodes) and the ith output, after smoothing, at time step t is denoted by $\tilde{\boldsymbol{h}}^i_t$. At the last time step t, the hidden units connect to a single unactivated output unit to give $\hat{y}_{t+m}$ (red node).

When the activation is the identity function $\sigma := I_d$, then we recover the AR(p) model

$$\hat{y}_{t+m} = \mu + \sum_{i=0}^{p-1} \phi_{i+1} L^i[y_t], \phi_i := \phi^i. \tag{3}$$

with geometrically decaying autoregressive coefficients when $|\phi| < 1$.

The α-RNN(p) is almost identical to a plain RNN, but with an additional scalar smoothing parameter, α, which provides the recurrent network with "long-memory"[4]. To see this, let us consider a one-step ahead univariate α-RNN(p) in which the smoothing parameter is fixed and $H = 1$.

This model augments the plain-RNN by replacing $\hat{h}_{s-1}$ in the hidden layer with an exponentially smoothed hidden state $\tilde{h}_{s-1}$. The effect of the smoothing is to provide infinite memory when $\alpha \neq 1$. For the special case when $\alpha = 1$, we recover the plain RNN with short memory of length $p \ll N$.

We can easily verify this informally by simplifying the parameterization and considering the unactivated case. Setting $b_y = b_h = 0$, $U_h = W_h = \phi \in \mathbb{R}$ and $W_y = 1$:

$$\hat{y}_{t+1} = \hat{h}_t, \tag{4}$$
$$= \phi\left(\tilde{h}_{t-1} + y_t\right), \tag{5}$$
$$= \phi\left(\alpha\hat{h}_{t-1} + (1-\alpha)\tilde{h}_{t-2} + y_t\right), \tag{6}$$

with the starting condition in each sequence, $\hat{h}_{t-p+1} = \phi y_{t-p+1}$. With out loss of generality, consider $p = 2$ lags in the model so that $\hat{h}_{t-1} = \phi y_{t-1}$. Then

$$\hat{h}_t = \phi\left(\alpha\phi y_{t-1} + (1-\alpha)\tilde{h}_{t-2} + y_t\right) \tag{7}$$

and the model can be written in the simpler form

$$\hat{y}_{t+1} = \phi_1 y_t + \phi_2 y_{t-1} + \phi(1-\alpha)\tilde{h}_{t-2}, \tag{8}$$

with auto-regressive weights $\phi_1 := \phi$ and $\phi_2 := \alpha\phi^2$. We now see that there is a third term on the RHS of **Eq. 8** which vanishes when $\alpha = 1$ but provides infinite memory to the model since $\tilde{h}_{t-2}$ depends on y_1, the first observation in the whole time series, not just the first observation in the sequence. To see this, we unroll the recursion relation in the exponential smoother:

$$\tilde{h}_{t+1} = \alpha\sum_{s=0}^{t-1}(1-\alpha)^s\hat{h}_{t-s} + (1-\alpha)^t y_1. \tag{9}$$

where we used the property that $\tilde{h}_1 = y_1$. It is often convenient to characterize exponential smoothing by the **half-life**[5]. To gain further insight on the memory of the network, Dixon (2021) study the partial auto-correlations of the process $\hat{y}_{t+m} + u_t$ to characterize the memory and derive various properties and constraints needed for network stability and sequence length selection.

[4]Long memory refers to autoregressive memory beyond the sequence length. This is also sometimes referred to as "stateful". For avoidance of doubt, we are not suggesting that the α-RNN has an additional cellular memory, as in LSTMs.

[5]The half-life is the number of lags needed for the coefficient $(1-\alpha)^s$ to equal a half, which is $s = -1/log_2(1-\alpha)$.

4. MULTIVARIATE DYNAMIC α-RNNS

We now return to the more general multivariate setting as in **Section 2**. The extension of RNNs to dynamical time series models, suitable for non-stationary time series data, relies on dynamic exponential smoothing. This is a time dependent, convex, combination of the smoothed output, $\tilde{\boldsymbol{h}}_t$, and the hidden state $\hat{\boldsymbol{h}}_t$:

$$\tilde{\boldsymbol{h}}_{t+1} = \boldsymbol{\alpha}_t \circ \hat{\boldsymbol{h}}_t + (1 - \boldsymbol{\alpha}_t) \circ \tilde{\boldsymbol{h}}_t, \tag{10}$$

where $\circ$ denotes the Hadamard product between vectors and where $\boldsymbol{\alpha}_t \in [0,1]^H$ denotes the dynamic smoothing factor which can be equivalently written in the one-step-ahead forecast of the form

$$\tilde{\boldsymbol{h}}_{t+1} = \tilde{\boldsymbol{h}}_t + \boldsymbol{\alpha}_t \circ \left(\hat{\boldsymbol{h}}_t - \tilde{\boldsymbol{h}}_t\right). \tag{11}$$

Hence the smoothing can be viewed as a dynamic form of latent forecast error correction. When $(\boldsymbol{\alpha}_t)_i = 0$, the i^{th} component of the latent forecast error is ignored and the smoothing merely repeats the i^{th} component of the current hidden state $(\tilde{\boldsymbol{h}}_t)_i$, which enforces the removal of the i^{th} component from the memory. When $(\boldsymbol{\alpha}_t)_i = 1$, the latent forecast error overwrites the current i^{th} component of the hidden state $(\tilde{\boldsymbol{h}}_t)_i$. The smoothing can also be viewed as a weighted sum of the lagged observations, with lower or equal weights, $\boldsymbol{\alpha}_{t-s} \circ \prod_{r=1}^{s} (1 - \boldsymbol{\alpha}_{t-r+1})$ at the $s \geq 1$ lagged hidden state, $\hat{\boldsymbol{h}}_{t-s}$:

$$\tilde{\boldsymbol{h}}_{t+1} = \boldsymbol{\alpha}_t \circ \hat{\boldsymbol{h}}_t + \sum_{s=1}^{t-1} \boldsymbol{\alpha}_{t-s} \circ \prod_{r=1}^{s} (1 - \boldsymbol{\alpha}_{t-r+1}) \circ \hat{\boldsymbol{h}}_{t-s} + g(\alpha),$$

where $g(\alpha) := \prod_{r=0}^{t-1} (1 - \boldsymbol{\alpha}_{t-r}) \circ \tilde{\boldsymbol{y}}_1$. Note that for any $(\boldsymbol{\alpha}_{t-r+1})_i = 1$, the i^{th} component of the smoothed hidden state $(\tilde{\boldsymbol{h}}_{t+1})_i$ will have no dependency on all the lagged i^{th} components of hidden states $\{(\hat{\boldsymbol{h}}_{t-s})_i\}_{s \geq r}$. The model simply forgets the i^{th} component of the hidden states at or beyond the r^{th} lag.

4.1. Neural Network Exponential Smoothing

While the class of α_t-RNN models under consideration is free to define how α is updated (including changing the frequency of the update) based on the hidden state and input, a convenient choice is use a recurrent layer. Remaining in the more general setup with a hidden state vector $\hat{\boldsymbol{h}}_t \in \mathbb{R}^H$, let us model the smoothing parameter $\hat{\alpha}_t \in [0,1]^H$ to give a filtered time series

$$\tilde{\boldsymbol{h}}_t = \hat{\boldsymbol{\alpha}}_t \circ \hat{\boldsymbol{h}}_t + (1 - \hat{\boldsymbol{\alpha}}_t) \circ \tilde{\boldsymbol{h}}_{t-1}. \tag{12}$$

This smoothing is a vectorized form of the above classical setting, only here we note that when $(\boldsymbol{\alpha}_t)_i = 1$, the i^{th} component of the hidden variable is unmodified and the past filtered hidden variable is forgotten. On the other hand, when $(\boldsymbol{\alpha}_t)_i = 0$, the i^{th} component of the hidden variable is obsolete, instead setting the current filtered hidden variable to its past value. The smoothing in **Eq. 12** can be viewed then as updating long-term memory, maintaining a smoothed hidden state variable as the memory through a convex combination of the current hidden variable and the previous smoothed hidden variable.

The hidden variable is given by the semi-affine transformation:

$$\hat{\boldsymbol{h}}_t = \sigma\left(U_h \tilde{\boldsymbol{h}}_{t-1} + W_h \boldsymbol{x}_t + \boldsymbol{b}_h\right), \tag{13}$$

which in turn depends on the previous smoothed hidden variable. Substituting **Eq. 13** into **Eq. 12** gives a function of $\tilde{\boldsymbol{h}}_{t-1}$ and $\boldsymbol{x}_t$:

$$\tilde{\boldsymbol{h}}_t = g\left(\tilde{\boldsymbol{h}}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\alpha}\right) \tag{14}$$

$$:= \hat{\boldsymbol{\alpha}}_t \circ \sigma\left(U_h \tilde{\boldsymbol{h}}_{t-1} + W_h \boldsymbol{x}_t + \boldsymbol{b}_h\right) + (1 - \hat{\boldsymbol{\alpha}}_t) \circ \tilde{\boldsymbol{h}}_{t-1}. \tag{15}$$

We see that when $(\boldsymbol{\alpha}_t)_i = 0$, the i^{th} component of the smoothed hidden variable $(\tilde{\boldsymbol{h}}_t)_i$ is not updated by the input $\boldsymbol{x}_t$. Conversely, when $(\boldsymbol{\alpha}_t)_i = 1$, we observe that the i^{th} hidden variable locally behaves like a non-linear autoregressive series. Thus the smoothing parameter can be viewed as the sensitivity of the smoothed hidden state to the input $\boldsymbol{x}_t$.

The challenge becomes how to determine dynamically how much error correction is needed. As in GRUs and LSTMs, we can address this problem by learning $\hat{\alpha} = F_{(W_\alpha, U_\alpha, b_\alpha)}(\underline{\boldsymbol{x}}_t)$ from the input variables with the recurrent layer parameterized by weights and biases $(W_\alpha, U_\alpha, \boldsymbol{b}_\alpha)$. The one-step ahead forecast of the smoothed hidden state, $\tilde{\boldsymbol{h}}_t$, is the filtered output of another plain RNN with weights and biases $(W_h, U_h, \boldsymbol{b}_h)$.

5. RESULTS

This section describes numerical experiments using financial time series data to evaluate the various RNN models. All models are implemented in v1.15.0 of TensorFlow (Abadi et al., 2016). Times series cross-validation is performed using separate training, validation and test sets. To preserve the time structure of the data and avoid look ahead bias, each set represents a contiguous sampling period with the test set containing the most recent observations. To prepare the training, validation and testing sets for m-step ahead prediction, we set the target variables (responses) to the $t + m$ observation, y_{t+m}, and use the lags from $t - p + 1, \ldots t$ for

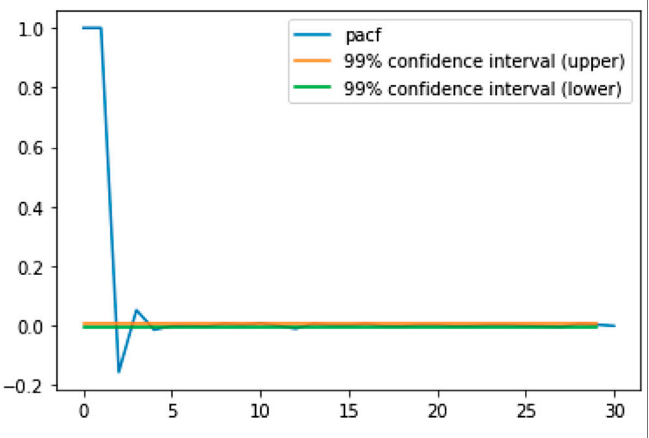


FIGURE 2 | The partial autocorrelogram (PACF) for 1 min snapshots of Bitcoin mid-prices (USD) over the period January 1, 2018 to November 10, 2018.

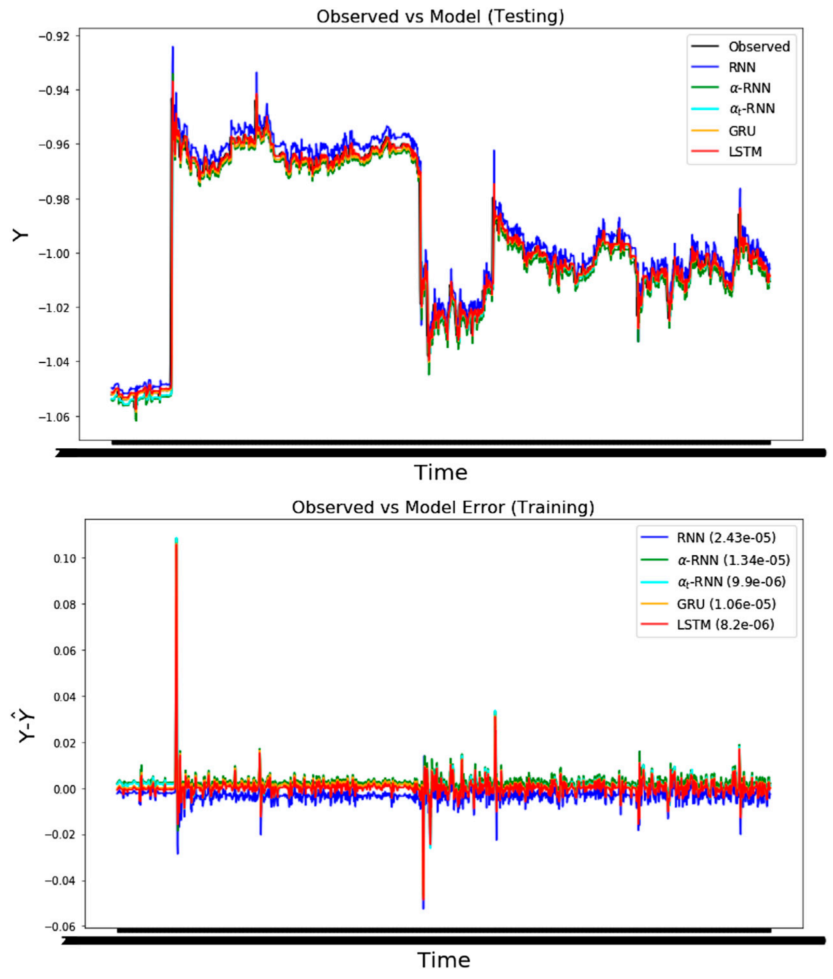


FIGURE 3 | The four-step ahead forecasts of temperature using the minute snapshot Bitcoin prices (USD) with MSEs shown in parentheses. (**top**) The forecasts for each architecture and the observed out-of-sample time series. (**bottom**) The errors for each architecture over the same test period. Note that the prices have been standardized.

each input sequence. This is repeated by incrementing t until the end of each set. In our experiments, each element in the input sequence is either a scalar or vector and the target variables are scalar.

We use the SimpleRNN Keras method with the default settings to implement a fully connected RNN. Tanh activation functions are used for the hidden layer with the number of units found by time series cross-validation with five folds to be $H \in \{5, 10, 20\}$ and L_1 regularization, $\lambda_1 \in \{0, 10^{-3}, 10^{-2}\}$. The Glorot and Bengio uniform method (Glorot and Bengio, 2010) is used to initialize the non-recurrent weight matrices and an orthogonal method is used to initialize the recurrence weights as a random orthogonal matrix. Keras's GRU method is implemented using version 1,406.1078v, which applies the reset gate to the hidden state before matrix multiplication. See Appendix 1.1 for a definition of the reset gate. Similarly, the LSTM method in Keras is used. Tanh activation functions are used for the recurrence layer and

TABLE 1 | *The **four-step** ahead Bitcoin forecasts are compared for various architectures using time series cross-validation. The half-life of the α-RNN is found to be 1.077 min ($\tilde{\alpha}$ = 0.4744).*

Architecture	Parameters	λ_1	H	MSE (test)	MSE (val)	MSE (train)
RNN	461	0	20	2.432×10^{-5}	1.921×10^{-5}	8.453×10^{-6}
α-RNN	132	0	10	1.342×10^{-5}	9.610×10^{-6}	7.664×10^{-6}
α_t-RNN	86	0	5	9.875×10^{-6}	8.614×10^{-6}	7.734×10^{-6}
GRU	371	0	10	1.055×10^{-5}	7.293×10^{-6}	6.293×10^{-6}
LSTM	491	0	10	8.164×10^{-6}	5.711×10^{-6}	4.922×10^{-6}

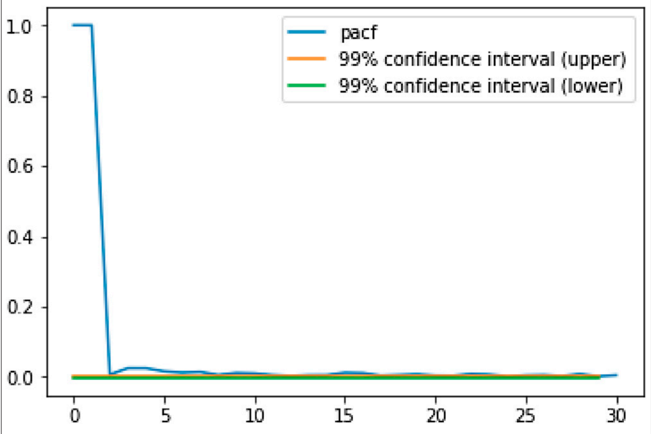


FIGURE 4 | The PACF of the tick-by-tick VWAP of ESU6 over the month of August 2016.

sigmoid activation functions are used for all other gates. The AlphaRNN and AlphatRNN classes are implemented by the authors for use in Keras. Statefulness is always disabled.

Each architecture is trained for up to 2000 epochs with an Adam optimization algorithm with default parameter values and using a mini-batch size of 1,000 drawn from the training set. Early stopping is implemented using a Keras call back with a patience of 50 to 100 and a minimum loss delta between 10^{-8} and 10^{-6}. So, for example, if the patience is set to 50 and the minimum loss delta is 10^{-8}, then fifty consecutive loss evaluations on mini-batch updates must each lie within 10^{-8} of each other before the training terminates. In practice, the actual number of epoches required varies between trainings due to the randomization of the weights and biases, and across different architectures and is typically between 200 and 1,500. The 2000 epoch limit is chosen as it provides an upper limit which is rarely encountered. No random permutations are used in the mini-batching sampling in order to preserve the ordering of the time series data. To evaluate the forecasting accuracy, we set the forecast horizon to up to ten steps ahead instead of the usual step ahead forecasts often presented in the machine learning literature—longer forecasting horizons are often more relevant due to operational constraints in industry applications and are more challenging when the data is non-stationary since the fixed partial auto-correlation of the process $\hat{y}_{t+m} + u_t$ will not adequately capture the observed

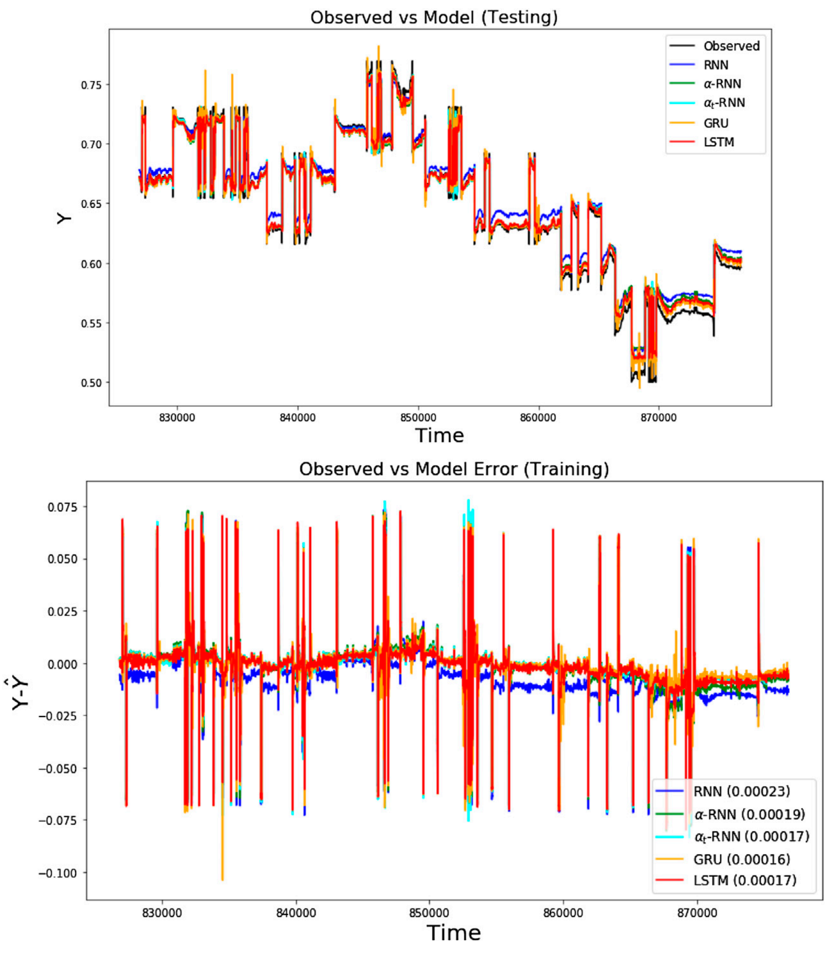


FIGURE 5 | The ten-step ahead forecasts of VWAPs are compared for various architectures using the tick-by-tick dataset. (**top**) The forecasts for each architecture and the observed out-of-sample time series. (**bottom**) The errors for each architecture over the same test period.

TABLE 2 | *The* ***ten-step*** *ahead forecasting models for VWAPs are compared for various architectures using time series cross-validation. The half-life of the α-RNN is found to be 2.398 periods ($\hat{\alpha}$ = 0.251).*

Architecture	Parameters	λ_1	H	MSE (test)	MSE (val)	MSE (train)
RNN	41	0	5	2.310×10^{-4}	1.843×10^{-4}	5.843×10^{-5}
α-RNN	132	0	10	1.926×10^{-4}	1.288×10^{-4}	3.456×10^{-5}
α_t-RNN	86	0	5	1.682×10^{-4}	1.311×10^{-4}	4.824×10^{-5}
GRU	1,341	0	20	1.568×10^{-4}	1.036×10^{-4}	2.488×10^{-5}
LSTM	491	0	10	1.685×10^{-4}	1.390×10^{-4}	3.154×10^{-5}

changing partial auto-correlation structure of the data. In the experiments below, we use $m = 4$ and $m = 10$ steps ahead. The reason we use less than $m = 10$ in the first experiment is because we find that there is little memory in the data beyond four lags and hence it is of little value to predict beyond four time steps.

5.1. Bitcoin Forecasting

One minute snapshots of USD denominated Bitcoin mid-prices are captured from Coinbase over the period from January 1 to November 10, 2018. We demonstrate how the different networks forecast Bitcoin prices using lagged observations of prices. The predictor in the training *and* the test set is normalized using the moments of the training data only so as to avoid look-ahead bias or introduce a bias in the test data. We accept the Null hypothesis of the augmented Dickey-Fuller test as we can not reject it at even the 90% confidence level. The data is therefore stationary (contains at least one unit root). The largest test statistic is −2.094 and the p-value is 0.237 (the critical values are 1%: -3.431, 5%: -2.862, and 10%: -2.567). While the partial autocovariance structure is expected to be time dependent, we observe a short memory of only four lags by estimating the PACF over the entire history (see **Figure 2**).

We choose a sequence length of $p = 4$ based on the PACF and perform a four-step ahead forecast. We comment in passing that there is little, if any, merit in forecasting beyond this time horizon given the largest significant lag indicated by the PACF. **Figure 3** compares the performance of the various forecasting networks and shows that stationary models such as the plain RNN and the α-RNN least capture the price dynamics—this is expected because the partial autocorrelation is non-stationary.

Viewing the results of time series cross validation, using the first 30,000 observations, in **Table 1**, we observe minor differences in the out-of-sample performance of the LSTM, GRU vs. the α_t-RNN, suggesting that the reset gate and extra cellular memory in the LSTM provides negligible benefit for this dataset. In this case, we observe very marginal additional benefit in the LSTM, yet the complexity of the latter is approximately 50x that of the α_t-RNN. Furthermore we observe evidence of strong over-fitting in the GRU and LSTM vs. the α_t-RNN. The ratio of training to test errors are respectively 0.596 and 0.603 vs. 0.783. The ratio of training to validation errors are 0.863 and 0.862 vs. 0.898.

5.2. High Frequency Trading Data

Our dataset consists of $N = 1,033,468$ observations of tick-by-tick Volume Weighted Average Prices (VWAPs) of CME listed ESU6 level II data over the month of August 2016 (Dixon, 2018; Dixon et al., 2019).

We reject the Null hypothesis of the augmented Dickey-Fuller test at the 99% confidence level in favor of the alternative hypothesis that the data is stationary (contains no unit roots. See for example (Tsay, 2010) for a definition of unit roots and details of the Dickey-Fuller test). The test statistic is −5.243 and the p-value is 7.16×10^{-6} (the critical values are 1%: −3.431, 5%: −2.862, and 10%: −2.567).

The PACF in **Figure 4** is observed to exhibit a cut-off at approximately 23 lags. We therefore choose a sequence length of $p = 23$ and perform a ten-step ahead forecast. Note that the time-stamps of the tick data are not uniform and hence a step refers to a tick.

Figure 5 compares the performance of the various networks and shows that plain RNN performs poorly, whereas and the α_t-RNN better captures the VWAP dynamics. From **Table 2**, we further observe relatively minor differences in the performance of the GRU vs. the α_t-RNN, again suggesting that the reset gate and extra cellular memory in the LSTM provides no benefit. In this case, we find that the GRU has 10x the number of parameters as the α_t-RNN with very marginal benefit. Furthermore we observe evidence of strong over-fitting in the GRU and LSTM vs. the α_t-RNN, although overall we observe stronger over-fitting on this dataset than the bitcoin dataset. The ratio of training to test errors are respectively 0.159 and 0.187 vs. 0.278. The ratio of training to validation errors are 0.240 and 0.226 vs. 0.368.

6. CONCLUSION

Financial time series modeling has entered an era of unprecedented growth in the size and complexity of data which require new modeling methodologies. This paper demonstrates a general class of exponential smoothed recurrent neural networks (RNNs) which are well suited to modeling non-stationary dynamical systems arising in industrial applications such as algorithmic and high frequency trading. Application of exponentially smoothed RNNs to minute level Bitcoin prices and CME futures tick data demonstrates the efficacy of exponential smoothing for multi-step time series forecasting. These examples show that exponentially smoothed RNNs are well suited to forecasting, exhibiting few layers and needing fewer parameters, than more complex architectures such as GRUs and LSTMs, yet retaining the most important aspects needed for forecasting non-stationary series. These methods scale to large numbers of covariates and complex data. The experimental design and architectural parameters, such as the predictive horizon and model parameters, can be determined by simple statistical tests and diagnostics, without the need for extensive hyper-parameter optimization. Moreover, unlike traditional time series methods such as ARIMA models, these methods are shown to be unconditionally stable without the need to pre-process the data.

DATA AVAILABILITY STATEMENT

The datasets and Python codes for this study can be found at https://github.com/mfrdixon/alpha-RNN.

AUTHOR CONTRIBUTIONS

MD contributed the methodology and results, and JL contributed to the results section.

FUNDING

The authors declare that this study received funding from Intel Corporation. The funder was not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

REFERENCES

Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., et al. (2016). "TensorFlow: a system for large-scale machine learning," in Proceedings of the 12th USENIX conference on operating systems design and implementation, Savannah, GA, November 2–4, 2016 (Berkeley, CA: OSDI'16) 265–283.

Akpinar N. J., Kratzwald B., Feuerriegel S. (2019). Sample complexity bounds for recurrent neural networks with application to combinatorial graph problems. Preprint repository name [Preprint]. Available at: https://arxiv.org/abs/1901.10289.

Bao W., Yue J., Rao Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS One* 12, e0180944–e0180924. doi:10.1371/journal.pone.0180944

Bayer J. (2015). Learning sequence representations. MS dissertation. Munich, Germany: Technische Universität München.

Borovkova S., Tsiamas I. (2019). An ensemble of LSTM neural networks for high-frequency stock market classification. *J. Forecast.* 38, 600–619. doi:10.1002/for.2585

Borovykh A., Bohte S., Oosterlee C. W. (2017). Conditional time series forecasting with convolutional neural networks. Preprint repository name [Preprint]. Available at: https://arxiv.org/abs/1703.04691.

Box G., Jenkins G. M. (1976). *Time series analysis: forecasting and control.* Hoboken, NJ: Holden Day, 575

Chen S., Ge L. (2019). Exploring the attention mechanism in LSTM-based Hong Kong stock price movement prediction. *Quant. Finance* 19, 1507–1515. doi:10.1080/14697688.2019.1622287

Chung J., Gülçehre Ç., Cho K., Bengio Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. Preprint repository name [Preprint]. Available at: https://arxiv.org/abs/1412.3555.

Dixon M. (2018). Sequence classification of the limit order book using recurrent neural networks. *J. Comput. Sci.* 24, 277. doi:10.1016/j.jocs.2017.08.018

Dixon M. F., Polson N. G., Sokolov V. O. (2019). Deep learning for spatio-temporal modeling: dynamic traffic flows and high frequency trading. *Appl. Stoch. Model. Bus Ind* 35, 788–807. doi:10.1002/asmb.2399

Dixon M. (2021). Industrial Forecasting with Exponentially Smoothed Recurrent Neural Networks, forthcoming in Technometrics.

Dixon M., London J. (2021b). Alpha-RNN source code and data repository. Available at: https://github.com/mfrdixon/alpha-RNN.

Glorot X., Bengio Y. (2010). "Understanding the difficulty of training deep feedforward neural networks," in Proceedings of the international conference on artificial intelligence and statistics (AISTATS'10), Sardinia, Italy, Society for Artificial Intelligence and Statistics, 249–256

Graves A (2013). Generating sequences with recurrent neural networks. Preprint repository name [Preprint]. Available at: https://arxiv.org/abs/1308.0850.

Hamilton J. (1994). *Time series analysis.* Princeton, NJ: Princeton University Press, 592

Hochreiter S., Schmidhuber J. (1997). Long short-term memory. *Neural. Comput.* 9, 1735–1780. doi:10.1162/neco.1997.9.8.1735

Kirchgässner G., Wolters J. (2007). *Introduction to modern time series analysis.* Berlin, Heidelberg: Springer-Verlag, 277

Li D., Zhu K. (2020). Inference for asymmetric exponentially weighted moving average models. *J. Time Ser. Anal.* 41, 154–162. doi:10.1111/jtsa.12464

Mäkinen Y., Kanniainen J., Gabbouj M., Iosifidis A. (2019). Forecasting jump arrivals in stock prices: new attention-based network architecture using limit order book data. *Quant. Finance* 19, 2033–2050. doi:10.1080/14697688.2019.1634277

Pascanu R., Mikolov T., Bengio Y. (2012). "On the difficulty of training recurrent neural networks," in ICML'13: proceedings of the 30th international conference on machine learning, 1310–1318. Available at: https://dl.acm.org/doi/10.5555/3042817.3043083.

Sirignano J., Cont R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quant. Finance* 19, 1449–1459. doi:10.1080/14697688.2019.1622295

Tsay R. S. (2010). *Analysis of financial time series.* 3rd Edn. Hoboken, NJ: Wiley

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.

APPENDIX

1. GRUS AND LSTMS

1.1. GRUs

A GRU is given by:

$$\begin{aligned} \text{smoothing} : \tilde{h}_t &= \hat{\alpha}_t \circ \hat{h}_t + (1 - \hat{\alpha}_t) \circ \tilde{h}_{t-1} \\ \text{smoother update} : \hat{\alpha}_t &= \sigma^{(1)}\left(U_\alpha \tilde{h}_{t-1} + W_\alpha x_t + b_\alpha\right) \\ \text{hidden state update} : \hat{h}_t &= \sigma\left(U_h \hat{r}_t \circ \tilde{h}_{t-1} + W_h x_t + b_h\right) \\ \text{reset update} : \hat{r}_t &= \sigma^{(1)}\left(U_r \tilde{h}_{t-1} + W_r x_t + b_r\right). \end{aligned}$$

When viewed as an extension of our α_t RNN model, we see that it has an additional reset, or switch, $\hat{r}_t$, which forgets the dependence of $\hat{h}_t$ on the smoothed hidden state. Effectively, it turns the update for $\hat{h}_t$ from a plain RNN to a FFN and entirely neglect the recurrence. The recurrence in the update of $\hat{h}_t$ is thus dynamic. It may appear that the combination of a reset and adaptive smoothing is redundant. But remember that $\hat{\alpha}_t$ effects the level of error correction in the update of the smoothed hidden state, $\tilde{h}_t$, whereas $\hat{r}_t$ adjusts the level of recurrence in the unsmoothed hidden state $\hat{h}_t$. Put differently, $\hat{\alpha}_t$ by itself can not disable the memory in the smoothed hidden state (internal memory), whereas $\hat{r}_t$ in combination with $\hat{\alpha}_t$ can. More precisely, when $\alpha_t = 1$ and $\hat{r}_t = 0$, $\tilde{h}_t = \hat{h}_t = \sigma(W_h x_t + b_h)$ which is reset to the latest input, x_t, and the GRU is just a FFN. Also, when $\alpha_t = 1$ and $\hat{r}_t > 0$, a GRU acts like a plain RNN. Thus a GRU can be seen as a more general architecture which is capable of being a FFN or a plain RNN under certain parameter values.

These additional layers (or cells) enable a GRU to learn extremely complex long-term temporal dynamics that a vanilla RNN is not capable of. Lastly, we comment in passing that in the GRU, as in a RNN, there is a final feedforward layer to transform the (smoothed) hidden state to a response:

$$\hat{y}_t = W_Y \tilde{h}_t + b_Y. \tag{A1}$$

1.2. LSTMs

LSTMs are similar to GRUs but have a separate (cell) memory, c_t, in addition to a hidden state h_t. LSTMs also do not require that the memory updates are a convex combination. Hence they are more general than exponential smoothing. The mathematical description of LSTMs is rarely given in an intuitive form, but the model can be found in, for example, Hochreiter and Schmidhuber (1997).

The cell memory is updated by the following expression involving a forget gate, $\hat{\alpha}_t$, an input gate $\hat{z}_t$ and a cell gate $\hat{c}_t$

$$c_t = \hat{\alpha}_t \circ c_{t-1} + \hat{z}_t \circ \hat{c}_t. \tag{A2}$$

In the terminology of LSTMs, the triple $(\hat{\alpha}_t, \hat{r}_t, \hat{z}_t)$ are respectively referred to as the forget gate, output gate, and input gate. Our change of terminology is deliberate and designed to provided more intuition and continuity with RNNs and the statistics literature. We note that in the special case when $\hat{z}_t = 1 - \hat{\alpha}_t$ we obtain a similar exponential smoothing expression to that used in our α_t-RNN. Beyond that, the role of the input gate appears superfluous and difficult to reason with using time series analysis.

When the forget gate, $\hat{\alpha}_t = 0$, then the cell memory depends solely on the cell memory gate update $\hat{c}_t$. By the term $\hat{\alpha}_t \circ c_{t-1}$, the cell memory has long-term memory which is only forgotten beyond lag s if $\hat{\alpha}_{t-s} = 0$. Thus the cell memory has an adaptive autoregressive structure.

The extra "memory", treated as a hidden state and separate from the cell memory, is nothing more than a Hadamard product:

$$h_t = \hat{r}_t \circ tanh(c_t), \tag{A3}$$

which is reset if $\hat{r}_t = 0$. If $\hat{r}_t = 1$, then the cell memory directly determines the hidden state.

Thus the reset gate can entirely override the effect of the cell memory's autoregressive structure, without erasing it. In contrast, the α_t-RNN and the GRU has one memory, which serves as the hidden state, and it is directly affected by the reset gate.

The reset, forget, input and cell memory gates are updated by plain RNNs all depending on the hidden state h_t.

$$\begin{aligned} \text{Reset gate} : \hat{r}_t &= \sigma(U_r h_{t-1} + W_r x_t + b_r) \\ \text{Forget gate} : \hat{\alpha}_t &= \sigma(U_\alpha h_{t-1} + W_\alpha x_t + b_\alpha) \\ \text{Input gate} : \hat{z}_t &= \sigma(U_z h_{t-1} + W_z x_t + b_z) \\ \text{Cell memory gate} : \hat{c}_t &= \tanh(U_c h_{t-1} + W_c x_t + b_c). \end{aligned}$$

The LSTM separates out the long memory, stored in the cellular memory, but uses a copy of it, which may additionally be reset. Strictly speaking, the cellular memory has long-short autoregressive memory structure, so it would be misleading in the context of time series analysis to strictly discern the two memories as long and short (as the nomenclature suggests). The latter can be thought of as a truncated version of the former.

METHODS
published: 22 March 2021
doi: 10.3389/fams.2021.598833

An Explainable Bayesian Decision Tree Algorithm

Giuseppe Nuti[1], *Lluís Antoni Jiménez Rugama*[1]* *and Andreea-Ingrid Cross*[2]

[1]UBS, New York, NY, United States, [2]UBS, London, United Kingdom

Bayesian Decision Trees provide a probabilistic framework that reduces the instability of Decision Trees while maintaining their explainability. While Markov Chain Monte Carlo methods are typically used to construct Bayesian Decision Trees, here we provide a deterministic Bayesian Decision Tree algorithm that eliminates the sampling and does not require a pruning step. This algorithm generates the greedy-modal tree (GMT) which is applicable to both regression and classification problems. We tested the algorithm on various benchmark classification data sets and obtained similar accuracies to other known techniques. Furthermore, we show that we can statistically analyze how was the GMT derived from the data and demonstrate this analysis with a financial example. Notably, the GMT allows for a technique that provides explainable simpler models which is often a prerequisite for applications in finance or the medical industry.

Keywords: explainable machine learning, Bayesian statistics, greedy algorithms, Bayesian decision trees, white box

OPEN ACCESS

Edited by:
Victor Wu,
Tilt Dev, United States

Reviewed by:
Xiangyu Chang,
Xi'an Jiaotong University, China
Chandan Gautam,
Indian Institute of Science, India
Marcela Svarc,
Universidad de San Andrés, Argentina

****Correspondence:***
Lluís Antoni Jiménez Rugama
lluis.jimenez-rugama@ubs.com

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *25 August 2020*
Accepted: *12 January 2021*
Published: *22 March 2021*

Citation:
Nuti G, Jiménez Rugama LIA and Cross A-I (2021) An Explainable Bayesian Decision Tree Algorithm. Front. Appl. Math. Stat. 7:598833. doi: 10.3389/fams.2021.598833

1 INTRODUCTION

The success of machine learning techniques applied to financial and medical problems can be encumbered by the inherent noise in the data. When the noise is not properly considered, there is a risk to overfit the data generating unnecessarily complex models that may lead to incorrect interpretations. Thus, there has been lot of efforts aimed at increasing model interpretability in machine learning applications [1–5].

Decision Trees (DT) are popular machine learning models applied to both classification and regression tasks with known training algorithms such as CART [6], C4.5 [7], and boosted trees [8]. With fewer nodes than other node-based models, DT are considered an explainable model. In addition, the tree structure can return the output with considerably fewer computations than other more complex models. However, as discussed by Linero in [9], greedely constructed trees are unstable. To improve the stability, new algorithms utilize tree ensembles such as bagging trees [10], Random Forests (RF) [11], and XGBoost (XG) [12]. But increasing the number of trees also increases the number of nodes and therefore the complexity of the model.

The Bayesian approach was introduced to solve the DT instability issue while producing a single tree model that accounts for the noise in the data. The first techniques, also known as Bayesian Decision Trees, were introduced in [13], BCART [14, 15], and BART [16]. The former article proposed a deterministic algorithm while the other three are based on Markov Chain Monte Carlo convergence. Some recent studies have improved upon these algorithms, for review see [9], and include a detailed interpretability analysis of the model, [17]. While most of the Bayesian work is based on Markov Chain convergence, here we take a deterministic approach that: 1) considers the noise in the data, 2) generates less complex models measured in terms of the number of nodes, and 3) provides a statistical framework to understand how the model is constructed.

The proposed algorithm departs from [13], introduces the *trivial partition* to avoid the pruning step, and generalizes the approach to employ any conjugate prior. Although this approach is

Bayesian, given the input data and model parameters the resulting tree is deterministic. Since it is deterministic, one can easily analyze the statistical reasons behind the choice of each node. We start with an overview of the Bayesian Decision Trees in **Section 2**. **Section 3** describes the building block of our algorithm, namely the partition probability space, and provides the algorithms to construct the greedy-modal tree (GMT). **Section 4** benchmarks the GMT vs. common techniques showing that the GMT works well for various publicly available data sets. Finally, a trading example is discussed in **Section 5** followed by some conclusive remarks in **Section 6**.

2 BAYESIAN DECISION TREES OVERVIEW

A Decision Tree is a directed acyclic graph. All its nodes have a parent node except the root node, the only one that has no parent. The level $\ell \in \mathbb{N}_0$ of a node is the number of ancestors of the node, starting from 0 at the root node. We classify the nodes as either *sprouts* or *leaves*. While sprouts point to two other child nodes in the case of binary trees, leaves are terminal nodes containing the model information. Each sprout contains a rule used to choose one of its children. To query the tree, we start at the root node and apply the rules to an input to select the child nodes until we reach a leaf.

We can use Decision Trees to partition $\mathbb{R}^d$ and assign an output to each subset of the partition. In this work, we restrict ourselves to finite partitions of $\mathbb{R}^d$. Each leaf of the tree will correspond to one of the subsets of the partition, one-to-one and onto. Our approach of Decision Trees departs from the Bayesian Decision Tree framework which provides a marginal likelihood to a Decision Tree based on some input data. Let's define the input data as $\mathcal{D} = [(x_i, y_i)]_{i=1}^n$ with n independent observations. A point $x = (x^1, \ldots, x^d)$ in $\mathbb{R}^d$ contains the features of each observation whose outcome y is randomly sampled from a random field Y_x. The Bayesian Decision Tree assumes the distribution of Y_x is constant at each leaf. Given x, the tree will return the posterior distribution of the parameters θ generating Y within the leaf x belongs to. In practice, the distribution of Y will determine the type of problem we are solving: a discrete random variable translates into a classification problem whereas a continuous random variable translates into a regression problem.

The probability of such a Bayesian Decision Tree, namely $\mathcal{T}$, can be computed with the usual Bayes approach,

$$p(\mathcal{T}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{T})}{p(\mathcal{D})} p(\mathcal{T}), \tag{1}$$

where $p(\mathcal{T})$ is the prior distribution over the tree space. To compute the marginal likelihood $p(\mathcal{D}|\mathcal{T})$, we consider the partition $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ induced by $\mathcal{T}$ and take the product of the marginal likelihoods at each leaf,

$$p(\mathcal{D}|\mathcal{T}) = \prod_{j=1}^{k} L(\mathcal{D}_j) = \prod_{j=1}^{k} \int_{\Theta} p(\mathcal{D}_j|\theta) p(\theta) d\theta \tag{2}$$

The probability $p(\theta)$ from **Eq. 2** is the prior distribution of the parameters θ. In this article, we will assume for simplicity that

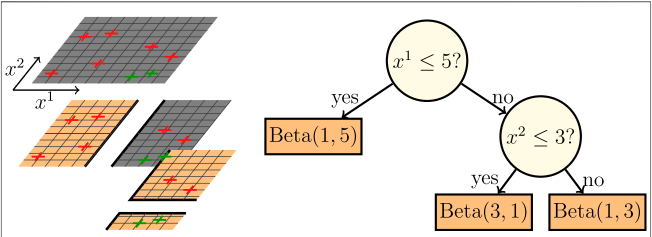


FIGURE 1 | Example of a Bayesian Decision Tree for a 2-categories example in $\mathbb{R}^2$. On the left: the data set is displayed three times. The first layer corresponds to the data set before any split. The second layer displays the two sets resulting from splitting along dimension 1. The third layer is an additional split of the right subset along dimension 2. The final leaves are displayed in orange. On the right: equivalent tree with posterior distributions for probability of being red assuming a Beta (1, 1) prior distribution. The marginal likelihood of the tree is $p(\mathcal{D}|\mathcal{T}) = B(1,5)B(3,1)B(1,3)/B(1,1)^3$, where B is the Beta function.

$p(\theta)$ is independent of $\mathcal{T}$ although this is not a requirement. The purpose of $p(\theta)$ is therefore two-fold:

- To obtain the tree probability from **Eq. 1**,
- To compute the posterior distribution of the parameters generating Y at each leaf.

Figure 1 shows a Bayesian Decision Tree that partitions $\mathbb{R}^2$ into $[(-\infty, 5) \times (-\infty, \infty), (5, \infty) \times (-\infty, 2.5), (5, \infty) \times (2.5, \infty)]$ and the corresponding posterior distributions Beta $(1, 5)$, Beta $(3, 1)$, and Beta $(1, 3)$. More information about conjugate priors and marginal likelihoods can be found in [18].

In an attempt to build explainable Bayesian Decision Trees, we define a greedy construction that does not apply Markov Chain Monte Carlo. This construction balances the greedy approach from [6] with the Bayesian approach discussed in [9, 14–17]. For this, we compute the probability of each split at every node and choose the modal split. This results in a model that performs well with different data sets as shown in **Section 4**.

3 FROM THE PARTITION PROBABILITY SPACE TO BAYESIAN DECISION TREES

The building block of the GMT algorithm is the *partition probability space*. For this space, we only consider binary partitions of the form $S_{r,h} = \{\{x \in \mathbb{R}^d \textit{ such that } x^r \le h\}, \{x \in \mathbb{R}^d \textit{ such that } x^r > h\}\}$ where $r \in \{1, \ldots, d\}$, $h \in \mathbb{R} \backslash \{x_1^r, \ldots, x_n^r\}$. Any partition of this form will induce a partition $\{\mathcal{D}_1, \mathcal{D}_2\}$ of $\mathcal{D}$. Note that any of these two subsets are allowed to be the empty set. Finally, for each dimension we identify all partitions that result in the same non-empty $\mathcal{D}_1$ and $\mathcal{D}_2$. All partitions that leave $\mathcal{D}_1$ or $\mathcal{D}_2$ empty are also identified as the *trivial partition* S_0. After identification, we will have $1 + \sum_{r=1}^{d} (n_r - 1)$ different partitions S, n_r tbeing the number of different features along dimension r. Following the minimum margin classifier idea, the partition representative location h will be placed at the mid-point between contiguous different features in a dimension.

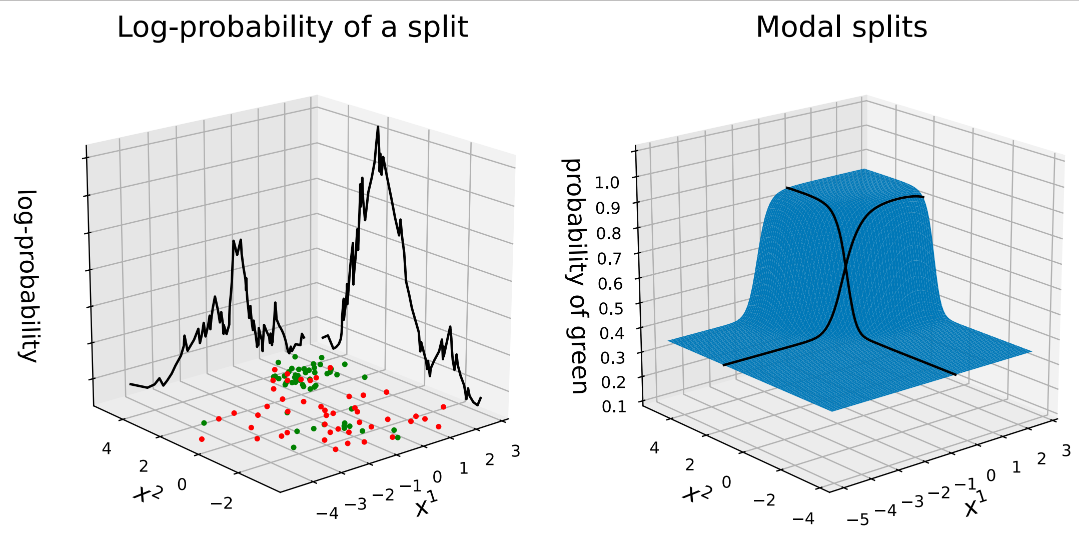


FIGURE 2 | Example of a data set whose outcomes are either green or red. The location of the points is sampled from a mixture of two Gaussian distributions with equal probability. One distribution draws outcomes from a Bernoulli distribution with probability 0.25, while the other from a Bernoulli with probability 0.75. On the left: log-probabilities of all possible non-trivial partitions given the data set. On the right: actual probability of a point being green and modal splits along each dimension.

The *partition probability space* is the finite probability space defined by partitions S and their probabilities $p(S|\mathcal{D})$. These probabilities can be computed using **Eqs 1, 2** when replacing $\mathcal{T}$ by S. In practice, we will work with $\ln[p(\mathcal{D}|S)p(S)]$ which are the log-probabilities from **Eq. 1** omitting the constant normalizing factor $p(\mathcal{D})$:

$$\ln[p(\mathcal{D}|S)p(S)] = \ln[p(S|\mathcal{D})] + \ln[p(\mathcal{D})] \tag{3}$$

We will also need the feature sorted indices of the input features for computation and visualization purposes, namely $i_1, \ldots, i_n$ such that $x^r_{i^r_j} \leq x^r_{i^r_{j+1}}$ for all $r = 1, \ldots, d$ and $j = 1, \ldots, n-1$. An example of the split probability space is shown in **Figure 2**. In this example, the inputs x live in $\mathbb{R}^2$ and the outcomes are drawn from a Bernoulli random variable. The points x are generated from two independent Gaussian distributions equally likely, i.e. we drew from each distribution with probability 0.5. The first distribution is a multivariate Gaussian with mean $(-1,-1)^t$ and covariance $2I$. Points sampled from this distribution have a probability of 0.25 of being green. The second distribution is another multivariate Gaussian with mean $(1,3)^t$ and covariance $0.5I$. In this case, the probability of being green is 0.75. Because the mean of these Gaussian distributions are further apart along the x^2 axis, the most probable partitions given the data are found along this dimension.

