frontiers | Frontiers in Applied Mathematics and Statistics

TYPE Original Research
PUBLISHED 22 September 2023
DOI 10.3389/fams.2023.1177081

Check for updates

OPEN ACCESS

EDITED BY
Quansheng Liu,
Université Bretagne Sud, France

REVIEWED BY
Rahib Abiyev,
Near East University, Cyprus
Yongjian Ouyang,
Independent Researcher, Houston, TX, United States
Xing-Ce Wang,
Beijing Normal University, China

*CORRESPONDENCE
Yinying Kong
✉ kongyy@gdufe.edu.cn

†These authors have contributed equally to this work and share second authorship

RECEIVED 01 March 2023
ACCEPTED 14 August 2023
PUBLISHED 22 September 2023

CITATION
Kong Y, Huang N, Deng H, Feng J, Liang X, Lv W and Liu J (2023) Text classification in fair competition law violations using deep learning. *Front. Appl. Math. Stat.* 9:1177081. doi: 10.3389/fams.2023.1177081

COPYRIGHT
© 2023 Kong, Huang, Deng, Feng, Liang, Lv and Liu. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Text classification in fair competition law violations using deep learning

Yinying Kong[1]*, Niangqiu Huang[1†], Haodong Deng[1†], Junwen Feng[2†], Xingyi Liang[2], Weisi Lv[2] and Jingyi Liu[2]

[1]School of Statistics and Mathematics, Guangdong University of Finance and Economics, Guangzhou, Guangdong, China, [2]School of Accounting, Guangdong University of Finance and Economics, Guangzhou, Guangdong, China

Introduction: Ensuring fair competition through manual review is a complex undertaking. This paper introduces the utilization of Long Short-Term Memory (LSTM) neural networks and TextCNN to establish a text classifier for classifying and reviewing normative documents.

Methods: The experimental dataset used consists of policy measure samples provided by the antitrust division of the Guangdong Market Supervision Administration. We conduct a comparative analysis of the performance of LSTM and TextCNN classification models.

Results: In three classification experiments conducted without an enhanced experimental dataset, the LSTM classifier achieved an accuracy of 95.74%, while the TextCNN classifier achieved an accuracy of 92.7% on the test set. Conversely, in three classification experiments utilizing an enhanced experimental dataset, the LSTM classifier demonstrated an accuracy of 96.36%, and the TextCNN classifier achieved an accuracy of 96.19% on the test set.

Discussion: The experimental results highlight the effectiveness of LSTM and TextCNN in classifying and reviewing normative documents. The superior accuracy achieved with the enhanced experimental dataset underscores the potential of these models in real-world applications, particularly in tasks involving fair competition review.

KEYWORDS

fair competition review, text classifier, LSTM, TextCNN, deep learning—artificial intelligence

1. Introduction

The application of computer technology to tackle complex challenges in human society has long been a central focus of machine learning. In pursuit of machines that can approach problem-solving in a manner closer to human thinking, the field of machine learning has witnessed the emergence of a novel research direction known as deep learning. Deep learning constructs neural networks that simulate the functioning of human brain neurons through the use of computer programs, enabling the prediction of outcomes following training with vast amounts of data. As deep learning continues to advance and computer hardware performance improves, this technology has found widespread implementation across industries such as language translation, image recognition, driverless cars, and online advertising. Deep learning holds distinctive advantages in handling unstructured data, which has led to its prominent role in addressing various natural language processing challenges.

Text recognition is widely welcomed by researchers and industry due to its wide-ranging applications. Currently, many scene text recognition methods perform well in benchmark tests. However, existing methods are unable to recognize unknown characters such as other language characters or special symbols. These characters have not appeared in the training set, so an effective method is needed to differentiate and map them to improve the level of scene text recognition. Chang et al. [1] proposed a method called "label-to-prototype learning framework," which can handle new character categories and is not affected by language. This method uses a "label-to-prototype mapping module" to generate character prototypes and uses them as classifier weights. Their approach achieved 13.98% improvement in character recognition accuracy compared with other recent methods on the CTW dataset.

