Sensors and Materials, Vol. 32, No. 8 (2020) 2659–2672
MYU Tokyo

S & M 2291

Convolutional-neural-network-based Multilabel Text Classification for Automatic Discrimination of Legal Documents

Ming Qiu,[1] Yiru Zhang,[1] Tianqi Ma,[1] Qingfeng Wu,[1*] and Fanzhu Jin[2**]

[1]School of Informatics, Xiamen University, Fujian 361005, China
[2]College of Mathematics and Information Engineering, Longyan University, Fujian 364012, China

(Received January 6, 2020; accepted May 25, 2020)

Keywords: multilabel learning, text classification, word embedding

Law courts spend too much time reading documents and judging the type of legal cases. This problem becomes more serious as a crime can be classified into several categories at the same time. Thus, legal documents need a multilabel classification. We propose a multilabel text classification model based on multilabel text convolutional neural network (MLTCNN). We scan legal documents and convert them to text data using optical character recognition (OCR) with a charge-coupled device (CCD) sensor. Then, we use Jieba, a word segmentation tool of Chinese letters, and TensorFlow VocabularyProcessor to generate vocabularies. Then, the case description after segmenting each word is mapped into a word index in the vocabularies. We use a word index vector as an input to the MLTCNN. Lastly, we adopt multiple sigmoid functions for multiple binary classifications. The result shows our method to be efficient in finding errors and deviations for similar cases among district courts. This study provides a new method to improve the legal service and to enable fairer law enforcement.

1. Introduction

Data has become a strategic resource and economic asset. There is now great demand for analyzing big data using various computing methods and solutions. A big data analysis improves management of governments and enterprises, and upgrades industrial capabilities through data mining and machine learning. However, these new technologies are rarely applied to the legal industry. Courts and legal professionals deal with many cases involving several crimes committed simultaneously, generating a large number of documents, often leading to mistakes in judicial decisions. The most common mistake is different verdicts for very similar cases. Thus, big data analysis can help the process of judicial decisions by more efficiently removing such mistakes for better enforcement of laws.

Big data analysis needs to digitize printed documents to convert them to a database. This requires the development of the sensor technologies of a charge-coupled device (CCD) and optical character recognition (OCR) conversion, along with artificial intelligence (AI). Scanned

*Corresponding author: e-mail: qfwu@xmu.edu.cn
**Corresponding author: e-mail: dennisbskim@hotmail.com
https://doi.org/10.18494/SAM.2020.2794

images of better quality from CCDs give clearer OCR-converted characters to improve the text classification. Thus, sensor technology is essential in this process.

In this study, we used an existing natural language process technology based on a deep learning method to process legal documents with classified texts. The technology attaches multiple labels to legal documents automatically. Then, it defines crime types and predicts legal provisions. We compared the results with the previous verdicts. At the same time, we tried to find any tendency or habitual deviations in verdicts for the same cases. We expect the results to improve the quality of trials and promote fairer law enforcement.

In this paper, Sect. 2 discusses relevant studies and Sect. 3 discusses the architecture and other details of our model multilabel text convolutional neural network (MLTCNN). Section 4 shows experimental results and a comparative analysis. Then, we compare this study's result with other models to prove the model's superiority in terms of performance in Sects. 5 and 6.

2. Text Classification

Text classification is a classical topic in the field of natural language processing. It categorizes, structures, and organizes texts to extract and query key words representing whole texts. Research on text classification dates back to the 1950s. Then, texts were classified by rule-based systems by knowledge engineering in the early 1980s. However, rule-based systems required a lot of time and had limited coverage with low accuracy. The methods enable large-scale text classification in a short period of time. The main technology used in the methods is feature engineering, that is, applied machine learning.

Feature engineering is often time- and energy-consuming, although extremely important. Feature engineering converts data to information, which is then converted to knowledge by classification. It does not have a strong generality and often needs to be combined with feature tasks. Text classification with a natural language process traditionally has its own feature processing logic. Feature engineering for texts is divided into three parts: text preprocessing, feature extraction, and text representation. The ultimate goal is to transform texts into numbers in the form of vectors that can be recognized by a computer. Then, the vectors can be processed by a machine learning algorithm and a classifier model.