Each partition S can be encoded into a tree node $\mathcal{N}$: if the partition is $\int_0$, the node becomes a leaf and stores the posterior hyper-parameters of θ; for any other $S_{r,h}$, the node becomes a sprout and stores the values r and h. Among all partitions, the

Algorithm 1 Returns the modal node for the general problem given $\mathcal{D}$ and prior hyper-parameters α.

```
1:  procedure FIND_MODAL_NODE($\mathcal{D}, \alpha$)
2:      $\alpha^* \leftarrow$ posterior_hyper($\mathcal{D}, \alpha$)        ▷ Obtain the posterior hyper-parameters using $\mathcal{D}$.
3:      $q \leftarrow \ln(L(\mathcal{D})) + \ln(p(\mathcal{S}_0))$        ▷ Compute (3) for $\mathcal{S}_0$.
4:      $\mathcal{N} \leftarrow$ create_leaf($\alpha^*$)        ▷ Initialize with the leaf.
5:      for each dimension $r$ in $1, \ldots, d$ do
6:          for each observation $j$ in $1, \ldots, n-1$ do
7:              if $x^r_{i^r_j} \neq x^r_{i^r_{j+1}}$ then
8:                  $h \leftarrow (x^r_{i^r_j} + x^r_{i^r_{j+1}})/2$        ▷ Compute the location of the split.
9:                  $\mathcal{D}_1 \leftarrow \{\boldsymbol{x}_{i^r_a}, y_{i^r_a}\}^j_{a=1}$        ▷ Generate $\mathcal{D}_1$ given $\mathcal{D}$, $r$, and $h$.
10:                 $\mathcal{D}_2 \leftarrow \{\boldsymbol{x}_{i^r_a}, y_{i^r_a}\}^n_{a=j+1}$        ▷ Generate $\mathcal{D}_2$ given $\mathcal{D}$, $r$, and $h$.
11:                 $q_{\text{new}} \leftarrow \ln(L(\mathcal{D}_1)) + \ln(L(\mathcal{D}_2)) + \ln(p(\mathcal{S}_{r,h}))$        ▷ Compute (3) for $\mathcal{S}_{r,h}$.
12:                 if $q_{\text{new}} > q$ then        ▷ Store the new node with higher log-probability.
13:                     $q \leftarrow q_{\text{new}}$
14:                     $\mathcal{N} \leftarrow$ create_sprout($r, h$)
15:                 end if
16:             end if
17:         end for
18:     end for
19:     Return $\mathcal{N}$
20: end procedure
```

mode and its node are of particular interest. **Algorithm 1** returns the modal node in the general case. For the classification problem, we also provide **Algorithm 2** with $\mathcal{O}(dn)$ cost assuming $p(\theta)$ follows a Dirichlet conjugate prior. Both algorithms start by computing $\ln[p(\mathcal{D}|S_0)p(S_0)]$ and initializing $\mathcal{N}$ to be a leaf. Then, they loop through each dimension and the sorted features to verify whether there exists a new node with higher log-probability. Because the features are sorted, there is at most one observation that moves from $\mathcal{D}_2$ to $\mathcal{D}_1$ when j increases by one.

With the partition space and modal node defined, we can introduce the GMT construction. We start by finding the modal node $\mathcal{N}$ for our initial data set $\mathcal{D}$. This node is the root of the tree and will be returned by the train method in **Algorithm 3**. If $\mathcal{N}$ is a leaf, the GMT is completed. Otherwise, we split $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$ according to $\mathcal{N}$. We repeat the process for the new input data sets $\mathcal{D}_1$ and $\mathcal{D}_2$, and link $\mathcal{N}_1$ and $\mathcal{N}_2$ to their parent $\mathcal{N}$. The recursion is defined in the grow_tree method from **Algorithm 3**. Note that **Algorithms 1 and 2** are just two implementations of find_modal_node, but one can replace this method by any other that returns the desired node based on the partition space. In addition, one can easily compute $\ln[p(\mathcal{D}|\mathcal{T})]$ for the GMT by adding the leaves' $\ln[L(\mathcal{D}_j)]$ calculated in **Algorithm 1** line 2, or **Algorithm 2** line 2. In practice, we realized that the GMT marginal log-likelihood $\ln[p(\mathcal{D}|\mathcal{T})]$ tends to be the highest when exploring for different possible roots.

The average cost of **Algorithm 3** is $\mathcal{O}[c(n)\ln(n)]$ where $c(n)$ is the cost of find_modal_node. If we choose find_modal_node to be **Algorithm 2**, the average cost of **Algorithm 3** becomes $\mathcal{O}[dn\ln(n)]$. While **Algorithms 1 and 2** only look at one successor ahead, we could improve the greedy exploration by looking at several levels ahead as suggested in [13]. Looking at m levels ahead comes at the expense of increasing the order of $c(n)$, for instance $c(n) = (dn)^m$ in the case of **Algorithm 2**. **Section 4** shows that the GMT constructed by looking at only one level ahead performs well in practice.

4 BENCHMARK

4.1 Decision Trees, Random Forests, XGBoost, and GMT

In this Section we use **Algorithm 2 and 3** to construct the GMT. We assume that the outcomes, 0 or 1, are drawn from Bernoulli random variables. The prior distribution $p(\theta)$ is chosen to be the Beta $(10, 10)$ and each tree will return the expected probability of drawing the outcome 0. The prior probabilities for each partition will be $p(S_0) = 1 - 0.9^{1+\ell}$ and $p(S_{r,h}) = 0.9^{1+\ell}/dn_r$, where ℓ is the level and n_r the number of non-trivial partitions along r. Note that the denominator d in $p(S_{r,h})$ is implicitly assuming a uniform prior distribution over the dimension space. One could also project the probabilities on each dimension to visualize which features are most informative. As an alternative to the suggested $p(S)$, one can use the partition margin weighted approach from [13].

The accuracy is measured as a percentage of the correct predictions. Each prediction will simply be the highest probability outcome. If there is a tie, we choose the category 0

Algorithm 2 Returns the modal node given $\mathcal{D}$ and Dirichlet prior hyper-parameters α.

```
1:  procedure FIND_MODAL_NODE($\mathcal{D}, \alpha$)
2:      $\alpha^* \leftarrow \alpha$                                          ▷ Initialize the posterior hyper-parameters.
3:      for each observation $j$ in $1, \dots, n$ do                         ▷ Obtain the posterior hyper-parameters for $\mathcal{D}$.
4:          $\alpha^*[y_j] \leftarrow \alpha^*[y_j] + 1$
5:      end for
6:      $q \leftarrow \ln(B(\alpha^*)) - \ln(B(\alpha)) + \ln(p(\mathcal{S}_0))$              ▷ Compute (3) for $\mathcal{S}_0$.
7:      $\mathcal{N} \leftarrow$ create_leaf($\alpha^*$)                        ▷ Initialize with the leaf.
8:      $s_\alpha \leftarrow$ sum($\alpha$)                                   ▷ Obtain pseudo count.
9:      for each dimension $r$ in $1, \dots, d$ do
10:         $q_1 \leftarrow 0$                                              ▷ Initialize $\ln(L(\mathcal{D}_1))$
11:         $q_2 \leftarrow \ln(B(\alpha^*)) - \ln(B(\alpha))$                   ▷ Initialize $\ln(L(\mathcal{D}_2))$
12:         $c_1 \leftarrow \alpha$                                         ▷ Initialize the posterior hyper-parameters for $\mathcal{D}_1$.
13:         $c_2 \leftarrow \alpha^*$                                       ▷ Initialize the posterior hyper-parameters for $\mathcal{D}_2$.
14:         for each observation $j$ in $1, \dots, n-1$ do
15:             $q_1 \leftarrow q_1 + \ln(c_1[y_{i_j}]/(j + s_\alpha - 1))$         ▷ Update $q_1$ adding $j$ to $\mathcal{D}_1$
16:             $c_1[y_{i^r_j}] \leftarrow c_1[y_{i^r_j}] + 1$                        ▷ Update $c_1$ adding $j$ to $\mathcal{D}_1$
17:             $c_2[y_{i^r_j}] \leftarrow c_2[y_{i^r_j}] - 1$                        ▷ Update $c_2$ removing $j$ from $\mathcal{D}_2$
18:             $q_2 \leftarrow q_2 - \ln(c_2[y_{i^r_j}]/(n + s_\alpha - j))$       ▷ Update $q_2$ removing $j$ from $\mathcal{D}_2$
19:             if $x^r_{i^r_j} \neq x^r_{i^r_{j+1}}$ then
20:                 $h \leftarrow (x^r_{i^r_j} + x^r_{i^r_{j+1}})/2$                  ▷ Compute the location of the split.
21:                 $q_{\text{new}} \leftarrow q_1 + q_2 + \ln(p(\mathcal{S}_{r,h}))$          ▷ Compute (3) for $\mathcal{S}_{r,h}$.
22:                 if $q_{\text{new}} > q$ then                             ▷ Store the new node with higher log-probability.
23:                     $q \leftarrow q_{\text{new}}$
24:                     $\mathcal{N} \leftarrow$ create_sprout($r, h$)
25:                 end if
26:             end if
27:         end for
28:     end for
29:     Return $\mathcal{N}$
30: end procedure
```

Algorithm 3 Train the GMT given $\mathcal{D}$ and prior hyper-parameters α.

```
1:  procedure TRAIN($\mathcal{D}, \alpha$)
2:      $\mathcal{N} \leftarrow$ find_modal_node($\mathcal{D}, \alpha$)          ▷ Apply Algorithm 1 or 2.
3:      if $\mathcal{N}$ is a sprout then
4:          grow_tree($\mathcal{D}, \mathcal{N}, \alpha$)
5:      end if
6:      Return $\mathcal{N}$
7:  end procedure

8:  procedure GROW_TREE($\mathcal{D}, \mathcal{N}, \alpha$)
9:      $\mathcal{D}_1, \mathcal{D}_2 \leftarrow$ split($\mathcal{D}, \mathcal{N}$)          ▷ Split $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$ using $\mathcal{N}$.
10:     $\mathcal{N}_1 \leftarrow$ find_modal_node($\mathcal{D}_1, \alpha$)          ▷ Apply Algorithm 1 or 2.
11:     set_left_child($\mathcal{N}, \mathcal{N}_1$)          ▷ Set $\mathcal{N}_1$ as the left child of $\mathcal{N}$.
12:     if $\mathcal{N}_1$ is a sprout then          ▷ We apply recursion.
13:         fill_tree($\mathcal{D}_1, \mathcal{N}_1, \alpha$)
14:     end if
15:     $\mathcal{N}_2 \leftarrow$ find_modal_node($\mathcal{D}_2, \alpha$)          ▷ Apply Algorithm 1 or 2.
16:     set_right_child($\mathcal{N}, \mathcal{N}_2$)          ▷ Set $\mathcal{N}_2$ as the right child of $\mathcal{N}$.
17:     if $\mathcal{N}_2$ is a sprout then          ▷ We apply recursion.
18:         fill_tree($\mathcal{D}_2, \mathcal{N}_2, \alpha$)
19:     end if
20: end procedure
```

by default. We compare the GMT results to DT [6, 19], RF [11, 19], and XG [12]. For reproducibility purposes, we set all random seeds to 0. In the case of RF, we enable bootstrapping to improve its performance. We also fix the number of trees to five for RF and XG. We provide the GMT *Python* module with integration into scikit-learn in [20].

We test the GMT on a selection of data sets from the University of California, Irvine (UCI) database [21]. We compute the accuracy of the DT, RF, XG, and GMT with a shuffled 10-fold cross validation.We do not perform any parameter tuning and keep the same $p(\theta)$ and $p(S)$ for all examples. Accuracy is shown in **Table 1** while training time and node count in **Table 2**.

The results reveal some interesting properties of the GMT. Noticeably, the GMT seems to perform well in general. In all cases, the DT accuracy is lower than the RF accuracy. The only case in which RF considerably outperforms the GMT is with the EEG data set. One reason may be that some information is hidden at the lower levels, i.e. feature correlation information that is hard to extract by looking at only one level ahead. The accuracy difference between GMT and RF indicates that these two techniques may work well for different data sets. Interestingly, the XG and GMT yield similar accuracies. Finally, in most cases the GMT takes more time to train than the other three techniques which is caused by the feature sorting overhead computation. Notably, the node count in **Table 2** shows that we successfully managed to simplify the models while producing similar accuracy. Note that for four of the seven data sets, the average number of nodes is less than ten and produces slightly better accuracies than RF. Ten nodes implies less than five sprouts in average which can be easily analyzed by a human. This highlights the importance of the priors $p(S)$ and $p(\theta)$ to avoid a pruning step. The strength of these two priors will determine how much statistical evidence do we require from our data to produce a meaningful split. In the following **Section 5**, we take a deeper look and explain the reasons behind the GMT construction with a finance application.

4.2 Bayesian Decision Trees and GMT

In this section we analyze the GMT on the Wisconsin breast-cancer data set studied in [9, 14] which is available at the University of California, Irvine (UCI) database [21]. Although this data set contains 699 observations, we are going to use the 683 that are complete. Each observation contains nine features and the outcome classifies the tumor as benign or malignant. We test the GMT for $p(S_0) = 1 - q^{1+\ell}, q \in \{0.75, 0.8, 0.85, 0.90, 0.95, 0.97\}$ and a Dirichlet prior with parameters $(\alpha_1, \alpha_2) \in \{1, 2, 3, 4, 5, 10\}^2$. For each of the 216 parameter sets we perform a 10-fold cross validation and plot the average accuracy in **Figure 3**. The results display a lower average accuracy compared to the 98.4% for BCART [14] with nine or more leaves, and the 96.8% for BART [9]. When we run the methods and parameters from Section 4.1 we obtain 95.3% for DT, 95.8% for RF, and 95.5% for XG. We were unable to compare the BCART and BART performance with the data sets from Section 4.1 due to the lack of software.

5 TRADING EXAMPLE

We consider three stocks, A, B, and C, whose price follows a multidimensional Ornstein-Uhlenbeck process, [22]. Using the

TABLE 1 | Accuracy of DT, RF, XG, and GMT for several data sets. We apply a shuffled 10-fold cross validation to each test. Results are sorted by relative performance, starting from hightest accuracy difference between GMT and RF.

			Accuracy				
	d	*n*	DT [6, 19]	RF [11, 19]	XG [12]	GMT	GMT – RF
Credit	23	30,000	72.5%	78.6%	82.0%	82.0%	3.4%
Diabetic	19	1,151	60.4%	63.1%	65.2%	65.3%	2.2%
Heart	20	270	72.6%	78.5%	80.4%	80.4%	1.9%
Seismic	18	2,584	87.8%	91.9%	93.0%	93.2%	1.3%
Haberman	3	306	59.8%	69.6%	69.9%	69.6%	0.0%
Gamma	10	19,020	81.7%	85.9%	86.2%	84.9%	−1.0%
EEG	14	14,980	83.8%	88.1%	80.5%	79.8%	−8.3%

TABLE 2 | Training time in milliseconds and average node count per fold. The node count includes the number of leaves.

	Train time (ms)				Node count			
	DT [6, 19]	RF [11, 19]	XG	GMT	DT [6, 19]	RF [11, 19]	XG	GMT
Credit	569.5	334.7	133.6	1,044.2	8,505.2	3,898.9	579.6	43.4
Diabetic	8.7	10.1	15.2	26.8	399.2	182.3	329.6	7.0
Heart	1.4	4.8	20.1	13.2	83.4	44.7	179.6	9.2
Seismic	10.5	11.7	20.5	18.5	410.2	177.7	304.6	6.4
Haberman	1.0	4.1	15.6	1.4	179.8	66.3	194.0	3.2
Gamma	245.2	232.2	92.2	519.1	3,564.0	1,514.9	551.6	111.4
EEG	123.4	101.2	117.9	550.8	2,553.4	1,374.7	472.8	203.4

notation from [22], we can sample the prices by applying the Euler's discretization, $X_{t+\Delta t} = \mu + e^{-\theta\Delta t}(X_t - \mu) + G$. We assume that σ is a unitary matrix, therefore the random vector G follows a normal distribution with mean 0 and covariance $(\theta + \theta^T)^{-1}[I - e^{-(\theta+\theta^T)\Delta t}]$. For this example, we set the parameters to,

$$\mu = \begin{pmatrix} 100 \\ 110 \\ 105 \end{pmatrix}, \qquad \theta = \begin{pmatrix} 4 & -1 & 0 \\ 0.4 & 2 & 0 \\ 0 & 0 & 0.2 \end{pmatrix}, \qquad \Delta t = 0.1. \tag{4}$$

Our goal is to train the GMT to predict the best portfolio configuration. Given that we have three stocks, we consider the following eight buy/sell configurations: +A/−B/−C (buy one stock A, sell one stock B, sell one stock C), −A/−B/−C, −A/+B/−C, +A/+B/−C, −A/−B/+C,+A/−B/+C, -A/+B/+C, +A/+B/+C. At each time step, we take the three stock prices X_{t_i} as inputs. The outcome is defined as the configuration that corresponds to the next price move, i.e. $sign(X^1_{t_{i+1}} - X^1_{t_i})A/sign(X^2_{t_{i+1}} - X^2_{t_i})B/sign(X^3_{t_{i+1}} - X^3_{t_i})C$. For example, if the prices are $(100, 105, 110)$ at t_i and $(110, 100, 120)$ at t_{i+1}, the features are $(100, 105, 110)$, the outcome is +A − B + C, and the profit between t_i and t_{i+1} for this portfolio is $+(110 - 100) - (100 - 105) + (120 - 110)$. Each portfolio configuration is identified to an integer from 0 to 7. We sample 10,000 time steps, train on the first 8,000 observations and test on the next 2,000.

We treat this problem as an eight class classification problem. The GMT is trained with the $p(S)$ from **Section 4** and a Dirichlet conjugate prior $p(\theta)$ with hyper-parameters (1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8). To benchmark the results, we train a 10×10. nodes neural network using the MLPClassifier from [19]. During the test phase, our predictions will be the expected modal portfolio configuration: if the input $\boldsymbol{x}$ returns the leaf posterior hyper-parameters α^*, we predict the portfolio $\arg\max_k\{\alpha^*_k/\sum\alpha^*\}$. The test results are shown in **Figure 4**.

The GMT we obtained by training on the first 8,000 observations has only four leaves: if the price of stock A is below 99.96 and the price of stock B below 109.85, we choose + A + B − C; if the price of A is below 99.96 and the price of B above 109.85, we choose + A-B-C; if the price of A is above 99.96 and the price of B below 110.14, we choose −A + B − C; and if the price of A is above 99.96 and the price of B above 110.14, we choose −A − B + C. Although the mean reversion for stock C is not captured in this model, we successfully recovered simple rules to trade the mean reversion of A and B. Since the price of C is more volatile by **Eq. 4**, the current price of C is not enough to recover the mean reversion decision logic. Some filtering of the price of C would allow to capture its mean reversion. In the neural network case, the over-parametrization makes it difficult to recover this simple model.

The deterministic nature of **Algorithm 3** provides a practical framework to explain how was the GMT constructed. We look at each of the nodes to understand how were the modal nodes chosen. The resulting GMT model contains three sprouts—node 0, node 1, node 2—and four leaves—node 3, node 4, node 5, node 6. **Figure 5** shows the log-probability 3) of splitting our data-set at a particular price by stock for the three sprouts. At the root level, node 0, we consider the whole data set. In this case, one can increase the GMT likelihood the most by choosing $S_{0,99.96}$, i.e., splitting the data according to Stock A's price at 99.96. After this node becomes a sprout, the input data is split into two subsets of sizes 3,640 (inputs with Stock A's price below 99.96), and 4,360 (inputs with Stock A's price above 99.96).

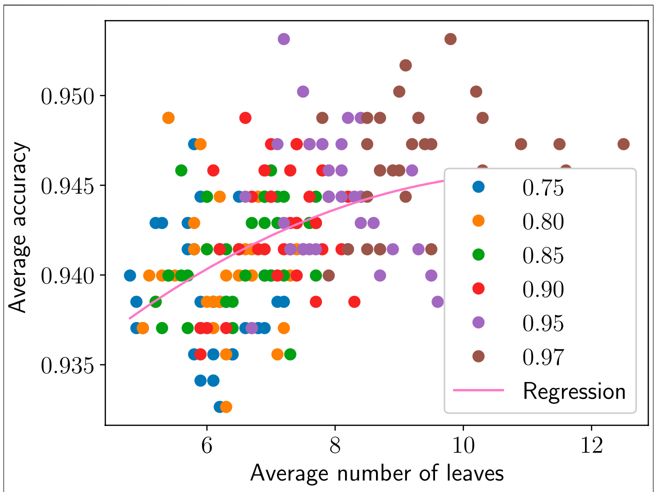


FIGURE 3 | Average accuracy vs. average number of leaves for each GMT parameter set applied to the Wisonson breast-cancer data. The color indicates which parameter $q \in \{0.75, 0.8, 0.85, 0.90, 0.95, 0.97\}$ was chosen. We include a quadratic regression of the results.

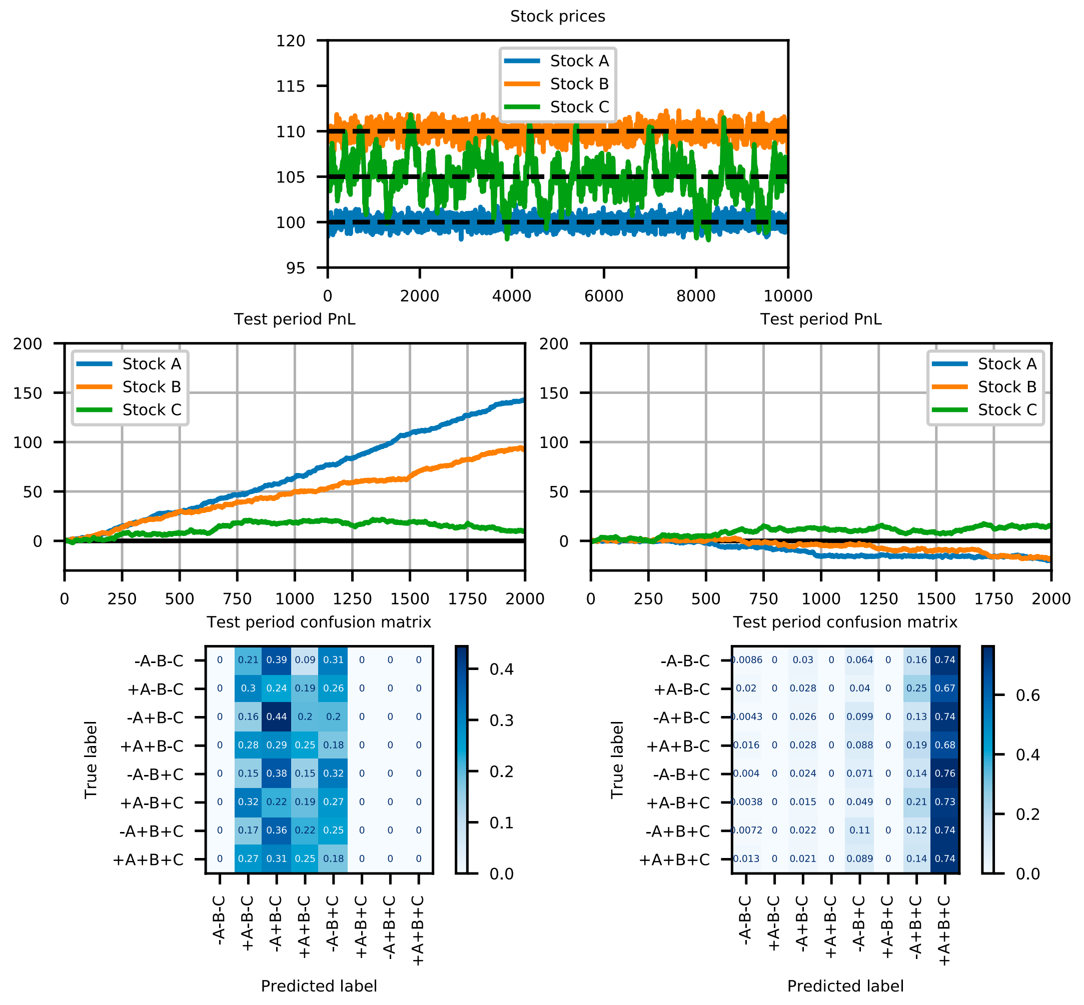


FIGURE 4 | From top to bottom, left to right: Simulated stock prices, test period PnL (Profit and Loss) for the GMT, test period PnL for the neural network, confusion matrix for the GMT, and confusion matrix for the neural network. The PnL is the cumulative profit achieved when the predicted portfolios are executed. The costs are omitted for simplicity.

These two subsets' partition log-probabilities are then shown in node one and node two plots respectively. By looking at the two figures, we conclude we can maximize the log-probability by splitting at Stock's B price 109.85 for node 1, and at Stock's B price 110.14 for node 2. The black horizontal line in each figure marks the log-probability of S_0, i.e.. the stopping condition. When any possible split log-probability is below this line, the node is chosen to be a leaf, as it happens in this example for nodes 3, 4, 5, and 6. Finally, note that by symmetry, the blue, orange, and green lines should look periodic because the extreme splits only separate one input point from the data set. In addition, the green line looks convex which indicates it is better not to split the data based on Stock C's price.

6 DISCUSSION AND FUTURE WORK

The proposed GMT is a deterministic Bayesian Decision Tree that reduces the training time by avoiding any Markov Chain Monte Carlo sampling or a pruning step. The GMT numerical example results show similar accuracies to other known techniques. This approach may be most useful where the ability to explain the model is a requirement. Hence, the advantages of the GMT are that it can be easily understood. Furthermore, the ability to specify $p(\theta)$ and $p(S)$ may be particularly suitable to noisy problems. However, it is not clear whether the hyper-parameters used in the examples are optimal for each data set. Future work will explore the sensitivity and parameter tuning for different prior distributions. It still remains to find a more efficient deterministic way to explore meaningful trees like Markov Chain Monte Carlo based Bayesian Decision Trees do.

As an extension, we would like to assess the performance of this algorithm on regression problems and experiment with larger partition spaces such as the SVM hyperplanes. Another computational advantage not explored is parallelization, which would allow for a more exhaustive exploration of the tree probability space from **Eq. 1**.

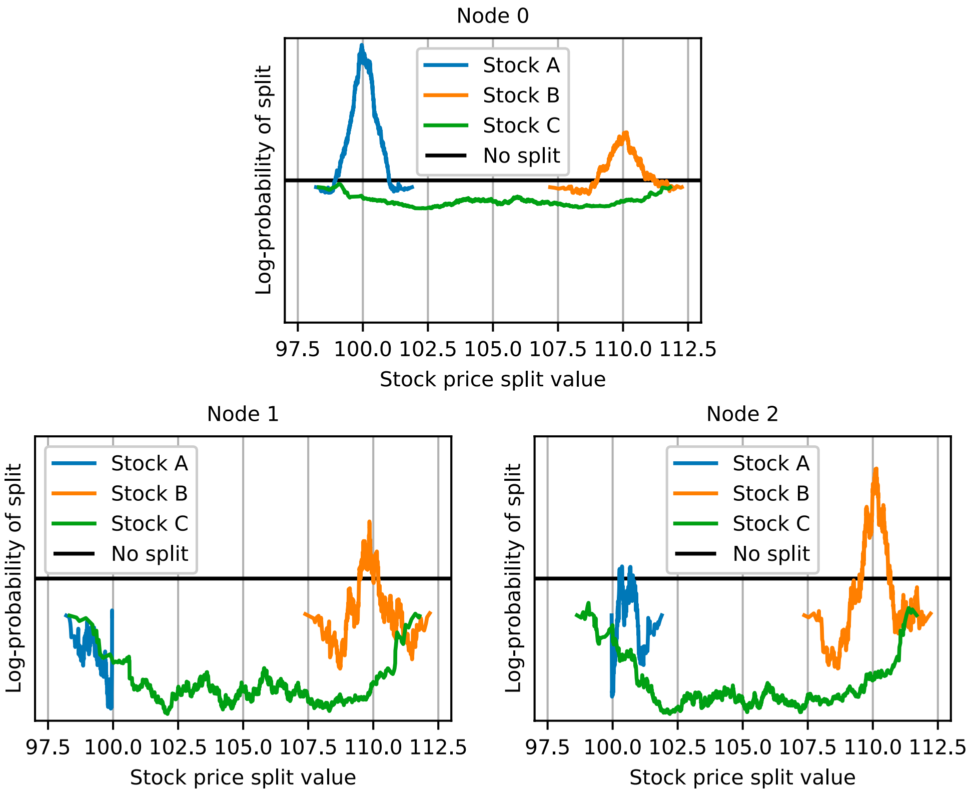


FIGURE 5 | Log-probability of $S_{r,h}$ at each sprout for the trading example. The black horizontal line indicates the S_0 log-probability.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found at https://archive.ics.uci.edu/ml/datasets.php. The code is available at https://github.com/UBS-IB/bayesian_tree/.

AUTHOR CONTRIBUTIONS

GN was the technical advisor leading the project. LlAJR was responsible for the technical details, first implementation, running the experiments, and manuscript preparation. AIC contributed in the manuscript preparation.

FUNDING

Authors were employed by UBS during the development of the project. UBS was not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

ACKNOWLEDGMENTS

The authors want to thank Kaspar Thommen for the *Python* module implementation with integration to scikit-learn and his suggestions to improve the quality of this project. This manuscript has been released as a pre-print at https://arxiv.org/pdf/1901.03214, [23].

REFERENCES

1. Lipton ZC. The mythos of model interpretability. *Queue* (2018). 16:31–57. doi:10.1145/3236386.3241340
2. Herman B. The promise and peril of human evaluation for model interpretability (2017). Preprint: arXiv: abs/1711.07414.
3. Doshi-Velez F, Kim B. Towards a rigorous science of interpretable machine learning (2017). Preprint: arXiv:1702.08608.
4. Lipton ZC. The doctor just won't accept that! (2017). Preprint: arXiv:1711.08037.
5. Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B. Definitions, methods, and applications in interpretable machine learning. *Proc Natl Acad Sci USA* (2019). 116:22071–80. doi:10.1073/pnas.1900654116
6. Breiman L, Friedman J, Stone C, Olshen R. Classification and regression trees. *The wadsworth and brooks-cole statistics-probability series*. Abingdon, UK: Taylor and Francis (1984).
7. Quinlan JR. *C4.5: programs for machine learning*. San Francisco, CA: Morgan Kaufmann Publishers Inc. (1993).
8. Friedman JH. Greedy function approximation: a gradient boosting machine. *Ann Stat* (2000). 29:1189–232. doi:10.1214/aos/1013203451

9. Linero AR. A review of tree-based Bayesian methods. *Csam* (2017). 24:543–59. doi:10.29220/csam.2017.24.6.543
10. Breiman L. Bagging predictors. *Mach Learn* (1996). 24:123–40. doi:10.1023/A:1018054314350
11. Breiman L. Random forests. *Machine Learn* (2001). 45:5–32. doi:10.1023/A:1010933404324
12. Chen T, Guestrin C. XGBoost: a scalable tree boosting system in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. New York, NY: ACM (2016). p. 785–94. doi:10.1145/2939672.2939785
13. Buntine W. Learning classification trees. *Stat Comput* (1992). 2:63–73. doi:10.1007/BF01889584
14. Chipman HA, George EI, McCulloch RE. Bayesian CART model search. *J Am Stat Assoc* (1998). 93:935–48. doi:10.1080/01621459.1998.10473750
15. Denison DGT, Mallick BK, Smith AFM. A Bayesian CART algorithm. *Biometrika* (1998). 85:363–77. doi:10.1093/biomet/85.2.363
16. Chipman HA, George EI, McCulloch RE. Bart: Bayesian additive regression trees. *Ann Appl Stat* (2010). 4:266–98. doi:10.1214/09-AOAS285
17. Schetinin V, Jakaite L, Jakaitis J, Krzanowski W. Bayesian decision trees for predicting survival of patients: a study on the us national trauma data bank. *Comput Methods Programs Biomed* (2013). 111:602–12. doi:10.1016/j.cmpb.2013.05.015
18. Gelman A, Carlin J, Stern H, Dunson D, Vehtari A, Rubin D. *Bayesian data analysis*. 3rd ed.. Boca Raton, FL: Chapman and Hall/CRC Texts in Statistical Science (Taylor & Francis) (2013).
19. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al.Scikit-learn: machine learning in Python. *J Machine Learn Res* (2011). 12: 2825–30. doi:10.5555/1953048.2078195
20. Thommen K, Goswami B, Cross AI. Bayesian decision tree (2019). Available at: https://github.com/UBS-IB/bayesian_tree.
21. Dheeru D, Karra Taniskidou E. *UCI machine learning repository* (2017).
22. Vatiwutipong P, Phewchean N. Alternative way to derive the distribution of the multivariate Ornstein-Uhlenbeck process. *Adv Differ Equ*, 2019 (2019). 2019. doi:10.1186/s13662-019-2214-1
23. Nuti G, Jiménez Rugama LlACross A-I. A Bayesian decision tree algorithm (2019). Preprint: arXiv: abs/1901.03214

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.

ORIGINAL RESEARCH
published: 26 April 2021
doi: 10.3389/fams.2021.641595

Identifying Actionable Serial Correlations in Financial Markets

Siew Ann Cheong[1,2]*, *Yann Wei Lee*[3†], *Ying Ying Li*[4†], *Jia Qing Lim*[4†], *Jiok Duan Jadie Tan*[3†] *and Xin Ping Joan Teo*[4†]

[1]*Division of Physics and Applied Physics, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore,* [2]*Complexity Institute, Nanyang Technological University, Singapore, Singapore,* [3]*Nanyang Girls High School, Singapore, Singapore,* [4]*River Valley High School, Singapore, Singapore*

Financial markets are complex systems where information processing occurs at multiple levels. One signature of this information processing is the existence of recurrent sequences. In this paper, we developed a procedure for finding these sequences and a process of statistical significance testing to identify the most meaningful ones. To do so, we downloaded daily closing prices of the Dow Jones Industrial Average component stocks, as well as various assets like stock market indices, United States government bonds, precious metals, commodities, oil and gas, and foreign exchange. We mapped each financial instrument to a letter and their upward movements to words, before testing the frequencies of these words against a null model obtained by reshuffling the empirical time series. We then identify market leaders and followers from the statistically significant words in different cross sections of financial instruments, and interpret actionable trends that can be traded upon.

Keywords: financial markets, serial correlations, complex systems, information processing, recurrent sequences

OPEN ACCESS

Edited by:
Sou Cheng Choi,
Illinois Institute of Technology,
United States

Reviewed by:
Halim Zeghdoudi,
University of Annaba, Algeria
Jiajia Li,
Pacific Northwest National Laboratory (DOE), United States

****Correspondence:***
Siew Ann Cheong
cheongsa@ntu.edu.sg

†These authors have contributed equally to this work

Specialty section:
This article was submitted to Mathematics of Computation and Data Science,
a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *14 December 2020*
Accepted: *29 March 2021*
Published: *26 April 2021*

Citation:
Cheong SA, Lee YW, Li YY, Lim JQ, Tan JDJ and Teo XPJ (2021) Identifying Actionable Serial Correlations in Financial Markets. Front. Appl. Math. Stat. 7:641595. doi: 10.3389/fams.2021.641595

1 INTRODUCTION

In his seminal 1970 paper, Fama introduced the notion of an *efficient market*, within which the prices of securities fully reflect all information from the past, as well as future expectations on their returns [1]. In his paper, the empirical evidence Fama cited as supporting this efficient market hypothesis is zero (or close to zero) serial correlation. However, if financial markets are truly efficient, then it would be impossible for traders to profit beyond the fundamental values of the securities. Naturally, the only trading strategy that makes sense in an efficient market would be buy-and-hold. This brings us then to the elephant in the room: why are there so many hedge funds (according to https://www.investopedia.com/terms/h/hedgefund.asp, more than 10,000 of them) in the world, and why are so many of them making money? Fundamentally, all hedge funds engage in some form of *technical trading* [2–4], frequently dismissed by financial economists as not founded on firm principles. The profitability of technical trading was first investigated by Lukac et al. [5] and Brock et al. [6]. Testing 12 technical trading rules for 12 commodities between 1978 and 1984, Lukac et al. found that seven rules produced significant gross returns, while four rules produced significant net returns and significant risk-adjusted returns, after taking into account transportation and storage costs. Testing the commonly used moving average and trading range break rules on the Dow Jones index from 1897 to 1986, Brock et al. found these technical trading rules generating significant positive returns, especially from buy signals. Later, Levich and Thomas [7], Parisi and Vasquez [8], Kwon and Kish [9], also showed that technical trading can be significantly profitable, for currency futures contracts between 1976 and 1990, for many stocks on the Chilean stock market between 1989 and 1998, and for the New York Stock Exchange value-weighted index over the period 1962 to 1996, respectively.

When the technical trading rules were tested within shorter subperiods in Refs. [7] and [9], their profitabilities were found to be lower for the last sub-periods, 1986 to 1990 and 1985 to 1996, respectively. Kwon and Kish suspected that this was due to the market becoming more efficient after computerization. But as they pondered this, a wave of criticism on technical trading started, led by the papers by Ready [10] and Bajgrowicz and Scaillet [11]. In these papers, as well as those by Fang et al. [12] and Taylor [13], technical trading rules found to be profitable in the earlier periods in Ref. [6] were tested for later periods, and found to have lost their magic. Taylor, who examined the performance of momentum-based technical trading rules over the cross section of Dow Jones Industrial Average component stocks between 1928 and 2012, found the profitability of these technical trading rules evolving slowly over time, but are most profitable between the mid-1960s to the mid-1980s. This phenomenon was also observed for the performance of hedge funds. For example, earlier studies by Ackermann et al. and Liang reported stellar performances of 9.2–16.1% annual return for 906 hedge funds between January 1994 and December 1995) [14], and monthly returns ranging from −0.10 to +1.35% for 385 hedge funds between January 1994 and December 1996 [15], respectively. However, a more recent study by Fung et al. of 1,603 funds between January 1995 and December 2004 found them delivering 14–24% annual return only between 1995 and 1999 [16]. In 1998, the average return was zero, presumably because of the Long Term Capital Management crisis, and generally anemic from 2000 onwards (except for 12% in 2003), because of the NASDAQ crash in 2000. In principle, these criticisms focused on technical trading rules shown to be profitable in earlier papers, and therefore do not constitute definitive proof that technical trading rules as a whole do not work. For example, it is entirely possible that some rules work well in a given period, but as they becomes less effective in another period, other rules would become more profitable. It is also possible, while a technical trading rule is profitable in a given period, another rule that we have not considered might do even better. This last problem of finding the optimal technical trading rule based on hidden temporal patterns is one ideally suited to machine learning. In one of the earliest studies, Allen and Karjalainen used a genetic algorithm to learn technical trading rules for the daily S&P 500 prices from 1928 to 1995 [17]. Unfortunately, the rules learned did not perform better than the simple buy-and-hold strategy in out-of-sample test periods, although some rules did performed better in some periods. Fernández-Rodríguez et al. had better luck, finding that the simple technical trading rule is superior to the buy-and-hold strategy for bear and bull markets [18].

Ultimately, through the literature survey above, we see that machine learning is also not exhaustive. It finds the best, but not all that are profitable. Also, technical trading rules discovered through machine learning (including those using artificial neural networks [19–21]) do not necessarily perform better than those learned by human traders. Here let us address the question why technical trading rules have only short-lived successes, from the context of information processing by complex systems. For example, a typical language like English contains more than 100,000 words, using which we construct sentences containing about 20 words. However, an overwhelming majority of the 20^{10^5} sentences that we form by randomly selecting words are unintelligible. For an English sentence to be meaningful, the sequence of words has to closely obey a set of rules that we call the English *grammar*, and further constrained to convey *meaning*. Because of this severe reduction of the space of all possible sentences to the space of all meaningful sentences, we expect in daily usage many sentences or sub-sentences to be repeated. Another way to look at this phenomenon, is that recurrent sequences are necessary for the transmission of meaning or information, and for information processing in general. Consequently, the rules of the language make it more likely for repetition to occur. Another example of information processing in complex systems is the Krebs cycle in our biological pathways, which gets activated more than 10^{13} times a day to produce adenosine triphospate (ATP) [22, 23], a molecule that we constantly consume to stay alive. If we could measure the concentrations of all transcribed species, the highly recurrent sequences associated with the Krebs cycle would be impossible to miss.

Financial markets are also complex systems, in which participants are constantly learning how to process the complex information coursing through the system. As they do so, they add to the complex information in the system. Therefore, efficient or not, we expect hidden rules and recurrent sequences to be present in financial markets. However, as financial agents act on the market, they are themselves acted upon. As such, no agent or strategy can dominate forever, even though a previously-dominant strategy may return to dominance time and again. This explains why technical trading rules can be profitable (because exploitable information always exists in the market), and why their profitabilities are short-lived (because they generate information that can be exploited by other technical trading rules). Therefore, when a group of technical trading rules become unprofitable, another group of technical trading rules become profitable. This tells us that to hunt for this shifting information, we should look not only for correlations in time, but also correlations in space, across different instruments and different asset classes. So far, technical trading focuses on temporal patterns representing high-order serial correlations in individual instruments, but spatio-temporal patterns involving multiple instruments should also exist, and can be exploited for technical trading. Surprisingly, after a broad survey of the literature, we found no previous studies on technical trading based on spatio-temporal patterns. In fact, when we search Google Scholar using “pattern recognition” and “multivariate time series”, we end up with two hits. In the 2011 conference paper by Spiegel et al. [24], time series segmentation was first used to define features in the individual time series, before these features were used to define patterns across the small number of car accelerator sensor time series. In their 2016 paper [25], Fontes and Pereira used a three-step method involving subsequence matching and fuzzy clustering, followed by PCA to

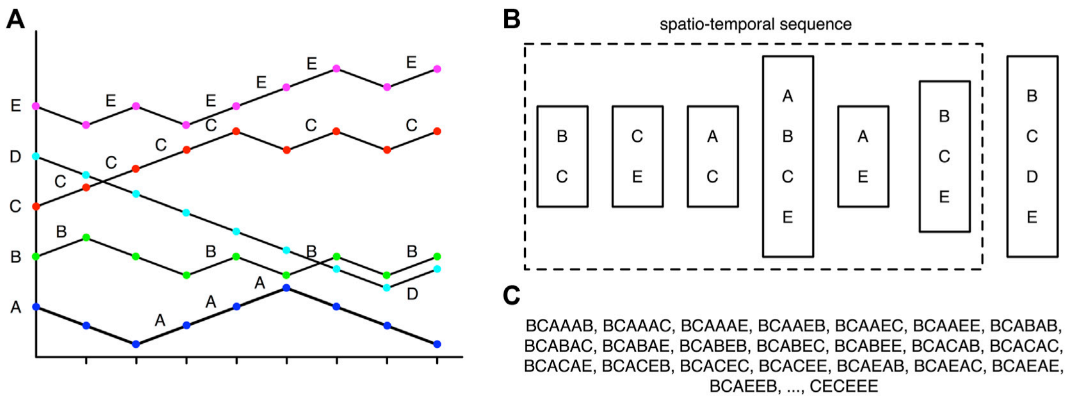

A
B
spatio-temporal sequence
C
BCAAAB, BCAAAC, BCAAAE, BCAAEB, BCAAEC, BCAAEE, BCABAB, BCABAC, BCABAE, BCABEB, BCABEC, BCABEE, BCACAB, BCACAC, BCACAE, BCACEB, BCACEC, BCACEE, BCAEAB, BCAEAC, BCAEAE, BCAEEB, ..., CECEEE

FIGURE 1 | (A) The price time series of five financial instruments, mapped to the alphabets 'A', 'B', 'C', 'D', 'E'. When the price of an instrument rises, the corresponding letter is added to the spatial cross section. **(B)** For days $t = 1$ to $t = 6$, the spatial cross sections are non-empty, but there are no price gains on day $t = 7$ (and therefore the spatial cross section on this day is empty). Thereafter, on day $t = 8$, we find a new non-empty spatial cross section. Hence we find a length-6 spatio-temporal sequence (of non-empty spatial cross sections) that ends on day $t = 6$, and another spatio-temporal sequence starting on day $t = 8$. In practice, because we set the price on a non-trading day to the price on the last trading day, the longest spatio-temporal sequence we have to deal with in this paper is length-5. **(C)** The length-6 spatio-temporal sequence shown in **(B)** can also be thought of as being equivalent to $2 \times 2 \times 2 \times 4 \times 2 \times 3 = 192$ length-6 temporal sequences. Some of these temporal sequences are shown here.

analyze sensor time series cross section from a gas turbine for monitoring and fault prediction.

In this paper, we take the natural next step to test the feasibility of technical trading using spatio-temporal patterns over the cross section of Dow Jones Industrial Average component stocks, as well as over cross sections of multiple asset classes including commodities, bonds, FOREX rates, indices, metals, and oil and gas. To find these recurrent spatio-temporal patterns, all these existing works relied on time series segmentation to first convert a real-valued time series into a symbolic time series. This is computationally heavy, so instead of time series segmentation, we describe in **Section 2** how we collected and cleaned our data, and how we map a specific choice of price movements to spatio-temporal cross sections of strings with lengths up to 5 days and comprising up to 10 alphabets. Ultimately, with this symbolic mapping all temporal patterns can be mapped to strings of alphabets. However, even after this simplification the extraction of actionable information on financial markets is not a trivial task, firstly because we have no prior knowledge what these signals would look like, and must thus analyze movements within the system, identify recurrent sequences that appear, and use these to infer the rules of information processing within financial markets. Such an analysis has been carried out in various fields to analyze various complex systems [26–30], and we ourselves have done so for natural languages [31] and teaching practices [32]. Secondly, the very many signals overlap in time to mask each other, and more importantly, participants hide their intentions as they trade. This leads to the financial markets becoming so "noisy" that one can guess price movements correctly only slightly more than 50% of the time (although Kelly showed in 1956 that this is sufficient to ensure a positive return betting on an outcome [33]). This second problem also occurs for our gene expression machinery, or the information processing machinery of other complex systems. Fortunately, network science has made great strides in systematically and independently identifying *spatial motifs* [34–36], which are collections of nodes that are co-activated much more frequently than we expected from random and uncorrelated activation of nodes, or *temporal motifs* [37, 38], which are sequences of nodes that are activated one after another. Spatial motifs can be very large, and we need a lot of data to be confident that they are not products of random fluctuations. Similarly, temporal motifs can be very long, making the space of sequences to search through very large indeed. As far as we know, there have been no efforts to develop methods for identifying *spatio-temporal motifs*, consisting of different cross sections of nodes at different lags. Therefore, in **Section 2**, we describe how to unpack spatio-temporal sequences into collections of temporal sequences, and thereafter test these empirical sequences against null models to identify sequences that are repeated more frequently than by chance. We then report in **Section 3** that in general, there are no actionable serial correlations for single instruments, but many recurrent multiple-instrument spatio-temporal sequences exist, which allow one to design trading strategies around them. Finally, we tested the feasibility of these trading strategies in **Section 4**, before summarizing our findings in **Section 5**.

2 DATA AND METHODS

2.1 Data

We downloaded two sets of time series data in the form of comma-separated values (CSV) files. The first set (see **Supplementary Table S1**) comprised daily prices of the 30 component stocks of the Dow Jones Industrial Average (DJI). These belong to the 30 largest publicly-owned United States companies, which are prominent brand names many people are familiar with. We used the maximum time period for each stock, so that we can compare them across the longest possible time period. The second set (see **Supplementary Table S2**) comprised daily prices of three to five instruments each from six different asset classes, including stock indices, precious metals, commodities, government bonds, energy materials, and foreign exchange. The 26 instruments in this second data set were selected primarily because data was readily available, and also because they are easily recognizable.

We then imported these CSV files into Python for cleaning. First, we removed empty cells or cells that contain errors, before saving the cleaned data as two separate numpy files. The first file contains the dates in International Organization for Standardization (ISO) format, while the second file contains the corresponding closing prices. For missing prices over weekends or public holidays, we set them equal to the prices of the previous days. As such, a financial instrument can only increase continuously for at most 5 days, as closing prices over the weekends are set to those on Friday.

2.2 Compiling Lists of Spatio-Temporal Sequences

To avoid having to deal with the full complexity of financial markets, but still be able to discover statistically significant patterns within the data, we map the day-to-day price change time series to symbolic sequences from a small alphabet. There are many ways this can be done, depending on what trading strategy we would like to adopt. For example, if we would like to watch for 2 days of positive price changes, and buy the instruments that are most likely to also experience positive price changes in the next one, two, or three days, we would choose to map the price changes to letters of an alphabet (one letter for each instrument), only when the price changes are positive. This example is illustrated in **Figure 1A**. Alternatively, if we would like to sell an instrument whose price is most likely to fall after 2 days of gain in the prices of two other instruments, we can map positive price changes to uppercase letters ‘A’, ‘B’, ‘C’, . . ., and negative price changes to lowercase letters ‘a’, ‘b’, ‘c’, ...

In **Figure 1B**, we show how we organize the symbolic sequences of the cross section of instruments into *spatio-temporal sequences*. A spatio-temporal sequence consists of *spatial cross sections* like (‘B’, ‘C’), (‘C’, ‘E’), (‘A’, ‘C’), (‘A’, ‘B’, ‘C’, ‘E’), (‘A’, ‘E’), (‘B’, ‘C’, ‘E’) at successive times. Spatial cross sections at different times need not be the same in size, like (‘B’, ‘C’) and (‘A’, ‘B’, ‘C’, ‘E’) for example. Spatio-temporal sequences also need not be equally long in time. For example, the spatio-temporal sequence (‘B’, ‘C’) → (‘C’, ‘E’) → (‘A’, ‘C’) → (‘A’, ‘B’, ‘C’, ‘E’) → (‘A’, ‘E’) → (‘B’, ‘C’, ‘E’) has a temporal length of 6. This spatio-temporal sequence stops here, because in the time series cross section, no instrument has an increasing price on day 7. The spatial cross section (‘B’, ‘C’, ‘D’, ‘E’) on day 8 then represents the start of the next spatio-temporal sequence, which may have a different temporal length. Going through the time series cross section $\{(\Delta p_{1,1}, \ldots, \Delta p_{1,t}, \ldots, \Delta p_{1,T}), \ldots, (\Delta p_{N,1}, \ldots, \Delta p_{N,t}, \ldots, \Delta p_{N,T})\}$, where $\Delta p_{i,t}$ is the price change of instrument $i = 1, \ldots, N$ on day $t = 1, \ldots, T$, we then obtain a list of spatio-temporal sequences $\{\Sigma_1, \ldots, \Sigma_k, \ldots, \Sigma_n\}$, where $\Sigma_k = \sigma_{k,1} \to \sigma_{k,2} \to \cdots \to \sigma_{k,m_k}$ consists of m_k spatial cross sections $\sigma_l = (s_{l,1}, s_{l,2}, \ldots, s_{l,p_l})$. In spatial cross section σ_l, the price changes of $1 \le p_l \le N$ instruments (whose symbols are $s_{l,1}, \ldots, s_{l,p_l}$) are positive.