Deep learning holds significant potential for widespread application in the realm of government governance. In China, the State Council has issued a series of policies aimed at dismantling local market protection, industry access barriers, and monopolies established by certain local governments through administrative means. These policies seek to establish a unified national market and deepen the reform of the economic system. While many departments have made progress in establishing and implementing fair competition review systems, there remains a need for government agencies to collect government documents from local governments' public websites and assess whether they violate the fair competition review system. Deep learning and natural language processing technologies can play a crucial role in facilitating this review process, thereby enhancing the efficiency of government governance.

Text examination inherently involves sequential models. When dealing with texts of varying lengths, it becomes necessary to map them to corresponding words or sentences that represent different responses. Standard neural networks often face challenges when addressing sequential model problems in natural language processing. This is primarily due to the requirement of sharing learned features from different parts of the text when processing natural language. Standard neural networks lack the ability to share features effectively and are thus ill-suited for examining comprehensive canonical files that possess logical and contextual linkages.

At present, the recurrent neural network (RNN) is commonly employed for Chinese text classification. RNNs possess a cyclic structure that enables the persistence of text information throughout the neural network's training, effectively capturing short-term dependencies within the text. However, RNNs encounter challenges in dealing with long-term dependencies, leading to issues such as vanishing gradients or exploding gradients. To address these problems, Sepp Hochreiter and Jurgen Schmidhuber introduced the Long Short-Term Memory (LSTM) neural network architecture. LSTM incorporates forgetting gates and update gates to regulate the flow of information within the cell state of the neural network, enabling the removal or addition of information as needed. This innovative approach successfully mitigates the vanishing gradient and exploding gradient issues inherent in RNNs [2].

While convolutional neural networks (CNNs) have traditionally been applied to image processing tasks, in recent years, researchers have started exploring their application in the field of natural language processing (NLP). Yoon Kim, among others, has made adaptations to CNN architectures by modifying the input data from images to text. This innovative approach has resulted in the development of the TextCNN model, which has demonstrated notable performance in the realm of text classification.

Government documents possess distinct characteristics that set them apart from general text. They typically exhibit lengthy content, formal writing styles, and limited emotional bias. Even manual review of these documents necessitates reviewers to possess extensive accumulated experience. Consequently, this paper utilizes unsupervised in-depth learning algorithms for classification review, while also comparing the performance of LSTM and TextCNN neural network classifiers.

2. Data processing

2.1. Data sources

The dataset of government documents used in this study is sourced from the Antitrust Office of Guangdong Province Market Supervision and Administration. The dataset comprises exclusively Chinese government documents. To facilitate the experiment, the dataset is divided into three subsets: a training set, a validation set, and a test set, distributed in proportions of 60%, 20%, and 20%, respectively.

In line with the Fair Competition Review System, manual labeling of the datasets is conducted and categorized into three distinct classes. The first category involves labeling text that pertains to market subjects but does not violate the Fair Competition Review, which is labeled as "relevant." The second category comprises labeling text that involves market subjects and violates the Fair Competition Review, and is thus classified as "suspect documents." Lastly, ordinary policy text that does not involve market players and does not violate the fair competition review is labeled as "irrelevant."

2.2. Text enhancement

Following the implementation of the Fair Competition Review system, policy measure documents that are deemed to violate the system are withdrawn from the government website. As a result, the dataset contains only 1,512 instances of the "Suspect Files" class. The remaining number of instances in the validation and test sets after dataset division is too small to achieve adequate training effectiveness. Therefore, in this experiment, the "suspect file" class is expanded using the EDA (Easy Data Augmentation) text enhancement technique.

The NLPCDA Chinese data enhancement tool, an open-source resource, is utilized to replace the text data with random entities, effectively augmenting the dataset. The change rate for each text is set to 0.3, resulting in three new augmented texts per original text. Consequently, the initial 1,512 instances of the "suspect file" class

are expanded to 4,536 instances. To validate the effectiveness of text enhancement, this experiment will conduct tests and comparisons on the dataset before and after augmentation.