2.1 Text preprocessing

As the legal documents in this study are written in Chinese, preprocessing is needed for the texts in the documents. The text processing mainly comprises two stages: text segmentation and deactivation. Text segmentation is important as Chinese characters have complex structures and a strict order when writing them (stroke order). As most text classification algorithms do not consider stroke order, much n-gram information is lost in Chinese text processing. As Chinese words in a sentence do not have spaces between them, they need a complex word segmentation algorithm. Traditional word segmentation algorithms have forward, reverse, or bidirectional maximum matchings based on string matching and mutual information(1) or a conditional random field (CRF) model.(2) Recent word-embedding methods with a deep learning algorithm recognize stop words (pronouns, conjunctions, prepositions) for text classification.

2.2 Feature extraction

Feature extraction in a vector space model consists of two parts: feature selection and feature weight calculation. The basic idea of feature selection is to rank original feature items according to an evaluation index, select some with high scores, and filter out the rest. Feature selection deletes words in the stop word list. Common evaluation indices include the frequency in texts, mutual information, information gain, and χ^2 statistics. Feature extraction assumes that important words appear more frequently in texts. Then, it forms a list of these words and transforms it to a feature set that can be used by a classifier.

2.3 Text representation

Text representation transforms preprocessed texts into information to be recognized by a computer. This is the most important part of determining the quality of text classification. A bag of words model(3) and a vector space model(4) are commonly used for text classification. However, these models do not tend to give similarity values in long texts and only provide semantic information. In addition, they require key words to be matched precisely in the texts and ignore the order of the words. The models have two major problems: high latitude and high sparsity. The bag of words model is the basis of the vector space model. The vector space model reduces the number of dimensions through feature selection, but increases the density through feature weight calculations.

In addition to these models, semantic-based text representation methods are also used, such as the latent Dirichlet allocation (LDA) topic model,(5) latent semantic indexing (LSI),(6) and probabilistic latent semantic analysis (PLSA).(7) While the semantic-based text representation method can give in-depth representations of documents, a word embedding distributed representation provides a more important foundation of deep learning methods.

Hinton(8) proposed the word embedding distributed representation in 1986. Later, after Yoshua *et al.*(9) proposed the neural probabilistic language model (NNLM) in 2003, the word vector became popular owing to research by Mikolov and coworkers in 2013.(10,11) Zhang and Zhou presented a Word2vec toolkit that is easier to use.(12) Through the expression of the word vector, the text data is transformed from a high-latitude and highly sparse neural network to continuous dense data similar to an image and a voice. Thus, a deep learning algorithm has strong ability of data migration. Many deep learning algorithms, such as the convolutional neural network (CNN), are highly applicable to images and migrate them to the text field easily.

2.4 Multilabel learning algorithm

Multilabel learning stems from text classification. In multilabel learning, each sample in a training set has multiple tags. There are many multilabel learning algorithms. In problem solving, these algorithms can be divided into two groups: one based on problem transformation and the other based on algorithm applicability.(13) The problem-transformation-based algorithm transforms problem data for an existing algorithm, while the algorithm-applicability-based one needs a specific algorithm to process multilabel data. The latter improves the algorithm applied to the data.

Some methods based on problem transformation consider the relevance of tags. The simplest algorithm, which does not consider relevance, treats each label as a single label and implements a common classification algorithm for each label. A traditional machine learning model adopts support vector machine (SVM),(14) decision tree (DT),(15) naive Bayes (NB),(16) or XGBoost.(17) The deep learning model trains text classification models such as convolutional neural networks for sentence classification (TextCNN)(18) or recurrent neural network for text classification (TextRNN).(19) Considering the relevance of multiple labels, the last output label can be regarded as the input of the next label classifier. In the traditional machine learning model, we can use the classifier chain. In this case, the first classifier is only trained on the input data, and then each classifier is trained on the input space and all previous classifiers on the chain.

In the deep learning model, a time series model is added to the output layer. It can add the output value of the previous time to the input data series at each time. In Ref. 20, after obtaining the overall semantics of the article (text feature vector), the text feature vector is input into an RNN sequence as the initial value, and the input at each time is the output of the previous time.

The label powerset has a uniform way to view labels.(21) It transforms a single problem into a multiclass problem in which a multiclass classifier is trained on all unique tag combinations in the training data. Therefore, the label powerset provides a unique class for every possible label combination in the training set. SVM, TextCNN, TextRNN, and other classification algorithms then train the model with the label combination. However, the label powerset can provide a small number of labels. When there are many labels (n labels), the possible datasets are distributed in a space with between 0 and $2n - 1$ dimensions. Thus, the data becomes sparse.