2.3 Null Model and Test of Statistical Significance

In general, when we expand the spatio-temporal sequences into temporal sequences, and count the number of times they appear, some temporal sequences will be frequent, while others will be rare. However, a frequent temporal sequence may be less informative than a rare temporal sequence, if the former contains many highly-frequent symbols. In other words, these frequent temporal sequences can occur by chance, because their symbols are so common. Therefore, the frequencies of different temporal sequences must be tested against appropriate null models, to ensure at the very least that they are not likely to be obtained by chance.

Depending on what information we are interested in, we can construct different null models. In **Section 3.1**, we will show that the probability of empirically finding positive price movements in an instrument for r consecutive days is p^r, where p is the probability of finding positive price movement for the instrument on any given day. This suggests that the appropriate null model to use for one instrument is independent price movements on each day. We can of course use this same null model for all N instruments. However, in this null model the N instruments would be uncorrelated in time (between different time lags) and also in space (between different instruments), when strong cross correlations between instruments are well known. Using such a null model, we will find many statistically significant spatio-temporal sequences with strong cross correlations between instruments on the same days. There is no gain trading these spatio-temporal sequences, since we cannot act on strong cross correlations within the same day. Therefore, we should choose a different null model that does not throw the baby out with the bath water.

A simple null model that preserves spatial cross correlations, but contains no temporal correlations, can be obtained by reshuffling the empirical spatio-temporal sequences, as shown in **Figure 2**. If this reshuffling is done within individual spatio-temporal sequences, we also preserve the distribution of lengths. With this null model, and some additional care, it is even possible to perform statistical testing at the level of spatio-temporal sequences. However, we chose for simplicity to perform statistical testing at the level of *temporal sequences*. A temporal sequence is a simple *word* (string of symbols), like ‘BCA’, ‘BCABA’, and so on. To do the test, we extract all possible words that can be generated from the list of spatio-temporal sequences. For example, for (‘B’, ‘C’) → (‘C’, ‘E’) → (‘A’, ‘C’), we can generate the words ‘BCA’, ‘BCC’, ‘BEA’, ‘BEC’, ‘CCA’, ‘CCC’, ‘CEA’, ‘CEC’. We then count the number of times each word appears after this *unpacking* of the spatio-temporal sequences. These are our *empirical frequencies*.

Next, we shuffle the empirical spatio-temporal sequences $S = 100$ times to create an ensemble of null-model spatio-temporal sequences. For a spatio-temporal sequence of length m_k, we can generate up to $m_k!$ null-model spatio-temporal sequences. Some short spatio-temporal sequences with $m_k < 5$ will be repeated if we shuffle them $S = 100$ times, but we need not worry about repetitions even if the data contains just $n = 10$ spatio-temporal sequences, as there will be $\langle m! \rangle^n \sim 10^{17}$ distinct combinations if $\langle m! \rangle = 50$ is the average number of null-model spatio-temporal sequences that can be generated from each empirical spatio-temporal sequence. Since we shuffle the empirical spatio-temporal sequences $S = 100$ times, after unpacking the null-model spatio-temporal sequences into temporal sequences, and counting the number of times different words appear, we will have a distribution of $S = 100$ null-model frequencies for each word.

Since we sampled the null model $S = 100$ times, it is convenient for us to perform the statistical test at the level $p < 0.01$. A word that is significant at this level would have an empirical frequency that is larger than all $S = 100$ null-model frequencies. Also, because the null-model frequencies of all words are obtained simultaneously from the shuffling of spatio-temporal sequences, we do not need to make Bonferroni [39] or similar corrections [40] for the multiple comparisons that we are making. All statistically significant words will truly be at the $p < 0.01$ level of confidence. In a sense, by testing observations against the null model, we are simultaneously testing all kinds of autocorrelations and cross correlations between the sign time series of the various instruments.

3 RESULTS

3.1 1-Letter Words

When we analyze each of the 56 instruments independently, by emitting a single letter when the price increases, we are looking at bull runs of different durations in their time series. We show the distributions of durations for stocks in the first data set in **Supplementary Figure S1**, and those for instruments in the second data set in **Supplementary Figure S2**. Plotted on a linear-log scale, these graphs are all close to being linear, suggesting that the distributions are exponential. Such an exponential distribution arises very naturally if we assume that the price increase on one day is uncorrelated with a price increase on any other day. Therefore, if the probability of a price increase is p, the probability of finding a bull run over r days is simply

$$p^r = \exp(r \ln p) = \exp(-r|\ln p|). \tag{1}$$

This result should not surprise us, since it is just an unconventional way to present a very well known observation in finance, namely the *serial correlation* or *autocorrelation* is nearly zero [41]. This also means that there is no signal for a trader to act on, when the 1-letter word lengths are so distributed, beyond betting on the probability p of getting a price increase on a given day, regardless of the number of days of price increases prior to it. According to Fama and others after him, the market is thus "efficient" [1].

3.2 2-Letter Words

However, this does not mean that there is no actionable price movement information in the financial markets. In fact, trying to understand this information by looking at the price movement of a single instrument is like trying to understand the first sentence of this paragraph by looking at the distribution {_,_, _, 'a', 'a', _, _, _, 'aa', _, _, 'a', _, _, 'aa', 'a'} of the letter 'a' appearing in the words. If we use two letters, say 'a' and 'e', the distribution {'ee', _, 'e', _, 'ea', 'a', 'ee', _, _, 'aae', 'e', 'ee', 'a', _, 'e', 'aa', 'ae'} is now more informative (though still not enough for us to comprehend the sentence). The distribution {'ee', 'i', 'e', _, 'ea', 'a', 'ee', 'i', _, 'aiae', 'ie', 'ee', 'iai', 'i', 'e', 'iaia', 'ae'} becomes even more informative if we include one more letter ('i'). In this subsection, let us demonstrate (as a proof of concept) how we can extract more information from the distribution of 2-letter words. To do this, let us examine two pairs of instruments, (A = HD, B = TRV) and (A = platinum, B = USD-EUR), which are chosen because individually, their distributions of 1-letter words are the least informative (in that the probability of finding a word with length-r is closest to the product of independently finding r length-1 words).

For the HD-TRV pair, we used data between Sep 22, 1981 and Mar 7, 2018. Going through the 9,194 closing prices, we found 1,019 trading days when there were no price increases in either HD or TRV. The rest of the trading days are partitioned into 1,974 spatio-temporal sequences. The shortest of these spatio-temporal sequences are {(A)}, {(B)}, and {(A, B)}, which are the three possible spatial cross-sections. The longest spatio-temporal sequence is length-22 (price increases over multiple holidays and weekends). We focused on the 1,676 spatio-temporal sequences length-5 and shorter. These unpack into 4,213 temporal sequences, with the distribution shown in **Table 1**. As expected, after statistical testing at the level of $p < 0.01$ most of the temporal sequences are insignificant, except for BABA and BABB. This tells us that after a price increase in TRV on day 1, followed by a price increase in HD on day 2, followed by a price increase in TRV on day 3, there is a very significant chance of price increases in either HD or TRV. We can find more actionable

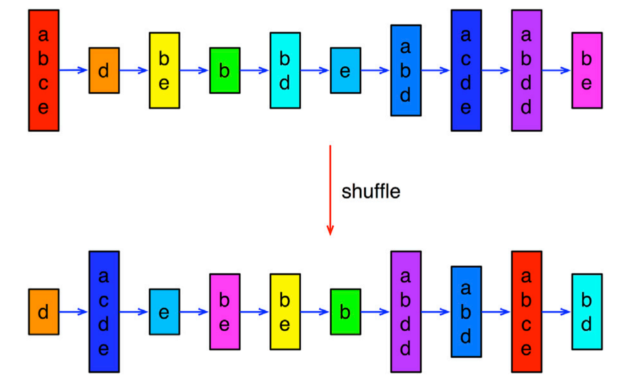


FIGURE 2 | Shuffling a length-10 spatio-temporal sequence with spatial cross sections of different sizes, to obtain a null-model ensemble of length-10 spatio-temporal sequences.

TABLE 1 | Empirical frequencies of 2-letter temporal sequences of up to length-5, corresponding to price increases in HD and TRV. In this table, an asterix indicates statistical significance at the level of $p < 0.01$.

Seq	Freq	Seq	Freq	Seq	Freq	Seq	Freq	Seq	Freq
A	453	AA	227	AAA	104	AAAA	54	AAAAA	35
								AAAAB	31
						AAAB	48	AAABA	28
								AAABB	24
				AAB	100	AABA	45	AABAA	30
								AABAB	30
						AABB	48	AABBA	21
								AABBB	24
		AB	197	ABA	99	ABAA	37	ABAAA	37
								ABAAB	34
						ABAB	39	ABABA	37
								ABABB	31
				ABB	91	ABBA	34	ABBAA	30
								ABBAB	27
						ABBB	38	ABBBA	28
								ABBBB	27
B	478	BA	232	BAA	106	BAAA	59	BAAAA	32
								BAAAB	23
						BAAB	45	BAABA	26
								BAABB	21
				BAB	90	BABA*	58	BABAA	25
								BABAB	23
						BABB*	60	BABBA	17
								BABBB	19
		BB	213	BBA	103	BBAA	40	BBAAA	32
								BBAAB	28
						BBAB	37	BBABA	30
								BBABB	27
				BBB	106	BBBA	45	BBBAA	27
								BBBAB	26
						BBBB	45	BBBBA	25
								BBBBB	27

temporal sequences if we relax the criterion of our statistical testing to $p < 0.05$.

For the platinum-USD-EUR pair, we used data between Dec 27, 1979 and Mar 13, 2018. Going through the 9,922 prices, we found 639 trading days when there were no price increases in either platinum or USD-EUR. The rest of the trading days were partitioned into 1,884 spatio-temporal sequences, and the longest spatio-temporal sequence is length-27 (price increases over multiple holidays and weekends). Focusing on the 1,449 spatio-temporal sequences length-5 and shorter, we find that these unpack into 3,185 temporal sequences, with the distribution shown in **Table 2**. In this case, we find BA occurring more frequently than expected from the null model, at the $p < 0.01$ level. No other temporal sequences occur more frequently than expected from the null model, even at the $p < 0.05$ level. This tells us that an increase in the USD-EUR exchange rate is very likely to be followed by an increase in the price in platinum the next day—an observation that traders can act on! Interestingly, ABBA, AAABB, and AAAAB occur less frequently than expected from the null model, the first two at the $p < 0.05$ level, while the last at the $p < 0.01$ level. The observations on AAABB and AAAAB suggest that following 3–4 days of increases in the platinum price, we are likely to find an ensuing decrease in the USD-EUR exchange rate. This is also an observation that traders can act on.

3.3 5-Letter Words

In **Section 3.2**, we illustrated how we can better understand the information contained in an English sentence by going from one-letter sequences to two-letter sequences, and how the information extraction improved with three-letter sequences. In this subsection, let us show that this is also true for financial markets, by going to a cross section of five stocks, (A) GE, (B) CSCO, (C) HD, (D) JPM, (E) MMM, using prices between Feb 16, 1990 and Mar 8, 2018. Out of the 10,248 trading days, there are 2,225 days on which there are no price increases in any of the five stocks. From the remaining 8,023 trading days, we found 1,987 spatio-temporal sequences of lengths between 1 and 5. After unpacking, we obtained 158,106 temporal sequences.

Out of $5 + 5^2 + 5^3 + 5^4 + 5^5 = 3,905$ distinct five-letter temporal sequences of length up to 5, we found 183 temporal sequences ($< 5\%$) that are statistically significant at the level of $p < 0.05$. Of these, 35 ($< 1\%$) are statistically significant at the level of $p < 0.01$. These are not small numbers. The distributions of dynamical motifs are skewed in favor of longer temporal sequences. For the $p < 0.05$ temporal sequences, four are length-

TABLE 2 | Empirical frequencies of 2-letter temporal sequences of up to length-5, corresponding to price increases in platinum and USD-EUR. In this table, an empirical frequency that is significantly higher than expected from the null model is indicated by an asterix ($p < 0.05$) or two asterixes ($p < 0.01$), whereas an empirical frequency that is significantly lower than expected from the null model is indicated by a dagger ($p < 0.05$) or a double dagger ($p < 0.01$).

Seq	Freq	Seq	Freq	Seq	Freq	Seq	Freq	Seq	Freq
A	343	AA	154	AAA	64	AAAA	47	AAAAA	17
								AAAAB‡	11
						AAAB	37	AAABA	15
								AAABB†	11
				AAB	77	AABA	48	AABAA	22
								AABAB	18
						AABB	46	AABBA	18
								AABBB	16
		AB†	147	ABA	83	ABAA	43	ABAAA	21
								ABAAB	13
						ABAB	35	ABABA	18
								ABABB	17
				ABB	78	ABBA†	34	ABBAA	22
								ABBAB	16
						ABBB	34	ABBBA	15
								ABBBB	13
B	317	BA**	181	BAA	83	BAAA	45	BAAAA	21
								BAAAB	14
						BAAB	41	BAABA	16
								BAABB	16
				BAB	86	BABA	49	BABAA	22
								BABAB	20
						BABB	45	BABBA	18
								BABBB	21
		BB	166	BBA	81	BBAA	47	BBAAA	22
								BBAAB	20
						BBAB	40	BBABA	14
								BBABB	22
				BBB	80	BBBA	39	BBBAA	23
								BBBAB	24
						BBBB	39	BBBBA	17
								BBBBB	23

2, five are length-3, 27 are length-4, and 147 are length-5. For the $p < 0.01$ temporal sequences, one is length-3, five are length-4, and 29 are length-5. Therefore, the identification of longer dynamical motifs seems to be easier. However, they are also less common overall. For trading, a compromise has to be found.

For this cross section of five stocks, we also found a very interesting statistic: of the 183 $p < 0.05$ temporal sequences, 39 starts with A (GE), 32 starts with B (CSCO), 27 starts with C (HD), 43 starts with D (JPM), and 42 starts with E (MMM). Restricting ourselves then to the 147 length-5 $p < 0.05$ temporal sequences, we found that 22 ends with A (GE), 47 ends with B (CSCO), 42 ends with C (HD), 14 ends with D (JPM), and 22 ends with E (MMM). This suggests that, even though we only look at five stocks, JPM, MMM, and GE are leaders in market-wide bull runs (and thus less likely to follow), whereas CSCO and HD are followers (and thus less likely to lead).

Finally, we find 11 of the $p < 0.05$ length-5 motifs containing ABB. These are AB*ABB*, BA*ABB*, CB*ABB*, CC*ABB*, DB*ABB*, EA*ABB*, E*ABB*B, EB*ABB*, EC*ABB*, ED*ABB*, EE*ABB*. Apart from E*ABB*B, ABB occurs at the end. This is an observation that traders can definitely act upon. However, since ABB is itself a $p < 0.01$ length-3 motif, perhaps it is not surprising that we find these length-5 motifs extending ABB. The situation is different with CBB, whose empirical frequency is exceeded by 15 null-model frequencies ($p = 0.15$), and thus not very significant. In spite of this, we find 8 length-5 motifs at $p < 0.05$ containing CBB. These are AB*CBB*, AD*CBB*, *CBB*BB, EA*CBB*, EB*CBB*, EC*CBB*, ED*CBB*, EE*CBB*. Except in *CBB*BB, CBB again occurs at the end of the other motifs, making them actionable. More importantly, if we compare the two series of length-5 motifs,

ABABB, BAABB, CBABB, CCABB, DBABB, **EA**ABB, **EB**ABB, **EC**ABB, **ED**ABB, **EE**ABB;

ABCBB, ADCBB, **EA**CBB, **EB**CBB, **EC**CBB, **ED**CBB, **EE**CBB,

we find that six prefixes match, and their empirical frequencies are close to each other. Therefore, we can write these 12 length-5 motifs as

$$\mathrm{AB}\begin{bmatrix}\mathrm{ABB}\\ \mathrm{CBB}\end{bmatrix},\ \mathrm{EA}\begin{bmatrix}\mathrm{ABB}\\ \mathrm{CBB}\end{bmatrix},\ \mathrm{EB}\begin{bmatrix}\mathrm{ABB}\\ \mathrm{CBB}\end{bmatrix},\ \mathrm{EC}\begin{bmatrix}\mathrm{ABB}\\ \mathrm{CBB}\end{bmatrix},\ \mathrm{ED}\begin{bmatrix}\mathrm{ABB}\\ \mathrm{CBB}\end{bmatrix},\ \mathrm{EE}\begin{bmatrix}\mathrm{ABB}\\ \mathrm{CBB}\end{bmatrix}, \tag{2}$$

and then further as

$$\begin{bmatrix} AB \\ EA \\ EB \\ EC \\ ED \\ EE \end{bmatrix} \begin{bmatrix} ABB \\ CBB \end{bmatrix} = \begin{bmatrix} AB \\ E \begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} \end{bmatrix} \begin{bmatrix} ABB \\ CBB \end{bmatrix}. \tag{3}$$

In doing so, we are repacking the 12 length-5 temporal motifs back into a spatio-temporal motif.

3.4 10-Letter Words

Ultimately, there are tens of thousands of stocks on the New York Stock Exchange and other US exchanges, so the 30 DJI component stocks, or even the 500 S&P 500 component stocks cannot provide a comprehensive picture on all information flowing through these stock markets. If we go beyond stock markets, to include assets from other financial markets (commodities, oil and gas, bonds, foreign exchange, . . .), it is clear a cross section of five instruments represents not even the tip of an iceberg. It is thus tempting to consider cross sections of many more instruments. However, as we have seen from **Section 3.2** and **Section 3.3**, while the numbers of spatio-temporal sequences remain comparable, the numbers of temporal sequences that we unpack going from a two-letter alphabet to a five-letter alphabet increased 50-fold. If we now go from a five-letter alphabet to a 10-letter alphabet, the numbers of temporal sequences is expected to increase another 30-fold, to approximately 5×10^6. If this number of temporal sequences gets any larger, testing them statistically will no longer be feasible on a desktop computer, so we must forget going to 50 letters (utilizing both uppercase and lowercase letters), where we would have to deal with 3×10^8 temporal sequences after unpacking, or larger alphabets.

At the same time, the information we can extract from financial markets become richer when we use larger alphabets. To illustrate this, and also highlight new problems encountered, let us analyze two large cross sections of instruments in this subsection: 1) a cross section of 10 DJI component stocks, and 2) a cross section of nine mixed assets. In the first cross section, (A) GE, (B) CSCO, (C) HD, (D) JPM, (E) MMM, (F) MRK, (G) UTX, (H) BA, (I) VZ, (J) XOM, we used prices between Feb 16, 1990 and Mar 8, 2018. Out of 10,248 trading days, we found 1,863 days on which there were no price increases in any of the stocks. For the rest of the trading days, price increases were organized into 1,738 spatio-temporal sequences up to length-5. For 10 stocks, there are 111,110 unique temporal sequences. Out of these, 3,580 temporal sequences are statistically significant at the $p < 0.05$ level, while 672 are significant at the $p < 0.01$ level. As functions of sequence length, these are distributed as

$$(F_1,\ F_2,\ F_3,\ F_4,\ F_5) = (2,\ 43,\ 42,\ 235,\ 3258) \tag{4}$$

for $p < 0.05$, and

$$(F_1,\ F_2,\ F_3,\ F_4,\ F_5) = (0,\ 20,\ 3,\ 33,\ 616) \tag{5}$$

for $p < 0.01$, where F_n is the frequency of length-n dynamical motifs. Of the $p < 0.05$ motifs,

$$\left(F_A^{pre},\ F_B^{pre},\ F_C^{pre},\ F_D^{pre},\ F_E^{pre},\ F_F^{pre},\ F_G^{pre},\ F_H^{pre},\ F_I^{pre},\ F_J^{pre}\right) = (214,\ 149,\ 207,\ 197,\ 363,\ 605,\ 105,\ 143,\ 516,\ 1081), \tag{6}$$

where F_σ^{pre} is the number of motifs starting with the letter $\sigma = $ A, . . ., J, whereas

$$\left(F_{5,A}^{post},\ F_{5,B}^{post},\ F_{5,C}^{post},\ F_{5,D}^{post},\ F_{5,E}^{post},\ F_{5,F}^{post},\ F_{5,G}^{post},\ F_{5,H}^{post},\ F_{5,I}^{post},\ F_{5,J}^{post}\right) = (426,\ 289,\ 830,\ 125,\ 270,\ 159,\ 204,\ 495,\ 191,\ 269), \tag{7}$$

F_σ^{post} being the number of motifs ending with the letter $\sigma = $ A, . . . , J. This tells us that XOM, MRK, VZ, MMM are the top four leaders, while HD, BA, GE, CSCO, MMM are the top five followers. Again, leaders are not followers (with the exception of MMM).

Because of the number of $p < 0.05$ motifs, we can no longer visually inspect individual sequences like we did for the 5-letter case to identify actionable patterns. This is why a visualization scheme is necessary. Since XOM is the strongest lead mover, we can choose to visualize only the 1,063 length-5 motifs that start with J, in the form of a tree rooted in J. From this root, we draw branches to the 10 letters in the first level (if such sequences exist), and from each of these letters, draw branches to the 10 letters in the second level (if such sequences exist), and so on and so forth until we reach the end of the sequences (the leaves). Because we draw only existing sequences, some branches will have more leaves while others will have fewer, as shown in **Figure 3**. The tree diagrams with other roots in this cross section of 10 DJI stocks are shown in

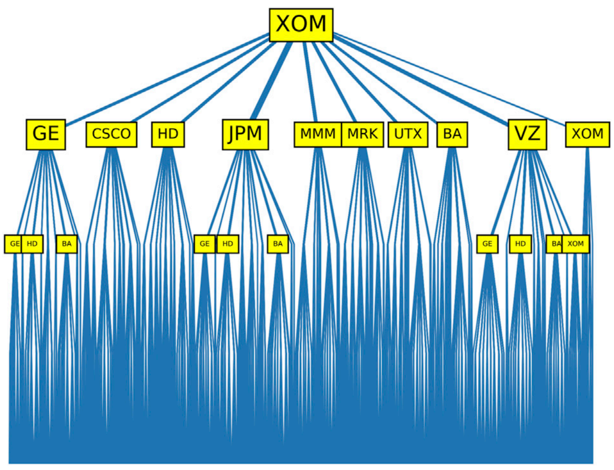


FIGURE 3 | Tree diagram of dynamical motifs rooted in (J) XOM (price increase on the first day). In this figure, we label all ten stocks with price increases on the second day, but use a larger font for (A) GE, (D) JPM, and (I) VZ, to indicate that price increases in these stocks are followed by the largest numbers of dynamical motifs. For price increases on the third day, we label only those stocks following XOM and GE/JPM/VZ, and are themselves followed by the most dynamical motifs. Except for (J) XOM following VZ, we find consistently (A) GE, (C) HD, and (H) BA following price increases on the second day. This is also true for the branches we did not highlight, as well as for the fourth day.

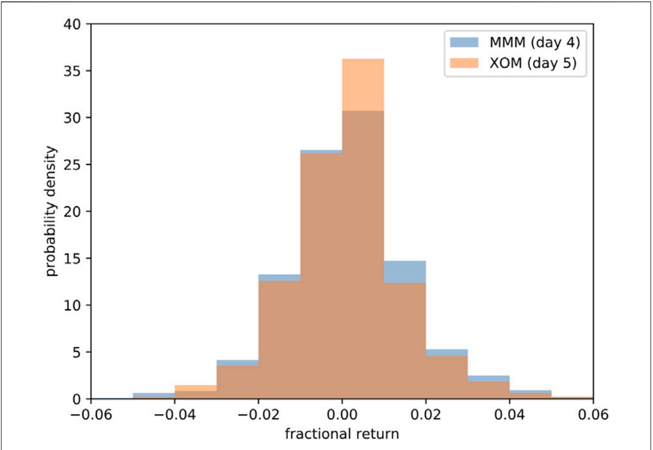


FIGURE 4 | Probability densities of the fractional returns for trading MMM on day 4 and XOM on day 5 based on the $p < 0.01$ length-5 motif XOM → VZ → BA → MMM → XOM, between Feb 16, 1990 and Mar 8, 2018.

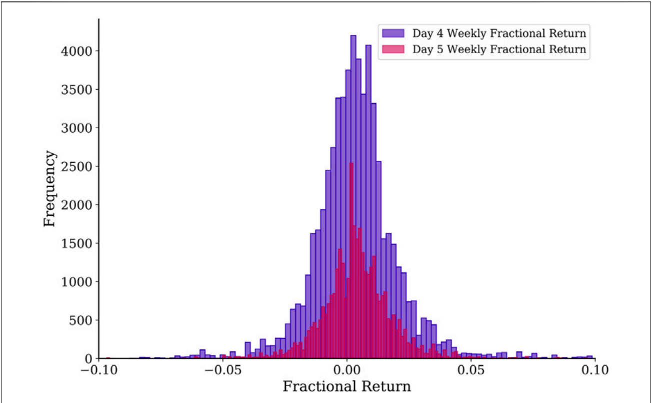


FIGURE 5 | Distributions of fractional returns for trading on day 4 and day 5 of $p < 0.01$ length-5 motifs in the first cross section of DJI component stocks, comprising (A) GE, (B) CSCO, (C) HD, (D) JPM, (E) MMM, (F) MRK, (G) UTX, (H) BA, (I) VZ, (J) XOM, between Feb 16, 1990 and Mar 8, 2018.

Supplementary Figure S3. In **Supplementary Figure S4**, we also show the tree diagrams for a second cross section of 10 DJI stocks.

The second cross section we feature here consists of indices and precious metals, namely (A) gold, (B) silver, (C) palladium, (D) S&P 500, (E) Hang Seng, (F) platinum, (G) Dow Jones, (H) Nikkei, and (I) NASDAQ. We used prices between Apr 2, 1990 and Jan 28, 2018. Of the 10,164 trading days, we find 1,604 days on which there were no price increases in any of the assets. For the rest of the trading days, price increases were organized into 1,607 spatio-temporal sequences up to length-5. For the nine assets, 9,145 temporal sequences are statistically significant at the $p < 0.01$ level. This is many more than the 672 $p < 0.01$ temporal sequences found for the cross section of 10 DJI stocks, suggesting that the DJI cross section is well exploited, and therefore there is less actionable information remaining. In contrast, the cross section of nine mixed assets investigated here is not well exploited, so there is more information that traders can act on. This is to be expected, since fewer funds and traders simultaneously trade indices and precious metals in the portfolios they manage.

We also found the 9,145 $p < 0.01$ temporal sequences distributed as

$$(F_1, F_2, F_3, F_4, F_5) = (0, 81, 697, 6463, 1904). \tag{8}$$

Unlike for the cross section of 10 DJI stocks, in this cross section of nine mixed assets, length-4 sequences outnumber length-5 sequences. For the length-4 sequences,

$$\begin{gathered}\left(F_{4,A}^{pre}, F_{4,B}^{pre}, F_{4,C}^{pre}, F_{4,D}^{pre}, F_{4,E}^{pre}, F_{4,F}^{pre}, F_{4,G}^{pre}, F_{4,H}^{pre}, F_{4,I}^{pre}\right)\\ = (719, 708, 727, 729, 727, 707, 729, 691, 724),\end{gathered} \tag{9}$$

while

$$\begin{gathered}\left(F_{4,A}^{post}, F_{4,B}^{post}, F_{4,C}^{post}, F_{4,D}^{post}, F_{4,E}^{post}, F_{4,F}^{post}, F_{4,G}^{post}, F_{4,H}^{post}, F_{4,I}^{post}\right)\\ = (711, 720, 724, 712, 712, 727, 724, 722, 711).\end{gathered} \tag{10}$$

None of the assets are particularly strong leaders or strong followers. For the length-5 sequences, from the distribution

$$\begin{gathered}\left(F_{5,A}^{pre}, F_{5,B}^{pre}, F_{5,C}^{pre}, F_{5,D}^{pre}, F_{5,E}^{pre}, F_{5,F}^{pre}, F_{5,G}^{pre}, F_{5,H}^{pre}, F_{5,I}^{pre}\right)\\ = (394, 115, 186, 281, 88, 209, 292, 71, 268),\end{gathered} \tag{11}$$

we find the strong leaders are (A) gold, (D) S&P 500, (G) Dow Jones, (I) NASDAQ, while the weak leaders are (B) silver, (E) Hang Seng, (H) Nikkei. Gold is well known to be a leading indicator of inflation [42, 43], so it would not be surprising for gold to also lead smaller-scale market movements. Using the Hilbert transform to complexify the return time series of major global indices, Vodenska et al. showed convincingly that FOREX markets lead equity markets, and the US equity market is one of the leaders of other equity markets [44]. Finally, from the distribution

$$\begin{gathered}\left(F_{5,A}^{post}, F_{5,B}^{post}, F_{5,C}^{post}, F_{5,D}^{post}, F_{5,E}^{post}, F_{5,F}^{post}, F_{5,G}^{post}, F_{5,H}^{post}, F_{5,I}^{post}\right)\\ = (102, 132, 437, 234, 358, 111, 164, 197, 169),\end{gathered} \tag{12}$$

we see that (C) palladium, (E) Hang Seng are strong followers, while (A) gold, (F) platinum are weak followers. As expected, (A) gold being a strong leader is a weak follower, whereas (E) Hang Seng being a weak leader is a strong follower. Surprisingly, (C) palladium is a strong follower, even though it is not weak as a leader. Similarly, (F) platinum is one of the weakest followers, even though it is not the strongest of leaders.

The tree diagrams for length-5 $p < 0.01$ dynamical motifs in this cross section of nine mixed assets are shown in **Supplementary Figure S5**. If we look at the tree diagram rooted in (A) gold, who is the strongest leader in this cross section, we find that price increases in (A) gold on the first day is followed most strongly by price increases in (D) S&P 500, (E) Hang Seng, (G) Dow Jones, (I) NASDAQ. This response by stock indices to rallies in the gold price is not at all surprising, apart from the weak response from (H) Nikkei. For subsequent days, price increases occur predominantly in (E) Hang Seng and (H) Nikkei. As it turned out, whoever the leader was on the first day (E) Hang Seng and (H) Nikkei were consistently the assets that

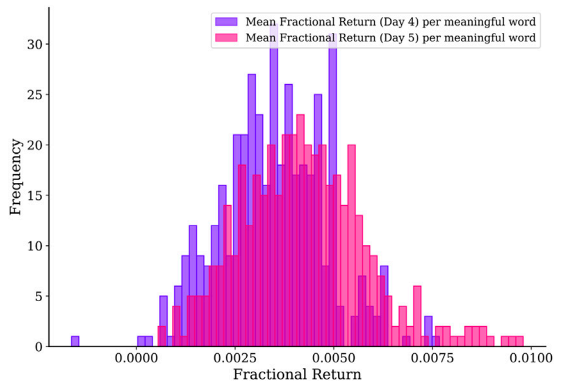


FIGURE 6 | Distributions of weekly average fractional returns for trading on day 4 and day 5 of all $p < 0.01$ length-5 motifs in the first cross section of DJI component stocks, comprising (A) GE, (B) CSCO, (C) HD, (D) JPM, (E) MMM, (F) MRK, (G) UTX, (H) BA, (I) VZ, (J) XOM, between Feb 16, 1990 and Mar 8, 2018.

responded on the second and third days. This behavior is not seen in the United States indices, (D) S&P 500, (G) Dow Jones, and (I) NASDAQ. It is well known that Nikkei follows United States indices [45–47]. The tree diagrams of two other cross sections of mixed assets are also shown in **Supplementary Figures S6, S7**.

4 FEASIBILITY

After identifying the dynamical motifs, let us check whether they can be traded profitably. We do this for length-5 motifs, which are the most informative. First, let us explain how a simple trading strategy can be developed using one specific $p < 0.01$ length-5 motif, say XOM → VZ → BA → MMM → XOM from the first DJI cross section. In this motif, price increase first occurs for XOM on day 1, then for VZ on day 2, for BA on day 3, for MMM on day 4, and finally for XOM on day 5. After observing a price increase in XOM at the end of day 1, we can of course buy VZ, BA, MMM, and XOM, to sell at the ends of days 2, 3, 4, and 5, respectively. However, this is risky, as we cannot be sure the price increase of XOM on day 1 is the start of the length-5 motif we are targeting. It could be the start of another motif, or just an idiosyncratic price movement that is not part of any motif. Therefore, a safer way to exploit this length-5 motif is to first observe the market for 3 days. If price increase occurs for XOM on day 1, VZ on day 2, and BA on day 3, there is a strong likelihood that we are in the midst of the length-5 motif. We can then buy MMM at the end of day 3, and since it is expected to experience a price increase, sell it at the end of day 4 to make a profit. Finally, if the price of MMM does increase on day 4, we can buy XOM at the end of day 4, and sell it at the end of day 5. In this way, we can execute one to two transactions every time XOM → VZ → BA occurs.

For this length-5 motif, we find that over the period Feb 16, 1990 to Mar 8, 2018, the price increase sequence XOM → VZ → BA appeared 964 times, while the price increase sequence XOM → VZ → BA → MMM appeared 477 times. Buying MMM 964 times at the end of day 3 and selling it at the of day 4, we compute for each transaction the fractional return

$$r_{\mathrm{MMM}}(t) = \frac{P_{\mathrm{MMM}}(t+3) - P_{\mathrm{MMM}}(t+2)}{P_{\mathrm{MMM}}(t+2)} \tag{13}$$

for the price increase sequence XOM → VZ → BA that started on day *t*. The normalized histogram for these 964 fractional returns is shown in **Figure 4**. Similarly, of the 477 times the price increase sequence XOM → VZ → BA → MMM appeared, price increase in XOM followed 236 times. If we wait for the price increase sequence XOM → VZ → BA → MMM to appear, buy XOM at the end of day 4, and sell it at the end of day 5, we find the fractional return

$$r_{\mathrm{XOM}}(t) = \frac{P_{\mathrm{XOM}}(t+4) - P_{\mathrm{MMM}}(t+3)}{P_{\mathrm{MMM}}(t+3)} \tag{14}$$

for the price increase sequence XOM → VZ → BA that started on day *t*. The normalized histogram for these 477 fractional returns is also shown in **Figure 4**.

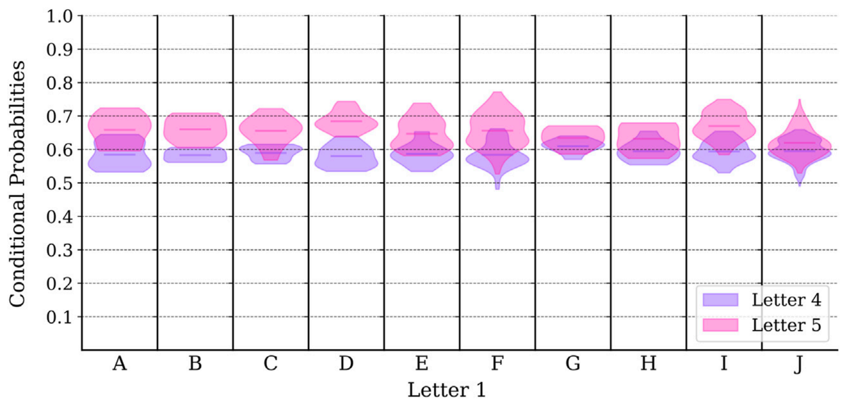


FIGURE 7 | Distributions of conditional probabilities for price increases on day 4 and day 5 for all $p < 0.01$ length-5 motifs in the first cross section of DJI component stocks, comprising (A) GE, (B) CSCO, (C) HD, (D) JPM, (E) MMM, (F) MRK, (G) UTX, (H) BA, (I) VZ, (J) XOM, between Feb 16, 1990 and Mar 8, 2018. Since we distinguish between strong and weak leaders, in this figure we show the conditional probabilities for motifs with different roots from (A) to (J).

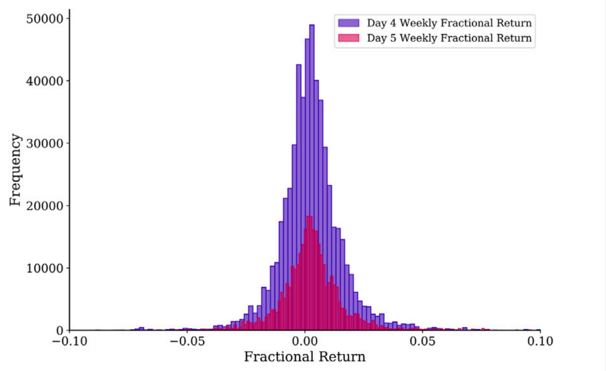


FIGURE 8 | Histograms of the fractional returns for day 4 and day 5 trading based on $p < 0.01$ length-5 motifs from the first cross section of nine mixed assets, comprising (A) gold, (B) silver, (C) palladium, (D) S&P 500, (E) Hang Seng, (F) platinum, (G) Dow Jones, (H) Nikkei, and (I) NASDAQ, using prices between Apr 2, 1990 and Jan 28, 2018.

The average fractional returns are 0.0012 for MMM, and 0.0007 for XOM. These are positive, but puny. More importantly, over the roughly 28-year period, we would have traded only $(964 + 477)/28 = 51.5$ times a year based on the motif XOM → VZ → BA → MMM → XOM. A human trader can easily trade many more times, let alone trading algorithms. Therefore, we next check the distribution of fractional returns shown in **Figure 5**, if we trade on day 4 (60,153 transactions) and day 5 (35,855 transactions) for all $p < 0.01$ length-5 motifs. The average fractional return on day 4 is 0.0035, while that on day 5 is 0.0042. While these average fractional returns are higher than those for XOM → VZ → BA → MMM → XOM alone, the distributions shown in **Figure 5** give us the wrong impression that trading in these motifs is high-risk and low-return. To put it more accurately, we get such dismal performance only if we choose to trade one random motif a week.

If we trade multiple motifs in the same week, then our prospects would be different. This is because we may fail to profit from some motifs, but still succeed in other motifs. Therefore, we should first average the fractional return over all motifs we trade in a given week, before compiling the histogram of 424 weekly average fractional returns shown in **Figure 6**. Unlike for the fractional return of individual transactions, when we trade all possible motifs and average the fractional returns over them, we find that we almost never lose money in any week. The average of the average fractional return per week is 0.0035 on day 4 and 0.0042 on day 5, the same as when we average over the distributions of fractional returns. However, since we now know the average fractional return per week is (almost) always positive, we can compound them to get an average fractional return of 0.0077 per week, or an average fractional return of 0.4004 per annum!

Another way to understand this profitability is in terms of $p(X_4 > 0|X_1 > 0, X_2 > 0, X_3 > 0)$, which is the conditional probability for a price increase $X_4 > 0$ to be observed on day 4, given that price increases $X_1 > 0$, $X_2 > 0$, $X_3 > 0$ have been observed on day 1, day 2, and day 3 for a given length-5 motif, and $p(X_5 > 0|X_1 > 0, X_2 > 0, X_3 > 0, X_4 > 0)$, which is the conditional probability for a price increase $X_5 > 0$ to be observed on day 5, given that price increases $X_1 > 0$, $X_2 > 0$, $X_3 > 0$, $X_4 > 0$ have been observed on day 1, day 2, day 3, and day 4 for the same length-5 motif. When we plot these conditional probabilities as a violin plot in **Figure 7**, we see that the conditional probability for day 4 is mostly larger than 0.5. There seems to be no correlation between the strength of this conditional probability and the strength of the stocks as leaders. Finally, we see that the conditional probability for day 5 is significantly larger than 0.5.

Before we conclude, let us also test the feasibility of our simple trading strategy for the $p < 0.01$ length-5 motifs in the first cross section of nine mixed assets, comprising (A) gold, (B) silver, (C) palladium, (D) S&P *P* 500, (E) Hang Seng, (F) platinum, (G) Dow Jones, (H) Nikkei, and (I) NASDAQ. If we trade one random motif each week, we would have the distributions of fractional returns shown in **Figure 8**. Based on these distributions, the average fractional return on day 4 (558,752 transactions) is 0.0025, while that on day 5 (318,509 transactions) is 0.0028. However, if we trade all possible motifs each week, we would have the distributions of 3,182 weekly fractional return shown in **Figure 9**. Just like for the first cross section of DJI stocks, the average weekly fractional returns on day 4 and day 5 are still 0.0025 and 0.0029, but as we can see from **Figure 9**, the downside risk of getting negative average weekly fractional returns is greatly reduced. As we can see from the violin plot in **Figure 10**, most of the conditional probabilities on day 5 are larger than their counterparts on day 4. However, only the day-5 conditional probabilities of motifs starting with (E) Hang Seng and (H) Nikkei are significantly larger than 0.5.

We also investigated a second cross section of DJI stocks, as well as a second and third cross sections of mixed assets. The distributions of fractional returns, weekly average fractional returns, and conditional probabilities of these cross sections are shown in **Supplementary Figures S8 and S9**.

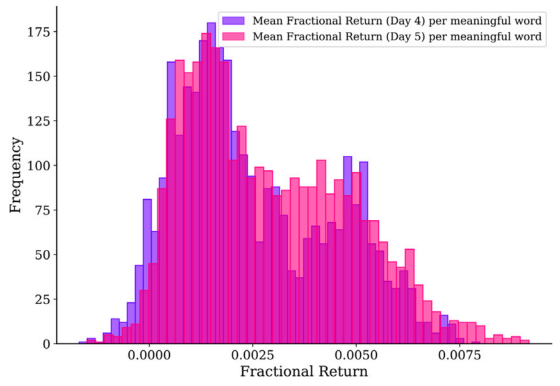


FIGURE 9 | Histograms of the weekly average fractional returns for day 4 and day 5 trading based on all $p < 0.01$ length-5 motifs from the first cross section of nine mixed assets, comprising (A) gold, (B) silver, (C) palladium, (D) S&P 500, (E) Hang Seng, (F) platinum (G) Dow Jones, (H) Nikkei, and (I) NASDAQ, using prices between Apr 2, 1990 and Jan 28, 2018.

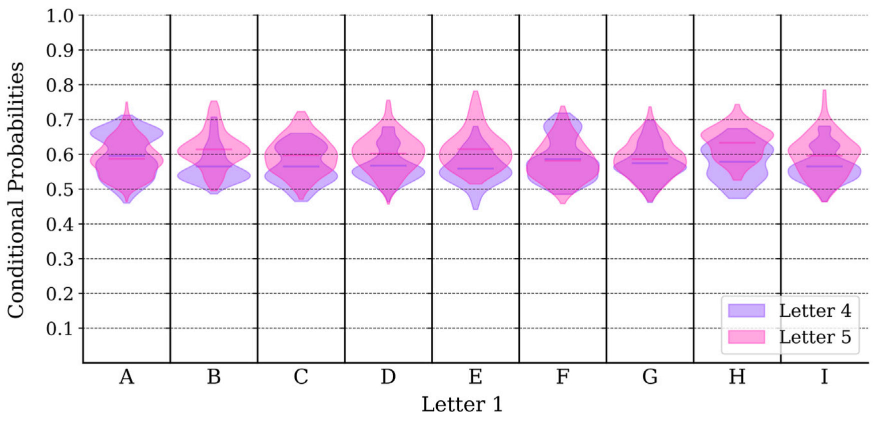


FIGURE 10 | Histograms of the conditional probabilities for day 4 and day 5 trading based on all $p < 0.01$ length-5 motifs from the first cross section of nine mixed assets, comprising (A) gold, (B) silver, (C) palladium, (D) S&P 500, (E) Hang Seng, (F) platinum, (G) Dow Jones, (H) Nikkei, and (I) NASDAQ, using prices between Apr 2, 1990 and Jan 28, 2018.

5 CONCLUSIONS AND OUTLOOK

In this paper, we explained how information processing by self-organized functions in complex systems lead to the existence of recurrent activity sequences or dynamical motifs. In financial markets, which are also complex systems, past and expected information that gets incorporated into prices must therefore be in the form of recurrent sequences. Thus far, technical traders have exploited temporal patterns corresponding to high-order serial correlations of individual instruments, but actionable spatio-temporal patterns (also called *dynamical motifs*) must also exist. To identify these dynamical motifs $\Sigma_k = \sigma_{k,1} \to \sigma_{k,2} \to \cdots \to \sigma_{k,m_k}$ with m_k spatial cross sections, $\sigma_{k,l} = (s_{k,l,1}, s_{k,l,2}, \ldots, s_{k,l,p_{k,l}})$, each of which containing $p_{k,l}$ activities, we first described a procedure for mapping price increases in a spatial cross section of financial instruments to an alphabet, so that price increases in the cross section can be mapped first to a symbolic spatio-temporal sequence, and then unpacked into a collection of temporal sequences represented as simple strings. We then described how statistically significant temporal sequences can be identified by testing the empirical frequencies of these frequencies against a null model obtained by reshuffling the spatio-temporal sequence (or collection of spatio-temporal sequences). Such a null model preserves equal-time spatial cross correlations, but completely destroys any serial correlations. Dynamical motifs that traders can act on are thus temporal sequences that occur more frequently then expected from the null model.

We tested the above procedure on the 30 DJI component stocks, as well as 26 instruments from various asset classes, to find the absence of serial correlations that traders can exploit, if they are traded individually. We then tested the procedure on two pairs of instruments, to find two length-4 dynamical motifs for (HD, TRV) that are statistically significant at the $p < 0.01$ level, and one length-2 dynamical motif for (platinum, USD-EUR) that is statistically significant at the $p < 0.01$ level. After testing the procedure next on a cross section of five DJI component stocks, and finding 35 dynamical motifs (29 of which are length-5) that are statistically significant at the $p < 0.01$ level, we proceeded to identify dynamical motifs in five cross sections containing eight to ten instruments. For a cross section of 10 DJI component stocks and a cross section of nine mixed assets, we reported in detail the 672 and 9,145 dynamical motifs that are statistically significant at the $p < 0.01$ level, how to identify leaders and followers, and more importantly, how to visualize these in the form of tree diagrams with different roots. Finally, we checked using our historical data whether these dynamical motifs can be traded feasibly, by targeting the price increases expected on day 4 and day 5 in length-5 motifs. We showed that if we trade only a single dynamical motif, the downside risk is appreciable, even though the expected fractional return is positive. While the expected fractional return is not greatly improved by trading all $p < 0.01$ length-5 motifs, we found that

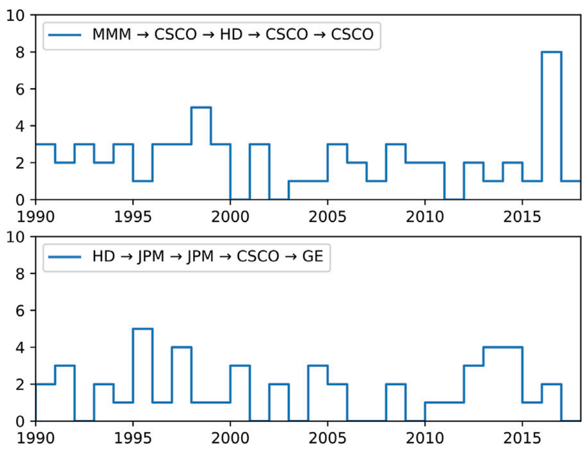


FIGURE 11 | The number of times two length-5 motifs ((top) MMM → CSCO → HD → CSCO → CSCO, and (bottom) HD → JPM → JPM → CSCO → GE) in the cross section of five DJI component stocks (A) GE, (B) CSCO, (C) HD, (D) JPM, (E) MMM, appear for each year between Feb 16, 1990 and Mar 8, 2018.

the downside risk is greatly reduced. This is true for the 10-DJI-component-stock cross section, as well as for the 9-mixed-assets cross section. For the 10-DJI-component-stock cross section with 616 $p < 0.01$ length-5 motifs, downside risk was practically non-existent, and an expected fractional return of 0.0077 per week, or equivalently 0.4004 per annum could be achieved.

In this study, we identified dynamical motifs consisting of price increases over five consecutive days from daily prices. For 616 length-5 motifs in the cross section of 10 DJI component stocks, we could execute 96,000 trades over 28 years, or about 3,400 trades per year. For 9,145 length-5 motifs in the cross section of nine mixed assets, we could execute about 877,300 trades, or about 31,300 trades per year. To get better returns, a trader would want to trade more frequently. This can be done by going to higher-frequency data, and use the high-frequency motifs identified for trading. We do not know how well such a strategy will perform, but imagine it doing better, since in high-frequency data, autocorrelations and cross correlations do not have time to die out, and therefore motifs would become easier to identify, and are also statistically more significant. In this paper, we also mapped all price increases in an instrument to a single letter. If we do not have many instruments, it is possible (and perhaps desirable) to use two letters per instrument, so that one would represent a small increase, while the other would represent a large increase. Alternatively, we can map the price increases in an instrument to more than one instance of the letter. For example, an increase of 0–1% can be mapped to A, an increase of 1–2% to AA, and an increase of 2–5% to AAA. Traders can then choose to act only if a large price increase is expected. Other variations are also possible.