3. Experiment design

For this experiment, PyTorch is selected as the framework for the neural network implementation. PyTorch is a lightweight distributed machine learning platform that offers several advantages. One notable feature is its "Dynamic Computational Graph," which enables faster model convergence [3]. The hardware utilized in the experiment consists of an Intel(R) Core(TM) i5-8300H CPU and NVIDIA GeForce GTX 1060 Max-Q GPU. The software environment is based on Python 3.6 programming language.

3.1. Natural language processing

Prior to conducting the classification experiment, the normative files in the dataset undergo preprocessing. The first step involves importing the THUCNews deactivated Thesaurus from the Tsinghua NLP group for data cleaning purposes. Subsequently, the Jieba word-breaking tool is utilized for word segmentation. The labels " irrelevant," " relevant," and " suspect documents" are encoded as 0, 1, and 2, respectively. The preprocessed data is then saved in the CSV file format, as shown in Table 1.

After preprocessing, the following steps are performed on each text in the dataset:

1. Removal of stop words: All stop words, punctuation marks, and numeric characters are removed from the text.
2. Word tokenization: The text is divided into individual words, with each word separated by spaces.

By removing stop words, punctuation, and numeric characters, the focus is placed on the essential content of the text, facilitating further analysis and classification.

As shown in Table 2, after word segmentation, the data in the CSV file is partitioned into three subsets: the training set, validation set, and test set, with proportions of 60%, 20%, and 20% respectively.

Next, a vocabulary is constructed based on the highest frequency of the first 5,000 words present in the dataset. This vocabulary is created using the torchtext tool and serves as the reference for word indexing and representation.

The constructed vocabulary is saved as the parent vocabulary for subsequent text processing and model training.

TABLE 1 Unprocessed dataset styles.

Type	Content (Chinese)
Irrelevant	10 月 29 日下午,省民族宗教委召开党组扩大会议,专题传达学习习近平总书记视察广东重要讲话精神和李希书记在全省干部大会上的讲话精神 … …
Relevant	广东省人民政府关于培育发展战略性支柱产业集群和战略性新兴产业集群的意见粤府函 … …
Suspect documents	各地级上述市财政局(委)，省属有关国有企业：现将《总后勤部有关国有资本加大对公益性行当落入的指导眼光》 … …

3.2. LSTM classifier setup

Long Short-Term Memory (LSTM) is a widely used type of recurrent neural network known for its effectiveness in handling sequential data. Over the years, several extensions and variations of the LSTM architecture have been proposed. While the pioneering work on LSTM was introduced by Sepp Hochreiter and Jurgen Schmidhuber, the current mainstream LSTM architecture, as popularized by Graves A and Schmidhuber J, forms the basis of this experiment.

In line with this, a classifier based on the LSTM neural network design proposed by Graves A and others is developed for this experiment.

3.2.1. LSTM cell structure

A single LSTM neuron comprises four essential components: an input gate, a forget gate, a cell state, and an output gate. These components work together to enable the LSTM cell to capture and retain relevant information over long sequences.

The LSTM cell structure can be visualized as shown in Figures 1, 2. The input gate regulates the flow of information into the cell, determining which input values are significant for updating the cell state. The forget gate controls the extent to which the previous cell state is retained or discarded. The cell state serves as the memory of the LSTM cell, storing relevant information throughout the sequence. Finally, the output gate controls the flow of information from the cell, allowing selective output based on the current cell state and input.

(1) The forget gate helps the network decide what information to retain and what to forget. Its input is the output from the previous timestep and the current input x_t, and it outputs a vector that controls what information should be retained in the previous timestep's cell state h_{t-1} .

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

(2) Then there is the input gate, which determines what type of new signal the cell state stores. It consists of two segments. The first

TABLE 2 Processed dataset styles.