In a traditional machine learning model, multilabel classification models of wear parts are a multilabel version of KNN (multilabel *k*-nearest neighbor, MLKNN,) or a multilabel version of SVM (rank-SVM).(22) In a deep learning model, the output layer of the multiple-classification model is often modified to adapt it for the classification of multiple labels. Berger used a sigmoid function to classify the output value of each label in the output layer, and there was no associated information between labels.(23) Mullenbach *et al.* used a general CNN to extract the semantic information of sentences by considering that different parts of a sentence representation were also classified in sentences.(24) These processes with different attention mechanisms make representations of different parts of sentences and play different roles in the classification.

3. Materials and Methods

3.1 Architecture

We designed a multilabel text classification model based on TextCNN. Figure 1 shows our model framework. The model is divided into two parts as follows.

(1) Text-based data preprocessing: we used Jieba for Chinese word segmentation and TensorFlow VocabularyProcessor to extract vocabulary, as Jieba is known to be the best

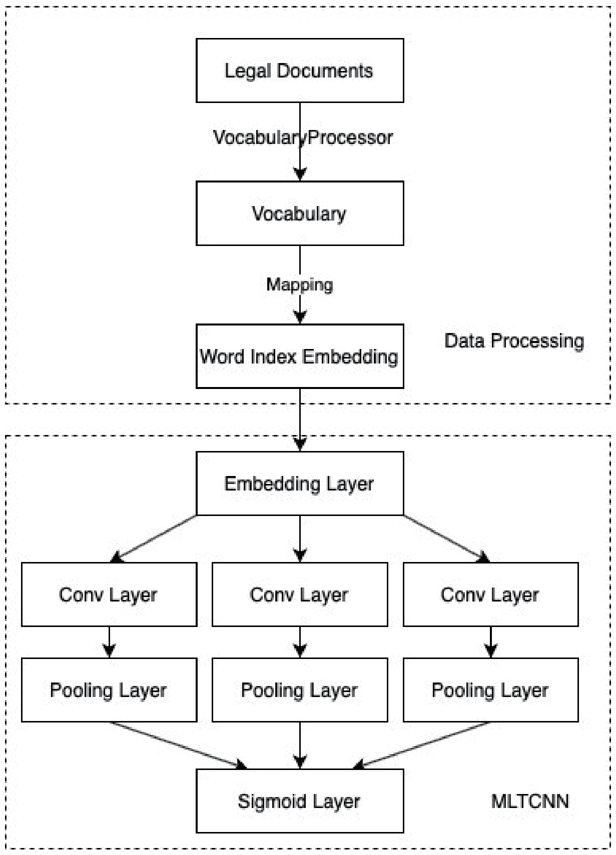


Fig. 1. Architecture of the proposed model.

Python Chinese word segmentation mode(25) and TensorFlow VocabularyProcessor is an open-source machine learning platform in Python.

(2) Multilabel text classification model based on TextCNN: The word vector is initialized and trained in the embedding layer of the model. Then, a final normal-type classification structure is output through the sigmoid layer.

3.2 Data preprocessing

3.2.1 Word segmentation

Word segmentation is a process of recombining consecutive word sequences according to a certain norm. Chinese words are segmented to Chinese character sequences. We used Jieba for Chinese word segmentation. There are three Chinese word segmentation patterns supported by Jieba. The standard patterns of Jieba enabled us to obtain words, which are then filtered by stop words.

3.2.2 Stop word filtering

Stop words can reduce noise in text to give more accurate text classification. Modal particles, adverbs, prepositions, conjunctions, and punctuations were not considered as stop words.

3.2.3 Vocabulary generation

Text processing establishes a vocabulary and a mapping relationship between a word and an index. When establishing the mapping relationship between documents and an index, we used TensorFlow VocabularyProcessor to establish the mapping relationship between a word and an index. Thus, VocabularyProcessor maps documents to word-index sequences. Through this function, we can obtain a categorical vocabulary object, save the object as a file, and then obtain the glossary file. Then, we generate the corresponding word index value by a segmented text data method. We embed these index values in a neural network.

3.3 MLTCNN

Figure 2 shows our model based on TextCNN. The structure of TextCNN is more complex than that of a fastText model. Kim proposed a convolution and maximum pooling, which are successful combinations in the image field.(18) As in Fig. 2, the first layer of the schematic diagram inputs a word vector matrix of 7 × 5, in which the word vector dimension is 5 and the sentence length is 7. The second layer uses three groups of convolution kernels of widths 2, 3, and 4.