As a final caveat, let us say that like for purely temporal patterns of single instruments, the profitabilities of spatio-temporal patterns containing multiple instruments are also expected to be short-lived, because once a spatio-temporal pattern becomes dominant it can be exploited by other spatio-temporal patterns. In **Figure 11** we show the frequencies of two length-5 motifs (at the $p < 0.01$ of statistical significance) over the period 1990 to 2018. In general, these frequencies are low, so for the most part it is difficult to tell visually whether they occurred uniformly over the period, or their occurrences were concentrated over certain subperiods. However, the frequency spike of MMM → CSCO → HD → CSCO → CSCO in 2016 will surely not be the product of a uniform probability that produced the frequencies in other years. Even though this has not happened in our data sets, we must be prepared for the eventuality of temporal motifs losing their statistical significance. Therefore, as we trade the significant sequences, we must at the same time be mining for new significant sequences. Once these latter sequences are discovered, they should be added to the trading pool, but we must also develop the criterion for discarding sequences that are no longer significant. This must be done in a way that maximizes the lifetime returns from such sequences, weighted against the potential for losses at the end of their usable lifetimes.

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 in the article/**Supplementary Material**.

AUTHOR CONTRIBUTIONS

SC conceived the study, wrote some of the Python programs, and wrote the manuscript. YWL, YYL, JL, JT, and XT downloaded the data sets and wrote the rest of the Python programs. All authors analyzed the results and reviewed the manuscript.

ACKNOWLEDGMENTS

YWL, YYL, JL, JT, and XT thank the Singapore Ministry of Education Science Mentorship Program for the opportunity to participate in this study.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fams.2021.641595/full#supplementary-material.

REFERENCES

1. Fama EF. Efficient Capital Markets: A Review of Theory and Empirical Work. *J Finance* (1970). 25:383–417. doi:10.2307/2325486
2. Brown DP, and Jennings RH. On Technical Analysis. *Rev Financ Stud* (1989). 2:527–51. doi:10.1093/rfs/2.4.527
3. Park CH, and Irwin SH. What Do We Know About the Profitability of Technical Analysis? *J Econ Surv* (2007). 21:786–826. doi:10.1111/j.1467-6419.2007.00519.x
4. Nazário RT, e Silva JL, Sobreiro VA, and Kimura H. A Literature Review of Technical Analysis on Stock Markets. *Q Rev Econ Finance* (2017). 66:115–26. doi:10.1016/j.qref.2017.01.014
5. Lukac LP, Brorsen BW, and Irwin SH. A Test of Futures Market Disequilibrium Using Twelve Different Technical Trading Systems. *Appl Econ* (1988). 20:623–39. doi:10.1080/00036848800000113
6. Brock W, Lakonishok J, and LeBaron B. Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. *J Finance* (1992). 47:1731–64. doi:10.2307/2328994
7. Levich RM, and Thomas LR. The Significance of Technical Trading-Rule Profits in the Foreign Exchange Market: A Bootstrap Approach. *J Int Money Finance* (1993). 12:451–74. doi:10.1016/0261-5606(93)90034-9
8. Parisi F, and Vasquez A. Simple Technical Trading Rules of Stock Returns: Evidence From 1987 to 1998 in Chile. *Emerg Mark Rev* (2000). 1:152–64. doi:10.1016/S1566-0141(00)00006-6
9. Kwon KY, and Kish RJ. Technical Trading Strategies and Return Predictability: NYSE. *Appl Financ Econ* (2002). 12:639–53. doi:10.1080/09603100010016139
10. Ready MJ. Profits From Technical Trading Rules. *Financ Manage* (2002). 31:43–61. doi:10.2307/3666314
11. Bajgrowicz P, and Scaillet O. Technical Trading Revisited: False Discoveries, Persistence Tests, and Transaction Costs. *J Financ Econ* (2012). 106:473–91. doi:10.1016/j.jfineco.2012.06.001

12. Fang J, Jacobsen B, and Qin Y. Predictability of the Simple Technical Trading Rules: An Out-of-Sample Test. *Rev Financ Econ* (2014). 23:30–45. doi:10.1016/j.rfe.2013.05.004
13. Taylor N. The Rise and Fall of Technical Trading Rule Success. *J Bank Finance* (2014). 40:286–302. doi:10.1016/j.jbankfin.2013.12.004
14. Ackermann C, McEnally R, and Ravenscraft D. The Performance of Hedge Funds: Risk, Return, and Incentives. *J Finance* (1999). 54:833–74. doi:10.1111/0022-1082.00129
15. Liang B. On the Performance of Hedge Funds. *Financ Anal J* (1999). 55:72–85. doi:10.2139/SSRN.89490
16. Fung W, Hsieh DA, Naik NY, and Ramadorai T. Hedge Funds: Performance, Risk, and Capital Formation. *J Finance* (2008). 63:1777–803. doi:10.1111/j.1540-6261.2008.01374.x
17. Allen F, and Karjalainen R. Using Genetic Algorithms to Find Technical Trading Rules. *J Financ Econ* (1999). 51:245–71. doi:10.1016/S0304-405X(98)00052-X
18. Fernández-Rodríguez F, González-Martel C, and Sosvilla-Rivero S. On the Profitability of Technical Trading Rules Based on Artificial Neural Networks: Evidence from the Madrid Stock Market. *Econ Lett* (2000). 69:89–94. doi:10.1016/S0165-1765(00)00270-6
19. Kimoto T, Asakawa K, Yoda M, and Takeoka M. Stock Market Prediction System with Modular Neural Networks. In: Proceedings of the 1990 International Joint Conference on Neural Networks (IJCNN 1990); 1990 June 17–21; San Diego, CA. New York, NY: IEEE. (2009). p. 1–6.
20. Zhang Y, and Wu L. Stock Market Prediction of S&P 500 via Combination of Improved BCO Approach and BP Neural Network. *Expert Syst Appl* (2009). 36:8849–54. doi:10.1016/j.eswa.2008.11.028
21. Guresen E, Kayakutlu G, and Daim TU. Using Artificial Neural Network Models in Stock Market Index Prediction. *Expert Syst Appl* (2011). 38: 10389–97. doi:10.1016/j.eswa.2011.02.068
22. Wise J, Roush R, and Fowler S. *Concepts of Biology*. Scotts Valley, CA: CreateSpace Independent Publishing Platform (2013).
23. Tymoczko JL, Berg JM, Stryer L, and Gatto G. *Biochemistry: A Short Course*. Austin, TX: Macmillan Learning (2019).
24. Spiegel S, Gaebler J, Lommatzsch A, De Luca E, and Albayrak S. Pattern Recognition and Classification for Multivariate Time Series. In: Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data; August 21, 2011, San Diego, CA. New York, NY: ACM (2011). p. 34–42.
25. Fontes CH, and Pereira O. Pattern Recognition in Multivariate Time Series – A Case Study Applied to Fault Detection in a Gas Turbine. *Eng Appl Artif Intell* (2016). 49:10–8. doi:10.1016/j.engappai.2015.11.005
26. Zhang YQ, Li X, Xu J, and Vasilakos AV. Human Interactive Patterns in Temporal Networks. *IEEE Trans Syst Man Cybern Syst* (2015). 45: 214–22. doi:10.1109/TSMC.2014.2360505
27. Hulovatyy Y, Chen H, and Milenković T. Exploring the Structure and Function of Temporal Networks With Dynamic Graphlets. *Bioinformatics* (2015). 31: i171–80. doi:10.1093/bioinformatics/btv227
28. Xuan Q, Fang H, Fu C, and Filkov V. Temporal Motifs Reveal Collaboration Patterns in Online Task-Oriented Networks. *Phys Rev E* (2015). 91:052813. doi:10.1103/PhysRevE.91.052813
29. Paranjape A, Benson AR, and Leskovec J. Motifs in Temporal Networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017); 2017 Feb 6–10; Cambridge, UK. New York, NY: ACM (2017). p. 601–10.
30. Liu P, Benson AR, and Charikar M. Sampling Methods for Counting Temporal Motifs. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining; 2019 Feb 11–15; Melbourne, Australia. New York, NY: ACM (2019). p. 294–302.
31. Goh WP, Luke KK, and Cheong SA. Functional Shortcuts in Language Co-Occurrence Networks. *PLoS ONE* (2018). 13:0203025. doi:10.1371/Fjournal.pone.0203025
32. Goh WP, Kwek D, Hogan D, and Cheong SA. Complex Network Analysis of Teaching Practices. *Eur Phys J Data Sci* (2014). 3:34. doi:10.1140/epjds/s13688-014-0034-9
33. Kelly JL. A New Interpretation of Information Rate. In: LC McClean, EO Thorp, and WT Ziemba, editors. *The Kelly Capital Growth Investment Criterion*. Singapore: World Scientific (2011). p. 25–34.
34. Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, and Alon U. Network Motifs: Simple Building Blocks of Complex Networks. *Science* (2002). 298:824–7. doi:10.1126/science.298.5594.824
35. Shen-Orr SS, Milo R, Mangan S, and Alon U. Network Motifs in the Transcriptional Regulation Network of *Escherichia coli*. *Nat Genet* (2002). 31:64–8. doi:10.1038/ng881
36. Wernicke S, and Rasche F. FANMOD: A Tool for Fast Network Motif Detection. *Bioinformatics* (2006). 22:1152–3. doi:10.1093/bioinformatics/btl038
37. Kovanen L, Karsai M, Kaski K, Kertész J, and Saramäki J. Temporal Motifs in Time-Dependent Networks. *J Stat Mech Theor Exp.* (2011). P11005. doi:10.1088/1742-5468/2011/11/P11005
38. Kovanen L, Kaski K, Kertész J, and Saramäki J. Temporal Motifs Reveal Homophily, Gender-Specific Patterns, and Group Talk in Call Sequences. *Proc Natl Acad Sci U S A* (2013). 110:18070–5. doi:10.1073/pnas.1307941110
39. Dunn OJ. Multiple Comparisons Among Means. *J Am Stat Assoc* (1961). 56: 52–64. doi:10.1080/01621459.1961.10482090
40. Šidák Z. Rectangular Confidence Regions for the Means of Multivariate Normal Distributions. *J Am Stat Assoc* (1967). 62:626–33. doi:10.1080/01621459.1967.10482935
41. Chakraborti A, Toke IM, Patriarca M, and Abergel F. Econophysics Review: I. Empirical Facts. *Quant Finance* (2011). 11:991–1012. doi:10.1080/14697688.2010.539248
42. Moore GH. Gold Prices and a Leading Index of Inflation. *Challenge* (1990). 33: 52. doi:10.1016/S0148-6195(97)00034-9
43. Mahdavi S, and Zhou S. Gold and Commodity Prices as Leading Indicators of Inflation: Tests of Long-Run Relationship and Predictive Performance. *J Econ Bus* (1997). 49:475–89. doi:10.1016/S0148-6195(97)00034-9
44. Vodenska I, Aoyama H, Fujiwara Y, Iyetomi H, and Arai Y. Interdependencies and Causalities in Coupled Financial Networks. *PloS One* (2016). 11:0150994. doi:10.1371/journal.pone.0150994
45. Kato K. Weekly Patterns in Japanese Stock Returns. *Manage Sci* (1990). 36: 1031–43. doi:10.1287/mnsc.36.9.1031
46. Becker KG, Finnerty JE, and Tucker AL. The Intraday Interdependence Structure Between US and Japanese Equity Markets. *J Financ Res* (1992). 15:27–37. doi:10.1111/j.1475-6803.1992.tb00784.x
47. Pan MS, and Hsueh LP. Transmission of Stock Returns and Volatility Between the US and Japan: Evidence from the Stock Index Futures Markets. *Asia-Pacific Finan Markets* (1998). 5:211–25. doi:10.1023/A:1010000606092

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.

ORIGINAL RESEARCH
published: 26 May 2021
doi: 10.3389/fams.2021.601105

Sentiment-Guided Adversarial Learning for Stock Price Prediction

*Yiwei Zhang[1], Jinyang Li[1], Haoran Wang[1] and Sou-Cheng T. Choi[2,3]**

[1]Beijing Institute of Technology, Haidian District, Beijing, China, [2]Kamakura Corporation, Honolulu, HI, United States, [3]Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, United States

Prediction of stock prices or trends have attracted financial researchers' attention for many years. Recently, machine learning models such as neural networks have significantly contributed to this research problem. These methods often enable researchers to take stock-related factors such as sentiment information into consideration, improving prediction accuracies. At present, Long Short-Term Memory (LSTM) networks is one of the best techniques known to learn knowledge from time-series data and to predict future tendencies. The inception of generative adversarial networks (GANs) also provides researchers with diversified and powerful methods to explore the stock prediction problem. A GAN network consists of two sub-networks known as generator and discriminator, which work together to minimize maximum loss on both actual and simulated data. In this paper, we developed a sentiment-guided adversarial learning and predictive models of stock prices, adopting a popular variation of GAN called conditional GAN (CGAN). We adopted an LSTM network in the generator and a multilayer perceptron (MLP) network in the discriminator. After extensively pre-processing historical stock price datasets, we analyzed the sentiment information from daily tweets and computed sentiment scores as an additional model feature. Our experiments demonstrated that the average forecast accuracies of the CGAN models were improved with sentiment data. Moreover, our GAN and CGAN models outperformed LSTM and other traditional methods on 11 out of 36 processed stock price datasets, potentially playing a part in ensemble methods.

Keywords: stock prediction, data processing, labels, regression, long short-term memory, sentiment variable, conditional generative adversarial net, adversarial learning

OPEN ACCESS

Edited by:
*Qingtang Jiang,
University of Missouri–St. Louis,
United States*

Reviewed by:
*Qi Ye,
South China Normal University, China
Qiang Wu,
Middle Tennessee State University,
United States*

****Correspondence:***
*Sou-Cheng T. Choi
schoi32@iit.edu*

Specialty section:
*This article was submitted to
Mathematics of Computation
and Data Science,
a section of the journal
Frontiers in Applied Mathematics and
Statistics*

Received: *31 August 2020*
Accepted: *01 February 2021*
Published: *26 May 2021*

Citation:
*Zhang Y, Li J, Wang H and Choi S-CT
(2021) Sentiment-Guided Adversarial
Learning for Stock Price Prediction.
Front. Appl. Math. Stat. 7:601105.
doi: 10.3389/fams.2021.601105*

1 INTRODUCTION

Stock prediction has been attached with great importance in the financial world. While stock fluctuation is very unpredictable, researchers have made every effort to simulate the stock variation because a relatively reasonable prediction can create massive profits and help reduce risks. Stock prediction researchers often consider two kinds of solution methods: classification and regression. Classification methods predict stock movement, while regression methods predict stock prices. Stock movement prediction can be seen as a simple classification task since it only predicts whether the stock will be up or down by a certain amount, or remain almost unchanged. However, many researchers focus their efforts on stock price prediction–a regression task that forecasts future prices with past values – since it could yield more profits than simple movement prediction. In a regression task, researchers have to tackle very complex situations to forecast the future price accurately. Like other time-series prediction problems, many factors should also be taken into consideration besides historical prices. Investors' sentiments, economic conditions, and public opinions are all critical variables that should be added into a stock prediction system. Particularly, sentiment analysis has been identified as a useful tool in recent years. Not just price and price-related values, researchers have paid more attention to the extraction of sentiment information from newspapers, magazines, and even social media. For example, Ref. [1] examined the relation between social media sentiment and stock prices, in addition to economic indicators in their research on stock market. Ref. [2] Focused on combining stock-related events with the sentiment information from reviews of financial critics. They analyzed sentiments and predicted the fluctuation of the stock market with a matrix and tensor factorization framework. Besides mathematical methods, the development of natural language processing also provides additional techniques for analyzing text and sentiment information. Machine learning, data mining, and semantic comprehension have made extracting large amounts of stock sentiments possible. With the development of social media, people are increasingly inclined to exchange information through the Internet platform. Real-time stock reviews contain a wealth of financial information that reflects the emotional changes of investors. Works such as [3, 4]; and [5] analyzed sentiment information from large numbers of real-time tweets, which were related to stocks and corresponding companies, then investigated the correlation between stock movements and public emotions with supervised machine learning principles. In particular, Ref. [3] utilized two mood tracking tools, OpinionFinder and Google-Profile of Mood States (GPOMS), to analyze the text content of daily tweets, which succeeded in measuring varying degrees of mood. These research results proved that modern techniques are mature enough to handle mountains of sentiment data and that Twitter is a valuable text resource for sentiment analysis.

In our work, we used VADER (Valence Aware Dictionary and sEntiment Reasoner) [6] as a mood tracking tool to help us analyze the sentiment information from tweets. We extracted sentiment features from the past several days of tweets and input them into stock prediction models to predict future stock prices. Applying sentiment analysis to nine stocks, we trained the models with two-month-long training sets with tweets and tested the model performance with the final five days' data. We applied several linear and nonlinear models such as LSTM to this regression task. Moreover, referring to the theories of recurrent neural networks and generative adversarial networks, we designed a sentiment-guided model to improve the accuracy of stock prediction further.

In the remainder of this article, the organization is set in the following order: First, in **Section 2** we review existing stock prediction methods and the development of GANs. Then, in **Section 3**, we introduce our methods in data collection and processing. In **Sections 4** and **5**, we propose our sentiment-guided adversarial learning model as well as comparisons between our models and baselines. Finally, conclusions and future work are discussed at the end of this paper in **Section 6**.

2 RELATED WORKS

In this section, we first present an overview of the relatively recent generative adversarial networks, followed by a brief review of some traditional methods for stock price prediction.

2.1 Adversarial Learning of Neural Networks

Generative adversarial networks (GANs) [7], which try to fool a classification model in an adversarial minimax game, have shown high potential in obtaining more robust results compared to traditional neural networks [8]. In a GAN framework, a generator produces fake data based on noisy samples and attempts to minimize the difference between real and fake distribution, which is maximized by a discriminator oppositely. The GAN framework and GAN-based research have attracted huge attention in various fields recently. Existing GAN works mainly focus on computer vision tasks like image classification [9] and natural language processing like text analysis [10]. With the inception and extensive applications of knowledge representation of natural or complex data such as languages and multi-media, the GAN framework is also widely applied to classification missions for data in representation spaces (e.g., vector spaces of matrices) rather than just for the original data in feature spaces. Some researchers also extended applications of GAN to more challenging problems such as recommendation systems [11] or social network alignment problems [12].

Conditional GANs (CGANs) [13] add auxiliary information to input data, guiding the training process to acquire expected results. Additional information (like class labels or extra data) is fed into both generator and discriminator to perform the conditioning. This architecture is mostly used in image generation [14] and translation [15]. However, we noticed that the CGAN framework has rarely been applied to time-series prediction, specifically, stock prediction problem. In our work, we added sentiment labels to guide the training process of our model and achieved a stronger performance.

Previous work [16] has shown the capacity of GANs for generating sequential data and to fulfill the adversarial training with discrete tokens (e.g., words). Recurrent neural networks (RNNs) have also been widely used to solve problems based on sequential data, such as speech recognition and image generation. Ref. [17] first combined RNNs with a GAN framework and successfully produced many kinds of continuous sequential data, including polyphonic music. Inspired by this work, researchers paid more attention to the potential of RNNs in adversarial training. Refs. [18, 19] succeeded in producing realistic real-valued multi-dimensional time-series data and concentrated their work on medical applications. They built models capable of synthesizing realistic medical data and developed new approaches to create predictive systems in the medical domain. Ref. [20] utilized prerecorded human motion data to train their models and applied their neural network in random motion synthesis, online or offline motion control, and motion filtering. Our work drew on the idea of the RNN-GAN framework and applied time-series analysis to the financial problem of stock prediction. Although similar application in stock prediction has been raised recently [21], the main difference was that our work also utilized text data from Twitter to create sentiment scores, which could gauge the market mood and guide our models to achieve more accurate prediction.

2.2 Stock Prediction Methods

The stock market prediction problem has a long history and is regarded as an essential issue in the field of financial mathematics. According to the development of research over the years, we can divide the prediction techniques into two categories: statistical and machine learning methods. Frequently used statistical techniques include the autoregressive method (AR), the moving average model (MA), the autoregressive moving average model (ARMA), and the autoregressive integrated moving average (ARIMA) [22]. These methods adopt the principles of random processes and essentially rely on past values of the sequences to analyze and predict the future data. Another technique is generalized autoregressive conditional heteroskedasticity (GARCH) [22], which takes the fluctuation of variance into account and nicely simulates the variation of financial variables. However, both kinds of techniques depend on strong prerequisites such as stationarity and independent and identically distributed (iid) random variables, which may not be satisfied by data in the real-world financial markets.

Recently, machine learning methods (including linear regression, random forest, and support vector machine) stimulate the interest of financial researchers and are applied to forecasting problems. Among them, support vector machine (SVM) and its application in the financial market revealed superior properties compared to other individual classification methods [23]. Since the emergence of deep learning networks, researchers have proved that neural networks can obtain better performance than linear models. Their capacity to extract features from enormous amount of raw data from diverse sources without prior knowledge of predictors makes deep learning a preferred technique for stock market prediction. Artificial neural networks and other advanced neural models like convolution neural networks (CNNs) are evidenced to be good at capturing the non-linear hidden relationships of stock prices without any statistical or econometric assumption [24]. Furthermore, neural networks have also been found to be more efficient in solving non-linear financial problems compared to traditional statistical methods like ARIMA [25]. Nowadays, RNNs are one of the most popular tools in time-series prediction problems. Notably, the LSTM network has been a great success due to its ability to retain recent samples and forget earlier ones [26]. Each LSTM unit has three different gates: forget gate, update gate, and output gate. LSTM units can change their states by controlling their inner operant, namely, their three gates. In our model, we utilized the LSTM network to extract features according to the timeline and to generate fake stock prices from real past price values. We also used sentiment measures to enhance the robustness of our prediction models and make the generative results approach the real distribution.

3 DATA AND METHODS

In this section, we document our data sources and preprocessing techniques. The latter part includes feature engineering, imputation of missing data, fast Fourier transform for denoising data, and last but not least, Isolation Forest for anomaly or outlier detection.

3.1 Data Collection and Sentiment Analysis

Before we started our work, we had the conviction that a collection of tweets related to the stocks could make comparatively accurate models of the investors' mood and thus reach better predictions of the stock market prices. Generally speaking, tweets have neutral, positive, or negative emotions; and we focused on those that contains one or several cashtags (unique identifiers for businesses), which could influence the stock's trend in the following day. If negative sentiment dominated a day, then the next day's stock prices would be expected to fall. The number of followers on one's Twitter account would also be a significant factor. The more followers of an account, the higher the influence of tweets from the account, and the more significant their impact would likely have on stock prices. Cashtags system is a particularly convenient feature of Twitter, allowing users to see what everyone is saying about public companies. The way this system works is similar to the well-known #hashtags of Twitter, except that a cashtag requires "$" followed by a stock symbol (e.g., $GOOG for Google, LLC; $FB for Facebook, Inc.; and $AAPL for Apple Inc.).

For our work, financial tweets and their retweets were downloaded from the website, https://data.world/kike/nasdaq-100-tweets. The time span of these tweets is 71 days from Apr 1–June 10, 2016. We could get all tweets mentioning any NASDAQ 100 companies from this source. The key daily stock price data includes Open prices (O), High prices (H), Low prices (L), and Close prices (C); they were subsequently crawled from Yahoo Finance with the Python package, pandas_datareader. We produced nine datasets, each one including tweets and price values for the nine companies as listed in **Table 1**. These companies are from different sectors/industry groups by the Global Industry Classification Standard (GICS). They could be classified into three price-trend categories: descending prices, mildly fluctuating prices, and ascending prices. **Figure 1** shows the Open price curves of nine stocks with the three distinct trends.

To analyze the sentiment of each tweet, we used VADER [6], which is available from vader-sentiment, a ready-made Python machine learning package for natural language processing. VADER is able to assign sentiment scores to various words and symbols (punctuations), ranging from extremely negative (−1) to extremely positive (+1), with neutral as 0. In this way, VADER could get an overall sentiment score for a whole sentence by combining different tokens' scores and analyzing the grammar frames. We assumed that the neutral emotion plays a much weaker role in the overall market's mood since neural sentiment tends to regard the stock market as unchanged in a specific period. Thus, we excluded the neutral sentiment scores in our analysis and only took the negative and positive moods into consideration. VADER places emphasis on the recognition of

TABLE 1 | Stock prices of nine companies selected for experiments in this study.

Company	Symbol	Sector	Industry group
Apple Inc	AAPL	Information Technology	Technology Hardware & Equipment
Cisco Systems Inc	CSCO	Information Technology	Communications Equipment
Electronic Arts Inc	EA	Communication Service	Media & Entertainment
Eastbay Inc	EBAY	Consumer Discretionary	Internet & Direct Marketing Retail
Endo International PLC	ENDP	Health Care	Pharmaceuticals
Kandy Hotels Co	KHC	Consumer Discretionary	Hotels Resorts & Cruise Lines
NVIDIA Corporation	NVDA	Information Technology	Semiconductors
Starbucks	SBUX	Consumer Discretionary	Consumer Services
Skyworks Solutions Inc.	SWKS	Information Technology	Semiconductors

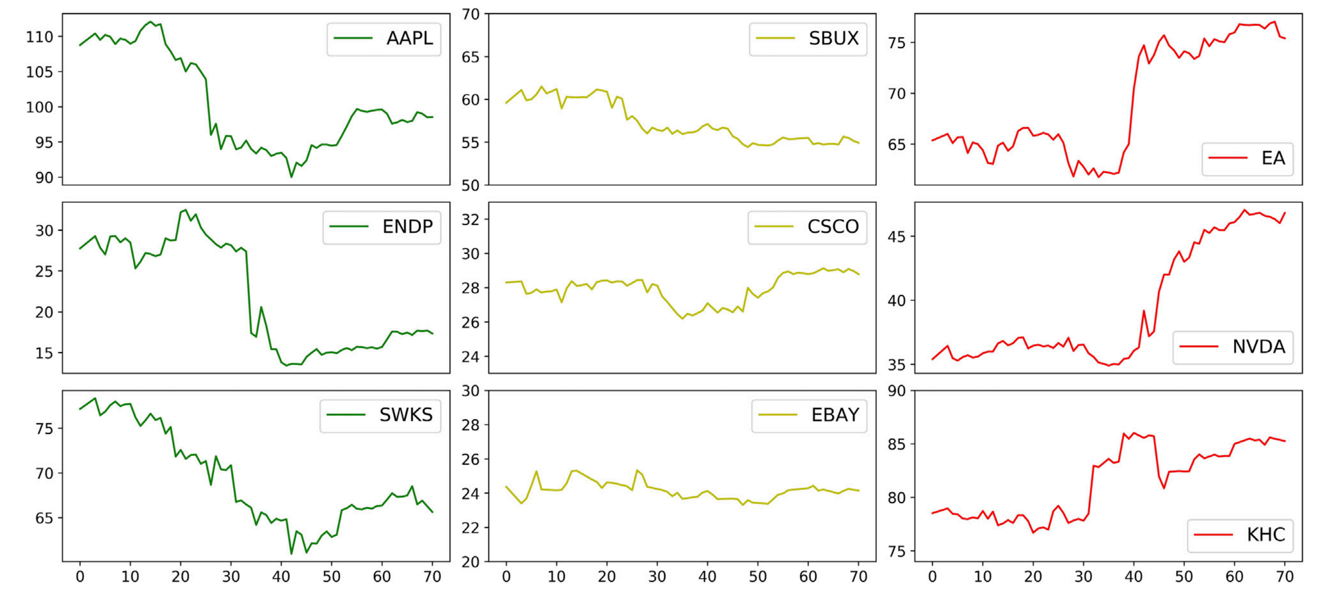


FIGURE 1 | The nine stocks classified into three classes with distinct price trends. The variations of Open price from April 1 to June 10, 2016 are displayed. The first column presents the stocks with descending prices; the second represents the stocks with mildly fluctuating prices; and the third represents the stocks with ascending prices.

uppercase letters, slang, exclamation marks, and the most common emojis. Tweet contents are not written academically or formally, so VADER is suitable for social media analysis. However, we needed to add some new words to the original dictionary of VADER, because VADER missed or misestimated some important words in financial world and therefore caused inaccuracy. For example, “bully” and “bullish” are negative words in the VADER lexicon, but they are positive words in the financial market. We updated the VADER lexicon with the financial dictionary Loughran-McDonald Financial Sentiment Word Lists [27], which include many words used in the stock market. This dictionary has seven categories, and we adopted the “negative” and “positive” lists. We further deleted about 400 existing words in VADER that overlap the Loughran-McDonald Financial Sentiment Word Lists. Finally, we added new negative and positive words to the VADER dictionary and attached sentiment scores to them, with +1.5 to each positive word and −1.5 to each negative word from the Loughran-McDonald lists.

To get one day’s overall sentiment score for one stock, we first analyzed all related tweets with VADER and gained the scores of each tweet. Considering that the more followers the bigger influence, we further regarded the number of followers as weights and calculated the weighted average of sentiment scores. The daily percentage change of this average was taken as a comprehensive factor, called compound_multiplied, for 1 day. One problem of our data was that we have only tweets’ data but no price data on the non-trading days. To make full use of the tweets’ data, we filled the gaps up with the price values from past trading days. We utilized moving averages to fill in the missing values.

We also normalized the compound_multiplied variable as neural networks generally perform better and more efficiently on scaled data. **Figure 2** shows the box plots of compound_multiplied and per_change (see Section 5.1.2 for details) for the nine stocks before the data was scaled.

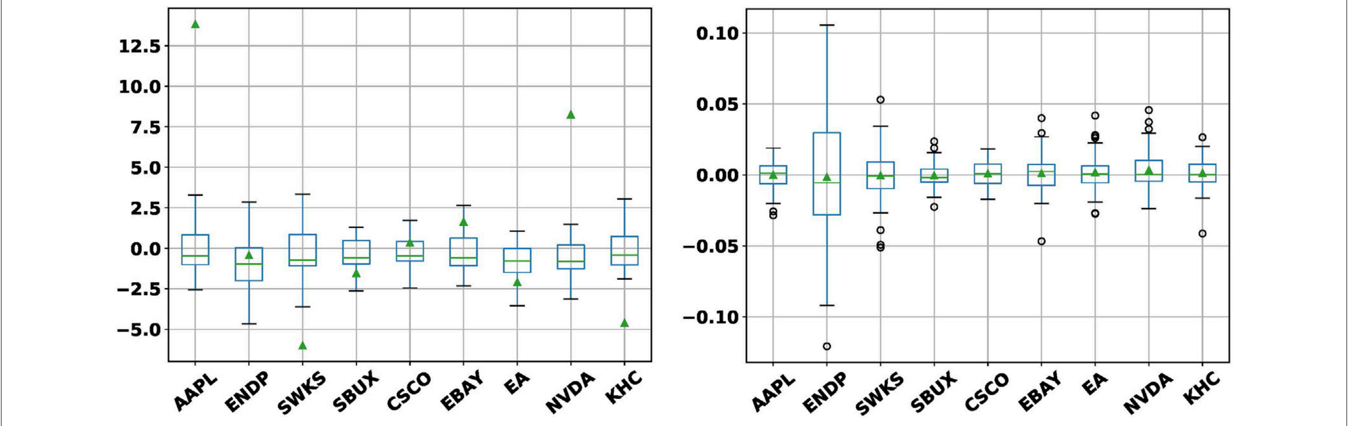


FIGURE 2 | Left: Box plots of changes in daily sentiment scores aggregated from tweets that contained cashtags of selected stocks over the time period Apr 2–Jun 10, 2016. Outliers are omitted to declutter the graph. The triangles mark the mean values. **Right**: Box plots of per_change with outliers shown as circles.

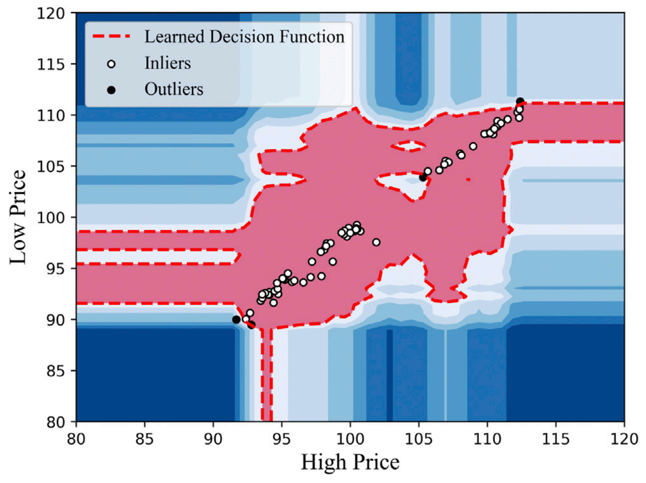


FIGURE 3 | Detecting outliers in the dimensions of High price and Low price with Isolation Forest on AAPL-Open in the training dataset. Only four points of outliers were flagged out by the algorithm.

3.2 Data Processing for Stock Prices

As discussed before, we collected stock prices from nine stock symbols, each of which contains four columns of time-series price data for each trading day: Open, High, Low, and Close. To change the sequential data into suitable input for our models, we did data cleaning and transformation for the training datasets, and approximate normalization for our both training and testing data. We present further details of these transformations below.

3.2.1 Data Cleaning

In consideration of the volatility of the stock market, we had to tackle the price data on days that showed abnormal trends or fluctuation. To accomplish the anomaly detection, we utilized the Isolation Forest (iForest) algorithm [28] to find such data points before treating them. Isolation Forest can directly describe the degree of isolation of data points based on binary trees without using other quantitative indicators. The main steps of distinguishing outliers were as follows: For a group of data with multiple dimensions, we attempted to build a binary search tree (BST). We first picked several dimensions, and then randomly selected one dimension and a value between the maximum and the minimum in that dimension as the root of the BST. Next, we divided the data into two groups (i.e., subtrees of the BST) according to the selected value. Likewise, we continued to subdivide the data according to another randomly selected value in another feature dimension—this step was repeated until the BST could not be further subdivided. Finally, we detected the anomaly data according to the path lengths of the nodes in the BST.

More specifically, in our work, we adopted only two dimensions in the iForest algorithm and detected the outliers within two steps. **(I)** We selected High price and Low price as the two dimensions to detect anomaly situations that stock prices fluctuated intensely in one day. **Figure 3** shows the anomaly detection results when we applied the iForest to the AAPL dataset. **(II)** For each one of the four price series (O, H, L, C), we respectively selected the differences $y_t - y_{t-1}$ and $y_{t+1} - y_t$, in which y_t was the price data on a particular date t, as the two input dimensions to iForest. In this way, we were able to detect anomaly local trends on the timeline. **Figure 4A** shows the detection result of 'Open Price' from the AAPL dataset, and **Figure 4B** displays the anomaly in the form of time series. After we identified each outlier y_t by considering its distances from $y_{t\pm1}$ using iForest, we replaced y_t with the rolling average $\frac{1}{3}(y_{t-2} + y_{t-1} + y_t)$. As a result, we prevented our model from being excessively influenced by outliers during the training process.

3.2.2 Data Transformation

We noticed the successful applications of RNNs in wave forms like sinusoids, so we further transformed our time-series datasets into a wave-like form in order to improve the training effect. In our prediction problem, we assumed that the changes in stock data were periodic, which gave us the opportunity to introduce Fourier Transform into our data processing work. Fourier transform can decompose a periodic function into a linear combination of

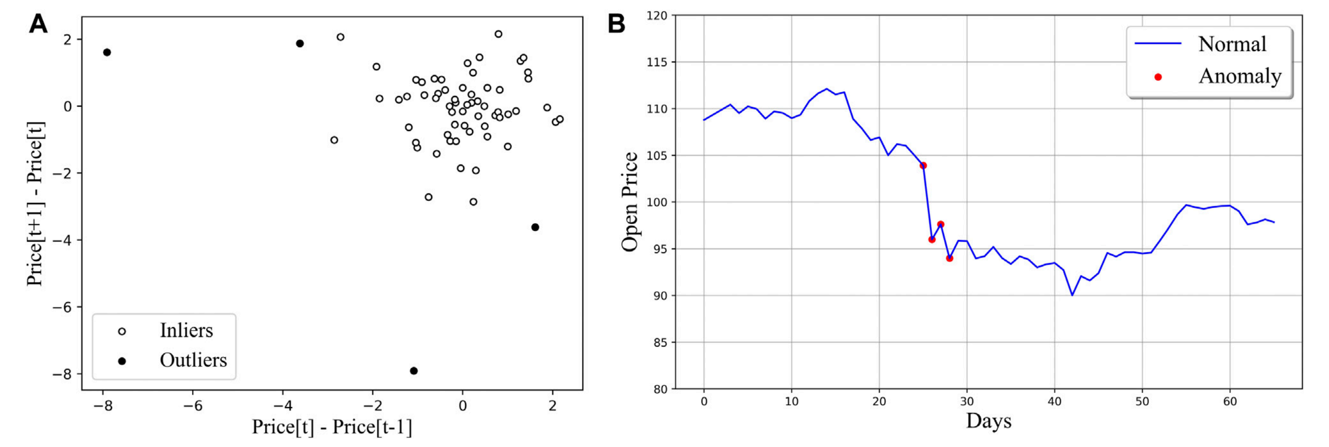


FIGURE 4 | (A) Detecting outliers according to $(y_t - y_{t-1})$ and $(y_{t+1} - y_t)$ with Isolation Forest using the AAPL-Open training set as an illustration. **(B)** Anomaly detection results in the form of time series on the same price values in **(A)**.

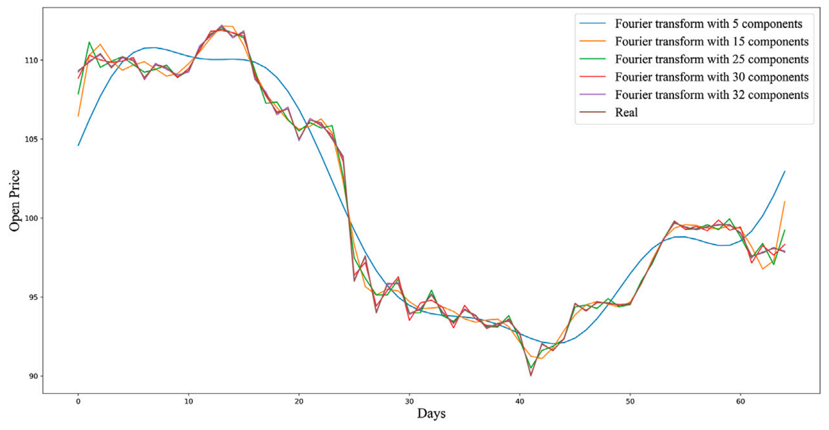


FIGURE 5 | Fourier transform curves with 5, 15, 25, 30, and 32 components on AAPL-Open training set.

orthogonal functions (such as sinusoidal and cosine functions) [29]. With Fast Fourier Transform (FFT), we could extract both global and local trends of our stock datasets and thus reduce their noise. For each of the four prices (O, H, L, C of a stock dataset), we respectively used FFT to create a series of sinusoidal waves (with different amplitudes and frames) and then combined these sinusoidal waves to approximate the original curve. In **Figure 5**, we can see that the pink curve, which was made by a combination of 32 components, closely approximates the original price graph of AAPL-Open training set. By using the derived curve, namely, the denoised sequential data, we could enhance the smoothness of our training set, and reduce noisiness in our data. Consequently, we smoothened the time-series data to make our training data more predictable.

3.2.3 Approximate Normalization

To make sure that our models could learn the features of time-series variables, we also modified the sequential price data to make it satisfy (roughly) normal distribution. In financial research, there is a traditional assumption that the simple daily returns of time-series data are iid normal samples, as long as the sequential data has constant mean and same variance over time; see [30]; for example. We calculated R_t, defined as one-period simple returns of the price values before inputting them into our supervised learning models as targets, where y_t represents one of the four prices for a given stock on day t that we are interested to predict:

$$R_t = \frac{y_t - y_{t-1}}{y_{t-1}}. \tag{1}$$

Let $\widehat{R}_t$ be an estimate of R_t returned by a machine learning model trained with predictors including historical returns, $R_{t-1}, R_{t-2}, \cdots, R_s$ for some $s < t$. Subsequently, we can obtain an estimate of the stock price, $\widehat{y}_t = y_{t-1}(1 + \widehat{R}_t)$.

4 MODEL THEORY

In this section, we present the architecture of our CGAN framework in relation to the sub-networks of our choice.

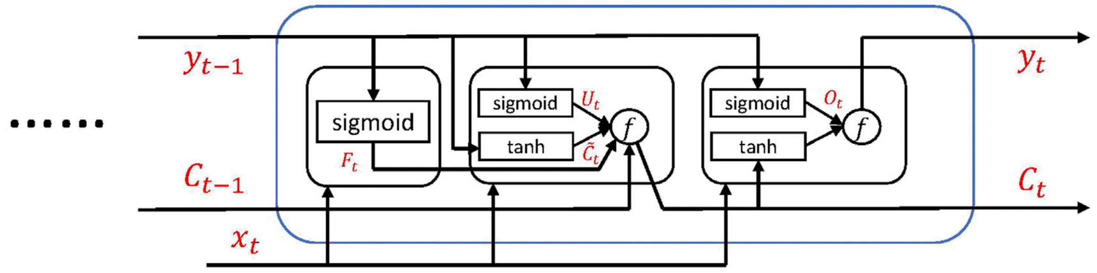


FIGURE 6 | Information flow through an LSTM unit.

4.1 Long Short-Term Memory Networks

Introduced by [31], LSTM is a special kind of RNN, which can learn both short-term and long-term correlations effectively from sequential data. An LSTM network usually consists of many repeating modules, the LSTM units, and we can tune the number of the units to improve the performance of the network. These LSTM units concatenate with each other in line with the information transmitting from one to another. Each unit contains three kinds of gates: Forget gate (decides what kind of old information to be discarded), Update gate (decides what kind of new information to be added), and Output gate (decides what to be output). **Figure 6** shows how the past information flows through an LSTM unit between two time steps, and how the LSTM unit transforms the data in both its long-term and short-term memory. The information transmission process is from C_{t-1} to C_t, which will be changed by inner operants. C_t is a latent variable; χ_t is the input; and y_t is the output. The mathematical operations in the three gates are defined as follows, where $W_.$ and $b_.$ are parameters to be estimated for minimizing some loss functions:

I. Forget gate:

$$F_t = \text{sigmoid}\left(W_f\left[y_{t-1}, \chi_t\right] + b_f\right). \tag{2}$$

II. Update gate:

$$\begin{aligned} U_t &= \text{sigmoid}\left(W_u\left[y_{t-1}, \chi_t\right] + b_u\right),\\ \tilde{C}_t &= \tanh\left(W_C\left[y_{t-1}, \chi_t\right] + b_C\right),\\ C_t &= F_t{}^*C_{t-1} + U_t{}^*\tilde{C}_t. \end{aligned} \tag{3}$$

III. Output gate:

$$\begin{aligned} O_t &= \text{sigmoid}\left(W_o\left[y_{t-1}, \chi_t\right] + b_o\right),\\ y_t &= O_t{}^*\tanh\left(C_t\right). \end{aligned} \tag{4}$$

Note that sigmoid and tanh are activation functions applied to an input vector elemenwise, whereas * represents elementwise multiplication of vectors. These three gates cooperate with each other and together determine the final information that is output from an individual unit. In this work, we took advantage of the memory property of LSTM and improved its accuracy with adversarial learning framework.

Nowadays, LSTM has been widely applied in many research fields such as machine translation, language modeling, and image generation.

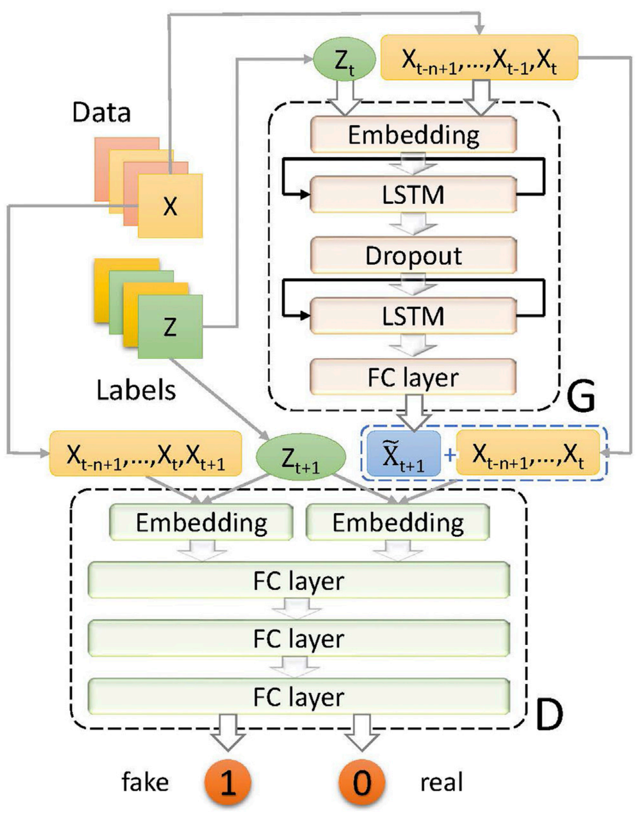


FIGURE 7 | The architecture of our conditional adversarial learning model.

4.2 Adversarial Learning Model

As the conditional GAN framework [13] has proved to be a great success, we adopted this idea in our GAN-based model to improve the training effect. **Figure 7** illustrates the architecture of our GAN-based model. Let $\mathbf{X}_t = \{X_{t-n+1}, \ldots, X_{t-1}, X_t\}$ represent the input data in the form of time series, in which each term is the historical data from the corresponding day from $n-1$ days ago to end of current day t. Note that X_t is a vector that includes all the daily factors on day t. In addition, let Z_t be the sentiment label on day t. Our generator is a multiple-input single-output system, which inputs n historical days' data with m factors from the stock market each day (plus,

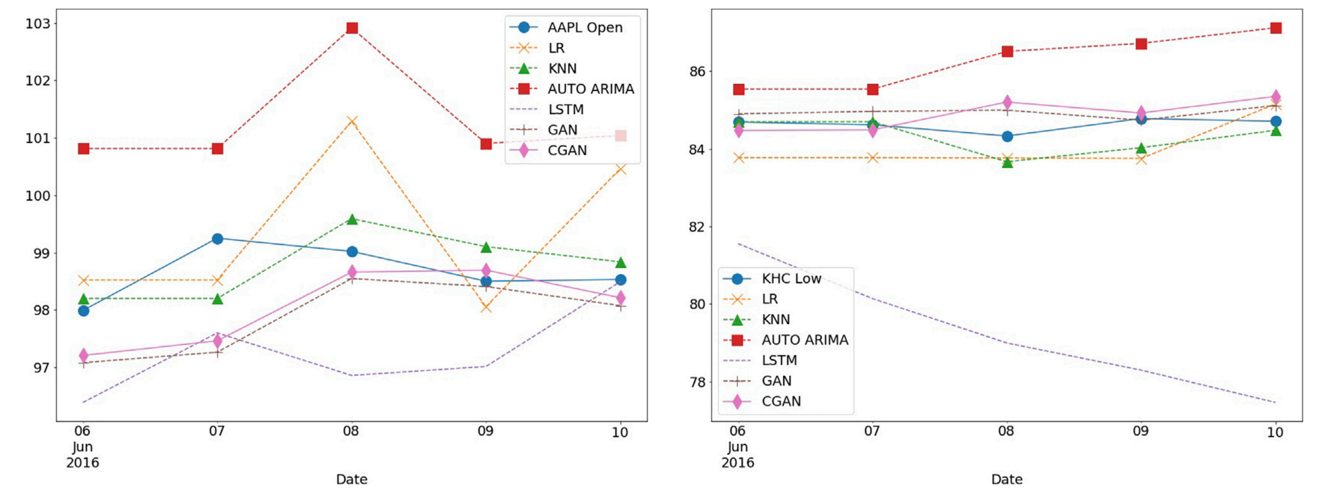


FIGURE 8 | Comparison of the prediction curves of different models on the AAPL-Open and KHC Low test sets. For AAPL Open, among all models, KNN achieved the minimum MAPE over the five-day test period; see **Table 3**. However, on each of the last three days, CGAN actually performed better than KNN. For KHC Low, GAN predictions were closest to the market values and CGAN came next.

the label) and outputs a specific kind of future price data (Open, High, Low, or Close) as the prediction. Since the stock data is typical time-series data, we adopted an LSTM network in our generative model denoted by G. LSTM was trained to extract the features of price data from the past n days, and then used to predict stock prices on day $t + 1$. The generator model G generally consists of three kinds of layers: embedding layer, LSTM layer, and fully connected (FC) layer. An embedding layer first embeds a label in the latent space and turns it into a dense vector, then combines the latent vector with the input matrix, which is the price data from the past in our application. Afterward, an LSTM layer generates some prediction result. A dropout layer could be added between two LSTM layers to make the network have a better generalization and less likely to overfit. Lastly, an FC layer with the Leaky Rectified Linear Unit (ReLU) activation function outputs simulated or fake data $\tilde{X}_{t+1}$, which approximates the real target data X_{t+1}. The output of our generator is defined as follows:

$$\tilde{X}_{t+1} = G(\mathbf{X}_t \mid Z_t). \tag{5}$$

In our work, $m = 4$ and $n = 5$. The four factors were historical Open price, High price, Low price and Close price, which could be seen as four features of the stock data for the price prediction problem. Specifically, we utilized all four kinds of stock prices in the past five business days to forecast each kind of price (O, H, L, or C) on the next day.

The purpose of our discriminator D is to classify input data as real or fake. This classification model is expected to output 0 when it receives real input data or 1 when it receives fake input data. For the input, we concatenated $\mathbf{X}_t$ with $\tilde{X}_{t+1}$ and X_{t+1} respectively to get the fake data $\tilde{\mathbf{X}}_{t+1} = \{X_{t-n+1}, \ldots, X_t, \tilde{X}_{t+1}\}$ and the real data $\mathbf{X}_{t+1} = \{X_{t-n+1}, \ldots, X_t, X_{t+1}\}$. In this way, we could make the discriminator learn the features of sequential data better. Our discriminator D consists of two parts: two embedding layers and a multilayer perceptron (MLP) network. The embedding layers worked in the way same as those in G, i.e., the embedding layers transformed the data and labels into suitable input for MLP. The MLP model then projected the data into higher dimensional space and effectively accomplished the classification task. The Leaky ReLU activation function was used in the hidden layers of the MLP and the sigmoid function was used in the output layer. Returning either 0 and class 1, the output of a discriminator while making a correct decision is as follows:

$$0 = D(\mathbf{X}_{t+1}|Z_t), \tag{6a}$$

$$1 = D\left(\tilde{\mathbf{X}}_{t+1}|Z_t\right). \tag{6b}$$

Specifically, if we select Open price to be predicted by G, we would concatenate the prediction result to Open price values from the past n days. Then, we distinguished this sequence from the corresponding one sampled from real data with D.