Label	Content (Chinese)
0	下午 民族宗教 召开 党组 扩大 会议 专题 传达 学习 习 近平 总书记 视察 广东 重要讲话 … …
1	广东省 人民政府 关于 培育 发展 战略性 支柱产业 集群 战略性 新兴产业 集群 意见 粤府 … …
2	各地 上述 财政局 省属 有关 国有企业 现将 总后勤部 有 关 国有资本 加大 公益性 行当 落入 指导 眼光 … …

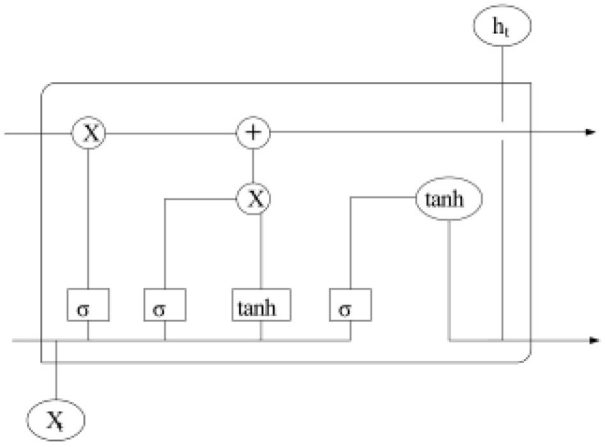


FIGURE 1
Structure diagram of LSTM unit.

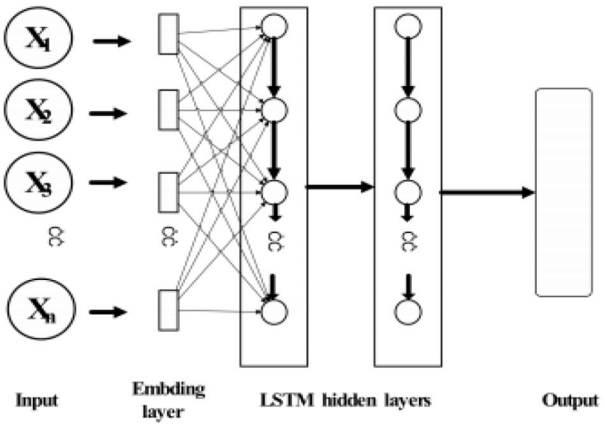


FIGURE 2
Schematic diagram of LSTM neural network structure

segment controls which input information at the current moment can be stored in the current memory unit; The second part is to create a new candidate value vector $\tilde{C}_t$ to add to the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\widetilde{C}_t \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

(3) Update the cell state. The future behavior depends on the previous steps and how our present goal is achieved.

We use the output of the oblivion gate to determine how much information can be retained from the previous cell state and how much of the current input can be stored in the new cell state. The updated formula is:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

(4) Finally, the output gate, whose output depends on the current state of the cell, consists of two segments. The first is the amount of information that h_{t-1} and x_t can be passed to the next moment by calculating the sum. The second fragment is the cell state processed by the tanh activation function and multiplied by the output of the first fragment, resulting in the information that needs to be passed on to the next moment.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

3.2.2. Network architecture

The LSTM neural network classifier is constructed with the following layers:

Input layer: The input layer loads the preprocessed datasets. It processes 64 samples at a time.

Embedding layer: The input samples are embedded into a dense vector representation using an embedding layer with a dimension of 100. This layer helps capture semantic relationships between words.

LSTM hidden layers: The network includes two LSTM hidden layers, each consisting of 128 LSTM neurons. These layers enable the network to capture long-term dependencies and temporal patterns in the input sequences.

Fully connected layer: The output from the LSTM layers is fed into a fully connected layer, which maps the LSTM hidden states to the desired output dimension. In this case, the output layer consists of 4 neurons, corresponding to the number of classes.

The loss function used in this experiment is the multiclass cross-entropy loss function, which is suitable for classification tasks involving multiple classes.

Additionally, an early stopping mechanism is implemented during the training phase of the LSTM. This mechanism automatically stops the training process when the loss function and accuracy on the validation set reach a stable state.