Each convolution kernel slides along a whole sentence to obtain an activation value. Padding is not used in the convolution kernel sliding, so the convolution kernel with a width of four slides along the sentence with a length of seven to obtain four eigenvalues. Then, global pooling is performed. A feature map composed of six feature values is obtained from the maximum value of each column vector. The vector has a feature value output of a convolution kernel in the whole sentence length.

The MLTCNN is divided into four parts: an embedding layer, a convolution layer, a pooling layer, and a sigmoid layer. A parameter vector is initialized in the embedding layer. The dimension is the size of the vocabulary multiplied by the embedding size. The corresponding word vector is selected according to the input word index vector to generate the batch size, which is the number of words multiplied by the word vector. The word vector is learned in the embedding layer. The convolution layer calculates the similarity with key words, and then uses the maximum pooling layer to ensure that the model focuses on the similarity of the key words and a convolution kernel in the whole input text. Finally, the sigmoid layer is applied to each dimension vector. A threshold selection is used to judge the prediction result.

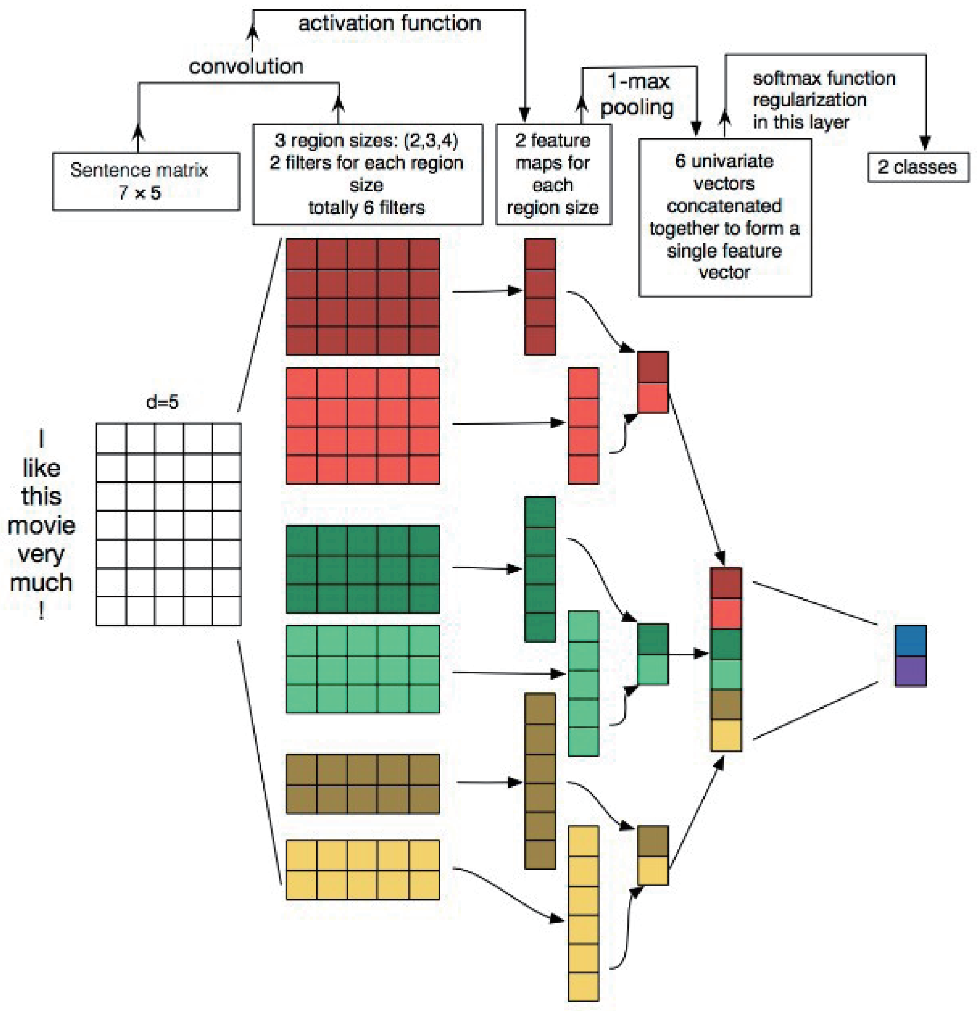

activation function
convolution
Sentence matrix
7 × 5
3 region sizes: (2,3,4)
2 filters for each region size
totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

Fig. 2. (Color online) Framework of the TextCNN model.