We alternatively trained G and D with binary cross-entropy loss (i.e., log loss), $L(\tilde{y}, y) = -y\log(\tilde{y}) - (1-y)\log(1-\tilde{y})$, where y is the target value and $\tilde{y}$ is a predicted value. On the one hand, our generator G simulated the fluctuations of the stock market and to minimize the difference between its prediction and the real data. On the other hand, our discriminator D tried to distinguish and maximize that difference. The training of G focused on making the discriminator D confused; it attempted to minimize the adversarial loss so that D could not discriminate the prediction easily. The adversarial loss at each data point on day $t + 1$ for the generator was

$$G_{loss}\left(\tilde{\mathbf{X}}_{t+1}\right) = L\left(D\left(\tilde{\mathbf{X}}_{t+1}|Z_t\right), 0\right). \tag{7}$$

The training of D focused on improving its ability to distinguish the difference between $\mathbf{X}_{t+1}$ and $\tilde{\mathbf{X}}_{t+1}$, so it attempted to maximize the adversarial loss at each data point on day $t + 1$:

$$D_{\text{loss}}\left(\mathbf{X}_{t+1}, \tilde{\mathbf{X}}_{t+1}\right) = L(D(\mathbf{X}_{t+1}|Z_t), 0) + L\left(D\left(\tilde{\mathbf{X}}_{t+1}|Z_t\right), 1\right). \tag{8}$$

In general, the generator tried to make $D(\tilde{\mathbf{X}}_{t+1}|Z_t) = 0$ while the discriminator tried to achieve $D(\mathbf{X}_{t+1}|Z_t) = 0$ and $D(\tilde{\mathbf{X}}_{t+1}|Z_t) = 1$. [7] defined the loss function of this specific binary classification task as cross-entropy loss. To achieve the training process described by **Eq. (7)** and **Eq. (8)**, we updated D by maximizing $E_{x\sim p_{\text{real}}(x)}[\log D(x|z)] + E_{\tilde{x}\sim p_{\text{fake}}(\tilde{x})}[\log(1 - D(G(\tilde{x}|z)))]$, and then updated G by minimizing $E_{\tilde{x}\sim p_{\text{fake}}(\tilde{x})}[\log(1 - D(G(\tilde{x}|z)))]$. In this way, the total loss function of the minimax game over all training samples was as follows:

$$\min_G \max_D L_{\text{RMSprop}}(D, G) = E_{x\sim p_{\text{real}}(x)}[\log D(x|z)] + E_{\tilde{x}\sim p_{\text{fake}}(\tilde{x})}[\log(1 - D(G(\tilde{x}|z)))], \tag{9}$$

where $p_{\text{real}}(x)$ was the distribution of real data, $p_{\text{fake}}(\tilde{x})$ was the distribution of fake data, and lastly, z represented the label vector.

The two models G and D were trained iteratively with the RMSprop optimizer, a stochastic gradient descent algorithm with mini batches [32]. In each iteration, we first trained the discriminator r times and then trained the generator one time, in which r was seen as a hyper-parameter to be tuned in our training process. The reason for training D first was that a trained discriminator would help improve the training effect of the generator. Besides, we also adopted gradient clipping (ensuring the norm of a gradient not too large in magnitude) for both G and D at the end of each iteration to avoid the potential problem of gradient explosion.

If Z_t is omitted from the discussion in this subsection, then we have simply a GAN model. In practice, we used Keras [33] with TensorFlow [34] version 2 as the backend to implement our LSTM, GAN, and CGAN models.

5 EXPERIMENT

In this section, we summarize the numerical results of our learning and prediction models in multiple error measures in response to hyperparameter search when applicable.

5.1 Experimental Settings

We collected price data and related tweets of nine companies from April 1st to June 10th, 2016. The data before June 5th was taken as the training set and the last five days' data as the test set. For each dataset, we respectively experimented on Open price, High price, Low price, and Close price, so we totally built and tested $9 \times 4 = 36$ models and datasets (36 groups of experiments).

We have already illustrated the main data processing work in **Section 3**. Here, we elaborate on specific configuration of our models.

5.1.1 Baseline Models

Linear Multiple Regression (***LMR***) is a classical statistical approach for analyzing the relationship between one dependent variable and one or more independent variables. Its simple formula in matrix notation is $Y = X\beta + \varepsilon$, where ε is iid normal with mean $\mathbf{0}$ and variance matrix, $\sigma^2\mathbf{I}$, and $\mathbf{I}$ is the identity matrix of order the same as number of independent observations in Y.

K-Nearest Neighbors (***KNN***) [35] can be used in both classification and regression problems. In our time-series regression problem, the input is the k closest training sequences from the feature space according to the timeline, and the output is the average of the values of the k nearest neighbors.

Autoregressive Integrated Moving Average (***ARIMA***) is composed of autoregression (AR), an integrated (I) model that calculates differences, and moving average (MA). Auto ARIMA [36, 37] can automatically perform grid search with parallel processing to find an optimal combination of p, q, and d, which are the parameters associated with order of AR, degree of differencing with I, and order of MA, respectively.

Long Short-Term Memory is a special kind of recurrent neural network. We tuned three hyper-parameters to improve its training effect: the number of LSTM units, step size, and training epochs.

5.1.2 Data Pre-Processing Techniques

Besides Open price, High price, Low price, and Close price, we also added two more engineered factors to the dataset as input to predict the price on day $t + 1$. One was the sentiment variable, compound_multiplied, which was obtained with the method illustrated in **Section 3.1**. The other was per_change, which was obtained by the equation: $\text{per\_change} := 100\% \times [\text{Close}(t) - \text{Open}(t)]/\text{Open}(t)$. **Table 2** shows the AAPL dataset sample with six columns, where the compound_multiplied was the sentiment variable.

Transformations for baseline models (***LMR, KNN, and auto ARIMA***) When predicting one type of stock price mentioned above, we utilized the two derived factors, per_change and compound_multiplied, as predictors.

Techniques for neural-network models (***LSTM, GAN, and CGAN***) **(I)** As for LSTM, we chose only three factors as input. For instance, we used 'Open price', per_change and compound_multiplied from day t as input when we predicted the Open price on day $t + 1$. However, for our adversarial learning model, we chose four kinds of prices as input and utilized the other two factors to create labels. The details about the labels were discussed below. **(II)** We converted time-series data to data that was fitting in supervised learning problems. Specifically, if we predicted the price on day $t + 1$ with the prices from the past 5 days, we would create a six-term sequence $X_{t+1} = \{x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t, x_{t+1}\}$. In other words, we used the first five time-lagged columns as input and the last column as the real target price on day $t + 1$.

5.1.3 Settings for Adversarial Learning Models

Labels We utilized the two variables, compound_multiplied and per_change, to create our final sentiment label in the CGAN models. The reason for adding the per_change was that we would like to take the local trend of past stock variation into account. Finally, the label was set to three classes: 0, 1, and 2. The three classes respectively represent three different tendencies of forecasting prices: up, down, and almost unchanged. Our

TABLE 2 | Stock price data and two engineered features on the first five days of the AAPL dataset, where NaN represents Not a Number.

Date	Open	High	Low	Close	per_change	compound_multiplied
4/1/16	108.78	110.00	108.20	109.99	1.1123	NaN
4/2/16	109.33	110.73	108.89	110.37	0.9529	1.1750
4/3/16	109.87	111.46	109.58	110.74	0.7934	-0.1666
4/4/16	110.42	112.19	110.27	111.12	0.6339	-0.1836
4/5/16	109.51	110.73	109.42	109.81	0.2739	0.2064

model was trained to learn the categories from labels with the training set and enhance its prediction accuracy. Let Z_t represent the label on day t. The process of creating the labels could be captured by the following min-max scaling of each $v_t \in [0, 3]$ before mapping it to $Z_t \in \{0, 1, 2\}$, meaning $[0, 1) \mapsto 0$, $[1, 2) \mapsto 1$, and $[2, 3] \mapsto 2$:

$$v_t = \lambda_1 \times \text{compound\_multiplied}(t) + \lambda_2 \times \text{per\_change}(t), \quad (10)$$

$$Z_t = \min\left(2, \left\lfloor 3\frac{v_t - \min(v_t)}{\max(v_t) - \min(v_t)} \right\rfloor\right),$$

where $\lambda_1 + \lambda_2 = 1$ and $0 \le \lambda_2, \lambda_2 \le 1$. Empirically, tuning λ_1 and λ_2 could change the training accuracies.

We also did experiments to evaluate the impact of the sentiment analysis had made in our work. Specially, we modified our CGAN models by taking out the embedding layers and not inputting the sentiment labels Z_t. Without the labels Z_t, we have GAN models. In this way, we could compare the training effect of our models with or without sentiment labels.

Hyper-parameters As discussed in Section 4, we alternately trained the D and the G. The epoch ratio, r, defined as epochs of D to epochs of G, was in the range of $[1, 4]$.

The learning rates for D and G, α_D and α_G, were searched in the range, $[5 \times 10^{-5}, 8 \times 10^{-4}]$, respectively, with the same decay, 10^{-8}. We also tuned α_D and α_G in our experiments to improve the training process. Besides, the gradient clipping thresholds were set to be the same at 0.8 for both the generator and the discriminator.

5.2 Results and Discussions

We used mean absolute percentage errors (MAPE) as the main metric to evaluate the performance of our model. The metric equation is

$$MAPE = \frac{100\%}{N} \sum_{t=1}^{N} \left| \frac{y_t - \tilde{y}_t}{y_t} \right|,$$

where y_t is the real price on day t, $\tilde{y}_t$ is the prediction price by a model, and N is number of data points. In our experiments, we first trained the CGAN model until the loss curves of both the generator and the discriminator converged. The training process was described in **Section 4**. After that, we tested the generator "LSTM model" to get the prediction results of testing data. **Figure 8** displays the comparison of prediction curves on the AAPL-Open and KHC-Low test sets. As our test sets contained five-day-long data, we calculated the average MAPEs for the five days' OHLC prices as the performance metric of the model.

Empirically, we tuned three kinds of hyper-parameters to improve the performance of our adversarial learning model: the learning rates of the sub-networks, α_D and α_G; the epoch ratio r; weightage λ_1 and λ_2 in **Eq. (10)**. **Table 3** show the MAPE results of the nine test sets. We did four groups of experiments, respectively on O, H, L, C, for each test set and then calculated the average MAPE to compare the performances of different models effectively. KNN achieved minimum mean MAPE for AAPL, SBUX, CSCO, SWKS, and KHC; linear models for EA and ENDP; GAN and CGAN for NVDA and EBAY, respectively. While KNN achieved minimum errors most of the time, GAN and CGAN were the best for 11 of the 36 stock prices and the second best for 18. In terms of overall average of all nine MAPEs (last column of **Table 4**), KNN was the best and CGAN came as a close second. Without sentiment labels, GAN models on average had a higher average of all MAPEs than CGAN, showing that our sentiment labels help generally in improving price prediction accuracy.

Root mean square errors (RMSE), mean square errors (MSE), mean absolute errors (MAE), and symmetric mean absolute percentage errors (SMAPE) were also used to verify the training results. Their defining equations are recapped here:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \tilde{y}_t)^2}, \qquad MSE = \frac{1}{N}\sum_{t=1}^{N}(y_t - \tilde{y}_t)^2,$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|y_t - \tilde{y}_t|, \qquad SMAPE = \frac{100\%}{N}\sum_{t=1}^{N}\frac{|y_t - \tilde{y}_t|}{(|y_t| + |\tilde{y}_t|)/2}.$$

To evaluate our model in a more comprehensive view, we selected AAPL and EBAY to illustrate these four metrics. We respectively obtained the results of O, H, L, C from all models and then calculate the average errors to compare the performance more carefully. **Table 5** for AAPL shows that KNN performed best on average while CGAN came second. **Table 5** for EBAY demonstrates that the CGAN models outperformed all the baselines not only in MAPE but also in RMSE, MSE, MAE, and SMAPE.

As discussed above, we treated λ_1 and λ_2 as hyper-parameters. Considering the importance of condition labels, adjusting the proportion of sentiment information would be necessary every time we trained the CGAN model. For short-term stock data, we need to train the model only once, since people's attitude toward a stock often remains more or less the same in a short period. However, readjusting λ_1 and λ_2 every month maybe useful. We collected another tweet dataset about Apple Inc. with a time span from Jan. 1, 2014 to Dec. 31, 2015 (available from stocknet-dataset by [5]. We selected three periods in these two-year-long datasets and calculated the MAPEs of prediction

TABLE 3 | Tables 3a–i present the MAPEs of forecasted O, H, L, and C prices from various models for nine stock symbols. The smallest MAPE is highlighted in bold across the models in each column for every stock price; the second smallest is underlined. The last column contains the average values of the MAPEs of the four price targets.

	Open	High	Low	Close	Mean/%
a). MAPEs of the AAPL test set.					
LMR	1.1958	2.9199	1.4223	2.1880	1.9315
KNN	**0.5546**	**0.8539**	**0.4884**	**0.4090**	**0.5765**
ARIMA	2.6741	1.4260	3.2027	2.6597	2.4906
LSTM	1.4061	4.5924	5.7139	5.2507	4.2408
GAN	0.7929	1.5085	1.2227	1.2868	1.2027
CGAN	0.7067	1.4688	0.6841	1.3076	1.0418
b). MAPEs of the SBUX test set.					
LMR	0.9019	1.1674	1.3397	1.2923	1.1753
KNN	0.5593	**0.4919**	0.6338	**0.5265**	**0.5529**
ARIMA	1.1861	1.0757	1.1539	1.0580	1.1184
LSTM	**0.5040**	3.8398	4.0186	3.7970	3.0399
GAN	0.6448	0.8664	0.8240	0.9556	0.8227
CGAN	0.6888	0.9053	**0.6280**	0.8969	0.7798
c). MAPEs of the EA test set.					
LMR	**0.5037**	0.8308	**0.9628**	0.9709	**0.8170**
KNN	0.5964	0.8414	0.9878	1.0162	0.8604
ARIMA	3.1441	3.8026	1.4712	2.4455	2.7158
LSTM	0.8987	16.9632	17.0647	16.4948	12.8554
GAN	0.9095	**0.8225**	1.0935	**0.7298**	0.8888
CGAN	1.2390	1.0105	1.3181	0.7448	1.0781
d). MAPEs of the ENDP test set.					
LMR	2.6631	2.892	**3.2137**	**2.8585**	**2.9068**
KNN	2.4014	**2.4446**	4.3038	3.4666	3.1541
ARIMA	7.5329	10.8514	7.8179	12.4049	9.6518
LSTM	**1.9243**	33.3106	27.6254	37.3947	25.0638
GAN	3.4240	53.2661	3.9620	4.8264	16.3696
CGAN	5.4597	3.6090	4.0107	3.1991	4.0696
e). MAPEs of the CSCO test set.					
LMR	1.1367	0.6497	0.8273	0.4343	0.7620
KNN	**0.3813**	**0.1555**	0.7219	**0.1840**	**0.3607**
ARIMA	1.3303	1.7011	2.7049	1.1942	1.7326
LSTM	0.5645	7.2849	7.9551	8.0766	5.9703
GAN	0.4081	0.2811	**0.6025**	0.2095	0.3753
CGAN	0.6318	0.4245	0.6968	0.2559	0.5022
f). MAPEs of the NVDA test set.					
LMR	2.4894	2.5058	2.0899	3.6247	2.6774
KNN	0.9341	1.0531	0.6399	1.5637	1.0477
ARIMA	2.1786	1.9938	1.5984	2.0929	1.9659
LSTM	**0.8402**	12.1243	10.8188	10.297	8.5201
GAN	1.0658	**0.7822**	**0.4776**	**1.2834**	**0.9022**
CGAN	1.1604	2.4538	0.4854	1.4711	1.3927
g). MAPEs of the SWKS test set.					
LMR	**1.3381**	1.6378	1.2815	**1.6442**	1.4754
KNN	1.5065	**1.1174**	1.2586	1.6564	**1.3847**
ARIMA	2.5976	2.0430	2.5419	3.4855	2.6670
LSTM	1.9081	6.2990	6.3457	7.1987	5.4379
GAN	1.9350	1.7313	1.2501	2.0234	1.7350
CGAN	1.7149	1.8520	**1.2180**	2.1641	1.7372
h). MAPEs of the EBAY test set.					
LMR	0.3222	1.0901	0.813	1.0681	0.8234
KNN	0.8402	0.5033	0.6336	0.7170	0.6735
ARIMA	1.1214	1.3163	1.5132	1.7715	1.4306
LSTM	**0.2329**	1.2547	**0.5792**	0.6469	0.6784
GAN	0.6034	0.5628	0.7878	**0.5867**	0.6352
CGAN	0.5781	**0.4940**	0.7231	0.6964	**0.6229**
i). MAPEs of the KHC test set.					
LMR	0.7283	0.7682	0.8998	1.4204	0.9542
KNN	**0.3715**	**0.1547**	0.4089	**0.4968**	**0.3580**
ARIMA	1.0659	1.0437	1.9594	0.6984	1.1919
LSTM	0.8691	5.9214	6.3042	5.9614	4.7640
GAN	0.5725	0.2316	**0.3953**	0.7931	0.4981
CGAN	0.5304	0.3763	0.4749	0.6463	0.5070

(Continued in next column)

TABLE 3 | (*Continued*) Tables 3a–i present the MAPEs of forecasted O, H, L, and C prices from various models for nine stock symbols. The smallest MAPE is highlighted in bold across the models in each column for every stock price; the second smallest is underlined. The last column contains the average values of the MAPEs of the four price targets.

TABLE 4 | Counts number of times a model is the best and second best in predicting the stock prices, and lists the mean of all mean values in the last column of Tables 3a–i.

Overall performance of all models

	Best	Second	Mean/%
LMR	6	4	1.5026
KNN	14	12	**0.9965**
ARIMA	0	1	2.7738
LSTM	5	1	7.8412
GAN	8	8	2.6033
CGAN	3	10	1.3035

TABLE 5 | Comparison of prediction results in different metrics on the AAPL and EBAY test sets using the average values of model errors on O, H, L, C prices. The smallest errors are displayed in bold font across the models we have built; the second smallest errors are underlined.

Model	RMSE	MSE	MAE	MAPE/%	SMAPE/%
a). Errors of the AAPL test set.					
LMR	2.2721	5.8504	1.9176	1.9315	1.9231
KNN	**0.7154**	**0.5680**	**0.5721**	**0.5764**	**0.5753**
ARIMA	2.6184	7.1359	2.4613	2.4906	2.4546
LSTM	5.1174	30.7145	4.2006	4.2408	4.1406
GAN	1.5321	2.5507	1.1970	1.2028	1.2152
CGAN	1.3464	2.0495	1.0378	1.0418	1.0511
b). Errors of the EBAY test set.					
LMR	0.2457	0.0689	0.1994	0.8234	0.8287
KNN	0.1970	0.0392	0.1628	0.6735	0.6765
ARIMA	0.4410	0.2071	0.3453	1.4306	1.4142
LSTM	0.2248	0.0617	0.1636	0.6784	0.6737
GAN	0.1903	0.0369	0.1534	0.6351	0.6368
CGAN	**0.1867**	**0.0353**	**0.1503**	**0.6229**	**0.6230**

results for the Open price. In the three groups of experiments, we chose three different (λ_1, λ_2) pairs in our CGAN model with $\lambda_2 := 1 - \lambda_1$. We also applied GAN as the baseline to highlight the influence of condition labels. As shown in **Table 6**, for the first test period, which follows immediately after the time span of the training data, the MAPEs from CGAN and GAN were least (see the first row of each group in **Table 6**). However, both models performed worse with increasingly larger MAPEs during the subsequent test periods except for GAN in the last training scenario. Thus, the learning models should be trained periodically, or whenever the test errors are larger than user-desired tolerances, especially when they are used in real-time prediction tasks.

TABLE 6 | Comparison of the MAPEs for AAPL-Open dataset from CGAN and GAN over three different testing periods for each training time span. The smallest error in each test group for each model is highlighted in bold.

Time span for training set	Time span for testing set	λ_1	CGAN/%	GAN/%
2014-01-01–2014-03-31	2014-04-01–2014-04-10	0.2	**3.81**	**3.73**
	2014-04-11–2014-04-20		5.52	4.93
	2014-05-01–2014-05-10		3.88	3.77
2014-07-01–2014-09-30	2014-10-01–2014-10-10	0.2	**0.77**	**0.71**
	2014-10-11–2014-10-20		1.03	0.97
	2014-11-01–2014-11-10		0.95	0.88
2015-01-01–2015-03-31	2015-04-01–2015-04-10	0.9	**2.90**	8.98
	2015-04-11–2015-04-20		9.58	**8.71**
	2015-05-01–2015-05-10		10.65	9.79

6 CONCLUSIONS AND FUTURE WORK

For individual investors and investment banks, rational prediction through statistical modeling helps decide stock trading schemes and increase their expected profits. However, it is well known that human trading decisions are not purely rational and the irrational drivers are often hard to observe. Introducing proxies of irrational decision factors such as relevant online discussion from Twitter followed by sentiment analysis has become an advanced approach in financial price modeling.

In this work, we have successfully built a sentiment-conditional GAN network in which LSTM served as the generator. We encoded Twitter messages, extracted sentiment information from the tweets, and utilized it to conduct our stock prediction experiments. The experiments showed, for about a third of our models, superior properties of our GAN and CGAN models compared to LMR and ARIMA models, KNN method, as well as LSTM networks. Our GAN and CGAN models could better adapt to the variations in the stock market and fluctuation trends of some real price data. Hence they could play a role in ensemble methods that combine strong models with small prediction errors to achieve better accuracies than any individual model when no one model would always perform best; see, for example, [38].

Even though our neural network models could outperform the simpler baseline models at times, it could still be enhanced. The main obstacle in our work was that we failed to find tweets' dataset that was longer than 70 days, which might largely weaken the training effect. For instance, we initially planned to add more labels to our input data, since it would help represent varying degrees of sentiment tendencies instead of only 'up', 'down', and 'almost unchanged'. Nevertheless, the GAN-based models could not learn better with more labels in such a short time-series dataset. Therefore, we finally chose only three classes to make sure that our model would be fully trained. The same consideration went when we created the sentiment variable, which was the reason why we only selected 'positive' and 'negative' lists when adding new words to the dictionary. If we could get a dataset with a longer time coverage, we would be able to do experiments with sentiment labels in more categories and potentially further improve our models.

Another shortcoming in our current approach is that we only took sentiment factors into account. The stock market is very complicated and there are potentially a great many factors to be considered. There are some other factors like economic growth, interest rates, stability, investor confidence and expectations. We also noticed that political events would have a huge impact on the variation of the stock market. We can extract political information from newspapers and news websites. If we added these factors to the input, our model may better learn the features of the stock market and make the prediction tendencies close to that in the real world.

In this work, we have assumed that the tweets are more or less truthful. However, social media sources could be contaminated with fake news or groundless comments, that are hard to be distinguished from the good ones with real signals. A fruitful area of future research is to look into GAN models for alleviating the problem.

It is also known that neural network models often need much bigger datasets to beat simpler models. To this end, there are multiple ways we could explore: consider more than nine stocks; join all stock prices into one single dataset, i.e., to train one model on panel data grouped by stocks and prices (O, H, L, C) instead of a single time series; sample our datasets at hourly frequency.

Lastly, many properties of neural networks are still active areas of research. For example, in LSTM, GAN, and CGAN models, the loss function values and the solution quality often appear to be sensitive to small changes in inputs, hyperparameters, or stopping conditions. Creating stable yet efficient numerical algorithms are necessary for reliable solutions. In addition, the success of neural networks are often associated with computer vision problems. Adopting them in finance may require different techniques or transformations.

In the future, we would like to explore the topic further in the following directions:

1) Obtaining larger datasets and creating labels in more classes to improve our model by examining hourly data, for example.
2) Building GAN models for detecting overhyped or deceitful messages about a stock before incorporating them into our models.
3) Attempting to extract more stock-related factors and adding them as predictors in our models.
4) Experimenting with more stocks or other financial assets, and considering having one model for all stock price data.

5) Utilizing more sophisticated natural-language processing (NLP) methods to analyze the financial or political information from news media and assessing the impact and the role they play in the stock market.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: https://data.world/kike/nasdaq-100-tweets.

AUTHOR CONTRIBUTIONS

YZ: Drafted the manuscript; developed code for data preprocessing and sentiment analysis; designed modeling methods and experiments. JL: Modified the manuscript and cooperated in the design of data processing. HW: Modified the manuscript and cooperated in the design of sentiment analysis. S-CC: Provided directions and supervision on scientific principles and methodology for the whole project; reviewed and modified code for computational scalability and reproducibility; edited manuscript for content accuracy and clarity.

ACKNOWLEDGMENTS

The authors would like to thank the editor, Prof. Qingtang Jiang, and reviewers, Prof. Qi Ye and Prof. Qiang Wu, for their invaluable support and feedback during the anonymous review process. In addition, we are grateful to the following seminar speakers for inspiration during the development of this work in the 2019 IIT Elevate Summer Program's research course, *SCI 498-106 Machine Learning Algorithms on Heterogeneous Big Data:* Victoria Belotti, Adam Ginensky, Joshua Herman, Rebecca Jones, Sunit Kajarekar, Hongxuan Liu, Lawrence K. H. Ma, Qiao Qiao, Jagadeeswaran Rathinavel, Aleksei Sorokin, Huayang (Charlie) Xie, and Zhenghao Zhao. In addition, the last author would like to thank the following colleagues for discussion and encouragement: Prof. Robert Ellis, Prof. Robert Jarrow, Jackson Kwan, Wilson Lee, Prof. Xiaofan Li, Prof. Lek-Heng Lim, Aleksei Sorokin, Don van Deventer, and Martin Zorn.

REFERENCES

1. Devi KN, and Bhaskaran VM. Impact of social media sentiments and economic indicators in stock market prediction. *Int J Comput Sci Eng Technol* (2015) 6.
2. Zhang X, Zhang Y, Wang S, Yao Y, Fang B, and Yu PS. Improving stock market prediction via heterogeneous information fusion. *Knowledge-Based Syst* (2018) 143:236–47. doi:10.1016/j.knosys.2017.12.025
3. Bollen J, Mao H, and Zeng X. Twitter mood predicts the stock market. *J Comput Sci* (2011) 2:1–8. doi:10.1016/j.jocs.2010.12.007
4. Pagolu VS, Reddy KN, Panda G, and Majhi B. Sentiment analysis of Twitter data for predicting stock market movements. In: International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES); 2016 Oct 3–5; Paralakhemundi, India. New York, US: IEEE (2016) p. 1345–50.
5. Xu Y, and Cohen SB. Stock movement prediction from tweets and historical prices. In Proceedings of the 56th annual meeting of the association for computational linguistics (Vol. 1: Long Papers); 2018 July; Melbourne, Australia. Stroudsburg, Pennsylvania: Association for Computational Linguistics (2018) 1970–9.
6. Hutto CJ, and Gilbert E. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth International AAAI Conference on Weblogs and Social Media; 2014 June 1-4; Ann Arbor, MI. The AAAI Press (2014).
7. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, and Ozair S. Generative adversarial nets. *Adv Neural Inf Process Syst* (2014) 2672–80.
8. Goodfellow IJ, Shlens J, and Szegedy C. Explaining and harnessing adversarial examples. In: Bengio A, LeCun Y, editors. 3rd International Conference on Learning Representations, {ICLR} 2015; 2015 May 7–9; San Diego, CA (2015). Available from: http://arxiv.org/abs/1412.6572.
9. Kurakin A, Goodfellow I, and Bengio S. (2017) Adversarial Machine Learning at Scale, 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings, Toulon, France, Apr 24–26, 2017, OpenReview.net. Available from: https://openreview.net/forum?id=BJm4T4Kgx
10. Miyato T, Dai AM, and Goodfellow I. Adversarial Training Methods for Semi-Supervised Text Classification, 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings Toulon, France, Apr 24–26, 2017, OpenReview.net. Available from: https://arxiv.org/abs/1605.07725
11. He X, He Z, Du X, and Chua TS. Adversarial personalized ranking for recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval; 2008 June. New York, US: ACM (2018) p. 355–64.
12. Li C, Wang S, Wang Y, Yu P, Liang Y, and Liu Y. Adversarial learning for weakly-supervised social network alignment. In: Proceedings of the AAAI Conference on Artificial Intelligence; 2019 Jan 27-Feb 1; Honolulu, Hawaii. Menlo park, California: AAAI (2019) 33:996–1003. doi:10.1609/aaai.v33i01.3301996
13. Mirza M, and Osindero S. Conditional generative adversarial nets. *CoRR* abs/1411.1784 (2014). Available from: http://arxiv.org/abs/1411.1784
14. Antipov G, Baccouche M, and Dugelay JL. Face aging with conditional generative adversarial networks. In: IEEE International Conference on Image Processing (ICIP); 2017 Sept 17–20; Beijing, China. New York, US: IEEE (2017) p. 2089–93.
15. Yang Z, Chen W, Wang F, and Xu B. Improving neural machine translation with conditional sequence generative adversarial nets. *CoRR* abs/1703.04887 (2017). Available from: http://arxiv.org/abs/1703.04887
16. Yu L, Zhang W, Wang J, and Yu Y. SeqGAN: Sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI conference on artificial intelligence. New York, US: ACM (2017) 2852–8.
17. Mogren O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *Constructive Machine Learning Workshop at NIPS 2016 in Barcelona, Spain*. Available from: https://arxiv.org/pdf/1611.09904.
18. Esteban C, Hyland SL, and Rätsch G. Real-valued (medical) time series generation with recurrent conditional GANs. *CoRR* (2017). Available from: http://arxiv.org/abs/1706.02633.
19. Zhu F, Ye F, Fu Y, Liu Q, and Shen B. Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network. *Sci Rep* (2019) 9: 6734. doi:10.1038/s41598-019-42516-z
20. Wang Z, Chai J, and Xia S. Combining recurrent neural networks and adversarial training for human motion synthesis and control. *IEEE Trans Visualization Comput Graphics* (2019) 27:14–28. doi:10.1109/TVCG.2019.2938520
21. Zhou X, Pan Z, Hu G, Tang S, and Zhao C. Stock market prediction on high-frequency data using generative adversarial nets. *Math Probl Eng* (2018) 2018: 1–11. doi:10.1155/2018/4907423
22. Wang JJ, Wang JZ, Zhang ZG, and Guo SP. Stock index forecasting based on a hybrid model. *Omega* (2012) 40:758–66. doi:10.1016/j.omega.2011.07.008
23. Huang W, Nakamori Y, and Wang S-Y. Forecasting stock market movement direction with support vector machine. *Comput Operations Res* (2005) 32: 2513–22. doi:10.1016/j.cor.2004.03.016

24. Chong E, Han C, and Park FC. Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies. *Expert Syst Appl* (2017) 83:187–205. doi:10.1016/j.eswa.2017.04.030
25. Adebiyi AA, Adewumi AO, and Ayo CK. Comparison of ARIMA and artificial neural networks models for stock price prediction. *J Appl Math* (2014) 2014: 1–7. doi:10.1155/2014/614342
26. Nelson DM, Pereira AC, and de Oliveira RA. Stock market's price movement prediction with LSTM neural networks. In: International Joint Conference on Neural Networks (IJCNN); 2017 May 14–19; Anchorage, AK, USA. New York, US: IEEE (2017) p. 1419–26.
27. Loughran T, and McDonald B. The use of word lists in textual analysis. *J Behav Finance* (2015) 16:1–11. doi:10.1080/15427560.2015.1000335
28. Li FT, Ting KM, and Zhou ZH. Isolation-based anomaly detection. *ACM Trans Knowledge Discov Data (TKDD)* (2012) 6:3. doi:10.1145/2133360.2133363
29. Bracewell RN. *The Fourier transform and its applications*. New York: McGraw-Hill (1986).
30. Tsay RS. *Analysis of financial time series*. New York, US: John Wiley & Sons (2005) Vol. 543.
31. Hochreiter S, and Schmidhuber J. Long short-term memory. *Neural Comput* (1997) 9:1735–80. doi:10.1162/neco.1997.9.8.1735
32. Tieleman T, and Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks Machine Learn* (2012) 4:26–31.
33. Chollet F. *Keras* (2015). Available from: https://keras.io.
34. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, and Citro C. TensorFlow: large-scale machine learning on heterogeneous distributed systems (2015). Available from: tensorflow.org.
35. Cover T, and Hart P. Nearest neighbor pattern classification. *IEEE Trans Inform Theor* (1967). 13:21–7. doi:10.1109/tit.1967.1053964
36. Hyndman RJ, and Khandakar Y. Automatic time series forecasting: The forecast Package for R. *J Stat Soft* (2008) 27:1–22. doi:10.18637/jss.v027.i03
37. Smith TG. *pmdarima: ARIMA estimators for Python* (2017).
38. Clemen RT. Combining forecasts: A review and annotated bibliography. *Int J Forecast* (1989) 5:559–83. doi:10.1016/0169-2070(89)90012-5

Conflict of Interest: Author S-CC was employed by the company Kamakura Corporation.

The remaining 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.

ORIGINAL RESEARCH
published: 10 June 2021
doi: 10.3389/fams.2021.632847

Auto Defect Detection Using Customer Reviews for Product Recall Insurance Analysis

Titus Hei Yeung Fong, Shahryar Sarkani and John Fossaceca*

Department of Engineering Management and Systems Engineering, The George Washington University, Washington, DC, United States

The challenge for Product Recall Insurance companies and their policyholders to manually explore their customer product's defects from online customer reviews (OCR) delays product risk analysis and product recall recovery processes. In today's product life cycle, product recall events happen almost every day and there is no practical method to automatically transfer the massive amount of valuable online customer reviews, such as defect information, performance issue, and serviceability feedback, to the Product Recall Insurance team as well as their policyholders' engineers to analyze the product risk and evaluate their premium. This lack of early risk analysis and defect detection mechanism often increases the risks of a product recall and cost of claims for both the insurers and policyholder, potentially causing billions of dollars in economic loss, liability resulting from the bodily injury, and loss of company credibility. This research explores two different kinds of Recurrent Neural Network (RNN) models and one Latent Dirichlet Allocation (LDA) topic model to extract product defect information from OCRs. This research also proposes a novel approach, combined with RNN and LDA models, to provide the insurers and the policyholders with an early view of product defects. The proposed approach first employs the RNN models for sentiment analysis on customer reviews to identify negative reviews and reviews that mention product defects, then applies the LDA model to retrieve a summary of key defect insight words from these reviews. Results of this research show that both the insurers and the policyholders can discover early signs of potential defects and opportunities for improvement when using this novel approach on eight of the bestselling Amazon home furnishing products. This combined approach can locate the keywords of these products' defects and issues that customers mentioned the most in their OCRs, which allows the insurers and the policyholders to take required mitigation actions earlier, proactively stop the diffusion of the detective products, and hence lower the cost of claim and premium.

Keywords: product recall insurance, machine learning, artificia lintelligence, natural language processing, neural network, opinion mining, product defect discovery, topic model

OPEN ACCESS

Edited by:
Sou Cheng Choi,
Illinois Institute of Technology,
United States

Reviewed by:
Halim Zeghdoudi,
University of Annaba, Algeria
Shengsheng Qian,
Institute of Automation, Chinese Academy of Sciences, China

****Correspondence:***
Titus Hei Yeung Fong
titusfong@gwmail.gwu.edu

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *24 November 2020*
Accepted: *31 May 2021*
Published: *10 June 2021*

Citation:
Fong THY, Sarkani S and Fossaceca J (2021) Auto Defect Detection Using Customer Reviews for Product Recall Insurance Analysis. Front. Appl. Math. Stat. 7:632847. doi: 10.3389/fams.2021.632847

INTRODUCTION

In the Product Recall Insurance landscape, as the number of product recall events increases every year in almost every industry, insurance companies are constantly faced with the difficulty of predicting recall risk, assessing the potential of loss, and estimating the premium. While manufacturers have been using advanced quality control tools for product development, defective products can still be found on the market, and product recall events often happen [1]. The motivation is this paper is to develop a novel approach, combined with RNN and LDA models, to provide the insurers and the manufacturers with an early view of product defects.

These product recall events can cost both the insurance companies and the manufacturers billions of dollars of loss and bring significant brand reputation impact that lasts for an extended period. A good example is the recent 2016 Samsung Galaxy Note 7 explosions recall event. This event cost Samsung more than $5 billion of loss and the subsequent loss of sales in the electronics industry [2]. Allianz Global Corporate & Specialty (AGCS) has analyzed the average value of each product recall insurance claim, excluding small value claims, to be about 1.4 million Euro between 2012 and 2017 [3]. Product recall insurance is intended to provide manufacturers with financial protection for the cost of the recall and their liabilities but when the number of cases increases the insurers have become the biggest victims.With the growth under the new era of Web 2.0, various social media and e-commerce sites such as Amazon.com and Twitter.com now provide online virtual communities for consumers to share their feedback on different products and services. Such feedback, which always includes customer complaints and defect information about the products, can provide valuable intelligence, such as the product's design, performance, and serviceability, for the insurers and the manufacturers to take remedial actions to avoid potential recalls. While there is a large amount of such information available on these sites, there is a lack of an effective method to automatically distill the information to the engineering team.

In the Samsung Galaxy Note 7 recall case, a study showed that there were early customer reports of overheating problems through online customer feedback before Samsung realized and took action [4–6]. It is, however, a challenge for the insurers and the manufacturers to manually read through a large amount of customer feedback available online and be able to discover early signs of product defects, delaying the manufacturers to take necessary recovery actions.

As technology progresses with faster computers and better computational algorithms, we are now able to collect, process, and extract useful information from a large amount of textual customer feedback. In the field of business and product design, studies [7, 8] show that machine learning predictive models can successfully extract useful information from OCRs, including customer buying patterns and customer requirements on future product design. This research provides the foundation for applying machine learning predictive models to detect defective product information from OCRs.

This research explores methods of extracting product defect information from online customer reviews (OCRs) and demonstrates a novel predictive model using Recurrent Neural Network (RNN) and Latent Dirichlet Allocation (LDA) topic model. The new predictive model provides the insurance companies and the manufacturers with an early view into product defects, enabling them to make an assessment of product recall risks, and take required mitigation actions early to proactively stop the spread of the defective products. This will help both sides to prevent further damages and economic losses.

The scope of this research studies the viability of a probabilistic model with RNN and LDA to analyze, extract, and identify defective product information from OCRs. RNN is a type of neural network model for analyzing time-series data. This model can solve problems involving sequences of word order textual data. LDA is a type of generative probabilistic model for discovering a set of topics best describes a collection of discrete data. This consists of the collection of customer online review data with manual labels for supervised learning, quantitative models for testing the hypothesizes associated with identifying product defects, and verification of models.

The new proposed approach includes two quantitative RNN classifiers and one LDA topic model. The first RNN classifier differentiates negative OCRs from non-negative OCRs. The second RNN classifier differentiates OCRs that contain defect information from OCRs that do not contain defect information. The LDA topic model, which combines the first and the second RNN classifiers, generates a set of key product defect topics for a product using OCRs.

The input data consists of 9,000 randomly selected OCRs, and their star ratings, from the furniture section of Amazon's Customer Review Public Dataset. This customer review dataset will be used as input for the RNN classifiers.

The rest of the paper is organized as follows: in *Data*, we discuss related literature reviews. Then, in *Methods* and *Results*, we present a detailed explanation of the data and our proposed model. After that, in *Conclusion* we present the results. Lastly, in *Discussion and Future Work*, and *Data Availability Statement*, we end with conclusion, discussion and future work, respectively.

RELATED WORK

Product Defect and Product Recalls

These defective product recall events can cost manufacturers as well as their insurers billions of dollars and bring significant impact that lasts for an extended period. The United States Consumer Product Safety Commission (CPSC) has concluded that deaths, injuries, and property damages from consumer product incidents cost the nation more than $1 trillion annually [9]. According to AGCS, these events have caused insured losses of over $2bn over the past five years, making them the largest generator of liability losses. An empirical, event-time analysis found that product recall events had a direct impact on the company's equity price for two months after the events' release [10]. These kinds of events expose companies not only to economic damages, but also negative consequences such as loss of goodwill, loss in product liability suits, and loss to their rivals [11].

Due to increased product complexity and more stringent product safety legislation [12], studies show that the trend of product recall events is increasing and no industry is immune from a product recall event [13]. Recalls happen when the manufacturer does not address the issue or was unaware of it before the product was distributed to the market. The main two reasons for recalls are 1) a consequence of design flaws, which make the product fail to meet required safety standards, and 2) manufacture defects in products that do not conform to specifications such as poor craftsmanship [14]. According to Beamish [15], in the toy manufacturing industry, design flaws contributed to 70.8% of the recalls, while only 12.2% of the recalls were from manufacture.

Time to Recall and Product Recall Strategy

The strategy and the timing for the recall of a defective product have a direct impact on both the finances and the reputation of manufacturers and insurers. Time is an essential factor during a defective product recall event. The longer the defective product remains in the marketing and distribution process, the harder it is for the company to take recovery action, and poses more potential for injuries [16, 17]. Studies showed that manufacturers with a preventive recall strategy in place, such as continuously identifying product defects and initiating voluntary recalls, have a shorter time to recall than companies with a reactive approach such as initiating recall after a hazard is reported [17]. There is also research proving that a positive customer and public relations impact on the company will result when a proactive recall strategy is implemented during a product defect event [18, 19]. While it is hard to avoid the existence of defective product risk, the aforementioned studies have shown that time to recall and a company's recall strategy have a direct impact on the company's reputation and losses, which supports the fact that when a company has an early view of product defect, it can reduce the loss of business and damage to reputation, and regain consumer trust.

Opinion Mining With Online Customer Reviews

With the rise of Web 2.0, social media and customer review sites have enabled companies to discover consumer feedback on their products with increased speed and accuracy. Information embedded in CORs has a direct impact on companies and their products [20]. Comparing to "Offline" word of mouth customer opinions, OCRs have a much more significant impact because of their persistent, easily accessible, and open-to-public format [21]. Companies have been looking at OCRs to improve their product and marketing strategies [22]. OCRs enable companies to monitor customer concerns and complaints, as well as to take corrective actions [20]. Some companies even respond to these customer text reviews personally to improve their service [23].

Product Defects Discovery

While there are several research studies and proposed algorithms for using the previous generation products' OCRs to provide valuable information to engineers on the next generation of product development and product design, there is little attention in academia for using OCRs in the later stages of the product cycle to discover customer complaints and product defects [24]. Abrahams et al. proposed a new algorithm using a sentiment analysis method to classify the type of product defect information (e.g., performance defects, safety defects, non-defect, etc.) embedded in OCRs for vehicles [24]. They recognized that while traditional sentiment analysis methods can successfully identify complaints in other industries, they fail to distinguish defects from non-defects and safety from performance defects in the automotive industry. This is because OCRs in the auto industry that mentions safety defects have more positive words, and fewer negative words and subjective expressions than other OCRs. Alternatively, their team spent 11 weeks building and tagging a set of automotive "smoke" words dataset from the OCRs before doing sentiment analysis. This method has shown success in defect discovery and classification, but it is also highly domain-specific for the automotive industry. Bleaney et al. studied and compared the performance of various classifiers, including Logistic Regression, k-nearest neighbors (k-NN), Support-Vector Machines (SVM), Naïve Bayes (NB) classifier, and Random Forest (Decision Tree) on identifying safety issues ("Mentions Safety Issue," "Does Not Mention Safety Issue") embedded in OCRs in the baby product industry [6]. They found that the Logistic Regression classifier had the highest precision, with 66% in the top 50 reviews surfaced. Zhang et al. proposed an unsupervised learning algorithm using the LDA topic model's method to group and identify key information and words in each type of defect from complaints and negative reviews [25].

While these studies have shown some success in using sentiment analysis methods to extract defective information from OCRs, these studies have not been able to locate defectives with OCRs from a product level or accept all OCRs from a single product.

Natural Language Processing

In the field of linguistics and computer science, there is active and ongoing research on how to improve how computers understand human behavior and language. The development of Natural Language Processing (NLP), which is a type of artificial intelligence concentrated on understanding and manipulating human language, has achieved practical successes over the years. NLP models have successfully helped researchers solve real-world human text processing problems, especially research on using OCRs to extract product defects [6, 24, 26]. In this research, Latent Dirichlet Allocation (LDA) and Neural Network and Recurrent Neural Network (RNN) are used to extract product defect information.

Text Representation

Word embedding is a widely adopted method in representing raw textual data to its low-dimension property. This encoding scheme transforms each word into a set of meaningful, real-valued vectors [27]. Instead of randomly assigning vectors to words, Mikolov et al. have created GloVe, one of the most widely-used pre-trained datasets among researchers for mapping words to vectors, which is to be used in this research [28]. This dataset maps words that have closer English meaning to a closer vector scale in a general sentiment analysis task. An example would be mapping the term "king" in a closer scale to "man" in vector, while further from "woman." Other studies have also built models on top of these two datasets to enhance word embedding in domain-specific sentiment analysis tasks [27, 29, 30].

Latent Dirichlet Allocation

Another unsupervised NLP method that can discover textual data insight is the Latent Dirichlet Allocation (LDA). LDA is a three-level hierarchical Bayesian model that can discover a set of unobserved groups, or topics, that best describe a large collection of observed discrete data. As Blei et al. suggested, the goal of LDA is to "find short descriptions of the members of a collection that enable efficient processing of large collections while preserving the essential statistical relationships that are useful for basic tasks, such as classification, novelty detection, summarization, and similarity and relevance judgments." [31]. LDA was first proposed and used for discovering population genetics structure in the field of bioengineering in 2000 [32] and further used in NLP processing on textual data in 2003 [31].

Researchers have been using LDA to discover topics from a large collection of OCRs to help the industry to gain insight into their product. Santosha et al. and Zhai et al. both suggested using LDA for grouping and producing an effective summary of product features from a large collection of OCRs. Santosha et al. successfully used the product features terms from the LDA topic model to build a Feature Ontology Tree for showing product features relationships [33], while Zhai et al. built a semi-supervised LDA with additional probability constraints to show product features linkage between products [34]. Researchers have also suggested LDA topic model can be used on serving industries to discover business insight such as a summary of OCRs on travel and hospitality review sites. Titov et al. and Calheiros et al. both have suggested using LDA outputted topic's terms to discover and analyze customer reviewer's sentiment for businesses to improve customer experience [35, 36]. In the field of product defect management, Zhang et al. used the LDA topic model to discover short summaries of product defects and solutions on a large amount of online product negative reviews and complaints to help engineers and customers discover product defect information [25].

Neural Network and Recurrent Neural Network

A neural network is a set of connected computational input/output units that loosely model the biological brain [37]. A neural network can be trained iteratively with supervised data, can recognize or "Learn" specific patterns embedded in this data, and perform prediction tasks based on these learned patterns without pre-programmed rules. In the context of this research, the neural network plays an important role in solving text classification problems, especially when using recurrent neural network architecture [38]. Based upon the Neural Network architecture, researchers in the 1980s have been proposing adding recurrent connections between nodes to solve problems involving sequential data, which is now called the Recurrent Neural Network (RNN) [39, 40]. While this type of network can solve sequential recognition, Bengio et al. found that it is difficult to solve problems where the sequences are getting longer and prediction depends on input presented at an earlier time [41]. This is due to vanishing gradients where the error gradient propagating back tends to vanish in time [42]. Hochreiter et al. also proposed the Long Short-Term Memory (LSTM) approach, a particular type of RNN architecture, which further improved the problem involving long data sequences. LSTM overcomes this problem by adding gates in RNN nodes to regulate the flow of information [42]. Researchers suggested that LSTM can greatly improve the accuracy of sequence learning, such as offline handwriting recognition [43], as well as text classification problems that involve word order [38]. An LSTM-RNN model can build the relations between sentences in semantic meaning on text classification, which can increase the model accuracy over that of the traditional methods [44, 45].

DATA

Data Collection and Data Labeling

The primary data source of this study is the OCRs and their associated metadata from the Amazom.com marketplace. This dataset contains customer text reviews, product information, star ratings, review dates, and other relevant information. The customer reviews dataset will be used as input for sentiment analysis to identify negative customer reviews, determine their usefulness in providing information about defects, classify the types of defects found in the OCRs, and extract product defect topics within the context of this research.

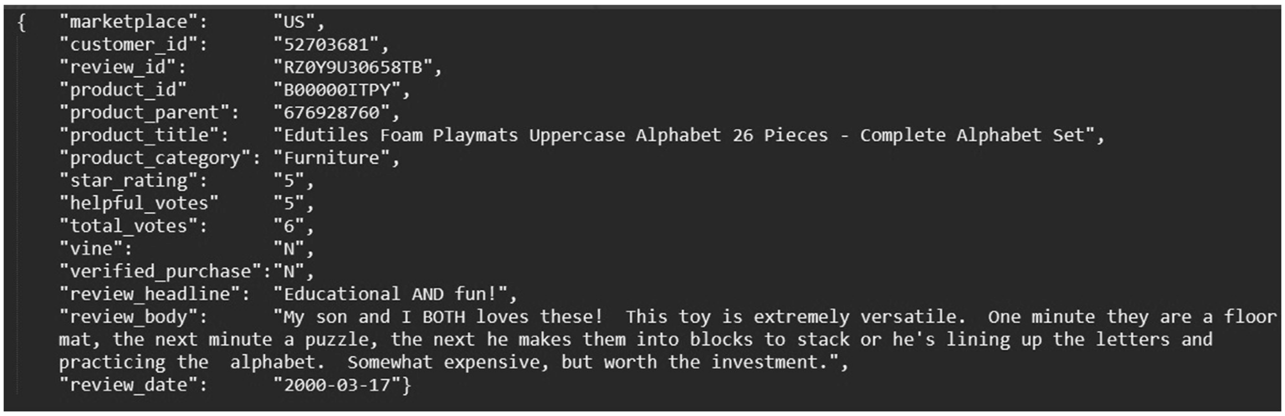
Amazon's Customer Reviews Public Dataset is an organized version of OCR data in a number of tab-separated values (TSV) files for researchers in the fields of Natural Language Processing (NLP), Information Retrieval (IR), and Machine Learning (ML). This data set contains more than 130 million OCRs and associated metadata in 43 product categories in the United States marketplace from 1995 until 2015.