The network structure, as described, involves the forward and backward propagation of each LSTM neuron in the hidden layers, capturing both the temporal dependencies and their reverse order.

3.3. Set up TextCNN classifier

TextCNN (Convolutional Neural Network for Text Classification) is a popular deep learning algorithm for natural language processing (NLP) tasks such as text classification and sentiment analysis. TextCNN is designed to capture local features from text data by applying convolutional filters to the input sentences. The convolutional filters act as feature detectors, scanning the input sentences to identify key patterns and features. The output of the filters is then fed into a max-pooling layer, which extracts and retains the most important features. The resulting feature vector is then passed through a fully connected layer and eventually used to classify the input text into relevant categories.

3.3.1. TextCNN frame

As shown in Figure 3, TextCNN contains an embedding layer, a convolution layer, a pooling layer and an output layer.

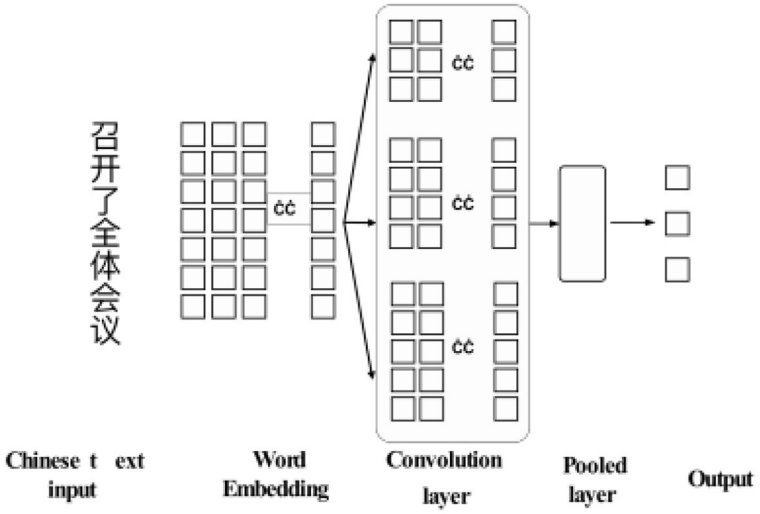


FIGURE 3
Schematic diagram of TextCNN neural network structure.

TABLE 3 Performance of LSTM and TextCNN before text enhancement.

	LSTM	TextCNN
Accuracy of test set	93.71%	92.84%
Accuracy of training set	99.82%	100%
Training set loss function value	0.002	0.0034
Accuracy of verify set	92.7%	92.4%
Validation set loss function value	0.19	0.3
Recall	88.5%	86.2%
F1 score	91.0%	89.4%

TABLE 4 Performance of LSTM and TextCNN after text enhancement.

	LSTM	TextCNN
Accuracy of test set	96.36%	96.19%
Accuracy of training set	99.84%	100%
Training set loss function value	0.0015	0.0012
Accuracy of Verify Set	95.96%	96.01%
Validation set loss function value	0.25	0.13
Recall	92.45%	93.71%
F1 score	94.36%	94.93%

3.3.1.1. Embedding layer

This layer converts the text data vector to the specified size, and each row in the matrix represents a word. If sentence X has n words, then the sentence can be expressed in matrix form:

$$X = X_1 \oplus X_2 \oplus .....X_n$$

Where, X_i represents the ith word and $\oplus$ is the concatenation symbol of vectors.

3.3.1.2. Convolution layer

The TextCNN model uses a convolution kernel filter with an input layer sentence matrix to calculate the unit node matrix. The matrix can extract different hierarchical features and then obtain the feature mapping matrix by nonlinear activation function $C = [c_1, c_2, , c_3..., c_n]$

$$C_i = f(W \cdot X_{i:i+h-1} + b)$$

Where W is the weight matrix of the output convolution kernel, b is the bias parameter, and f is the activation function.