4. Experiments

4.1 Dataset description

To evaluate the performance of our model, we used legal documents from regional courts, which included 600000 words in 143 articles. We adopted an official evaluation metric based on F1 scores. The F1 score in our model was 98%.

4.2 Baselines

We used a problem transformation method in this study. Problem transformation methods include binary association, the classifier chain, and the label powerset. Common classifiers include the logistic regression (LR), SVM, KNN, and NB.

4.2.1 Problem transformation

1) Binary relevance

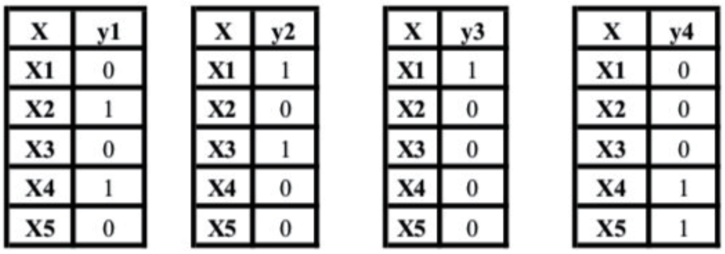
In this method, we transformed a multilabel problem into a single-label problem in three different ways. The method classifies each tag as a separate class. In a binary association, this problem is decomposed into four different class classification problems, as shown in Fig. 3. Here, *X*s are independent variables and *y*s are target variables.

2) Classifier chain

The first classifier is only trained on the input data, while the other classifiers are trained on all previous classifiers in the input space and the chain. In the dataset given in Fig. 4, we use *X*s as the input space and *y*s as the label.

In the classifier chain, this problem is converted into four different label problems, as shown in Fig. 5. This is very similar to binary association. The only difference is that the classifier chain is formed to maintain label relevance.

In the dataset of Fig. 5, *X*s are input spaces and *y*s are the labels. In the classifier chain, this problem is converted into four different label problems. The last column is the target space and the other columns are the input space.



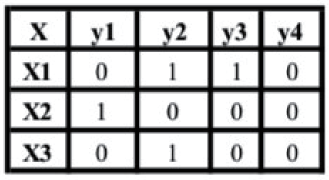
X	y1
X1	0
X2	1
X3	0
X4	1
X5	0

X	y2
X1	1
X2	0
X3	1
X4	0
X5	0

X	y3
X1	1
X2	0
X3	0
X4	0
X5	0

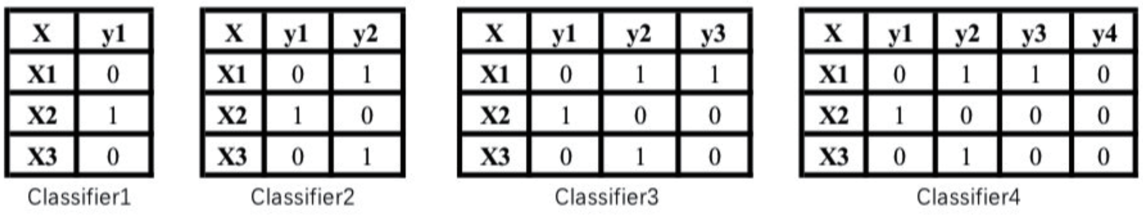
X	y4
X1	0
X2	0
X3	0
X4	1
X5	1

Fig. 3. Binary relevance.



X	y1	y2	y3	y4
X1	0	1	1	0
X2	1	0	0	0
X3	0	1	0	0

Fig. 4. Dataset of the classifier chain.



X	y1
X1	0
X2	1
X3	0

Classifier1

X	y1	y2
X1	0	1
X2	1	0
X3	0	1

Classifier2

X	y1	y2	y3
X1	0	1	1
X2	1	0	0
X3	0	1	0

Classifier3

X	y1	y2	y3	y4
X1	0	1	1	0
X2	1	0	0	0
X3	0	1	0	0

Classifier4

Fig. 5. Example of classifier chain method.

3) Label powerset

We transformed the problem into a multiclass problem, where a multiclass classifier was trained on all unique tag combinations found in the training data. Data characteristics and data labels are shown in Fig. 6. We found that *x*1, *x*4, and *x*3, *x*6 both had the same label. Thus, the label powerset turned this problem into a single multiclass problem. The only disadvantage is that the number of classes increases with increasing training data. There are 143 categories in the dataset.