Furniture Customer Reviews and Data Format

The furniture section subset of this dataset was used in this study. This data subset contains 792,114 OCRs and associated metadata of customer options on Amazon furniture products. Due to the constraint of manual labor in creating a supervised dataset, this study randomly selected 3,000 Amazon OCRs with a rating of three or more stars and 6,000 Amazon OCRs with a rating of one or two stars for model building, training, and testing. A sample data of this dataset is shown in **Figure 1**. The focus of this study is the textual analysis in the "review_body" column of the dataset.

Data Labeling

Successful machine learning models are built on large volumes of high-quality training data. To build RNN models that can extract defective information embedded in each OCR, each OCR needs

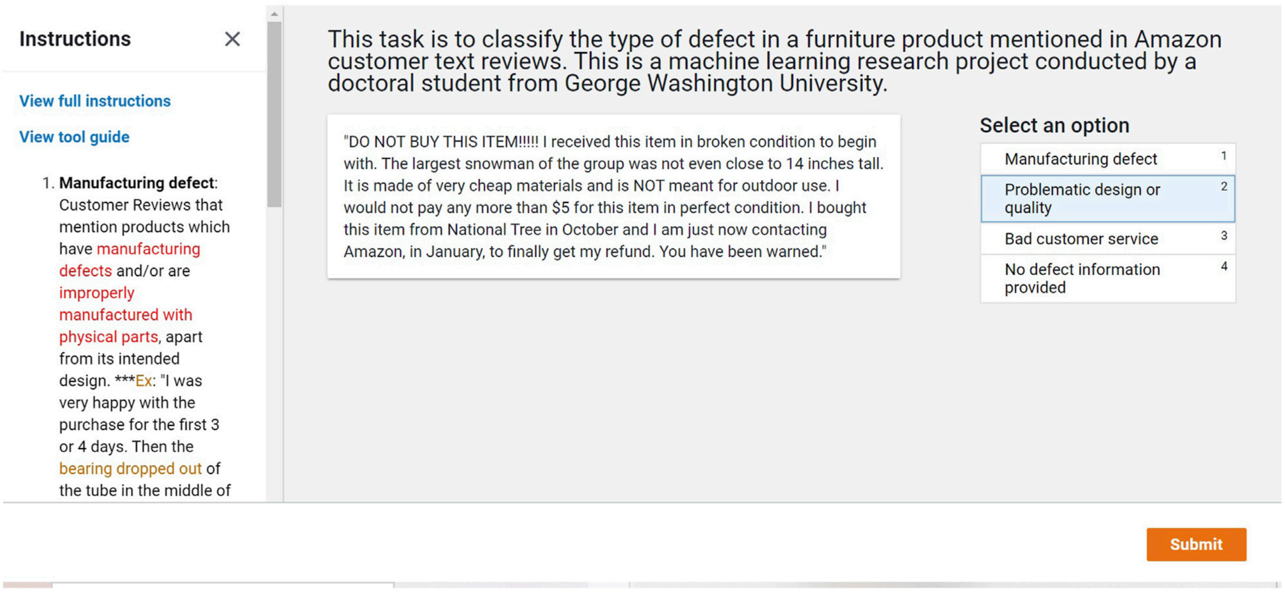


```
{   "marketplace":      "US",
    "customer_id":      "52703681",
    "review_id":        "RZ0Y9U30658TB",
    "product_id"        "B00000ITPY",
    "product_parent":   "676928760",
    "product_title":    "Edutiles Foam Playmats Uppercase Alphabet 26 Pieces - Complete Alphabet Set",
    "product_category": "Furniture",
    "star_rating":      "5",
    "helpful_votes"     "5",
    "total_votes":      "6",
    "vine":             "N",
    "verified_purchase":"N",
    "review_headline":  "Educational AND fun!",
    "review_body":      "My son and I BOTH loves these!  This toy is extremely versatile.  One minute they are a floor
    mat, the next minute a puzzle, the next he makes them into blocks to stack or he's lining up the letters and
    practicing the  alphabet.  Somewhat expensive, but worth the investment.",
    "review_date":      "2000-03-17"}
```

FIGURE 1 | Simple amazon raw data.

TABLE 1 | Definition of AWS SageMaker ground truth labels.

Label	Definitions of Labels	Example
Manufacturing defect	OCRs that mention products that have manufacturing defects and/or are improperly manufactured with physical parts, apart from its intended design	I Was very happy with the purchase for the first 3 or 4 days. Then the bearing dropped out of the tube in the middle of it. (review_id: R238K8EITCNRZZ
Problematic design or quality	OCRs that mention products that have a problematic design or quality issue, with no mention of physical parts falling off	The knee cushion is not comfortable for sitting any longer than about 10 min. The chair is clunky and hard to move on the floor without picking it up. (review_id: R31BYJESH8F2DO)
Bad customer service	OCRs related to the frustration of delivery of the product or customer support process, while not related to the physical products themselves	I Have since called twice more with no returned call. This is the worst customer service I have ever received (review_id: R3VG1CFNR60ED)
No defect information provided	OCRs did not mention any of the defect information from the last three categories	We bought four of these, they are just some real cheap chairs that are overpriced. (review_id: R2F8RCR0LFI7SS)



Instructions

View full instructions

View tool guide

1. **Manufacturing defect**: Customer Reviews that mention products which have manufacturing defects and/or are improperly manufactured with physical parts, apart from its intended design. ***Ex: "I was very happy with the purchase for the first 3 or 4 days. Then the bearing dropped out of the tube in the middle of

This task is to classify the type of defect in a furniture product mentioned in Amazon customer text reviews. This is a machine learning research project conducted by a doctoral student from George Washington University.

"DO NOT BUY THIS ITEM!!!!! I received this item in broken condition to begin with. The largest snowman of the group was not even close to 14 inches tall. It is made of very cheap materials and is NOT meant for outdoor use. I would not pay any more than $5 for this item in perfect condition. I bought this item from National Tree in October and I am just now contacting Amazon, in January, to finally get my refund. You have been warned."

Select an option

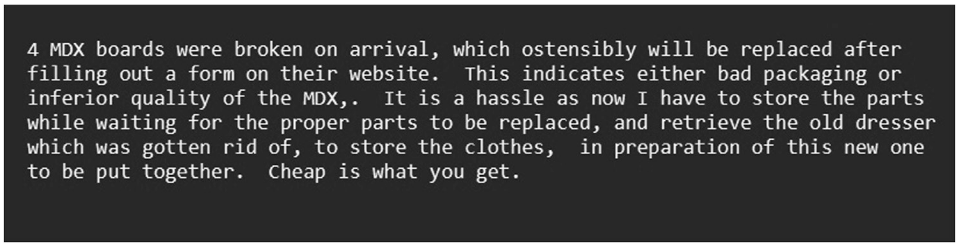
Manufacturing defect 1

[✓ Problematic design or quality 2]

Bad customer service 3

No defect information provided 4

FIGURE 2 | AWS SageMaker interface for labelers.



```
4 MDX boards were broken on arrival, which ostensibly will be replaced after
filling out a form on their website.  This indicates either bad packaging or
inferior quality of the MDX,.  It is a hassle as now I have to store the parts
while waiting for the proper parts to be replaced, and retrieve the old dresser
which was gotten rid of, to store the clothes,  in preparation of this new one
to be put together.  Cheap is what you get.
```

FIGURE 3 | Simple OCR before text preprocessing step.

to be manually labeled by human labelers for RNN model training and testing purposes. The "Amazon Web Services (AWS) SageMaker Ground Truth" data labeling service was procured to manually label OCRs to build the required supervised dataset. AWS SageMaker Ground Truth provides a platform for independent labelers to label machine learning tasks, and each of the 9,000 OCRs was reviewed by three human labelers to ensure the accuracy of the data. Human-labeled results are also generated with a confidence score for each label to ensure high-quality data.

Labelers were provided with a detailed definition of each label with an example (as shown in **Table 1**) to categorize each OCR on the AWS SageMaker Ground Truth interface in **Figure 2**.

Each of the OCRs was labeled with one of the following four labels: "Manufacturing defect," "Problematic design or quality," "Bad customer service," and "No defect information." The labels of "Manufacturing defect" and "Problematic design or quality" are identified as the two main reasons for a company to initiate product recalls [14, 15]. "Bad customer service" is also added as one of the labels because a customer service issue is also where customers often report issues, especially while shopping on an online platform. The human label is able to label and determine which label is the OCRs related to the most in the minimalist and clean interface as shown in **Figure 2**.

Data Preprocessing

Data preprocessing is an essential step in turning raw text data from OCRs into useful information that the computer can read, and machine learning can process. After the OCRs have been extracted from the review body, the OCRs were cleaned and preprocessed, and then turned into digitized text representation vectors.

Text Preprocessing and Cleaning

Human-created text data often contains inconsistent wording, special characters, and contractions, which can contribute to inaccurate data analysis and affect model performance [37, 44, 46]. For this reason, text preprocessing and cleaning is an essential step in ensuring input data quality with normalizing words and removing unnecessary characters. The following three steps are taken in this study to improve text data quality: 1) Stemming, 2) Stop-word removal, and 3) Special character, numeric, and empty text removal.

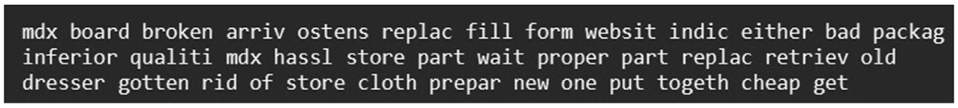
Stemming is an NLP technique to groups and reduces different words with the same root and linguistic meaning into the same word stem or root form. This study employed the Python Natural Language Toolkit (NLTK) package algorithms for the stemming process. The computer would treat these different words with the same word stem as an equal text vector representation. Stop-word removal can remove over commonly used words in English that give no or very little linguistic meaning to the overall context of the given text. To avoid losing the textual message in translation, this study used a custom-written Stop-word removal function. Special character, numeric, and empty text removal is an NLP technique to remove the non-text characters to improve data quality such as "!", "@", "#", and empty space. This increases the machine learning model's performance by only focusing on real textual data.

The following two figures, **Figures 3**, **4** show an OCR before and after these three steps were done.

The two figures, **Figures 3**, **4** show an OCR before and after these three steps were done. The special character, empty text and stop-words are removed to improve data quality. Stemming process is also done in the text. Words such as "replaced" are replaced with the root "replac" and "arrived" are replaced with "arriv". This allows the models later in the process treat words with the same root with the same meaning.

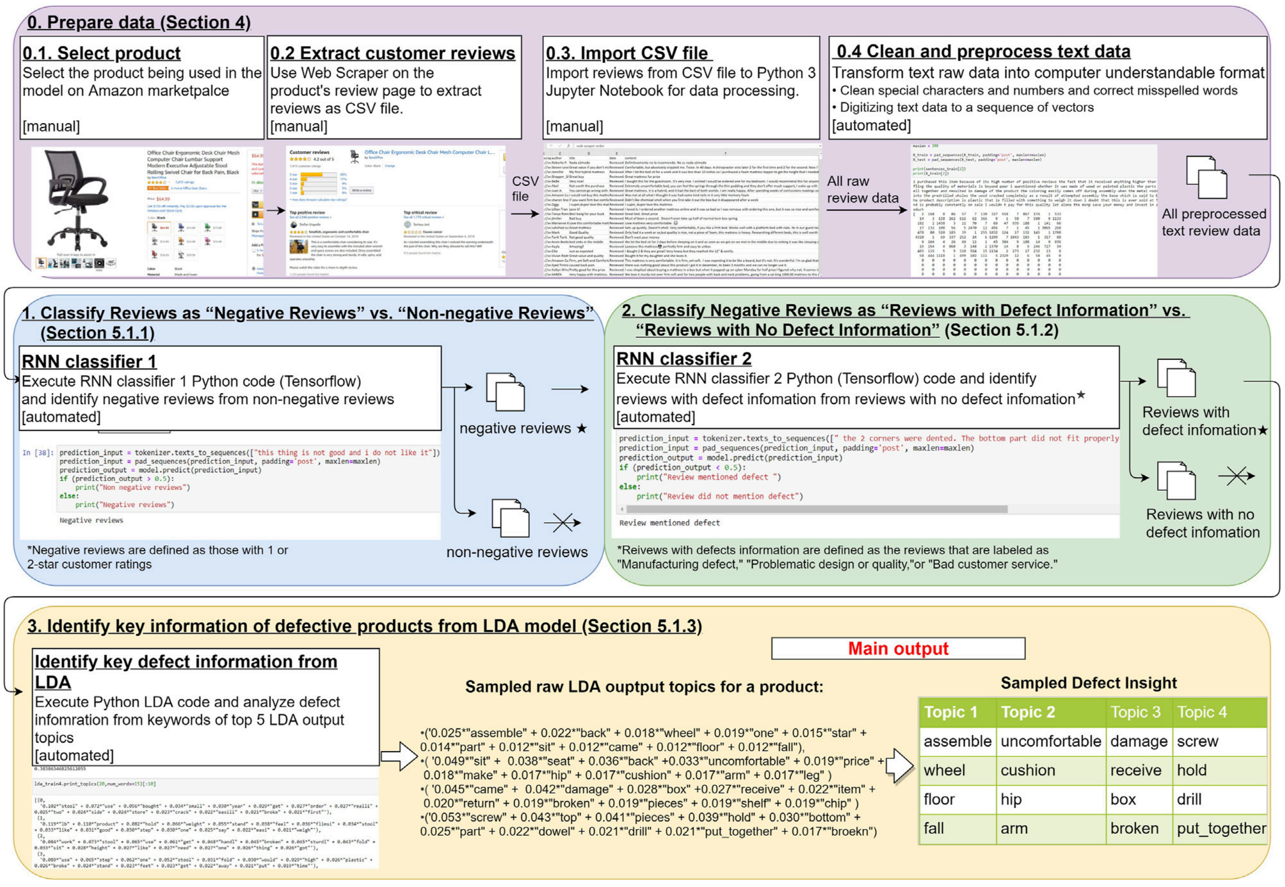
Text Representation

As mentioned in *Text Representation*, OCRs textual data have to be turned into text representation in a digitalized format inputting to machine learning models for computers to recognize the information. This encoding scheme transforms each word into a set of meaningful, real-valued vectors to represent each word in a given text [27] This study uses Word2Vector pre-trained datasets for mapping real number vectors to words with similar linguistic meaning to a closer vector scale [28]. This increases the efficiency of the training process for the machine learning model. The data preprocess also takes a text padding step to normalize the length of the ORCs by padding "0" after the end of each text representation before the OCRs are fed to the machine learning model.



```
mdx board broken arriv ostens replac fill form websit indic either bad packag
inferior qualiti mdx hassl store part wait proper part replac retriev old
dresser gotten rid of store cloth prepar new one put togeth cheap get
```

FIGURE 4 | Simple OCR after text preprocessing step.


0. Prepare data (Section 4)
0.1. Select product
Select the product being used in the model on Amazon marketpalce
[manual]
0.2 Extract customer reviews
Use Web Scraper on the product's review page to extract reviews as CSV file.
[manual]
0.3. Import CSV file
Import reviews from CSV file to Python 3 Jupyter Notebook for data processing.
[manual]
0.4 Clean and preprocess text data
Transform text raw data into computer understandable format
• Clean special characters and numbers and correct misspelled words
• Digitizing text data to a sequence of vectors
[automated]
CSV file
All raw review data
All preprocessed text review data
1. Classify Reviews as "Negative Reviews" vs. "Non-negative Reviews" (Section 5.1.1)
RNN classifier 1
Execute RNN classifier 1 Python code (Tensorflow) and identify negative reviews from non-negative reviews
[automated]
negative reviews ★
non-negative reviews
*Negative reviews are defined as those with 1 or 2-star customer ratings
2. Classify Negative Reviews as "Reviews with Defect Information" vs. "Reviews with No Defect Information" (Section 5.1.2)
RNN classifier 2
Execute RNN classifier 2 Python (Tensorflow) code and identify reviews with defect infomation from reviews with no defect infomation*
[automated]
Reviews with defect infomation★
Reviews with no defect infomation
*Reivews with defects information are defined as the reviews that are labeled as "Manufacturing defect," "Problematic design or quality,"or "Bad customer service."
3. Identify key information of defective products from LDA model (Section 5.1.3)
Identify key defect information from LDA
Execute Python LDA code and analyze defect infomration from keywords of top 5 LDA output topics
[automated]
Main output
Sampled raw LDA ouptput topics for a product:
Sampled Defect Insight

```
In [38]: prediction_input = tokenizer.texts_to_sequences(["this thing is not good and i do not like it"])
         prediction_input = pad_sequences(prediction_input, padding='post', maxlen=maxlen)
         prediction_output = model.predict(prediction_input)
         if (prediction_output > 0.5):
             print("Non negative reviews")
         else:
             print("Negative reviews")

         Negative reviews
```

```
prediction_input = tokenizer.texts_to_sequences([" the 2 corners were dented. The bottom part did not fit properly
prediction_input = pad_sequences(prediction_input, padding='post', maxlen=maxlen)
prediction_output = model.predict(prediction_input)
if (prediction_output < 0.5):
    print("Review mentioned defect ")
else:
    print("Review did not mention defect")

Review mentioned defect
```

```
0.38386346825612055

lda_train4.print_topics(20,num_words=15)[:10]

[(0,
  '0.102*"stool" + 0.072*"use" + 0.056*"bought" + 0.034*"small" + 0.030*"year" + 0.029*"get" + 0.027*"order" + 0.027*"realli" +
0.025*"two" + 0.024*"side" + 0.024*"store" + 0.023*"crack" + 0.022*"easili" + 0.021*"broke" + 0.021*"first"'),
 (1,
  '0.119*"lb" + 0.110*"product" + 0.082*"hold" + 0.066*"weight" + 0.055*"stand" + 0.038*"feel" + 0.036*"flimsi" + 0.034*"stool"
+ 0.033*"like" + 0.031*"good" + 0.030*"step" + 0.030*"one" + 0.025*"say" + 0.022*"easi" + 0.021*"weigh"'),
 (2,
  '0.084*"work" + 0.073*"stool" + 0.065*"use" + 0.061*"get" + 0.048*"handl" + 0.045*"broken" + 0.045*"sturdi" + 0.043*"fold" +
0.033*"sit" + 0.028*"height" + 0.027*"like" + 0.027*"need" + 0.027*"one" + 0.026*"thing" + 0.026*"got"'),
 (3,
  '0.089*"use" + 0.065*"step" + 0.062*"one" + 0.052*"stool" + 0.031*"fold" + 0.030*"would" + 0.029*"high" + 0.026*"plastic" +
0.026*"broke" + 0.024*"stand" + 0.023*"feet" + 0.023*"get" + 0.022*"away" + 0.021*"put" + 0.019*"time"'),
```

•('0.025*"assemble" + 0.022*"back" + 0.018*"wheel" + 0.019*"one" + 0.015*"star" + 0.014*"part" + 0.012*"sit" + 0.012*"came" + 0.012*"floor" + 0.012*"fall"),
•('0.049*"sit" + 0.038*"seat" + 0.036*"back" +0.033*"uncomfortable" + 0.019*"price" + 0.018*"make" + 0.017*"hip" + 0.017*"cushion" + 0.017*"arm" + 0.017*"leg")
•('0.045*"came" + 0.042*"damage" + 0.028*"box" +0.027*"receive" + 0.022*"item" + 0.020*"return" + 0.019*"broken" + 0.019*"pieces" + 0.019*"shelf" + 0.019*"chip")
•('0.053*"screw" + 0.043*"top" + 0.041*"pieces" + 0.039*"hold" + 0.030*"bottom" + 0.025*"part" + 0.022*"dowel" + 0.021*"drill" + 0.021*"put_together" + 0.017*"broekn")

Topic 1	Topic 2	Topic 3	Topic 4
assemble	uncomfortable	damage	screw
wheel	cushion	receive	hold
floor	hip	box	drill
fall	arm	broken	put_together

FIGURE 5 | End-to-end process.

METHODS

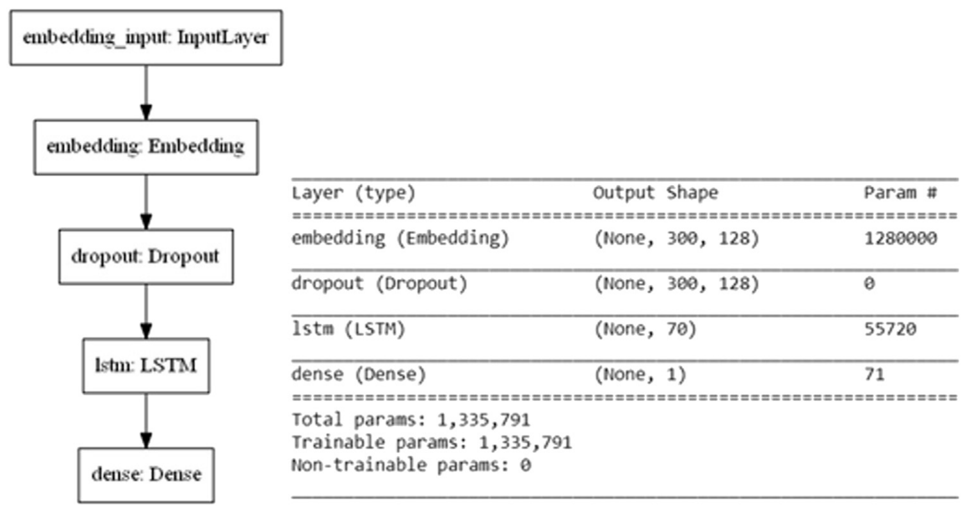
Model Development and Testing Procedures

The goal of this study is to develop and evaluate how the recurrent neural network (RNN) classifiers and the Latent Dirichlet Allocation (LDA) topic model can extract product defect information from online customer reviews (OCRs). After preprocessing the data, there were three targeted statistical models used in the research in order to accomplish the thesis statement of providing engineers with an early view of product defects. They include two quantitative RNN classifiers and one Latent Dirichlet Allocation (LDA) topic model. The first RNN classifier categorizes negative OCRs from non-negative OCRs. The second RNN classifier categorizes OCRs that mention defects from those that do not. The LDA topic model provides engineers a view on groups of word items or topics that best describe the type of defects found in a single product.

Figure 5, shows the end-to-end process as to how OCRs from a single product is processed to extract product defect insight using the fully trained and built models. The user of the model would first select the specific product and do data preparation on the dataset as described in *Methods*. After the data is prepared, the data would then go though the three models as described in the following passage in *Results*.

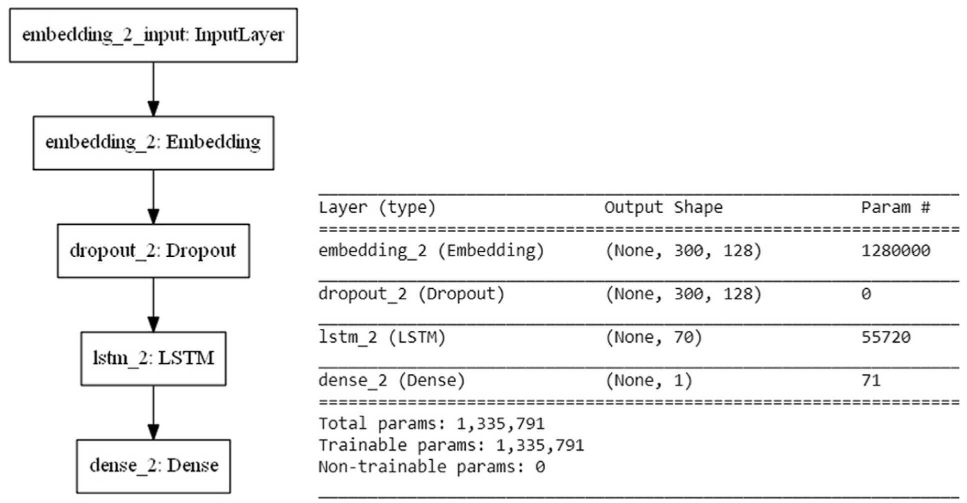
Model 1: Recurrent Neural Network Classifier for Classifying Negative Revies From Non-Negative Reviews

OCRs with negative sentiment has a much higher chance to include complaints and defective issues about products as



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 300, 128)          1280000
_________________________________________________________________
dropout (Dropout)            (None, 300, 128)          0
_________________________________________________________________
lstm (LSTM)                  (None, 70)                55720
_________________________________________________________________
dense (Dense)                (None, 1)                 71
=================================================================
Total params: 1,335,791
Trainable params: 1,335,791
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 6 | RNN model architecture and params for negative and non-negative OCRs classification.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 300, 128)          1280000
_________________________________________________________________
dropout_2 (Dropout)          (None, 300, 128)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 70)                55720
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 71
=================================================================
Total params: 1,335,791
Trainable params: 1,335,791
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 7 | RNN model architecture and params for classifying OCRs that mention product defects vs. do not mention product defects.

compared to OCRs with a non-negative sentiment. A classifier built to distinguish negative OCRs from non-negative OCRs is useful for giving engineers insight into the evidence of a product defect. This RNN model contains five layers, as shown in **Figure 6**.

The first layer is the input layer that takes input vectors. The second layer is for word embedding, as was described in *Text Representation*. The third layer is a dropout layer. The dropout layer randomly sets the neuron's output to 0 during each iteration of training to avoid overfitting. The fourth layer is an LSTM layer with 128 LSTM neurons chained in sequence for recurrently processing information during the training step. After the information is processed with the chained 128 LSTM units, information is then sent to the output layer for classification. The output layer uses one Sigmoid to separate the output into two classes, either near 1 or near 0. This identifies the OCRs as either negative OCRs or non-negative OCRs. This model uses the Adaptive Moment Estimation (Adam) optimizer for backpropagation to update the hidden LSTM layer with a learning rate of 0.01. The whole dataset it consists of 9,000 OCRs with one-third non-negative OCRs and two-thirds negative OCRs.

The model is then trained with 8,100 OCRs, which is 90% of the total selected data with 50 epochs and a batch size of 32 OCRs in each batch. A successfully trained RNN model is able to automatically identify negative and non-negative OCRs with no given label. This allows engineers to identify negative

OCRs from online forums or markets that only contain text reviews but no ratings. To test the accuracy of the RNN model on classifying negative OCRs from non-negative OCRs, a set of 900 OCRs are separated for testing, which is 10% of the total selected data, with one-third non-negative OCRs and two-thirds negative OCRs. The RNN would then predict the labels of the testing data and compare it against the actual label.

Model 2: Recurrent Neural Network Classifier for Classifying Reviews That Mention Product Defects From Reviews That do not Mention Product Defects

To further investigate which negative OCRs contain product defect or issue information, a second RNN classifier is built to identify OCRs that provide defective product information from OCRs that do not. This RNN model is similar to the first one that contains five layers, as shown in **Figure 7**.

Some negative OCRs do not contain defective product information. For example, review id #R2F8RCR0LFI7SS states: "We bought four of these, they are just some real cheap chairs that are overpriced." This OCR only mentions the product is overpriced with no other information about the product issue. The second example is review id #R3W0KKHC5LK7K5 with the review title as "One Star," and a two-word review in the text body as "Pure junk." These do not give engineers any information about the product itself. The RNN model is trained with the two labels: "Defect information provided" and "No defect information provided". The OCRs that are manually labeled as "Manufacturing defect," "Problematic design or quality," or "Bad customer service" are considered with defective product information provided. Otherwise, OCRs with human labels of "No defect information" is considered as no defect information provided.

A successfully-trained RNN model is able to automatically identify OCRs that provide defect information from OCRs that provide no defect information without a given label. This allows engineers to identify and filter out OCRs that are of value for their engineering design and product correction. To test the accuracy of the RNN model on classifying OCRs that mention product defects from OCRs that do not mention product defects, a set of 900 negative OCRs, which is 10% of the total, was used for testing. The RNN would then predict the labels of the testing data and compare it against the actual label.

Model 3: Latent Dirichlet Allocation Topic Model for Providing Product Defect Insight

The Latent Dirichlet Allocation (LDA) topic model is able to automatically offer engineers a view on groups of words, items, or topics that best describe OCRs with defect information. The LDA topic model builds a probabilistic text model by viewing a document, or OCR, as a mixture of topics, each with its distribution of topics [47]. This allows manufacturers to have an early view of where the problem is.

The LDA topic model assumes that OCRs are represented as random mixtures over k latent topics, where each topic is characterized by a distribution over words [31] After a large number of the iterative training process of reassigning words to topics according to the multinomial distribution, the model is converged with the K numbers of topics each with word distribution that best describe the set of OCRs. The words that make up the word distribution are able to tell the information about the topics among the group of selected OCRs. This allows engineers to have an early view on the OCRs that provided defective information, before manually looking at each one of them.

While LDA models are able to give a view of a list of topics that a group of OCRs is mentioning, for its unsupervised learning property, there is no actual label for the model to test against. The most common evaluation metric that is used on LDA models is topic coherence, which measures semantic similarity among the top words that appear in the word distribution for a single topic. This method might not include the actual meaning of the words that tie back to the defective information. Chang et al. suggested these traditional topic coherence metrics are negatively correlated with the measures of topic quality developed, and they agree that human judgments and manual determinations remain a better way to determine if the LDA model is giving out informative topics among a set of documents or OCRs [48].

To test out this LDA approach along with RNN models that filter out OCRs that mention defective product information, a set of OCRs of the eight top-selling home and furniture products on Amazon were selected for testing. This process developed eight test cases. The OCRs from each product were first filtered using the first RNN model for selecting negative OCRs and then filtered using the second RNN model for selecting OCRs that mention defective product information. These OCRs were then inputted into LDA for discovering essential topic information on defective product information. The 10 highest-scoring words that build upon the five LDA topics were used for testing purposes. Amazon SageMaker labelers were then asked to identify if these topics, or the group of words, are related back to the defectiveness of a product or the details of a product itself. This method verifies whether the output topic words are linked back to the product or the defect itself.

RESULTS

This section provides comprehensive results of the three presented models conducted during this study to give engineers an early view of product defects and issues. The first two models are recurrent neural network (RNN) classifiers that categorize OCRs that contain product defect information. This section also demonstrates the results of these models against the human labels of both the model training and the model testing stages. The last model is the Latent Dirichlet Allocation (LDA) topic model that can extract product defect information from OCRs. A total of eight test cases for the LDA model using the eight bestselling products on the Amazon.com home furniture section was used. This section presents the output of the LDA model along with its relevance to the defective product information.

TABLE 2 | Prediction statistics and hypothesis test of for RNN model classifier for classifying negative OCRs from non-negative OCRs.

Prediction Statistics			
True Positives	509	Testing accuracy	0.8500
False Positives	48	Testing precision	0.9156
True Negative	256	Testing F1 score	0.8851
False Negative	87	Testing recall	0.8593
		One sample Z-tests for a proportion H_0: $p = 0.7$ H_1: $p > 0.7$	
Sample proportion	0.850000	95% confidence interval for proportion	(0.826672, 0.873328)
Z-value	9,82	*p*-value	0.000

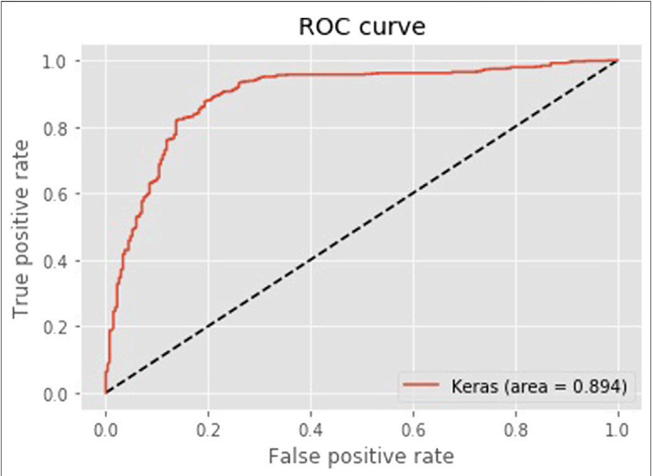


FIGURE 8 | ROC curve for RNN model for classifying negative OCRs from non-negative OCRs.

Model 1: Recurrent Neural Network Classifier for Classifying Negative Reviews From Non-Negative Reviews

The first model is the RNN classifier for sorting negative OCRs from non-negative OCRs. Negative OCRs are defined as those OCRs with 1 or 2-star customer ratings. Non-negative OCRs are those defined as OCRs with three or more stars customer ratings. The model was trained with 8,100 OCRs and was validated with 900 OCRs, with one-third non-negative OCRs and two-thirds negative OCRs. The 8,100 OCRs were the training dataset reserved for training the RNN model. The 900 OCRs, which had never been trained by the model, were the testing dataset. The model was executed for 50 epochs, with a batch size of 32 OCRs in each batch, at a learning rate of 0.01.

This test evaluates the performance of the RNN Model for classifying negative reviews from non-negative reviews. The test data that was not trained by the model was predicted after the 50 epochs compared with the output against the actual label. **Table 2** shows the prediction metrics and hypothesis test of this model. This table shows the hypothesis test of the one sample Z-tests for a proportion at alpha 0.05 with 70% accuracy. **Figure 8** shows the ROC curve of this model.

The model correctly predicted 765 OCRs out of 900 OCRs. This was tested with a one-sample Z-test for a proportion to evaluate if the model has an accuracy of 70% accuracy. With the area under a ROC Curve at 0.930, the model has good predictive power. With the *p*-value at 0, the hypothesis test successfully rejected the null hypothesis, and thus this RNN classifier is sufficient in classifying negative reviews from non-negative reviews.

Model 2: Recurrent Neural Network Classifier for Classifying Reviews that Mention Product Defects From Reviews that do not Mention Product Defects

The second model is the RNN classifier for categorizing OCRs that mention product defects from OCRs that do not mention product defects. OCRs that mention product defects are defined as the OCRs that are manually labeled as "Manufacturing defect," "Problematic design or quality," or "Bad customer service." OCRs that do not mention product defects are defined as the OCRs that are manually labeled as "No defect information". This model was trained with 8,100 OCRs and was validated with 900 OCRs. The training data is the data used for training and fitting to the RNN model. The testing data is the data used for testing the model, which was not adjusted by the model. This model was executed for 50 epochs, with a batch size of 32 OCRs in each batch, at a learning rate of 0.01.

This test evaluates the performance of the RNN classifier for classifying OCRs that mention product defects from OCRs that do not mention product defects. The test data that was not fitted by the model was predicted after the 50 epochs and results were compared against the actual label. **Table 3** shows the prediction metrics and hypothesis test of this model. This table shows the hypothesis test of the one sample Z-tests for a proportion at alpha 0.05 with 70% accuracy. **Figure 9**shows the ROC curve of this model.

The model correctly predicted 779 OCRs out of 900 OCRs. The predictive power of this model is good with the area under a ROC Curve at 0.894. The hypothesis test was tested with one-sample Z-tests for a proportion to evaluate if the model has an accuracy of 70%. With the *p*-value at 0, the hypothesis test successfully rejected the null hypothesis, and thus this RNN classifier is sufficient in classifying reviews that mention product defects from reviews that do not mention product defects.

TABLE 3 | Prediction metrics and hypothesis test of RNN model for classifying OCRs that mention product defects from OCRs that do not mention product defects.

Prediction Statistics			
True positives	579	Testing accuracy	0.8659
False positives	69	Testing precision	0.8985
True negative	200	Testing F1 score	0.9208
False negative	52	Testing recall	0.9208
		One sample Z-tests for a proportion H_0: $p = 0.7$ H_1: $p > 0.7$	
Sample proportion	0.865556	95% confidence interval for proportion	(0.843269, 0.887842)
Z-value	10.84	*p*-value	0.000

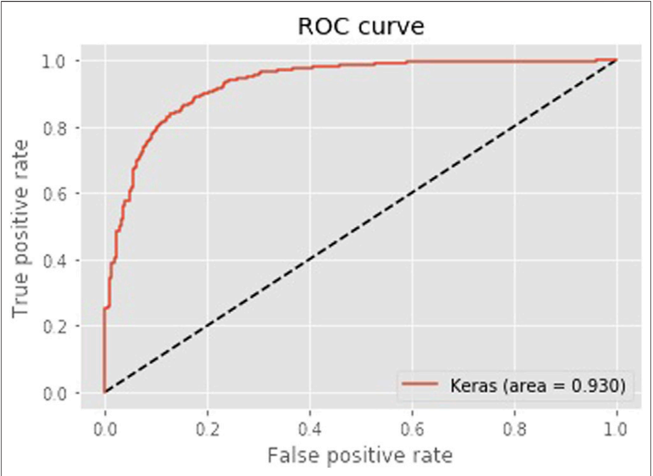


FIGURE 9 | ROC curve for RNN model for classifying OCRs that mention product defects from OCRs that do not mention product defects.

Model 3: Latent Dirichlet Allocation Topic Model for Providing Product Defect Insight

The LDA topic model was used in this study to identify groups of words, or topics, that best describe OCRs that have already been identified as negative and embedded with defect information by previous models. The selection provides the aggregated LDA topic modeling result of the top eight furniture products on Amazon.com as of 12th April 2020. Each product was provided with five topics with 10 words each. Those groups of words were validated by Amazon SageMaker human labelers to verify their relevance to the product. Amazon SageMaker human labelers were asked to evaluate if at least half of the group of words retrieved by the LDA topics are relevant to detail, buying process, usage, or defect of a furniture product.

Example Test Product: Linenspa 2 Inch Gel Infused Memory Foam Mattress Topper, Twin

This is an example of one of the eights that were used in this research. Linenspa 2 Inch Gel Infused Memory Foam Mattress Topper is a polyurethane memory foam mattress topper that adds softness to the top of mattresses to enhance the sleeping experience. This product received 12,684-star ratings at an average of 4.7 out of five stars and 7,202 valid OCRs. The RNN classifier for classifying negative OCRs from non-negative OCRs identified 1,658 negative OCRs among the total valid OCRs. The second RNN classifier for classifying reviews that mention product defects from reviews that do not mention product defects identified 1,363 OCRs that mention product defects.

The LDA topic model was used to identify the five topics with 10 words each that best describe the OCRs which mention product defects and generated a topic Coherence Value (CV) of 0.4930. The following **Table 4** shows the topics, their supporting words, and their support weight. The topics were also given to Amazon SageMaker human labelers for their relevance to a furniture product or a product defect, along with its corresponding confidence level.

Amazon SageMaker human judgment indicated five out of five topics show words related to furniture products or product issues. The five topics summarized the terms used in OCRs that mention defects. Words that support topic number five might indicate a body support problem with the topper being too soft and the foam sinks. The following list shows three sampled OCRs that indicate body support problems using the word search function with the word "support" in the data.

> "Six months later the foam is squashed and offers no support at all. She weighs less than 110 lbs. I will not buy this pad again." (review ID # R2MV2APVP8P9CX)
>
> "This topper goes completely flat once you lay on it and offers zero support." (review ID # RYTH07FT8W7XS)
>
> "This thing provided no support, it crushes down to nothing anywhere there is the weight (I'm 5′9" 155 lbs … Shouldn't be an issue)" (review ID # RRPFIQ56HYMVU)

A total of eight test product was evaluated with a one-sample sign test in this research with a total of 32,301 ORCs ran through this model. Amazon SageMaker human labelers were asked to evaluate if at least half of the group of words retrieved by the LDA topics are relevant to detail, buying process, usage, or defect of a furniture product. Since there are eight test cases demonstrated in this paper, one sample sign test was used for the hypothesis test. **Table 5** shows the number of relevant topics generated by each of

TABLE 4 | Topic model words for LINENSPA Mattress topper.

Topic	Words supporting the topic (support weight)						Is this topic relevant to the product or a defect? (Confidence)
	Words	Weight	Words	Weight	Words	Weight	
1	Sleep	0.029	Pain	0.028	Bed	0.027	Yes (0.95)
	Pad	0.027	Night	0.021	Use	0.016	
	Topper	0.014	Back_ Pain	0.014	Hip	0.013	
	Help	0.013					
2	Topper	0.035	Sleep	0.027	Bed	0.023	Yes (0.95)
	Night	0.017	Get	0.016	Cool	0.016	
	Hot	0.014	Purchase	0.013	Bought	0.012	
	memory_foam	0.012					
3	Bed	0.035	Topper	0.026	Smell	0.023	Yes (0.94)
	Hour	0.021	Like	0.018	inch	0.017	
	Size	0.016	Open	0.016	Air	0.016	
	Order	0.015					
4	Back	0.036	Return	0.027	Box	0.022	Yes (0.95)
	Topper	0.020	Would	0.019	One	0.019	
	Try	0.015	Review	0.011	Amazon	0.010	
	Disappoint	0.010					
5	Soft	0.036	Topper	0.026	Bed	0.022	Yes (0.95)
	Sink	0.020	Support	0.020	Like	0.019	
	Foam	0.018	Firm	0.014	Really	0.014	
	Body	0.014					

TABLE 5 | Summary of test cases.

Summary of test cases			
Test cases	Relevance (Human Label)	Topic CV	Average confidence
Test case one	5 out of 5 topics are related to a furniture product	0.4930	0.948
Test case two	4 out of 5 topics are related to a furniture product	0.3583	0.950
Test case three	5 out of 5 topics are related to a furniture product	0.5162	0.948
Test case four	5 out of 5 topics are related to a furniture product	0.4256	0.950
Test case five	5 out of 5 topics are related to a furniture product	0.3959	0.944
Test case six	3 out of 5 topics are related to a furniture product	0.3503	0.946
Test case seven	4 out of 5 topics are related to a furniture product	0.3698	0.930
Test case eight	5 out of 5 topics are related to a furniture product	0.3942	0.950

TABLE 6 | Hypothesis test for LDA topic model.

One sample sign test for median			
H_0: $\eta = 0.7$			
H_1: $\eta > 0.7$			
Sample (N)	8	Median	5
95% confidence interval for proportion	(3.936, 5)	*p*-value	0.035

the eight products. The following **Table 6** shows the result of the one-sample sign test for the median.

The test products have a mean of 4.375 out of five topics and a median of five out of five topics evaluated as relevant to a furniture product or problematic product information. They generated a mean topic CV of 0.4129 with all of them more than 0.3, which indicates the words in topics are somewhat coherent. Some test products generated a topic CV of higher than 0.5, which indicates that they have a good coherence between topics. This was tested with a one-sample sign test to evaluate if the model can retrieve 70% of the topic relevant to a furniture product or problematic product information. With the *p*-value at 0.035, the hypothesis test successfully rejected the null hypothesis, and thus, the LDA model can successfully retrieve key product defect topics.

CONCLUSION

This research explored the best method to provide insurers and manufacturers with an early view into product defects. While there is a large number of OCRs available on social media and e-commerce sites, it is difficult for insurers and manufacturers to manually inspect those OCRs for defective information, which will delay product recall. This research has demonstrated a novel predictive model using RNN and LDA topic models to extract product defects from OCRs in a fast and automatic way. As the time to recall action increases during a product defect event, the recovery will be more challenging [17]. This new predictive model provides them with an early risk analysis on defective products and defect detection mechanisms.

In summary, this research has proposed and constructed two RNN classifiers and one LDA topic to determine if these models can retrieve product defect information. The first RNN model was able to identify negative OCRs, where most of these OCRs consist of complaints and issues about the products. The second RNN model was able to identify if the OCRs consist of one of the following information labels about the products' defect: "Manufacturing defect," "Problematic design or quality," and "Bad customer service." The Latent Dirichlet Allocation (LDA) model successfully combined the first and the second RNN models to retrieve key defect information on OCRs that were negative and mentioned product defect. This combined model was able to locate the keywords of the problems and issues that customers mentioned the most in their OCRs, which in turn achieved the goal of providing the insurers and manufacturers with an early view into potential product defect events.

DISCUSSION AND FUTURE WORK

The purpose of this research was to investigate the possibility of using probabilistic models for online customer reviews to retrieve product defect information and provide insurers and manufacturers with insight into the defects. Other than the two RNN models discussed earlier, this research also attempted to build a third RNN model to identify what kind of defect type ("Manufacturing defect," "Problematic design or quality," and "Bad customer service.") was mentioned in the OCRs. Due to the similarity of the use of words among the OCRs mentioning product defects, the RNN model was not able to classify OCRs down to the defect type. It was overfitted with the classification type that has the heaviest weight. The assumption is that the root cause is because the words used in all three classes of labels were so similar that the model was unable to separate the probability space. The initial assumption was that the model could identify the difference among these words, but that was proven not to be the case. This limitation with this RNN approach provided an excellent space to use the LDA model. The LDA model, combined with the first and the second RNN models, successfully retrieved key information on OCRs that are negative and mention product defects.

Future research may consider improving and refining the accuracy of both the RNN classifiers and the LDA topic model and addressing the main outstanding problem identified in this research: the RNN classifier model for identifying defect types. RNN classifiers may be improved by adding multiple LSTM layers to construct a deeper neural network architecture. The accuracy may be improved this way, although the training time may increase. RNN classifiers may also be improved by training on a larger supervised dataset, while this would involve more manual labor work. The other neural network architectures such as Convolutional Neural Network layers can also be explored in future research for increasing the performance of identifying defective products. The deficiency for the RNN classifier that was attempted to identify defect types could be solved by constructing term frequency word lists for each defect type. Abrahams et al. have developed a set of smoke words, specifically for auto defect and auto safety issues, to identify whether OCRs mentioned defect or safety issues [24]. This method may be harder for particular classifiers to differentiate "Manufacturing defect" from "Problematic design or quality," since they have a very similar use of words in the OCRs.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

TF was responsible for running all the initial set of experiments and dataset preparation. SS and JF were the technical advisors for driving this project to completion.