3.3.1.3. Pooling layer

Through the maximum pooling of feature vectors, a one-dimensional vector is formed to represent the aggregation of important features, so as to reduce the number of neural network parameters and feature vectors, achieve dimensionality reduction, and prevent overfitting phenomenon to a certain extent.

$$\bar{c} = \max(c)$$

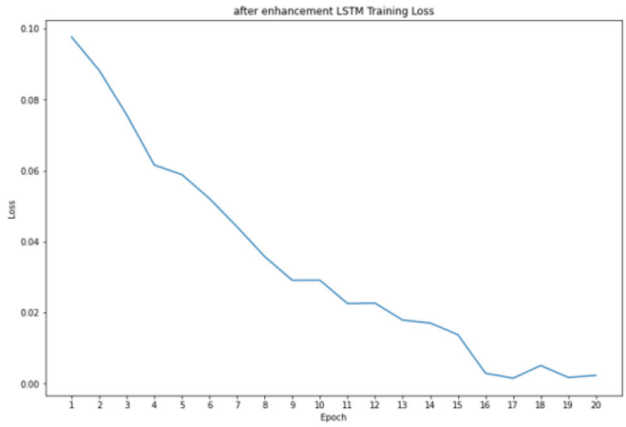
3.3.1.4. Output layer

The convolution operation using different convolution kernels can extract different features and then combine these features into a one-dimensional vector by pooling operation. Finally, the vector is entered into a fully connected softmax layer to get a probability distribution that the sample belongs to different categories.

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{i=1}^{n} \exp(x_i)}$$

3.3.2. Building network

The dimension setting of word vector and the size of batch_size of the model in this section are the same as those in 3.2.2 of this paper. The height of convolution kernel is Kim [4], Zhang [5], and Shu et al. [6], the number of convolution kernel is 100,



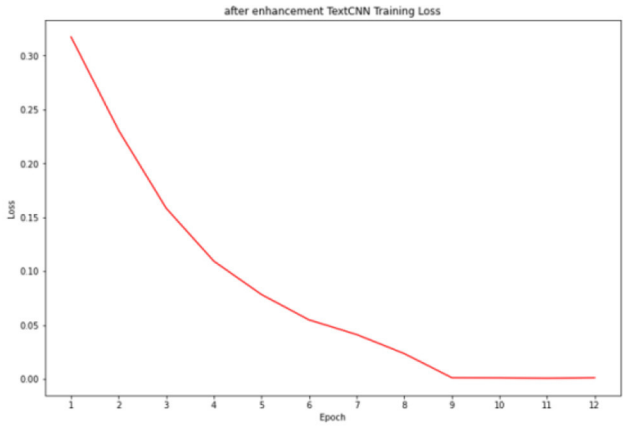
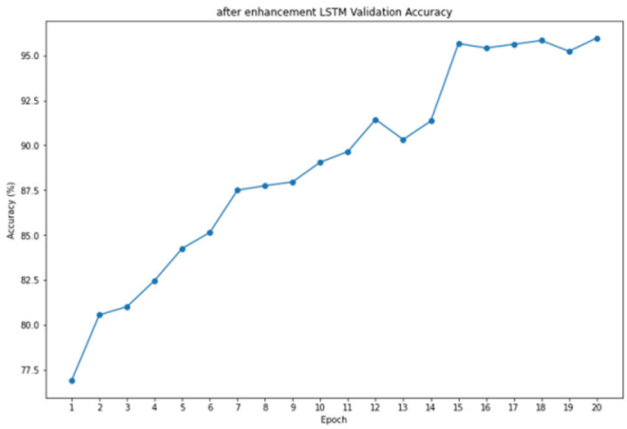


FIGURE 4
Loss function of LSTM and TextCNN after text enhancement.



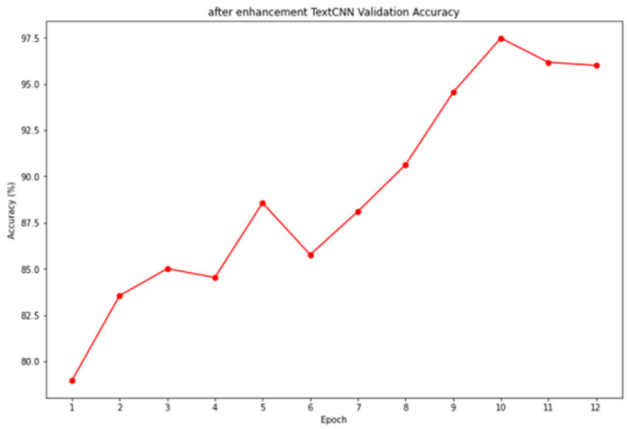


FIGURE 5
Validation accuracy of LSTM and TextCNN after text enhancement.

ReLU activation function is used, and multi-class cross entropy loss function is used for loss function.

4. Experiment analysis

4.1.Content analysis

This experiment focuses on analyzing and comparing the performance of the LSTM classifier and TextCNN classifier using the same normalized file dataset. The aim is to assess their effectiveness in text classification. Furthermore, the experiment investigates the robustness [7, 8] of the classifiers by comparing their performance on two datasets: one before text enhancement and the other after text enhancement. This analysis aims to evaluate how well the classifiers can handle enhanced data.

Additionally, the experiment examines and compares the training speed and the number of training iterations required for each classifier. This analysis provides insights into the practicality and efficiency [9] of the two approaches.

4.2. Data comparison

In this experiment, several evaluation criteria are used to assess the effectiveness of the classification review: accuracy, recall rate, and multi-class cross-entropy loss function. The formulas for these metrics are as follows:

Accuracy: The percentage of correctly classified instances in the test set.

LSTM accuracy on the test set for the canonical file datasets without text enhancement in the three-class experiments was 93.71%.

TextCNN accuracy on the test set was 92.84%.

Recall rate: The proportion of true positive instances identified correctly out of all relevant instances.

TABLE 5 Training cost of LSTM and TextCNN.

	LSTM	TextCNN
Epoch	15–20	10–12
Training time	30 min	10 min
Correct number of unrelated classifications	1,963	1,990
Correct number of related classifications	340	310
Correct number of suspect files classifications	1,179	1,182

The recall rates for LSTM and TextCNN classifiers were 88.5% and 86.2% respectively.

Multi-class cross-entropy loss function: A measure of the dissimilarity between the predicted and actual class probabilities.

The loss function values for LSTM were stable at 0.002 for the training set and 0.19 for the validation set.

The loss function values for TextCNN eventually stabilized at 0.0034 for the training set and 0.3 for the validation set.

Additionally, the training set accuracy for LSTM stabilized at 99.82%, while the training set accuracy for TextCNN reached 100%. These results indicate that both classifiers achieved high accuracy on the training set.

After text enhancement, the LSTM classifier achieved an accuracy of 96.36% on the test set. The training set accuracy and loss function values were 99.84% and 0.0015 respectively, while the validation set accuracy and loss function values were 95.96% and 0.25.

Similarly, the TextCNN classifier achieved an accuracy of 96.19% on the test set. The training set accuracy and loss function values were 100% and 0.001, and the validation set accuracy and loss function values were 96.01% and 0.13.

Furthermore, the recall rates for the two classifiers on the test set were 92.45% for LSTM and 93.71% for TextCNN.

These results indicate that both LSTM and TextCNN classifiers showed improved performance after text enhancement, with higher accuracy and relatively lower loss values. The recall rates also suggest the ability of the classifiers to identify relevant instances correctly.

Tables 3, 4 show that both the LSTM and TextCNN classifiers achieved high accuracy on the validation and test sets, surpassing 90% even before text enhancement. After text enhancement, the accuracy further increased to over 95%. This indicates that the robustness of the classification model was maintained, as the synonym conversion and random word replacement operations did not negatively impact accuracy.

Figures 4, 5 illustrate the changes in the loss function of the LSTM and TextCNN models on the training set, as well as the accuracy changes on the validation set, following text enhancement. The results show that the LSTM model reaches convergence at epoch 15, whereas the TextCNN model achieves convergence at epoch 8. Furthermore, the accuracy of the validation set shows a clear upward trend during the training process, indicating a positive impact of model optimization.