REFERENCES

1. Ahsan K. Trend Analysis of Car Recalls: Evidence from the US Market. *Ijmvsc* (2013). 4(4):1–16. doi:10.5121/ijmvsc.2013.4401
2. Jeong E-Y. Samsung's Conclusions on Galaxy Note 7 Recall Backed up by Regulator. *Wall Street J* (2017). Retrieved from https://www.wsj.com/articles/samsungs-conclusions-on-galaxy-note-7-recall-backed-up-by-regulator-1486348636 (Accessed February, 2020).
3. Allianz Global Corporate & Specialty SE. AGCS Product Recall Report (2017). Retrieved from https://www.agcs.allianz.com/content/dam/onemarketing/agcs/agcs/reports/AGCS-Product-Recall-Report.pdf (Accessed March, 2020).
4. Mack R. San Francisco: CNET (2016). Another Samsung Galaxy Note 7 up in Smoke as Users Ignore Recalls. Retrieved from https://www.cnet.com/news/another-samsung-galaxy-note-7-up-in-smoke-recalled-phone-not-put-down-data-shows/ (Accessed February, 2020).
5. Tibken S, and Cheng R. Samsung Answers Burning Note 7 Questions, Vows Better Batteries. San Francisco: CNET (2017). Retrieved from https://www.cnet.com/news/samsung-answers-burning-note-7-questions-vows-better-batteries/ (Accessed March, 2020).
6. Bleaney G, Kuzyk M, Man J, Mayanloo H, and Tizhoosh HR. Auto-detection of Safety Issues in Baby Products. *Lecture Notes Comp Sci Recent Trends Future Tech Appl Intelligence* (2018). 505–16. doi:10.1007/978-3-319-92058-0_49
7. Lee T, and Bradlow E. Automated Marketing Research Using Online Customer Reviews. *SSRN J* (2010) 48:881–894. doi:10.2139/ssrn.1726055
8. Suryadi D, and Kim HM. A Data-Driven Approach to Product Usage Context Identification from Online Customer Reviews. *J Mech Des* (2019). 141(12): 121104. doi:10.1115/1.4044523

9. CPSC. *U.S. Consumer Product Safety Commission*. Bethesda, MD: About CPSC (2019). Retrieved from https://www.cpsc.gov/About-CPSC (Accessed February, 2020).
10. Pruitt SW, and Peterson DR. Security price Reactions Around Product Recall Announcements. *J Financial Res* (1986). 9(2):113–22. doi:10.1111/j.1475-6803.1986.tb00441.x
11. Jarrell G, and Peltzman S. The Impact of Product Recalls on the Wealth of Sellers. *J Polit Economy* (1985). 93(3):512–36. doi:10.1086/261313
12. Dawar N, and Pillutla MM. Impact of Product-Harm Crises Onbrand Equity: The Moderating Role of Consumer Expectations'. *Marketing Res* (2000). 37(May):215–26. doi:10.1509/jmkr.37.2.215.18729
13. Matos CA, and Rossi CA. Consumer Reaction to Product Recalls: Factors Influencing Product Judgement and Behavioral Intentions. *Int J Consumer Stud* (2007). 31(1). doi:10.1111/j.1470-6431.2006.00499.x
14. Lyles MA, Flynn BB, and Frohlich MT. All Supply Chains Don't Flow through: Understanding Supply Chain Issues in Product Recalls. *Manag Organ Rev* (2008). 4(2):167–82. doi:10.1111/j.1740-8784.2008.00106.x
15. Beamish PW, and Bapuji H. Toy Recalls and China: Emotion vs. Evidence. *Manag Organ Rev* (2008). 4(2):197–209. doi:10.1111/j.1740-8784.2008.00105.x
16. Berman B. Planning for the Inevitable Product Recall. *Business Horizons* (1999). 42(2):69–78. doi:10.1016/s0007-6813(99)80011-1
17. Hora M, Bapuji H, and Roth AV. Safety hazard and Time to Recall: The Role of Recall Strategy, Product Defect Type, and Supply Chain Player in the U.S. Toy Industry. *J Operations Manag* (2011). 29(7-8):766–77. doi:10.1016/j.jom.2011.06.006
18. Siomkos GJ, and Kurzbard G. The Hidden Crisis in Product-harm Crisis Management. *Eur J Marketing* (1994). 28(2):30–41. doi:10.1108/03090569410055265
19. Souiden N, and Pons F. Product Recall Crisis Management: the Impact on Manufacturer's Image, Consumer Loyalty and purchase Intention. *Jnl Prod Brand Mgt* (2009). 18(2):106–14. doi:10.1108/10610420910949004
20. Karakaya F, and Ganim Barnes N. Impact of Online Reviews of Customer Care Experience on Brand or Company Selection. *J Consumer Marketing* (2010). 27(5):447–57. doi:10.1108/07363761011063349
21. Dellarocas C, Zhang X, and Awad NF. Exploring the Value of Online Product Reviews in Forecasting Sales: The Case of Motion Pictures. *J Interactive Marketing* (2007). 21(4):23–45. doi:10.1002/dir.20087
22. Barton B. Ratings, Reviews & ROI. *J Interactive Advertising* (2006). 7(1):5–50. doi:10.1080/15252019.2006.10722125
23. Chan NL, and Guillet BD. Investigation of Social Media Marketing: How Does the Hotel Industry in Hong Kong Perform in Marketing on Social Media Websites?. *J Trav Tourism Marketing* (2011). 28(4):345–68. doi:10.1080/10548408.2011.571571
24. Abrahams AS, Jiao J, Wang GA, and Fan W. Vehicle Defect Discovery from Social media. *Decis Support Syst* (2012). 54(1):87–97. doi:10.1016/j.dss.2012.04.005
25. Zhang X, Qiao Z, Ahuja A, Fan W, Fox EA, and Reddy CK. Discovering Product Defects and Solutions from Online User Generated Contents. In: The World Wide Web Conference on - WWW 19 Editors L. Liu and R. White (2019), San Francisco, CA, United States, 13 May, 2019–17 May, 2019, (New York: . ACM). doi:10.1145/3308558.3313732
26. El-Dehaibi N, Goodman ND, and Macdonald EF. Extracting Customer Perceptions of Product Sustainability from Online Reviews. *J. Mech. Des.* (2019). 141:121103. doi:10.1115/detc2019-98233
27. Rezaeinia SM, Rahmani R, Ghodsi A, and Veisi H. Sentiment Analysis Based on Improved Pre-trained Word Embeddings. *Expert Syst Appl* (2019). 117: 139–47. doi:10.1016/j.eswa.2018.08.044
28. Mikolov T, Sutskever I, Chen K, Corrado G, and Dean J. Distributed Representations of Words and Phrases and Their Compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Red Hook, NY, USA: Curran Associates Inc. (2013). 2. p. 3111–9. doi:10.5555/2999792.2999959
29. Ren Y, Wang R, and Ji D. A Topic-Enhanced Word Embedding for Twitter Sentiment Classification. *Inf Sci* (2016). 369:188–98. doi:10.1016/j.ins.2016.06.040
30. Tang D, Wei F, Yang N, Zhou M, Liu T, and Qin B. Learning Sentiment-specific Word Embedding for Twitter Sentiment Classification. *Proc 52nd Annu Meet Assoc Comput Linguistics* (2014). 1:1014–1023. doi:10.3115/v1/p14-1146
31. Blei DM, Ng AY, and Jordan MI. Latent Dirichlet Allocation. *J Machine Learn Res* (2003). 3:993–1022. doi:10.1162/jmlr.2003.3.4-5.993
32. Pritchard J, Stephens M, and Donnelly P. *Inference of Population Structure Using Multilocus Genotype Data*. Rockville, MD: Faculty Opinions – Post-Publication Peer Review of the Biomedical Literature (2003). doi:10.3410/f.1015548.197423
33. Santosh DT, Babu KS, Prasad S, and Vivekananda A. Opinion Mining of Online Product Reviews from Traditional LDA Topic Clusters Using Feature Ontology Tree and Sentiwordnet. *Int J Edu Manag Eng* (2016). 6(6):34–44. doi:10.5815/ijeme.2016.06.04
34. Zhai CX, and Massung S. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. New York, NY: ACM Books (2016). doi:10.1145/2915031
35. Titov I, and Ryan M. *A Joint Model of Text and Aspect Ratings for Sentiment Summarization*. ACL/HLT (2008).
36. Calheiros AC, Moro S, and Rita P. Sentiment Classification of Consumer-Generated Online Reviews Using Topic Modeling. *J Hospitality Marketing Manag* (2017). 26(7):675–93. doi:10.1080/19368623.2017.1310075
37. Han J, Kamber M, and Pei J. *Data Mining: Concepts and Techniques*. Waltham, MA: Morgan Kaufmann (2012). p. 398.
38. Liu P, Qiu X, and Huang X. Recurrent Neural Network for Text Classification with Multi-Task Learning. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence IJCAI'16 Editors S. Kambhampati, New York, United States, 9–15 July 2016. (Palo Alto, CA, United States: . AAAI Press) (2016). p. 2873–9. doi:10.5555/3060832.3061023
39. Rumelhart DE, Hinton GE, and Williams RJ. Learning Representations by Back-Propagating Errors. *Nature* (1986). 323(6088):533–6. doi:10.1038/323533a0
40. Pearlmutter. Learning State Space Trajectories in Recurrent Neural Networks. In: International Joint Conference on Neural Networks Editors S.-I. Amari, Washington, DC, Unite States, June 18–22, 1989, (Washington, DC: IEEE) (1989). doi:10.1109/ijcnn.1989.118724
41. Bengio Y, Simard P, and Frasconi P. Learning Long-Term Dependencies with Gradient Descent Is Difficult. *IEEE Trans Neural Netw* (1994). 5(2):157–66. doi:10.1109/72.279181
42. Hochreiter S. The Vanishing Gradient Problem during Learning Recurrent Neural Nets and Problem Solutions. *Int J Unc Fuzz Knowl Based Syst* (1998). 06(02):107–16. doi:10.1142/s0218488598000094
43. Graves A. Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks. *Guide to OCR for Arabic Scripts* (2012). 297–313. doi:10.1007/978-1-4471-4072-6_12
44. Tang J, Li H, Cao Y, and Tang Z. Email Data Cleaning. In: Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining - KDD 05 Editors R. Grossman, Chicago, IL, United States, 21 August, 2005–24 August, 2005, (New York: Springer) (2005). doi:10.1145/1081870.1081926
45. Li J, Luong T, Jurafsky D, and Hovy E. When Are Tree Structures Necessary for Deep Learning of Representations?. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015). doi:10.18653/v1/d15-1278
46. Agarwal S, Godbole S, Punjani D, and Roy S. How Much Noise Is Too Much: A Study in Automatic Text Classification. In: Seventh IEEE International Conference on Data Mining Editors N. Ramakrishnan, O. R. Zaïane, Y. Shi, C. W. Clifton, and X. Wu, Omaha, NE, United States, 28–31 Oct 2007. (Washington, DC: . ICDM 2007) (2007). doi:10.1109/icdm.2007.21
47. Russell SJ, and Norvig P. *Artificial Intelligence: A Modern Approach*. Hoboken: Pearson (2011).
48. Chang J, Boyd-Graber J, Wang J, Gerrish S, and Blei DM. Reading tea Leaves: How Humans Interpret Topic Models. Neural Information Processing Systems (2009). Available from: https://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.pdf (Accessed May, 2020). doi:10.1145/1557019.1557044

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.

BRIEF RESEARCH REPORT
published: 02 August 2021
doi: 10.3389/fams.2021.649501

Using Machine Learning to Analyze Merger Activity

Tiffany Jiang[1,2]*

[1]Department of Economics, Stanford University, Stanford, CA, United States, [2]Department of Economics, University of California, Davis, Davis, CA, United States

An unprecedented amount of access to data, "big data (or high dimensional data)," cloud computing, and innovative technology have increased applications of artificial intelligence in finance and numerous other industries. Machine learning is used in process automation, security, underwriting and credit scoring, algorithmic trading and robo-advisory. In fact, machine learning AI applications are purported to save banks an estimated $447 billion by 2023. Given the advantages that AI brings to finance, we focused on applying supervised machine learning to an investment problem. 10-K SEC filings are routinely used by investors to determine the worth and status of a company–Warren Buffett is frequently cited to read a 10-K a day. We sought to answer–"Can machine learning analyze more than thousands of companies and spot patterns? Can machine learning automate the process of human analysis in predicting whether a company is fit to merge? Can machine learning spot something that humans cannot?" In the advent of rising antitrust discussion of growing market concentrations and the concern for decrease in competition, we analyzed merger activity using text as a data set. Merger activity has been traditionally hard to predict in the past. We took advantage of the large amount of publicly available filings through the Securities Exchange Commission that give a comprehensive summary of a company, and used text, and an innovative way to analyze a company. In order to verify existing theory and measure harder to observe variables, we look to use a text document and examined a firm's 10-K SEC filing. To minimize over-fitting, the L2 LASSO regularization technique is used. We came up with a model that has 85% accuracy compared to a 35% accuracy using the "bag-of-words" method to predict a company's likelihood of merging from words alone on the same period's test data set. These steps are the beginnings of tackling more complicated questions, such as "Which section or topic of words is the most predictive?" and "What is the difference between being acquired and acquiring?" Using product descriptions to characterize mergers further into horizontal and vertical mergers could eventually assist with the causal estimates that are of interest to economists. More importantly, using language and words to categorize companies could be useful in predicting counterfactual scenarios and answering policy questions, and could have different applications ranging from detecting fraud to better trading.

Keywords: supervised training, machine learning, finance, mergers and acquisations, text

OPEN ACCESS

Edited by:
Glenn Fung,
American Family Insurance,
United States

Reviewed by:
Simon Grima,
University of Malta, Malta
Marco Corazza,
Ca' Foscari University of Venice, Italy

Correspondence:
Tiffany Jiang
tiffj56@gmail.com

Specialty section:
This article was submitted to Mathematical Finance, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *04 January 2021*
Accepted: *16 July 2021*
Published: *02 August 2021*

Citation:
Jiang T (2021) Using Machine Learning to Analyze Merger Activity. Front. Appl. Math. Stat. 7:649501. doi: 10.3389/fams.2021.649501

1 INTRODUCTION

AI applications are changing the financial services industry. Fraud detection and compliance, banking chatbots and robo-advisory services, and algorithmic trading are just a few applications that leverage artificial intelligence in their solutions. Consumer finance has benefited from increased security provided by AI's ability to prevent fraud and cyber-security threats in a quantity unnoticed and unrecognizable by humans. Corporate finance has benefited through AI's ability to answer prediction questions–whether it is the future of loan risks or stock price, AI has been able to power strong portfolios and minimize risk of loan underwriting. AI has been key in improving process automation, where intelligent character recognition makes it possible to automate routine chores that are time-consuming and prone to mistakes. JP Morgan Chase, a leading financial firm, developed Robotic Process Automation to extract data and a Contract Intelligence (COiN) platform to leverage natural language processing (NLP), processing legal documents and extracting essential data. 360,000 labor hours reading 12,000 contracts was cut down to just a few hours. To better detect accounting fraud, the SEC is now using topic models. They draw on text mining and NLP to help us understand the behavioral incentives of different market participants. These are unsupervised learning techniques, and these applications can be used to learn from past violations of regulations and predict new ones. Insider trading and cartel detection are examples. In our case, we will be using supervised machine learning to determine new irregularities in company behavior–merging and acquiring.

From 1982 to 2012, industries in manufacturing, retail trade, wholesale trade, services, finance, and utilities and transportation reported remarkably consistent upward trends in concentration in each sector. The relationship between trends in concentration and competition have presented two hypotheses: an increase in concentration indicates a decline in competition, or a concentration increase reflects the forces of competition and has presented economies of scale. To begin to answer the question of which hypothesis reflects the current trends of large incumbent firms and their gains in market share, we sought to examine characteristics of a firm that predicted merger activity. Prediction is particularly useful to answer a merger question in two ways–first, it allows the examination of a counter-factual, or in this case, a merger retrospective. Second, predicting scenarios of mergers that might lessen competition in the future aids with preventing the loss of potential competition.

Anti-trust economists are generally interested in enforcing stricter merger policy to protect consumers. For instance, a firm's growing market power could drive up prices and lower wages and standards of living. It is in societal interest that antitrust authorities are able to identify and punish collusion in order to promote competition. Vertical mergers are often seen as socially desirable, while horizontal mergers are not. The answer of how to enforce horizontal mergers is an empirical question that is answered by examining merger retrospectives–which mergers harmed customers by lessening competition?

Gentzkow, Kelly, and Taddy (2018) detailed the importance of causal inference compared to prediction in economics. Commonly, economists are more interested in the "how" of a question. They detail how increased data sources could supplement traditional data, especially in finance economics. "For social scientists, the information encoded in text is a rich complement to the more structured kinds of data traditionally used in research, and recent years have seen an explosion of empirical economics research using text as data. In finance, text from financing news, social media, and company filings is used to predict asset price movements and study the causal impact of new information." In our case, we are using company filings in order to predict merger activity and study the causal impact of this new data set.

With the newly available quantities of digital text, we sought textual data that could complement current financial data of firm characteristics. Often times, firm characteristics have been measured using financial data and stock prices. Unique aspects of a firm, we conjectured, such as culture, heavy legal involvement, or product descriptions are not as easy to identify through just numerical values alone. A financial value is one number, and it is hard to distinguish specific characteristics of a firm other than profits. A firm's 10-K SEC filing, an annual document required by the government for all firms to complete, details around 15 items and five distinct sections that includes business, risk factors, selected financial data, management's discussion and analysis of financial condition and results of operations and financial statements and supplementary data. Using text data to supplement current available datasets of a firm, we hypothesized, could gain more insight to the qualities of a firm that predict merger activity. As a next step, we plan on categorizing the acquisitions to horizontal or vertical mergers using detailed product descriptions, or determining which section of a 10-K document has the most predictive ability.

To analyze the text, we modeled the methods after those described by Gentzkow, Kelly, and Taddy (2017). After transforming the text into arrays of words, we further reduce the dimensionality by using LASSO techniques to select words that are promising prediction variables. To select the best model penalized by the LASSO, we used cross-validation techniques to select the best performing regressor variables.

2 MOTIVATION

The application of machine learning and textual analysis is unique to economics in two ways.

First, economics commonly does not look at prediction problems. Text data differs from other kinds of economics data in that it is inherently high dimensional. While this is useful for prediction, this makes it difficult for any sort of causal analysis because the high dimensionality of the data makes ordinary least squares (OLS) and other traditional economics techniques infeasible. The main category of interest is causal inference, or identification. "Economics journals emphasize the use of methods with formal properties of a type that many of the ML methods do not naturally deliver. This includes large sample properties of estimators and tests, including consistency, normality, and efficiency." [1–4] The object of interest is a causal effect, which can be quantified with semi-parametric estimates, or when the number of covariates are large relative to the observation. However, ML and prediction could still provide benefits to economic analysis.

ML could improve empirical analysis to selection functional form flexibly. ML could estimate and compare many models, which is different from economics, where a research will pick a model based on hypothesis and estimate it once. ML also could evaluate the simpler questions of prediction and classification tasks. More work needs to be done to apply an algorithmic approach to economic problems, as they says, but using ML "could provide the best of both worlds: the model selection is data driven, systematic, and a wide range of models are considered; yet, the model selection is fully documented, and confidence intervals take into account the entire algorithm. ML is a very powerful tool for data-driven model selection."

Second, an increased amount of data and the different types of data will become available to economists. Text data and increasing digitization suggest new identification questions to be answered, and digitization is leading the amounts of big data that can be used in economics. According to Gentzkow, Kelly, and Taddy, "In industrial organization and marketing, text from advertisements and product reviews is used to study the drivers of consumer decision making. In political economy, text from politicians' speeches is used to study the dynamics of political agendas and debate. The most important way that text differs from the kinds of data often used in economics is that text is inherently high dimensional." In our case, text is a new form of data in economics that was previously unused. 10-K SEC filings are just a case example of how text can be used to supplement the traditional economic database. The 10-K document also has significance in the role of an investment banker or analyst–they are commonly read over and used to determine the worth of a company and to evaluate the landscape of the business. The document includes pages and pages of what the company itself, the expert in their own business, summarizes and even discloses a risk factor analysis. As to abide by legal guidelines, companies are incentivized to be as thorough and exhaustive as possible as to avoid misleading stakeholders. Previously, only financial data and stock prices were examined in order to analyze a company. To measure harder to observe variables that are not just captured in financial variables, we propose using a text document and examine a firm's 10-K SEC filing. Statistical learning technique also offers an advantage by taking into account many aspects of the data.

Our procedure uses big data. We use tens of thousands of disclosures, and even more words in order to obtain predictive regressors from the text. We propose to use LASSO as a regularization technique, which has been more successful than the bag-of-words method and RIDGE regression. We used the K-fold cross validation in order to obtain the best hyper-parameter.

Three common ways of looking at merger types are to categorize them into horizontal, vertical and conglomerate mergers. Merger motives are primarily due to financial considerations for the profit maximizing firm, and these considerations are driven by increasing market power, exploiting economies of scale, and eliminating managerial inefficiency. Other motivations include risk reduction by diversifying activity, government policy, and principal-agent problems in which company managers have different interests from the stakeholders and prefer to instead maximize their own income. To examine theoretical models of mergers we have three groups. The first are neoclassical models, which propose that merger waves come from political, economic, industrial, or regulatory shocks. The second are models that demonstrate herding, hubris or agency problems and propose takeovers are led by managerial inefficiency. The third are models that reflect capital market development and attribute mergers to market timing. The second may be hard to measure through words alone.

2.1 Neoclassical Models–A Look at Merger Theory

Coase (1937) is an early proponent of the model suggesting that takeover activity is driven by technological change. A later model by Gort (1969) claims that economic disturbances, such as market disequilibrium, may cause wholesale industry restructuring.

Jovanovic and Rousseau (2001, 2002) builds on Gort's theory, and developed the Q-theory of takeovers, which posits that economic and technological changes cause a higher degree of corporate growth opportunities. Such changes may cause capital to be reallocated to more productive and efficient firms. What about situations that do not fall into any of these categories? These is where the regularization and the selection of predictive variables could reveal situations that have been unprecedented in previous patterns.

2.2 Related Work

Gentzkow and Shapiro popularized the use of text as data by measuring a previously difficult to observe variable, *media slant*. They determined whether a newspaper was more Republican or Democratic, and then used words that captured slant to incorporate in a demand function that maximizes newspaper profits to predict consumer behavior. This was compared with an actual profit maximizing choice to validate economic preference, and it was found that consumers had a preference for newspapers that were like-minded.

[5] determine firm similarity and product differentiation through textual analysis and found that estimating patterns of similarities in this method performed better than SIC or NAICS codes. However, SIC and NAICS codes have major drawbacks of being too broad. First, neither reclassifies firms significantly over time as the product market evolves. Second, neither can easily accommodate innovations that create entirely new product markets. In the late 1990s, hundreds of new technology and web-based firms were grouped into a large and nondescript SIC-based "business services" industry. Third, SIC and NAICS impose transitivity even though two firms that are rivals to a third firm might not be rivals. Hoberg and Phillips further examined asset complementaries as a way to analyze merger pairs and further predict merger activity.

[6] use market conditions to test the effects of uncertainty on acquisition by using a firm's financial data that includes size, stock returns, and dividends to find that assets and being in a high

acquisition industry increases the probability of being acquired. The model yields R-squared of around 0.02.

Previous attempts to explain merger activity have yielded a Psuedo R-squared that ranges from 0.01 to 0.09. [7–9].

[10] are the first to use words alone to predict merger activity. The authors cite a number of papers that use various financial information to predict merger activity and note "predicting target firms with any accuracy has proven difficult" (Betton, Eckbo, and Thorburn, 2008). In their study, they use one specific section of a company's 10-K SEC filing, the firm's Management Discussion and Analysis and two-word phrases to fit their model. Their text uses the frequency of words appearing on a document and transform the counts with a logarithm function to account for any right-hand skew. Their main discovery supports the Q-theory of takeovers and find that firms that are struggling financially are more likely to be acquired. Their models range from a 0.01 to a 0.07 R-squared.

None of these past works, however, uses the entire document. Our dataset is also the most recent. Our dataset is from 2013 to 2017, and the landscape for technology and business is arguably much different from 10 years ago. Previous works have also attempted to evaluate the accuracy of a model by using Pseudo R-squared, still attempting to retain some interpretability from the data. We propose using accuracy to measure predictive success. Future studies could include different measures of accuracy, such as sensitivity and specificity, or measuring an ROC curve. No method also uses tf-idf to transform the text data. Tf-idf accounts for common and rare words and considers how important a word is to the document. This adds some transformation to text data that could reduce noise and add significant information to the regression.

Theoretically, neoclassical models have pointed to the ways that increase the likelihood of a firm merging with another. These changes have been hard to measure in the past, and accordingly have not had confident estimates about the magnitude of an effect or a conclusion on the sign. Thus, we turn to a different empirical investigation.

The central contributions of this study are 1) an application of machine learning in economics to uncover a new field of policy and prediction questions previously unanswered, and 2) using text as a new dataset previously unused in economics.

3 DATA

The estimates presented below are based on US SEC filings for the periods of 2013–2016. Merger activity are based on aggregate U.S. data from 2013 to 2017 (**Table 1**).

TABLE 1 | Observation count broken down, from years 2013–2017.

Observation count	
Involved in a merger (Total)	Not involved (Total)
5,379	17,073
Training observations	Test observations
4,303	13,658

3.1 Representing Text as Data

3.2.1 Cleaning

We performed the standard cleaning to transform the 10-K filings into a text corpus data set. Punctuation and common words were removed, white space was stripped, and all words were transformed for lower characters. We defined each observation as a 10-K SEC filing - this is our document for cleaning.

3.2.2 Tf-Idf

The method used to represent the documents was filtering first by "tf-idf", or "term-frequency inverse-document-frequency". This method excludes both common ("a", "the", "and") and rare ("sesquipedalian", "phantasm") words. Rare words are excluded to optimize model fit, as their marginal value of meaning often exceeds adding more features to the model. Common words that appear with most documents will have a low tf-idf score, and words that have a low tf-idf score will be cut off from the document. This technique has proven to be very useful in practice as it reduces the number of features to something more manageable [11].

Term Frequency[1]: Measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization.

Inverse Document Frequency: Measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scaling up the rare ones.

$$Term.Frequency(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total Number of terms in the document}} \quad (1)$$

$$Inverse.Document.Frequency(t) = \log\left(\frac{\text{Total Number of documents}}{\text{Number of documents with term t in it}}\right) \quad (2)$$

$$tfidf_{ij} = tf_{ij} \times idf_{ij} \quad (3)$$

3.3 10-K Filings

We start off with textual data because of the use of 10-K's by investment bankers and investors to determine company value. The 10-K differs from other documents such the annual shareholders report in length, detail, and scrutiny and are meant to be lengthy, detailed, and not easily digestible. Successful fund managers have cited reading the 10-K as a way to gauge worthwhile investments and have listed notable sections in the Management Discussion and Analysis, the chairman's letter, the risk factor analysis, proxy statements, earnings adjustments and even footnotes. See **Table 2** for the full 10-K description.

We use the entire document in our findings. Total observations are 22,418. We try to test whether there is a way

[1]Source of definition of tf-idf comes from http://www.tfidf.com/

TABLE 2 | Detailed description of a 10-K filing.

Name	Section description
10-K Section descriptions	
Item 1–Business	This describes the business of the company: Who and what the company does, what subsidiaries it owns, and what markets it operates in. It may also include recent events, competition, regulations, and labor issues. (Some industries are heavily regulated, have complex labor requirements, which have significant effects on the business). Other topics in this section may include special operating costs, seasonal factors, or insurance matters
Item 1A–Risk factors	Here, the company lays anything that could go wrong, likely external effects, possible future failures to meet obligations, and other risks disclosed to adequately warn investors and potential investors
Item 1B–Unresolved staff comments	
Item 2–Properties	This section lays out the significant properties, physical assets, of the company. This only includes physical types of property, not intellectual or intangible property
Item 3–Legal proceedings	Here, the company discloses any significant pending lawsuit or other legal proceeding. References to these proceedings could also be disclosed in the risks section or other parts of the report
Item 4–Mine safety disclosures	This section requires some companies to provide information about mine safety violations or other regulatory matters
Item 5–Market	Gives highs and lows of stock, in a simple statement. Market for Registrant's common equity, related stockholder matters and issuer purchases of equity securities
Item 6–Consolidated financial data	In this section financial data showing consolidated records for the legal entity as well as subsidiary companies
Item 7–Management's discussion and analysis of financial condition and results of operations	Here, management discusses the operations of the company in detail by usually comparing the current period versus prior period. These comparisons provide a reader an overview of the operational issues of what causes such increases or decreases in the business
Item 8–Financial statements	Here, also, is the going concern opinion. This is the opinion of the auditor as to the viability of the company. Look for "unqualified opinion" expressed by auditor. This means the auditor had no hesitations or reservations about the state of the company, and the opinion is without any qualifications (unconditional) 1. Independent auditor's report 2. Consolidated statements of operation 3. Consolidated balance sheets 4. Other accounting reports and notes

10-K Section names–items 9–15

Item 9. Changes in and disagreements with accountants on accounting and financial disclosure	Item 9A. Controls and procedures	Item 9B. Other information
Item 10. Directors, executive officers and corporate governance	Item 11. Executive compensation	Item 12. Security ownership of certain beneficial owners and management and related stockholder matters
Item 13. Certain relationships and related transactions, and director independence	Item 14. Principal accounting fees and services	Item 15. Exhibits, financial statement schedules signatures

to develop an automated decision support system for helping the human process determining the characteristics of a company and then use these new predictors to predict merger activity. Natural language processing techniques are ways to quantify business phrases such as "synergies" in a more robust manner or at least in a different way compared to previous methods.

Merger events are drawn through Thomson Reuters' SDC Platinum database. Firms that have been recorded as involved with merger activity are labeled, and those that have not are also marked accordingly.

4 MODEL

Three major steps were taken, in the following order, as modeled by [11]:

1. Represent raw text (documents) as numerical array **C.**
2. Map **C** to predicted values of $\hat{V}$ of unknown outcomes **V.**
3. Use $\hat{V}$ in subsequent descriptive of causal analysis.

The first step was completed using the cleaning and tf-idf method described in **section 3**.

4.1 The LASSO

We decided on using the LASSO over ridge or elastic net. In our case, we wanted to come up with a limited set of words that predicted merger activity. The high dimensionality.

The LASSO, or the Least Absolute Shrinkage and Selection Operator, imposes a restriction on a high-dimensional linear model. The model is penalized by the size of the model through the absolute values of the coefficients [12]. define the LASSO as

$$\hat{\beta} = \arg \min_{\beta} \sum_{n=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{i,j} b_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \gamma_j \tag{4}$$

where $\lambda > 0$ is the "penalty level" and γ_j are the "penalty loadings." [13, 14] discuss how the LASSO corresponds to 1) a quadratic loss function, 2) a class of linear functions (over some fixed set of possible variables), and 3) a regularizer which is the sum of absolute values of coefficients. This absolute-value regularizer shrinks many coefficients to zero, yielding an approximately sparse linear framework.

With regularization techniques such as the LASSO comes the advantage of allowing for "wide" data, where there are more regressor variables than observations themselves, such as the case

TABLE 3 | Words picked by the LASSO, tf-idf.

iii	Reasonable	Director	Security	But	Chief
−263.634	−21.447	−9.736	−7.983	−4.556	1.599
Indicate	Controls	Disclosures	Procedures	Registered	Disclosure
39.172	44.864	146.363	192.412	390.211	1, 648.075

With this model, an accuracy rate of 85% was obtained. Further versions will include sensitivity and specificity.

with the text data from the SEC filings. Using such machine learning techniques effectively allows for the data to speak for itself–the LASSO could potentially uncover generalizable patterns that were not specified by the economist in advance.

However, there are limitations to using the LASSO [13]. detail how correlations between variables must be limited. There are two particularly strong assumptions for the LASSO to hold - that first, only a few variables are relevant, and second, none of the irrelevant covariates can be even moderately related to the set of relevant ones. This leaves danger of interpreting the β parameters naively.

As increasingly flexible methods are used, variance will increase and bias decreases. Traditional models want no bias. Machine learning allows some bias and reduces variability (e.g., Lasso, Ridge). The model is penalized for size, i.e., how many coefficients is put into the equation. To adjust, a different sample once a model is selected to test for goodness of fit. In order to ensure that the model we created has external validity, we use cross-validation training techniques and separate the dataset to a training set and a test set.

4.1.1 Cross-Validation

In order to select the optimal λ, we used the K-fold cross-validation technique. This splits our sample into 10 subsets, and then fit a model 10 times excluding each subset in turn. With this, 10 mean squared errors were obtained by verifying the accuracy of the model fit on the subset that was left out for 100 values of λ. The value of λ which minimizes the average error was selected as the λ in the final prediction model.

4.2 Final Prediction Model

Given our explanatory variables, our prediction y will have a binary value: 1 for a prediction that a firm will be involved in the merger as either a target or be the takeover company, 0 for a prediction that will not. The final formula is:

$$\hat{\beta} = \arg \max_{\beta} \sum_{n=1}^{N} \log p(y_i|x_i) + \lambda_1 \|\beta\|_1 \tag{5}$$

To model the relationship between $p(X) = Pr(Y = 1|X)$ and X, we use the logistic regression:

$$p(y = 1|\mathbf{x}) = \frac{\exp(\beta_0 + \beta^{\tau} x)}{1 + \exp(\beta_0 + \beta^{\tau} x)} = \frac{1}{\exp(-\beta_0 - \beta^{\tau} x)} \tag{6}$$

To fit this logistic regression, the parameters are fit through a maximum likelihood function:

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_i'=0} (1 - p(x_i')) \tag{7}$$

5 RESULTS

Seeing the low prediction rate of our original theory, we decided to try to find the best prediction model instead. With this result, the LASSO selected 12 words as predictors (**Table 3**). Given the size of the estimates and the nature of tf-idf, the results are difficult to interpret and the best method to find the predictors of a firm's potential to merge would probably be to feed the individual 10-K document into the code and observe each result on a case by case basis. However, perhaps the word "but" could signify a struggling firm - perhaps management needs to explain poor performance by using the word "but" a lot to argue in favor of the firm.

5.1 N-Grams Representation

Originally, we represented words using the *bag-of-words* method (**Table 4**). This technique represents words in terms of frequency, where $word_{ij}$ is an element in the dictionary vector that appears j times in document i.

Example[2], where the text of document i is:

Good night, good night! Parting is such sweet sorrow.

After stemming, removing stop words, and removing punctuation, we might be left with "good night good night part sweet sorrow." The bag-of-words representation would then have $c_{ij} = 2$ for j ∈ {good, night}, $c_{ij} = 1$ for j ∈ {part, sweet,sorrow}, and $c_{ij} = 0$ for all other words in the vocabulary.

5.2 Comparing Words to Financial Variables

To test whether words supplemented financial datasets when describing a firm or could completely replace financial variables entirely when looking to predict merger activity, we used financial variables to predict merger accuracy to compare it to textual accuracy (**Table 5**). Financial variable accuracy yielded a 85% success rate. However, the interpretability of the variables is different from the text regression, and offers different insight. This suggests text supplementing financial variables could be a powerful combination.

5.3 Attempts to Verify Existing Merger Theory

Originally, we had personally narrowed down a dictionary of words without using tf-idf in order to attempt to use some

[2]Example from [17].

TABLE 4 | LASSO results, Bag of Words method.

Test	2007	Unrecognized	Improvements	2005	Testing
−0.086	−0.080	−0.079	−0.069	−0.065	−0.064
Out	2006	2018	Accounted	Allocated	Component
−0.062	−0.061	−0.057	−0.044	−0.041	−0.041
Contributions	Competition	Respective	Longlived	Locations	102
−0.039	−0.038	−0.035	−0.032	−0.031	−0.030
Discounted	Combined	Next	Areas	Strategy	Yield
−0.030	−0.030	−0.029	−0.029	−0.027	−0.025
Then	Trends	Retirement	Earned	Environment	Par
−0.025	−0.025	−0.023	−0.023	−0.023	−0.023
Treasury	Protection	Final	Investing	Cumulative	Low
−0.023	−0.022	−0.022	−0.020	−0.020	−0.020
Performed	Impaired	Several	Maturities	Leased	Supplemental
−0.020	−0.020	−0.019	−0.018	−0.018	−0.018
Weightedaverage	Assumed	Reflected	Outside	Primary	Directly
−0.017	−0.017	−0.017	−0.017	−0.016	−0.015
Domestic	Registration	Acquired	Summary	Competitors	Institutions
−0.013	−0.013	−0.011	−0.011	−0.010	−0.010
Termination	Taxable	Active	Developed	North	Resulted
−0.009	−0.009	−0.008	−0.007	−0.007	−0.006
Designed	Generated	Classified	Delivery	Measures	Therefore
−0.006	−0.005	−0.004	−0.004	−0.004	−0.003
Excluding	Although	Sources	Highly	Law	Building
−0.003	−0.003	−0.003	−0.002	−0.002	−0.001
Restrictions	Completion	Lives	Projected	Recovery	Detriment
−0.001	−0.001	−0.001	−0.0003	−0.0001	−0.0001
Right	Professional	Covered	Authorized	Consist	Need
−0.00003	−0.00002	0.0001	0.0003	0.001	0.002
Individual	Reflect	10q	1934	Application	Dependent
0.003	0.005	0.007	0.007	0.009	0.009
Unless	Actions	Treatment	Organizations	Ending	Contained
0.010	0.010	0.011	0.011	0.012	0.016
Approach	Proxy	Major	Recognize	name	Proprietary
0.016	0.017	0.018	0.022	0.022	0.026
Timing	Variable	Strategic	Gaap	Transfer	Standard
0.027	0.032	0.035	0.045	0.048	0.059
Evaluate	2017	Entitled	Adoption	Early	Fasb
0.075	0.113	0.136	0.142	0.155	0.290
Goods	2019				
0.204	0.604				

Using this method, an accuracy rate of 35% was yielded.

amount of theory to drive the variables that would go into the model.

To verify Jovanic and Rousseau's theory on take-overs for the financially deteriorating firm, we hypothesized that one would look for characteristics that signal a poorly performing firm. To study whether firms abide by diversification incentives or horizontal and vertical merger incentives, one would prefer information on the specifics on a firm such as product descriptions and equipment. While examining a principal agent problem, one would look for information detailing management and executive leadership.

We looked for this information in the 10-K sections that described who and what the company does, subsidiaries it owns, and what markets it operates in, recent events, competition, regulations, and labor issues, operating costs, season factors, and insurance matters as well as a section describing the properties and physical assets of the company. To aid with examining management concerns, the documents contain two

TABLE 5 | Financial regression.

	Dependent variable
	Merged (boolean)
Assets	0.002
	(0.001)
Cash	−0.0004
	(0.002)
Net receivables	−0.020*
	(0.012)
Retained earnings	0.001*
	(0.001)
Current liabilities	0.001
	(0.001)
Constant	−4.518***
	(0.620)
Observations	884
Log Likelihood	−21.866
Akaike Inf. Crit.	55.733

Note: *p < 0.1; **p < 0.05; ***p < 0.01.

pertinent sections: certain relationships and related transactions and director independence and directors, executive officers and corporate governance.

With this kind of motivation in mind, we used the bag of word techniques in attempt to verify existing theory. We assume that phrases such as "technological", "commercial", "marketing", "integrated", "data", "development", "electronically", "technical", and "support" are proxies for technological change as proposed by Coase (1937). "Liabilities", "loan", "losses", "expense", "adversely", "adverse", "negatively", "fail", "deteriorate", "risk", and "depreciation" are used to test Q-theory of takeovers. "Global", "established", "minimum", "competitive", "holders", "forward looking", "comparable", "health", "international", "respect", "power", "properties", "longterm", "exceed", and "trends" are used to test for market power theory. "Promotion", "training", "managerial", "finance" are phrases that test for management inefficiency. We also test directly for mention of acquisition with "acquired", "consolidated", "accumulated", "aggregate", "integrated", "cumulative", "portfolio", "spread". For market conditions and any anti-trust considerations, we examine words such as "fluctuations" and "sarbanesoxley". These proxy variables have limitations of having poor correlation with my intended variable of interest. As these are one-word phrases, it is difficult to directly measure the extent in which it describes the characteristic we wish to examine. For instance, "risk" is used to test for signs of a financially deteriorating firm, but perhaps the phrase "risk" was used in the phrase "little risk". This method, however, did not achieve a high accuracy rate.

Traditionally, machine learning estimation works best in creating a predictive model. The trade-offs of creating a flexible, nonparametric predictive model are that causal interpretations are often lost. Linear regression is relatively inflexible approach but easy to interpret. Flexible models avoid assumptions of a particular functional form for a model, but require a large number of observations and are more difficult to interpret.

6 LIMITATIONS

The limitations of this work are primarily definitional–defining accuracy in a more rigorous way, such as using sensitivity and specificity. Sensitivity would determine the accuracy of the amount of mergers correctly identified that would have merged. Specificity rates would determine the accuracy of determining the rate of non-mergers–the number of companies who are predicted to not merged over the amount that did not merge. This requires increased programming capabilities that a future paper could take into account. This limits how we can exactly we can interpret the prediction model. Using 2-word tokens, or two-word phrases, might also provide more information that we'd like to take as a next step. However, as Gentzkow, Kelly, and Taddy (2018) note, this might create memory and computational limitations, but could provide more insight on merger activity.

There are four potential next steps that we hope to take this research.

The first is to attempt to find the best predictive model by using different machine learning techniques. So far, we have used a penalized linear model to predict merger activity. However, there are also nonlinear regression methods such as generalized linear models, support vector machines, regression trees, and deep learning. There are also Bayesian regression methods such as the spike-and-slab, as well as topic modeling techniques such as Linear Discriminant Analysis. So far, the LASSO method is just one technique amongst the numerous machine learning methods.

The next direction is to attempt to predict horizontal and vertical mergers through product descriptions [15]. have discovered the limitations to categorize firms through traditional SIC and NAICS descriptions. For example, the descriptions do not reclassify firms over time as the product market evolves, or accommodate innovations that create entirely new product markets. Using product descriptions could potentially better determine whether a merger was horizontal or vertical, a pressing antitrust issue.

The third is to use the words to proxy characteristics of a firm in an index, the same way [16–19] measured slant.

This would be to see if there's a way to predict specific types of mergers to prevent "loss of future competition occurred when a large incumbent firm acquires a highly capable firm operating in an adjacent space."

The equation would look like:

$$P(y = 1|x) = \text{Financial variables} + \text{Location} + \text{HHI} + \text{Heavy Legal Involvement} + \text{Product Type} \quad (8)$$

To give an example, we took a look at two 10-k's in 2013–Dish Network and Echostar. we predicted that maybe Dish would acquire Echostar eventually after reading the 10-K, and sure enough, they bought Echostar's DBS and OTT assets in 2017. A few phrases that stood out: "ku-band payload," "xrbl taxomony extension," "high thorough geostationary," "satellite orbit," "transponder service agreements" showed up, both were a "colorado corporation", and given the variations of "subscriber services" and "customer subscriptions" that showed up in Dish's 10-k, Dish probably has more incentive to acquire to satisfy their customer base. "Treble damages" also showed up frequently in Echostar's 10-K.

Lastly, merger types are usually well defined. However, a lot of "unusual" merger activity have occurred recently. For example: Amazon and Whole Foods, Intel and Caring.com, Delta Airlines and Refinery Phillips 66. Perhaps the "culture" of a firm would have to do with those who choose to engage in conglomerate mergers. This would be a textual exercise, where one would search for stylistic preferences and firm values. The equation might look like:

$$p(y = 1|\mathbf{x}) = \beta \times Words(Culture) + \alpha \times Financial.Variables + \gamma \times Industry.Type + \epsilon \quad (9)$$

7 CONCLUSION

Previously, merger activity has been hard to define and predict. We believe, however, with more granular data that captures information about a company that financial variables could not, NLP and applied machine learning could begin to chip away at harder to understand cases of mergers. Our use of LASSO and tf-idf on a text regression produced favorable results and interpretability different from financial variables alone. We found that poor performance predicted merger likelihood because of a struggling firm becoming likely to become a takeover target. We found that tf-idf was favorable for prediction purposes compared to bag-of-words, and that financial variables also added prediction power. Machine learning, like in our case, can be applied to problems that were traditionally difficult to solve and offer insight that past datasets could not. Text is a powerful data source that could capture a lot of missing information that limited past research endeavors. We believe that an important transformation of economics and applied finance is underway.

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

The author confirms being the sole contributor of this work and has approved it for publication.

ACKNOWLEDGMENTS

We would like to thank Giovanni Peri, Colin Cameron, and the faculty of UC Davis Economics.

REFERENCES

1. Athey S. The Impact of Machine Learning on Economics. *The Econ Artif Intelligence: Agenda* (2019) 21:507–52. doi:10.7208/chicago/9780226613475.003.0021
2. Athey S. Machine Learning and Causal Inference for Policy Evaluation. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2015) p. 5–6. doi:10.1145/2783258.2785466
3. Athey S, and Imbens GW. The State of Applied Econometrics: Causality and Policy Evaluation. *J Econ Perspect* (2017) 31(2):3–32. doi:10.1257/jep.31.2.3
4. Angrist JD, and Pischke J-S. Undergraduate Econometrics Instruction: Through Our Classes, Darkly. *J Econ Perspect* (2017) 31(2):125–44. doi:10.1257/jep.31.2.125
5. Hoberg G, and Phillips G. Text-Based Network Industries and Endogenous Product Differentiation. *J Polit Economy* (2017) 124(5):1423–65. doi:10.1086/688176
6. Harford J. What Drives Merger Waves? *J Financial Econ* (2005) 77:529–60. doi:10.1016/j.jfineco.2004.05.004
7. Edmans A, Goldstein I, and Jiang W. The Real Effects of Financial Markets: The Impact of Prices on Takeovers. *J Finance* (2012) 67:933–71. doi:10.1111/j.1540-6261.2012.01738.x
8. Chatterjee S, John K, and Yan A. Takeovers and Divergence of Investor Opinion. *Rev Financ Stud* (2012) 25(1):227–77. doi:10.1093/rfs/hhr109
9. Cocco JF, and Volpin PF. Corporate Pension Plans as Takeover Deterrents. *J Financ Quant Anal* (2013) 48(4):1119–44. doi:10.1017/s0022109013000355
10. Routledge B, Sacchetto S, and Smith N. *Predicting Merger Targets and Acquirers from Text*. Carnegie Mellon University Working Paper (2018).
11. Gentzkow M, Kelly B, and Taddy M. Text as Data. *J Econ Lit* (2019) 57(3):535–74. available at: https://web.stanford.edu/~gentzkow/research/text-as-data.pdf. doi:10.1257/jel.20181020
12. Belloni A, Chernozhukov V, and Hansen C. High-Dimensional Methods and Inference on Structural and Treatment Effects. *J Econ Perspect* (2014) 28(2):29–50. doi:10.1257/jep.28.2.29
13. Mullainathan S, and Spiess J. Machine Learning: An Applied Econometric Approach. *J Econ Perspect* (2017) 31(2):87–106. doi:10.1257/jep.31.2.87
14. Tibshirani R. Regression Shrinkage and Selection via the Lasso. *J R Stat Soc Ser B (Methodological)* (1996) 58(1):267–88. doi:10.1111/j.2517-6161.1996.tb02080.x
15. Hoberg G, and Phillips G (2018). *Product Integration and Merger Success. Tuck School of Business Working Paper No. 2933283*, Marshall School of Business Working Paper No. 17–21. Available at: https://ssrn.com/abstract=2933283.
16. Gentzkow M, and Shapiro J. What Drives Media Slant? Evidence from U.S. Daily Newspapers. *Econometrica* (2010) 78(1):35–71. doi:10.3982/ecta7195
17. Kleinberg J, Ludwig J, Mullainathan S, and Obermeyer Z. Prediction Policy Problems. *Am Econ Rev* (2015) 105(5):491–5. doi:10.1257/aer.p20151023
18. Gregoriou GN, and Renneboog L. Understanding Mergers and Acquisitions. *Quantitative Finance* (2007) 1:1–20. doi:10.1016/b978-075068289-3.50003-4
19. Shapiro C. Antitrust in a Time of Populism. *Int J Ind Organ* (2018) 61:714–48. doi:10.1016/j.ijindorg.2018.01.001

Conflict of Interest: The author declares 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 Applied Mathematics and Statistics

METHODS
published: 26 August 2021
doi: 10.3389/fams.2021.604842

AVA: A Financial Service Chatbot Based on Deep Bidirectional Transformers

Shi Yu, Yuxin Chen and Hussain Zaidi*

The Vanguard Group, Malvern, PA, United States

We develop a chatbot using deep bidirectional transformer (BERT) models to handle client questions in financial investment customer service. The bot can recognize 381 intents, decides when to say I don't know, and escalate escalation/uncertain questions to human operators. Our main novel contribution is the discussion about the uncertainty measure for BERT, where three different approaches are systematically compared with real problems. We investigated two uncertainty metrics, information entropy and variance of dropout sampling, in BERT, followed by mixed-integer programming to optimize decision thresholds. Another novel contribution is the usage of BERT as a language model in automatic spelling correction. Inputs with accidental spelling errors can significantly decrease intent classification performance. The proposed approach combines probabilities from masked language model and word edit distances to find the best corrections for misspelled words. The chatbot and the entire conversational AI system are developed using open-source tools and deployed within our company's intranet. The proposed approach can be useful for industries seeking similar in-house solutions in their specific business domains. We share all our code and a sample chatbot built on a public data set on GitHub.

Keywords: chabot, BERT, rasa, bayesian learning, intent classification

OPEN ACCESS

Edited by:
*Glenn Fung,
American Family Insurance,
United States*

Reviewed by:
*Qian You,
University of Wisconsin-Madison,
United States
Amin Salighehdar,
Stevens Institute of Technology,
United States*

****Correspondence:***
*Shi Yu
shi.yu@hotmail.com*

Specialty section:
*This article was submitted to
Mathematical Finance,
a section of the journal
Frontiers in Applied Mathematics and
Statistics*

Received: *10 September 2020*
Accepted: *12 May 2021*
Published: *26 August 2021*

Citation:
*Yu S, Chen Y and Zaidi H (2021) AVA:
A Financial Service Chatbot Based on
Deep Bidirectional Transformers.
Front. Appl. Math. Stat. 7:604842.
doi: 10.3389/fams.2021.604842*

1 INTRODUCTION

Since their first appearances decades ago [1–3], chatbots have always been marking the apex of artificial intelligence as forefront of all major AI revolutions, such as human–computer interaction, knowledge engineering, expert system, natural language processing, natural language understanding, deep learning, and many others. Open-domain chatbots, also known as *chitchat* bots, can mimic human conversations to the greatest extent in topics of almost any kind, thus are widely engaged for socialization, entertainment, emotional companionship, and marketing. Earlier generations of open-domain bots, such as those mentioned in Ref [3, 4], relied heavily on hand-crafted rules and recursive symbolic evaluations to capture the key elements of human-like conversation. New advances in this field are mostly data-driven and end-to-end systems based on statistical models and neural conversational models [5] aim to achieve human-like conversations through a more scalable and adaptable learning process on free-form and large data sets [5], such as those given in Ref [6–9] and [10].

Unlike open-domain bots, closed-domain chatbots are designed to transform existing processes that rely on human agents. Their goals are to help users accomplish specific tasks, where typical examples range from order placement to customer support; therefore, they are also known as *task-oriented bots*

[5]. Many businesses are excited about the prospect of using closed-domain chatbots to interact directly with their customer base, which comes with many benefits such as cost reduction, zero downtime, or no prejudices. However, there will always be instances where a bot will need a human's input for new scenarios. This could be a customer presenting a problem it has never expected for [11], attempting to respond to a naughty input, or even something as simple as incorrect spelling. Under these scenarios, expected responses from open-domain and closed-domain chatbots can be very different: a successful open-domain bot should be "*knowledgeable, humorous, and addictive*," whereas a closed-domain chatbot ought to be "*accurate, reliable, and efficient*." One main difference is the way of handling unknown questions. A chitchat bot would respond with an adversarial question such as *Why do you ask this?* and keep the conversation going and deviate back to the topics under its coverage [12]. A user may find the chatbot is out-smarting, but not very helpful in solving problems. In contrast, a task-oriented bot is scoped to a specific domain of intents and should terminate out-of-scope conversations promptly and escalate them to human agents.

This article presents AVA (a Vanguard assistant), a task-oriented chatbot supporting phone call agents when they interact with clients on live calls. Traditionally, when phone agents need help, they put client calls on hold and consult experts in a support group. With a chatbot, our goal is to transform the consultation processes between phone agents and experts to an end-to-end conversational AI system. Our focus is to significantly reduce operating costs by reducing the call holding time and the need of experts, while transforming our client experience in a way that eventually promotes client self-provisioning in a controlled environment. Understanding intents correctly and escalating escalation intents promptly are key to its success. Recently, the NLP community has made many breakthroughs in context-dependent embeddings and bidirectional language models like ELMo, OpenAI, GPT, BERT, RoBERTa, DistilBERT, XLM, and XLNet [1, 13–21]. In particular, the BERT model [1] has become a new NLP baseline including sentence classification, question answering, named-entity recognition and many others. To our knowledge, there are few measures that address prediction uncertainties in these sophisticated deep learning structures, or explain how to achieve optimal decisions on observed uncertainty measures. The off-the-shelf softmax outputs of these models are predictive probabilities, and they are not a valid measure for the confidence in a network's predictions [22–25], which are important concerns in real-world applications [11].