However, there are differences between the LSTM and TextCNN classifiers. As shown in Table 5, the LSTM network required more training epochs and had a longer training time compared to TextCNN. Additionally, the accuracy of each classification varied between the two models. Both classifiers performed well, but the LSTM classifier had a longer training time and required more epochs.

In summary, both the LSTM classifier and TextCNN classifier achieve high levels of accuracy and recall. The difference between them is minimal. The standardized file dataset used in this study presents more challenges compared to general short text classification due to its larger size, complex structure, and reduced emotional expression. However, both classifiers outperform manual review methods in terms of efficiency. Notably, TextCNN demonstrates better speed due to its ability to process text in parallel, whereas LSTM processes text linearly.

5. Summary

In conclusion, the application of deep learning technology to the fair competition review system is an interdisciplinary endeavor that encompasses computer technology, government governance, and legal treatment. The field of deep learning is rapidly advancing, offering endless possibilities for neural network configurations. Incorporating advanced technologies into government decision-making processes is crucial for fostering top-level design and planning, ultimately contributing to the development of a "smart government." Chinese text classification based on deep learning has also made significant progress, and its integration with the fair competition system is a natural progression [10, 11].

However, it is important to acknowledge the limitations of this study. The paper focuses on commonly used technologies for combining fair competition review and deep learning, but more advanced techniques and practical implementations still require further development. Both classification models have room for improvement, and future research should aim to address these shortcomings.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

YK contributed to the conception and design of the study. JF wrote the initial draft of the manuscript. NH and HD significantly contributed to the major revision of the manuscript. All authors participated in the revision process, read, and approved the final submitted version.

Funding

The authors express gratitude for the support provided by the Guangdong Basic and Applied Basic Research Foundation (2022A1515012429) and the Guangdong Provincial Department of Education Innovation Team Project (2022WCXTD009). The support from these organizations has

been instrumental in conducting this study and has contributed to its success.

Conflict of interest

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

Publisher's note

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

References

1. Chang, L., Chun, Y., Hai-bo, Q., Xiaobin, Z., and Xu-Cheng, Y. (2021). Towards open-set text recognition via label-to-prototype learning. *Pattern Recog.* 134, 109109. doi: 10.1016/j.patcog.2022.109109

2. Zhang S. Economic law analysis of fair competition examination system. *Polit Sci Law*. (2017) 11:2–10. doi: 10.15984/j.cnki.1005-9512.2017.11.001

3. Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM networks proceedings. *2005 IEEE International Joint Conference on Neural Networks. IEEE*. (2005) 4:2047–52. doi: 10.1109/IJCNN.2005.1556215

4. Kim Y. Convolutional neural networks for sentence classification. *Eprint Arxiv. arXiv preprint arXiv*. (2014) 1408.5882. doi: 10.48550/arXiv.1408.5882

5. Zhou FY, Jin LP, Dong J. Review of convolutional neural network. *Chin J Comp*. (2017) 40:1229–51.

6. Shu N, Liu B, Lin W, Li PF. Survey of distributed machine learning platforms and algorithms. *Comp Sci*. (2019) 46:9–18. doi: 10.11896/j.issn.1002-137X.2019.03.002

7. Zhu J. An empirical study on the implementation of the fair competition review system-sample of 59 cases published by the national development and reform commission. *Compet Policy Res.* (2018) 04:121–35.

8. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput.* (1997) 9:1735–80. doi: 10.1162/neco.1997.9.8.1735

9. Yang L, Wu Y, Wang J, Liu Y. Research on recurrent neural network. *J Comp App*. (2018) 38:1–26.

10. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceed of the IEEE.* (1998) 86:2278–324. doi: 10.1109/5.726791

11. Wei J, Zou K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv*. (2019) 1901:11196. doi: 10.18653/v1/D19-1670