Our main contribution in this study is applying advances in Bayesian deep learning to quantify uncertainties in BERT intent predictions. Formal methods like stochastic gradient (SG)-MCMC [23, 26–30] and variational inference (VI) [22, 31–33] extensively discussed in the literature may require modifying the network. In conventional neural networks, the parameters are estimated by a single point value obtained using backpropagation with stochastic gradient descent (SGD), whereas Bayesian deep learning assumes a prior over model parameters and then data are used to compute a distribution over each of these parameters. However, for BNNs with thousands of parameters, computing the posterior is intractable due to the complexity in computing the marginal likelihood [34]. SG-MCMC and VI methods propose two different solutions to address the aforementioned complexity. SG-MCMC mitigates the need to compute gradients on full data set by using mini-batches for gradient computation, which enables easier computation (with the same computational complexity as SGD), but still lacks the ability to capture complex distributions in the parameter space. VI performs Bayesian inference by using a computationally tractable "variational" distribution $q(\theta)$ to approximate the posterior, and the capacity of uncertainty representation is limited by the variational distribution. Re-implementation of the entire BERT model for Bayesian inference is a non-trivial task, so here we took the Monte Carlo dropout (MCD) approach [22] to approximate variational inference, whereby dropout is performed at training and test time, using multiple dropout masks. Our dropout experiments are compared with two other approaches (entropy and dummy class), and the final implementation is determined among the trade-off between accuracy and efficiency. Recently, similar MCD dropout approach has been proposed for transformer models to calibrate speech detection outcomes [35].

We also investigate the usage of BERT as a language model to decipher spelling errors. Most vendor-based chatbot solutions embed an additional layer of service, where device-dependent error models and N-gram language models [36] are utilized for spell checking and language interpretation. At the representation layer, WordPiece model [37] and byte pair rncoding (BPE) model [38, 39] are common techniques to segment words into smaller units; thus, similarities at the sub-word level can be captured by NLP models and generalized on out-of-vocabulary (OOV) words. Our approach combines efforts of both sides: words corrected by the proposed language model are further tokenized by the WordPiece model to match pretrained embeddings in BERT learning.

Despite all advances of chatbots, industries like finance and health care are concerned about cyber security because of the large amount of sensitive information entered during chatbot sessions. Task-oriented bots often require access to critical internal systems and confidential data to finish specific tasks. Therefore, 100% on-premise solutions that enable full customization, monitoring, and smooth integration are preferable than cloud solutions. In this study, the proposed chatbot is designed using RASA open-source version and deployed within our enterprise intranet. Using RASA's conversational design, we hybridize RASA's chitchat module with the proposed task-oriented conversational systems developed on Python, TensorFlow, and PyTorch. We believe our approach can provide some useful guidance for industries to contemplate adopting chatbot solutions in their business domains.

2 BACKGROUND

Recent breakthroughs in NLP research are driven by two intertwined directions: Advances in distributed representations, sparked by the success of word embeddings [40, 41], character embeddings [42–44], and contextualized word embeddings [1, 19, 45], have successfully tackled the curse of dimensionality in modeling complex language models. Advances of neural network architecture, represented by CNN [46–48], attention mechanism [49], and transformer as the seq2seq model with parallelized attentions [50], have defined the new state-of-the-art deep learning models for NLP.

Principled uncertainty estimation in regression [51], reinforcement learning [52], and classification [53] are active areas of research with a large volume of work. The theory of Bayesian neural networks [54, 55] provides the tools and techniques to understand model uncertainty, but these techniques come with significant computational costs as they double the number of parameters to be trained. The authors of Ref [22] showed that a neural network with dropout turned on at test time is equivalent to a deep Gaussian process, and we can obtain model uncertainty estimates from such a network by multiple-sampling the predictions of the network at test time. Non-Bayesian approaches to estimate the uncertainty are also shown to produce reliable uncertainty estimates [56]; our focus in this study is on Bayesian approaches. In classification tasks, the uncertainty obtained from multiple sampling at test time is an estimate of the confidence in the predictions similar to the entropy of the predictions. In this study, we compare the threshold for escalating a query to a human operator using model uncertainty obtained from dropout-based chatbot against setting the threshold using the entropy of the predictions. We choose dropout-based Bayesian approximation because it does not require changes to the model architecture, does not add parameters to train, and does not change the training process as compared to other Bayesian approaches. We minimize noise in the data by employing spelling correction models before classifying the input. Further, the labels for the user queries are human-curated with minimal error. Hence, our focus is on quantifying epistemic uncertainty in AVA, rather than aleatoric uncertainty [57]. We use mixed-integer optimization to find a threshold for human escalation of a user query based on the mean prediction and the uncertainty of the prediction. This optimization step, once again, does not require modifications to the network architecture and can be implemented separately from model training. In other contexts, it might be fruitful to have an integrated escalation option in the neural network [58], and we leave the trade-offs of integrated reject option and non-Bayesian approaches for future work.

Similar approaches in spelling correction, besides those mentioned in **Section 1**, are reported in Deep Text Corrector [59] that applies a seq2seq model to automatically correct small grammatical errors in conversational written English. Optimal decision threshold learning under uncertainty is studied in Ref [60] as reinforcement learning and iterative Bayesian optimization formulations.

3 SYSTEM OVERVIEW AND DATA SETS

3.1 Overview of the System

Figure 1 illustrates system overview of AVA. The proposed conversational AI will gradually replace the traditional human–human interactions between phone agents and internal experts and eventually allow clients self-provisioning interaction directly to the AI system. Now, phone agents interact with AVA chatbots deployed on Microsoft Teams in our company intranet, and their questions are preprocessed by a sentence completion model (introduced in **Section 6**) to correct misspellings. Then, inputs are classified by an intent classification model (**Sections 4**, **Sections 5**), where relevant questions are assigned predicted intent labels, and downstream information retrieval and questioning answering modules are triggered to extract answers from a document repository. Escalation questions are escalated to human experts following the decision thresholds optimized using methods introduced in **Section 5**. This article only discusses the intent classification model and the sentence completion model.

3.2 Data for Intent Classification Model

Training data for AVA's intent classification model is collected, curated, and generated by a dedicated business team from interaction logs between phone agents and the expert team. The whole process takes about one year to finish. In total, 22,630 questions are selected and classified to 381 intents, which compose the *relevant question set* for the intent classification model. Additionally, 17,395 questions are manually synthesized as *escalation questions*, and none of them belongs to any of the aforementioned 381 intents. Each *relevant question* is hierarchically assigned with three labels from Tier 1 to Tier 3. In this hierarchy, there are five unique Tier-1 labels, 107 Tier-2 labels, and 381 Tier-3 labels. Our intent classification model is designed to classify relevant input questions into 381 Tier-3 intents and then trigger downstream models to extract appropriate responses. The five Tier-1 labels and the numbers of intents included in each label are *account maintenance* (9,074), *account permissions* (2,961), *transfer of assets* (2,838), *banking* (4,788), *tax FAQ* (2,969). At Tier-1, general business issues across intents are very different, but at the Tier-3 level, questions are quite similar to each other, where differences are merely at the specific responses. Escalation questions, compared to relevant questions, have two main characteristics:

- Some questions are relevant to business intents but unsuitable to be processed by conversational AI. For example, in **Table 1**, question "*How can we get into an account with only one security question?*" is related to *call authentication* in *account permission*, but its response needs further human diagnosis to collect more information. These types of questions should be escalated to human experts.
- Out-of-scope questions. For example, questions like "*What is the best place to learn about Vanguard's investment philosophy?*" or "*What is a hippopotamus?*" are totally outside the scope of our training data, but they may still occur in real-world interactions.

3.3 Textual Data for Pretrained Embeddings and Sentence Completion Model

Inspired by the progress in computer vision, transfer learning has been very successful in NLP community and has become a common practice. Initializing deep neural network with pretrained embeddings and fine-tuning the models toward

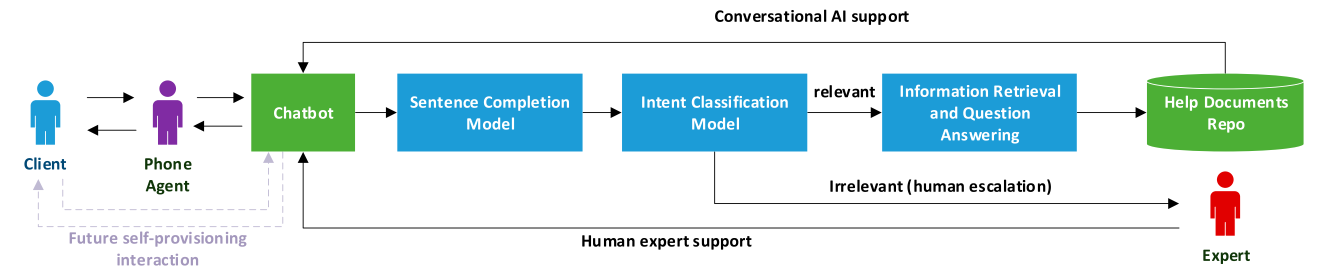

Conversational AI support
Client
Phone Agent
Chatbot
Sentence Completion Model
Intent Classification Model
relevant
Information Retrieval and Question Answering
Help Documents Repo
Irrelevant (human escalation)
Expert
Human expert support
Future self-provisioning interaction

FIGURE 1 | End-to-end conceptual diagram of AVA.

TABLE 1 | Example questions used in AVA intent classification model training.

T1 label	T2 label	T3 label	Questions
Account maintenance	Call authentication	Type 2	Am I allowed to give the client their social security number?
	Call authentication	Type 5	Do the web security questions need to be reset by the client if their web access is blocked?
	Web reset	Type 1	How many security questions are required to be asked to reset a client's web security questions?
Account permission	Call authentication	Type 2	How are the web security questions used to authenticate a client?
	Agent incapactiated	Type 3	Is it possible to set up agent certification for an incapacitated person on an individual Roth 401 k?
TAX FAQ	Miscellaneous	What is	Do I need my social security number on the 1099MISC form?
Transfer of asset	Unlike registrations	Type 2	Does the client need to provide special documentation if they want to transfer from one account to another account?
	Brokerage transfer	Type 3	Is there a list of items that need to be included on a statement to transfer an account?
Banking	Add owner	Type 4	Once a bank has been declined how can we authorize it?
	Add/change/delete	Type 3	Does a limited agent have authorization to adjust bank info?
Escalation	–	–	How can we get into an account with only one security question?
	–	–	Am I able to use my Roth IRA to set up a margin account?
	–	–	What is the best place to learn about Vanguard's investment philosophy?

TABLE 2 | Comparison of intent classification performance. BERT and XLNet models were all trained for 30 epochs using batch size 16.

Model	Performance
BERT small + SharePoint embeddings	0.944
BERT small + Google embeddings	0.949
BERT large + Google embeddings	0.954
XLNet large + Google embeddings	0.927
LSTM with attention + Word2Vec	0.913
LSTM + Word2Vec	0.892
Logistic regression + TFIDF	0.820
Xgboost + TFIDF	0.760
Naive Bayes + TFIDF	0.661

task-specific data are proven methods in multitask NLP learning. In our approach, besides applying off-the-shelf embeddings from Google BERT and XLNet, we also pretrain BERT embeddings using our company's proprietary text to capture special semantic meanings of words in the financial domain. Three types of textual data sets are used for embeddings training:

- SharePoint text: About 3.2G bytes of corpora scraped from our company's internal SharePoint websites, including Web pages, Word documents, ppt slides, pdf documents, and notes from internal CRM systems.
- Emails: About 8G bytes of customer service emails are extracted.
- Phone call transcriptions: We apply AWS to transcribe 500 K client service phone calls, and the transcription text is used for training.

All embeddings are trained in case-insensitive settings. Attention and hidden layer dropout probabilities are set to 0.1, hidden size is 768, attention heads and hidden layers are set to 12, and vocabulary size is 32,000 using SentencePiece tokenizer. On AWS P3.2xlarge instance, each embeddings is trained for one million iterations and takes about one week CPU time to finish. More details about parameter selection for pretraining are available in the GitHub code. The same pretrained embeddings are used to initialize BERT model training in intent classification and also used as language models in sentence completion.

4 INTENT CLASSIFICATION PERFORMANCE ON RELEVANT QUESTIONS

Using only relevant questions, we compare various popular model architectures to find one with the best performance on 5-fold validation. Not surprisingly, BERT models generally

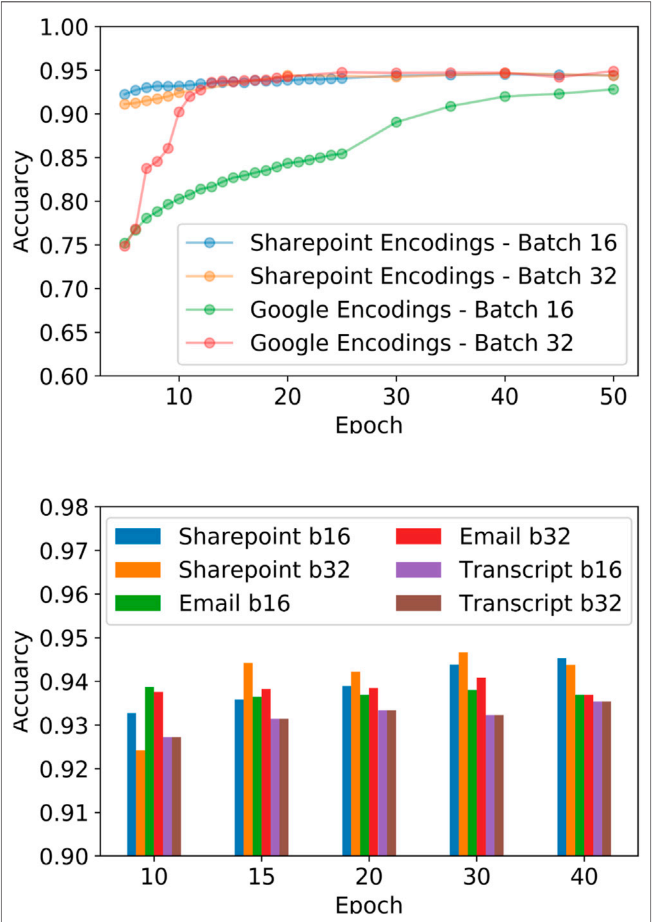


FIGURE 2 | Comparison of test set accuracy using different embeddings and batch sizes.

produce much better performance than other models (**Table 2**). Large BERT (24-layer, 1024-hidden, and 16-heads) has a slight improvement over small BERT (12-layer, 768-hidden, and 12-heads) but less preferred because of expensive computations. To our surprise, XLNet, a model reported outperforming BERT in multitask NLP, performs 2 percent lower on our data.

BERT models initialized by proprietary embeddings converge faster than those initialized by off-the-shelf embeddings (**Figure 2A**). And embeddings trained on company's SharePoint text perform better than those built on Emails and phone call transcriptions (**Figure 2B**). Using larger batch size 32) enables models to converge faster and leads to better performance.

5 INTENT CLASSIFICATION PERFORMANCE INCLUDING ESCALATION QUESTIONS

We have shown how the BERT model outperforms other models on real data sets that only contain relevant questions. The capability to handle 381 intents simultaneously at 94.5% accuracy makes it an ideal intent classifier candidate in a chatbot. This section describes how we quantify uncertainties on BERT predictions and enable the bot to detect escalation questions. Three approaches are compared:

- Predictive entropy: We measure uncertainty of predictions using Shannon entropy $H = -\sum_{k=1}^{K} p_{ik} \log p_{ik}$, where p_{ik} is the prediction probability of ith sample to kth class. Here, p_{ik} is softmax output of the BERT network [56]. A higher predictive entropy corresponds to a greater degree of uncertainty. Then, an optimally chosen cutoff threshold applied on entropies should be able to separate the majority of in-sample questions and escalation questions.
- Dropout: We apply Monte Carlo (MC) dropout by doing 100 Monte Carlo samples. At each inference iteration, a certain percent of the set of units drop out. This generates random predictions, which are interpreted as samples from a probabilistic distribution [22]. Since we do not employ regularization in our network, τ^{-1} in Eq. 7 in Ref [22] is effectively zero and the predictive variance is equal to the sample variance from stochastic passes. We could then investigate the distributions and interpret model uncertainty as mean probabilities and variances.
- Dummy class: We simply treat escalation questions as a dummy class to distinguish them from original questions. Unlike entropy and dropout, this approach requires retraining of BERT models on the expanded data set including dummy class questions.

5.1 Experimental Setup

All results mentioned in this section are obtained using BERT small + SharePoint embeddings (batch size 16). In entropy and dropout approaches, both relevant questions and escalation questions are split into five folds, where four folds (80%) of relevant questions are used to train the BERT model. Then, among that 20% held-out relevant questions, we further split them into five folds, where 80% of them (equal to 16% of the entire relevant question set) are combined with four folds of escalation questions to learn the optimal decision variables. The learned decision variables are applied on BERT predictions of the remaining 20% (906) of held-out relevant questions and held-out escalation questions (4,000), to obtain the test performance. In the dummy class approach, the BERT model is trained using four folds of relevant questions plus four folds of escalation questions and tested on the same amount of test questions as entropy and dropout approaches.

5.2 Optimizing Entropy Decision Threshold

To find the optimal threshold cutoff b, we consider the following quadratic mixed-integer programming problem

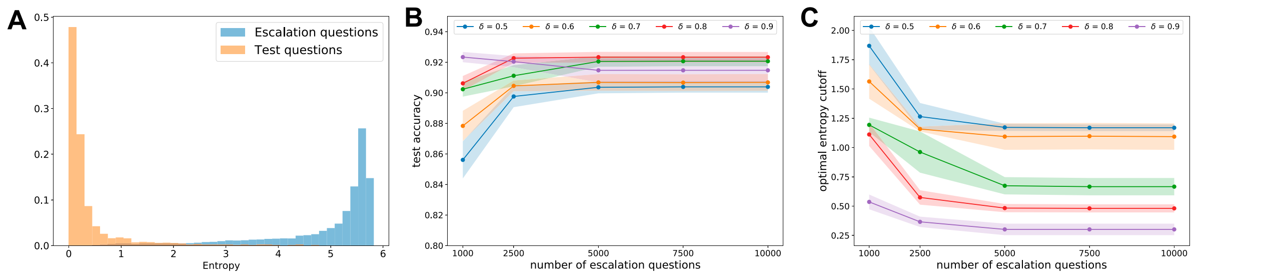


FIGURE 3 | Optimizing the entropy threshold to detect escalation questions. As shown in **(A)**, in-sample test questions and escalation questions have very different distributions of predictive entropies. Subfigure **(B)** shows how test accuracies, evaluated using decision variables b solved by **(1)** on BERT predictions on test data, change when different numbers of escalation questions are involved in training. Subfigure **(C)** shows the impact of δ on the optimized thresholds when the number of escalation questions increase optimization. Usually, to safeguard making wrong predictions in client-facing applications, δ is set to a value smaller than 1 because 1 means the cost of making wrong predictions is the same as spending human effort on a question. In contrast, a value 0.5 means the cost of wrong predictions is two times larger than human answering cost. Such a cost is guided by business reasons, and different δ could lead to different optimal thresholds.

$$\begin{array}{lll} \min\limits_{x,b} & \sum_{i,k}(x_{ik}-l_{ik})^2 & \\ s.t. & x_{ik}=0 & \textit{if } E_i \ge b, \text{ for } k \text{ in } 1,\ldots,K \\ & x_{ik}=1 & \textit{if } E_i \ge b, \text{ for } k = K+1 \\ & x_{ik}\in\{0,1\} & \\ & \sum_{k=1}^{K+1} x_{ik}=1 & \forall i \text{ in } 1,\ldots,N \\ & b\ge 0 & \end{array} . \quad (1)$$

to minimize the quadratic loss between the predictive assignments x_{ik} and true labels l_{ik}. In **Eq. 1**, i is the sample index, k is class (intent) indices, x_{ik} is $N\times(K+1)$ binary matrix, and l_{ik} is also $N\times(K+1)$, where the first K columns are binary values and the last column is a uniform vector δ, which represents the cost of escalating questions. Normally, δ is a constant value smaller than 1, which encourages the bot to escalate questions, rather than making mistaken predictions. The first and second constraints of **Eq. 1** force an escalation label when entropy $E_i \ge b$. The third and fourth constraints restrict x_{ik} as binary variables and ensure the sum for each sample is 1. Experimental results (**Figure 3**) indicate that **Eq. 1** needs more than 5,000 escalation questions to learn a stabilized b. The value of escalation cost δ has a significant impact on the optimal b value and in our implementation is set to 0.5.

5.3 Monte Carlo Dropout

In the BERT model, dropout ratios can be customized at encoding, decoding, attention, and output layers. A combinatorial search for optimal dropout ratios is computationally challenging. Results reported in the article are obtained through simplifications with the same dropout ratio assigned and varied on all layers. Our MC dropout experiments are conducted as follows:

1. Change dropout ratios in encoding/decoding/attention/output layer of BERT
2. Train the BERT model on 80% of relevant questions for 10 or 30 epochs
3. Export and serve the trained model by TensorFlow serving
4. Repeat inference 100 times on questions, average the results per each question to obtain mean probabilities and standard deviations, and then average the deviations for a set of questions.

According to the experimental results illustrated in **Figure 4**, we make three conclusions: 1) Epistemic uncertainty estimated by MCD reflects question relevance: when inputs are similar to the training data, there will be low uncertainty, while data are different from the original, training data should have higher epistemic uncertainty. 2) Converged models (more training epochs) should have similar uncertainty and accuracy no matter what drop ratio is used. 3) The number of epochs and dropout ratios are important hyper-parameters that have substantial impacts on uncertainty measure and predictive accuracy and should be cross-validated in real applications.

$$\begin{array}{ll} \min\limits_{x,c,d} & \sum_{i,k}(x_{ik}-l_{ik})^2 \\ \text{s.t.} & \alpha_{ik}=\begin{cases}0 & \text{if } P_{ik}\le c, \text{ for } k \text{ in } 1,\ldots,K \\ 1 & \text{if otherwise}\end{cases} \\ & \beta_{ik}=\begin{cases}0 & \text{if } V_{ik}\ge d, \text{ for } k \text{ in } 1,\ldots,K \\ 1 & \text{if otherwise}\end{cases} \\ & x_{ik}=0 \quad \text{if } \alpha_{ik}=0 \text{ OR } \beta_{ik}=0 \\ & x_{ik}=1 \quad \text{if } \alpha_{ik}=1 \text{ AND } \beta_{ik}=1 \\ & \sum_k^{K+1} x_{ik}=1 \quad \forall i \text{ in } 1,\ldots,N \\ & 1\ge c\ge 0 \\ & 1\ge d\ge 0 \end{array} . \quad (2)$$

We use mean probabilities and standard deviations obtained from models where dropout ratios are set to 10% after 30 epochs of training to learn optimal decision thresholds. Our goal is to optimize lower bound c and upper bound d and designate a question as relevant only when the mean predictive probability P_{ik} is larger than c and standard deviation V_{ik} is lower than d. Optimizing c and d, on a 381-class problem, is much more computationally challenging than learning entropy threshold because the number of constraints is proportional to class number. As shown in **Eq. 2**, we introduce two variables α and β to indicate the status of mean probability and deviation conditions, and the final assignment variable x is the logical AND of α and β. Solving 2) with more than 10 k samples is very slow (shown in **Supplementary Appendix**), so we use 1,500 original relevant questions and increase the number of escalation questions from 100 to 3,000. For performance testing, the optimized c and d are applied as decision variables

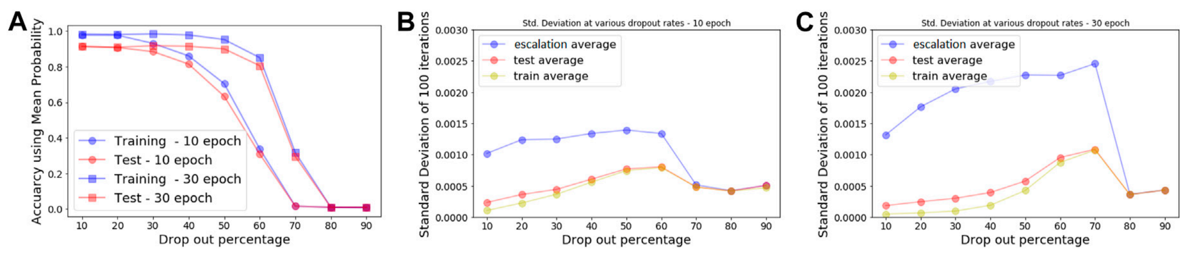


FIGURE 4 | Classification accuracy and uncertainties obtained from Monte Carlo dropout.

TABLE 3 | Performance of cross-comparison of three approaches evaluated on test data of the same size (906 relevant questions plus 4,000 escalation questions). Precision/recall/F1 scores were calculated assuming relevant questions are true positives. In entropy and dropout optimization processes, δ is set to 0.5. Other delta values for the dropout approach are listed in **Supplementary Appendix**.

Number of escalation questions in training	**Entropy**				**Dropout**				**Dummy class**			
	1,000	**5,000**	**8,000**	**10,000**	**100**	**1,000**	**2000**	**3,000**	**1,000**	**5,000**	**8,000**	**10,000**
Optimal entropy cutoff *b*	2.36	1.13	0.85	0.55	–	–	–	–	–	–	–	–
Optimal mean probability cutoff *c*	–	–	–	–	0.8172	0.6654	0.7921	0.0459	–	–	–	–
Optimal standard cutoff *d*	–	–	–	–	0.1533	0.0250	0.0261	0.0132	–	–	–	–
Mean accuracy in 381 classes	91.9%	88.3%	85.6%	81.7%	88.41%	80.13%	80.24%	74.72%	94.2%	**93.7%**	87.7%	82%
Accuracy of the dummy class	79.25%	91.2%	93.25%	95.2%	86.69%	91.83%	91.95%	92.57%	73.6%	**94.5%**	99.4%	99.6%
Precision (binary classification)	51.4%	70.2%	74.7%	79.8%	90.7%	68.8%	68.9%	63.7%	81%	**95.3%**	99.5%	99.6%
Recall (binary classification)	96.7%	91.3%	88.1%	83.5%	93.9%	82.7%	83.2%	84.7%	99.7%	**98.7%**	92.6%	86%
F1 score (binary classification)	0.671	0.794	0.808	0.816	0.738	0.751	0.754	0.727	0.894	**0.967**	0.959	0.923

Bold values represent best performance.

on samples of BERT predictions on test data. Performance from dropout is presented in **Table 3** and **Supplementary Appendix**. Our results showed a decision threshold optimized from **Eq. 2** involving 2000 escalation questions and gave the best F1 score (0.754), and we validated it using the grid search and confirmed its optimality (shown in **Supplementary Appendix**).

5.4 Dummy Class Classification

Our third approach is to train a binary classifier using both relevant questions and escalation questions in the BERT model. We use a dummy class to represent those 17,395 escalation questions and split the entire data sets, including relevant and escalation, into five folds for training and test.

Performance of the dummy class approach is compared with entropy and dropout approaches (**Table 3**). Deciding an optimal number of escalation questions involved in threshold learning is non-trivial, especially for entropy and dummy class approaches. Dropout does not need as many escalation questions as entropy does to learn the optimal threshold mainly because the number of constraints in **Eq. 2** is proportional to the class number (381), so the number of constraints is large enough to learn a suitable threshold on small samples. (To support this conclusion, we present extensive studies in **Supplementary Appendix** on a 5-class classifier using Tier one intents.) The dummy class approach obtains the best performance, but its success assumes the learned decision boundary can be generalized well to any new escalation questions, which is often not valid in real applications. In contrast, entropy and dropout approaches only need to treat a binary problem in the optimization and leave the intent classification model intact. The optimization problem for entropy approach can be solved much more efficiently and is selected as the solution for our final implementation.

It is certainly possible to combine dropout and entropy approach, for example, to optimize thresholds on entropy calculated from the average mean of MCD dropout predictions. Furthermore, it is possible that the problem defined in **Eq. 2** can be simplified by proper reformulation and can be solved more efficiently, which will be explored in our future works.

6 SENTENCE COMPLETION USING LANGUAGE MODEL

6.1 Algorithm

We assume misspelled words are all OOV words, and we can transform them as [MASK] tokens and use bidirectional language models to predict them. Predicting masked word within sentences is an inherent objective of a pretrained bidirectional model, and we utilize the masked language model API in the Transformer package [61] to generate the ranked list of candidate words for each [MASK] position. The sentence completion algorithm is illustrated in Algorithm 1.

6.2 Experimental Setup

For each question, we randomly permutate two characters in the longest word, the next longest word, and so on. In this way, we generate one to three synthetic misspellings in each question. We investigate intent classification accuracy changes on these questions, and how our sentence completion model can prevent performance changes. All models are trained using relevant data (80%) without misspellings and validated on synthetic misspelled test data. Five settings are compared: 1) no correction: classification performance without applying any autocorrection; 2) no LM: autocorrections made only by word edit distance without using masked language model; 3) BERT SharePoint: autocorrections made by masked LM using pretrained SharePoint embeddings together with word edit distance; 4) BERT Email: autocorrections using pretrained email embeddings together with word edit distance; and 5) BERT Google: autocorrections using pretrained Google small uncased embedding data together with word edit distance.

Algorithm 1 Auto-correction of Multiple OOV tokens. Assuming there are d OOV tokens, we first find top B combinations using beam search that maximizes joint probability $\prod_{i=1}^{d} P_i$, then among them select the combination that has the smallest accumulated Levenshtein distances to thoses OOV tokens.

```
1:  procedure AUTO-CORRECT($< t_0, .., t_N >, I$)          ▷ $I$ is the list of OOV indices
2:      initialize $P \in D_{M \times d}$      ▷ Prob. Matrix of candidate tokens
3:      initialize $\hat{P} \in D_{B \times d}$                ▷ Optimal prob. Matrix
4:      for $i$ in $I_{1xd}$ do
5:          $oov_i \leftarrow t_i$
6:          $t_i \leftarrow$ [MASK]
7:      end for
8:      $W, P \leftarrow$ MaskedLM($t_0, .., t_N, M$)
9:      for $w_{j,-}$ in $\hat{W}$ do             ▷ iterate top M candidates
10:         $d_{j,-} = \sum_i$ WordEditDistance($w_{ji}, oov_i$)
11:     end for
12:     $\hat{j} = \arg\min_j d_{j,-}$       ▷ The candidate with shortest edit distance
13:     for $i$ in $I_{1xd}$ do
14:         $t_i = w_{\hat{j}i}$                  ▷ Autocorrect the oov token
15:     end for
16:     return $< t_0, .., t_N >$ ▷ Return the autocompleted sentence
17: end procedure
```

We also need to decide what is an OOV or what should be included in our vocabulary. After experiments, we set our vocabulary as words from four categories: 1) All words in the pretrained embeddings; 2) all words that appear in training questions; 3) words that are all capitalized because they are likely to be proper nouns, fund tickers, or service products; 4) all words start with numbers because they can be tax forms or specific products (e.g., 1099b and 401 k). The purposes of including 3) and 4) are to avoid autocorrection on those keywords that may represent significant intents. Any word falls outside these four groups is considered as an OOV. During our implementation, we keep monitoring the OOV rate, defined as the ratio of OOV occurrences to total word counts in recent 24 h. When it is higher than 1%, we apply manual intervention to check chatbot log data.

We also need to determine two additional parameters M, the number of candidate tokens prioritized by masked language model, and B, the beam size in our sentence completion model. In our approach, we set M and B to the same value, and it is benchmarked from 1 to 10 k by test sample accuracy. Notice that when M and B are large, and when there are more than two OOVs, beam search becomes very inefficient in Algorithm 1. To simplify this, instead of finding the optimal combinations of candidate tokens that maximize the joint probability $\text{argmax}\prod_{i=1}^{d} p_i$, we assume they are independent and apply a simplified algorithm (shown in **Supplementary Appendix**) on single OOV separately.

In additional to BERT, we also implemented a conventional spelling correction algorithm using Google Web 1 T n-gram [62]. We used the longest common subsequence (LCS) string matching algorithm [63] and compared a variety of best combinations of n-grams report in the article. The experimental setting is identical as the one we set up for BERT models: We apply auto-spelling correction algorithms on synthetic misspelled test data (20%), and then the intent classification accuracy performance is evaluated using the BERT SharePoint model trained on 80% relevant data without misspellings for 10 epochs. As shown in **Table 4**, n-gram models do not provide comparable performance as BERT language models, and the most complicated hybrid n-gram models (5-4-3 g and 5-4-3-2 g) [63] is not comparable to Google BERT model and far worse than BERT SharePoint model.

An further improved version of sentence completion algorithm to maximize joint probability is our future research. In this article, we have not considered situations when misspellings are not OOV. Detecting improper words or improper grammar in a sentence may need evaluation of metrics such as perplexity or sensibleness and specificity average (SSA) [10], and the simple word matching algorithm can be much generalized as reinforcement learning–based approach [64].

TABLE 4 | Comparison of intent classification accuracy using the best BERT model vs. conventional n-gram models.

Model	Single OOV	Two OOVs	Three OOVs
BERT SharePoint	0.934	0.882	0.849
5 G	0.755	0.719	0.622
5-4 G	0.817	0.731	0.643
5-4-3 G	0.823	0.752	0.646
5-4-3-2 G	0.826	0.755	0.643

TABLE 5 | Benchmark of intent classification API performance across different models in real-time application. Each model is tested using 10 threads, simulating 10 concurrent users, for a duration of 10 min. In this test, models are not served as Monte Carlo sampling, so the inference is done only once. All models are hosted on identical AWS m5.4xlarge CPU instances. As seen, the simplest model (6A-6H, six attention layers and six hidden layers) can have a double throughput rate and half latency than the original BERT small model, and the accuracy performance only drops 1.6%. The performance is evaluated using JMeter at the client side, and APIs are served using Domino Lab 3.6.17 Model API. Throughput indicates how many API responses being made per second. Latency is measured as time elapse between request sent till response received at client side.

Model	Performance	Throughput	Average latency (ms)
12A-12H	0.944	8.9/s	1,117
6A-12H	0.941	9.0/s	1,108
12A-9H	0.934	11.8/s	843
3A-9H	0.933	12.0/s	831
3A-12H	0.930	9.1/s	1,097
6A-6H	0.928	18.1/s	552

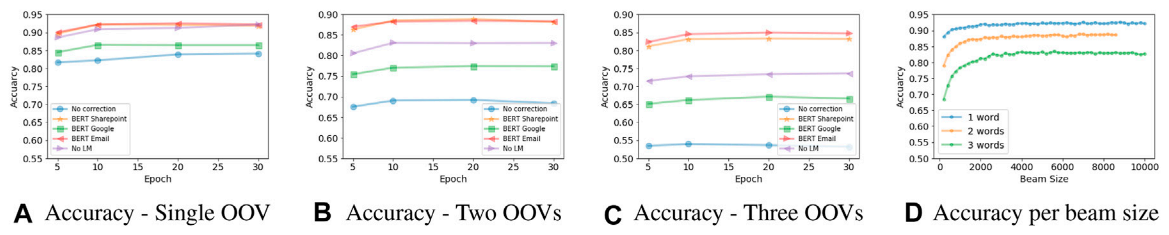


FIGURE 5 | As expected, misspelled words can significantly decrease intent classification performance. The same BERT model that achieved 94% on clean data, dropped to 83.5% when a single OOV occurred in each question. It further dropped to 68 and 52%, respectively, when two and three OOVs occurred. In all experiments, LM models proved being useful to help correcting words and reduce performance drop, while domain-specific embeddings trained on Vanguard SharePoint and Email text outperform off-the-shelf Google embeddings. The beam size *B* (*M*) was benchmarked as results shown in subfigure **(D)** and was set to 4,000 to generate results in subfigure **(A–C)**.

6.3 RESULTS

According to the experimental results illustrated in **Figure 5**, pretrained embeddings are useful to increase the robustness of intent prediction on noisy inputs. Domain-specific embeddings contain much richer context-dependent semantics that helps OOVs get properly corrected and leads to better task-oriented intent classification performance. Benchmark shows B $\geq$ 4000 leads to the best performance for our problem. Based on this, we apply SharePoint embeddings as the language model in our sentence completion module.

7 IMPLEMENTATION

The chatbot has been implemented fully inside our company network using open-source tools including RASA [65], TensorFlow, and PyTorch in Python environment. All backend models (sentence completion model, intent classification model, and others) are deployed as RESTful APIs in AWS SageMaker. The front end of chatbot is launched on Microsoft Teams, powered by Microsoft Bot Framework and Microsoft Azure Directory, and connected to backend APIs in AWS environment. All our BERT model trainings, including embeddings pretraining, are based on BERT TensorFlow running on AWS P3.2xlarge instance. The optimization procedure uses Gurobi 8.1 running on AWS C5.18xlarge instance. The BERT language model API in the sentence completion model is developed using Transformer 2.1.1 package on PyTorch 1.2 and TensorFlow 2.0.

During our implementation, we further explore how the intent classification model API can be served in real applications under budget. We gradually reduce the numbers of attention layer and hidden layer in the original BERT small model (12 hidden layers and 12 attention heads) and create several smaller models. By reducing the number of hidden layers and attention layers in half, we see a remarkable 100% increase in performance (double the throughput and half the latency) with the cost of only 1.6% drop in intent classification performance (**Table 5**).

8 CONCLUSION

Our results demonstrate that optimized uncertainty thresholds applied on BERT model predictions are promising to escalate escalation questions in task-oriented chatbot implementation, meanwhile the state-of-the-art deep learning architecture provides high accuracy on classifying into a large number of intents. Another feature we contribute is the application of BERT embeddings as the language model to automatically correct small spelling errors in noisy inputs, and we show its effectiveness in reducing intent classification errors. The entire end-to-end conversational AI system, including two machine learning models presented in this article, is developed using open-source tools and deployed as in-house solution. We believe those discussions provide useful guidance to companies that are motivated to reduce dependency on vendors by leveraging state-of-the-art open-source AI solutions in their business.

We will continue our explorations in this direction, with particular focuses on the following issues: 1) Current fine-tuning and decision threshold learning are two separate parts, and we will explore the possibility to combine them as a new cost function in BERT model optimization. 2) Dropout methodology applied in our article belongs to approximated inference methods, which is a crude approximation to the exact posterior learning in parameter space. We are interested in a Bayesian version of BERT, which requires a new architecture based on variational inference using tools like TFP TensorFlow Probability. 3) Maintaining chatbot production system would need a complex pipeline to continuously transfer and integrate features from deployed model to new versions for new business needs, which is an uncharted territory for all of us. 4) Hybridizing "chitchat" bots, using state-of-the-art progresses in deep neural models, with task-oriented machine learning models is important for our preparation of client self-provisioning service.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusion of this article is available at https://github.com/cyberyu/ava.

AUTHOR CONTRIBUTIONS

SY: main author, deployed the model and wrote the paper. Corresponding author YC: deployed the model and wrote the paper HZ: deployed the model and wrote the paper.

ACKNOWLEDGMENTS

We thank our colleagues in Vanguard CAI (ML-DS team and IT team) for their seamless collaboration and support. We thank colleagues in Vanguard Retail Group (IT/Digital, Customer Care) for their pioneering effort collecting and curating all the data used in our approach. We thank Robert Fieldhouse, Sean Carpenter, Ken Reeser and Brain Heckman for the fruitful discussions and experiments.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fams.2021.604842/full#supplementary-material

REFERENCES

1. Devlin J, Chang M-W, Lee K, and Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the NACL, Vol 1 (2019). p. 4171–86.
2. Colby KM, Weber S, and Hilf FD. Artificial Paranoia. *Artif Intelligence* (1971). 2(1):1–25. ISSN. doi:10.1016/0004-3702(71)90002-6
3. Weizenbaum J. ELIZA-a Computer Program for the Study of Natural Language Communication between Man and Machine. *Commun ACM* (1966). 9(1):36–45. ISSN. doi:10.1145/365153.365168
4. Worswick S. Mitsuku (2019). Available at: http://www.mitsuku.com.
5. Gao J, Galley M, and Li L. *Neural Approaches to Conversational Ai*. SIGIR '18 (2018).
6. Fedorenko DG, Smetanin N, and Rodichev A (2017). Avoiding echo-responses in a Retrieval-Based Conversation System. In: Conference on Artificial Intelligence and Natural Language. p. 91–7.
7. Microsoft (2019). Microsoft. Zo. Available at: https://www.zo.ai.
8. Serban IV, Sankar C, Germain M, Zhang S, Lin Z, Subramanian S, et al. (2017). A Deep Reinforcement Learning Chatbot. *CoRR* Available at: http://arxiv.org/abs/1709.02349.
9. Zhou L, Gao J, Li D, and Shum H (2018). The Design and Implementation of Xiaoice, an Empathetic Social Chatbot. *CoRR, abs/1812.08989*.
10. Adiwardana D, Luong M-T, So DR, Hall J, Fiedel N, Thoppilan R, et al. (2020). *Towards a Human-like Open-Domain Chatbot*. Available at: https://arxiv.org/abs/2001.09977
11. Larson S, Mahendran A, Peper JJ, Clarke C, Lee A, Hill P, et al. (2019). An Evaluation Dataset for Intent Classification and Out-Of-Scope Prediction. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics.
12. Sethi S (2019). The State of Chatbots in 2019. Available at: https://hackernoon.com/the-state-of-chatbots-in-2019-d97f85f2294b.
13. Dai AM, and Le QV (2015). Semi-supervised Sequence Learning. In: In Proceedings of Advances in Neural Information Processing Systems 28. p. 3079–87.
14. Howard J, and Ruder S (2018). Universal Language Model fine-tuning for Text Classification. In: Proceedings of the 56th Annual Meeting of the ACL, Melbourne, Australia, July 2018. p. 328–39.
15. Lample G, and Conneau A (2019). Cross-lingual Language Model Pretraining. *CoRR, abs/1901.07291*.
16. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, et al. (2019). Roberta: A Robustly Optimized BERT Pretraining Approach. *CoRR, abs/1907.11692*.
17. Peters ME, Neumann M, Zettlemoyer L, and Yih W (2018). Dissecting Contextual Word Embeddings: Architecture and Representation. *CoRR, abs/1808.08949*.
18. Peters M, Ammar W, Bhagavatula C, and Power R. Semi-supervised Sequence Tagging with Bidirectional Language Models. In: Proceedings of the 55th ACL. Vancouver, Canada (2017). p. 1756–65.
19. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, et al. (2018). Deep Contextualized Word Representations. In: Proceedings of the 2018 Conference of the NAACL. New Orleans, Louisiana. p. 2227–37.
20. Tang R, Lu Y, Liu L, Mou L, Vechtomova O, and Lin J (2019). Distilling Task-specific Knowledge from BERT into Simple Neural Networks. *CoRR, abs/1903.12136*.
21. Yang Z, Dai Z, Yang Y, Carbonell JG, Salakhutdinov R, and Le QV (2019). Xlnet: Generalized Autoregressive Pretraining for Language Understanding. *CoRR, abs/1906.08237*.
22. Gal Y, and Ghahramani Z (2016). Dropout as Bayesian Approximation: Representing Model Uncertainty in Deep Learning 48:1050.
23. Maddox WJ, Garipov T, Izmailov P, Vetrov DP, and Wilson AG (2019). A Simple Baseline for Bayesian Uncertainty in Deep Learning. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett. editors. *Advances in Neural Information Processing Systems*. Red Hook, NYCurran Associates, Inc.
24. Pearce T, Zaki M, Brintrup A, and Neely A (2018). Uncertainty in Neural Networks: Bayesian Ensembling. *ArXiv, abs/1810.05546*.
25. Shridhar K, Laumann F, and Liwicki M (2019). A Comprehensive Guide to Bayesian Convolutional Neural Network with Variational Inference. *CoRR, abs/1901.02731*.
26. Li C, Stevens A, Chen C, Pu Y, Gan Z, and Carin L (2016). Learning Weight Uncertainty with Stochastic Gradient Mcmc for Shape Classification. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). p. 5666–75.
27. Park C, Kim J, Ha SH, and Lee J (2018). Sampling-based Bayesian Inference with Gradient Uncertainty. *CoRR, abs/1812.03285*.
28. Rao Q, and Frtunikj J (2018). Deep Learning for Self-Driving Cars: Chances and Challenges. In: 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS). Los Alamitos, CA: IEEE Computer Society. p. 35–8.
29. Seedat N, and Kanan C (2019). Towards Calibrated and Scalable Uncertainty Representations for Neural Networks. *ArXiv, abs/1911.00104*.
30. Welling M, and Teh YW (2011). Bayesian Learning via Stochastic Gradient Langevin Dynamics. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, ISBN 9781450306195. Madison, WI, USA: Omnipress. p. 681–8.
31. Blundell C, Cornebise J, Kavukcuoglu K, and Wierstra D (2015). Weight Uncertainty in Neural Network. In: B. Francis and B. David. editors. Proceedings of the 32nd International Conference on Machine Learning. PMLR p. 1613–22.
32. Graves A (2011). Practical Variational Inference for Neural Networks. *Adv Neural Inf Process Syst* 24:2348–56. doi:10.1016/s0893-6080(10)00238-8
33. Hernández-Lobato JM, and Adams RP (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In: Proceedings of the 32nd ICML, Vol 37, ICML'15, Lille, France. p. 1861–9.
34. Ye N, and Zhu Z (2019). Functional Bayesian Neural Networks for Model Uncertainty Quantification. Available at: https://openreview.net/forum?id=SJxFN3RcFX.
35. Miok K, Skrlj B, Zaharie D, and Robnik-Sikonja M (2020). To Ban or Not to Ban: Bayesian Attention Networks for Reliable Hate Speech Detection. *arXiv: Appl.*
36. Lin Y, Michel J-B, Aiden EL, Orwant J, Brockman W, and Petrov S (2012). Index. In: Proceedings of the ACL 2012 System Demonstrations. USA. p. 169–74. doi:10.2307/j.ctv18pgr3b.13
37. Schuster M, and Nakajima K (2012). Japanese and Korean Voice Search. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). p. 5149–52. doi:10.1109/ICASSP.2012.6289079
38. Gage P (1994). A New Algorithm for Data Compression. *C Users J* 12(2): 23–38.
39. Sennrich R, Haddow B, and Birch A. Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th ACL, Berlin, Germany. Stroudsburg, PA: Association for Computational Linguistics (2016). p. 1715–25.
40. Mikolov T, Sutskever I, Chen K, Corrado G, and Dean J (2013). Distributed Representations of Words and Phrases and Their Compositionality. *CoRR, abs/1310.4546*.
41. Mikolová T, Karafit M, Burget L, Cernocký J, and Khudanpur S (2010). Recurrent Neural Network Based Language Model. In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*. Makuhari, Japan 2:1045–8.
42. dos Santos C, and Gatti M (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In: Proceedings of the 25th International Conference on Computational Linguistics. Dublin, Ireland. p. 69–78.

43. Dos Santos CN, and Zadrozny B (2014). Learning Character-Level Representations for Part-Of-Speech Tagging. In: Proceedings of the 31st International Conference on Machine Learning - Volume 32, ICML'14, Beijing, China. p. II–1818–II–1826.
44. Kim Y, Jernite Y, Sontag DA, and Rush AM (2015). Character-aware Neural Language Models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 2741–9.
45. Radford A, and Sutskever I (2018). Improving Language Understanding by Generative Pre-training. Available at: https://openai.com/blog/language-unsupervised/.
46. Collobert R, and Weston J. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In: Proceedings of the 25th ICML, Helsinki, Finland. ACM Press (2008). p. 160–7. doi:10.1145/1390156.1390177
47. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, and Kuksa PP (2011). Natural Language Processing (Almost) from Scratch. *J Mach Learn Res* 12:2493–537.
48. Elman JL (1990). Finding Structure in Time. *Cognitive Science* 14(2):179–211. doi:10.1207/s15516709cog1402_1
49. Bahdanau D, Cho K, and Bengio Y (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In: 3rd International Conference on Learning Representations. San Diego, CA, USA: ICLR 2015.
50. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. (2017). Attention Is All You Need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY: Curran Associates Inc. P. 6000–10.
51. Kuleshov V, and Ermon S (2018). Accurate Uncertainties for Deep Learning Using Calibrated Regression. *Proceedings of the 35th International Conference on Machine Learning*. PMLR 80:2796–804.
52. Ghavamzadeh M, Mannor S, Pineau J, and Tamar A (2016). Bayesian Reinforcement Learning: A Survey. *Found Trends Mach Learn* 8(5-6): 359–483. doi:10.1561/2200000049
53. Guo C, Pleiss G, Sun Y, and Weinberger KQ (2017). On Calibration of Modern Neural Networks. In: P. Doina and T. Yee Whye. editors. Proceedings of the 34th International Conference on Machine Learning 70:1321–30. . PMLR.
54. MacKay D (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation* 4:448–72. doi:10.1162/neco.1992.4.3.448
55. Neal R (1995). Bayesian Learning for Neural Networks. Berlin, Heidelberg: Springer-Verlag.
56. Lakshminarayanan B, Prtizel A, and Blundell C (2017). *Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles*. p. 6405.
57. Kendall A, and Gal Y (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?. In: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, and S. Vishwanathan. editors. *Advances in Neural Information Processing Systems*. Curran Associates, Inc. 30.
58. Geifman Y (2019). Selective Net: A Deep Neural Network with an Integrated Reject Option. In: K. Chaudhuri and R. Salakhutdinov. editors. *Proceedings of the 36th International Conference on Machine Learning*. PMLR. p. 2151–2159. Available at: http://proceedings.mlr.press/v97/geifman19a/geifman19a.pdf.
59. Atpaino (2017). Selective Net: A Deep Neural Network with an Integrated Reject Option. Available at: https://github.com/atpaino/deep-text-corrector.
60. Lepora NF (2016). Threshold Learning for Optimal Decision Making. *Adv Neural Inf Process Syst* 29:3763–71.
61. HuggingFace (2017). Transformers. Available at: https://github.com/huggingface/transformers.
62. Brants T, and Franz A (2006). *Web 1T 5-gram Version 1*. Philadelphia: Linguistic Data Consortium. p. LDC2006T13.
63. Islam A, and Inkpen D (2009). Real-word Spelling Correction Using Google Web it 3-grams. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3, EMNLP '09. USA: Association for Computational Linguistics. p. 1241–9.
64. Chen JZ, Yu S, and Wang H (2020). Exploring Fluent Query Reformulations with Text-To-Text Transformers and Reinforcement Learning. *ArXiv, abs/2012.10033*.
65. Bocklisch T, Faulkner J, Pawlowski N, and Nichol A (2017). Rasa: Open Source Language Understanding and Dialogue Management. *CoRR, abs/1712.05181*. Available at: http://arxiv.org/abs/1712.05181.

Conflict of Interest: All authors were employed by the company The Vanguard Group.

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