4.2.2 Used algorithm

We used an existing algorithm to perform multilabel classification directly, rather than transform the problem to different problem subsets. For example, the multilabel version of the KNN is represented by the MLKNN. For each test sample, its *k*-nearest neighbor was found in the training set. Then, on the basis of the statistical information of neighbor samples, such as the number of neighbors belonging to the same category, the maximum a posteriori probability principle (MAP) was used to determine the label set of test samples.

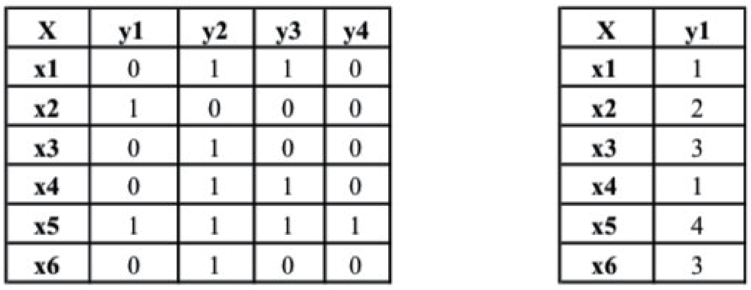
The algorithm has four steps:

1. Find the nearest *k* samples using the KNN algorithm.
2. Count the number of each category in the *k* samples.
3. On the basis of the statistics in the second step, use the NB algorithm to calculate the probability of each tag.
4. Output the category probability.

4.2.3 Base binary classifiers

1) LR

LR is a common supervised learning algorithm, which is essential to solve the classification problem.[26] Because of its advantages such as easy implementation, good explanation, and easy expansion, it is widely used in click rate prediction, advertising calculations, and recommending systems. A logistic function is selected as the activation function in LR, which is an important representative of the sigmoid function.



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3

Fig. 6. Example of label powerset.

2) SVM

The SVM was proposed in 1964 and developed rapidly after the 1990s. It has been applied to human image recognition and text classification. The decision boundary of the generalized linear classifier is the maximum-margin hyperplane. The SVM uses the hinge loss function to calculate the empirical risk and adds a regularization term to the solution system to optimize the structural risk. It is a classifier with sparsity and robustness. The SVM can be classified as nonlinear by the kernel method, which is one of the common kernel learning methods.

3) KNN

The KNN classification algorithm is one of the simplest methods in data mining. Each sample is represented by its *k* nearest neighbors.(27) The core idea of the KNN algorithm is that if most of the *k* nearest samples in the feature space belong to a certain category, then the sample also belongs to this category and has the characteristics of the samples in the category. In deciding the classification, this method only determines the category of the samples to be classified according to the category of the nearest one. When deciding the class, the KNN method is applied to a small number of adjacent samples. Because the KNN method mainly depends on a limited number of adjacent samples, it is more suitable than the other methods for dividing a sample set with overlapping class domains.

4) Gaussian NB

There are three kinds of NB models: the Gaussian model, polynomial model, and Bernoulli model. The Gaussian model is suitable for features with continuous variables. Assuming that the features follow a Gaussian distribution, it can be used for the classification of continuous variables. Gaussian NB is a classification method based on Bayes' theorem. This is a generative model. It first learns the joint probability distribution $P(X, Y)$, and then calculates the posterior probability distribution $P(Y|X)$.

4.2.4 fastText

fastText is an open-source word vector and a text classification tool of Facebook Research launched in 2016.(28) It is used for rapid text representation and text classification tasks. fastText provides a simple and efficient method for text categorization and representational learning. Its performance is comparable to that of a deep learning algorithm. However, fastText is faster. It combines the most successful ideas in natural language processing and machine learning. The softmax hierarchy (taking advantage of the unbalanced distribution of categories) is used to speed up the process. fastText also takes advantage of the fact that categories are unbalanced (some occur more than others) by using Huffman's algorithm to build tree structures that represent categories. Therefore, the depth of a tree structure with frequent categories is smaller than that with infrequent categories, which makes further calculation more efficient.

4.3 Parameter setting

The whole TextCNN architecture described in Sect. 3.3 and shown in Fig. 2 was implemented using TensorFlow, a Python deep learning framework. We set the input document word number to 100, which was the average number of words per document. Thus, we obtained a matrix of 100 as the embedding dimension size for the representation of the texts.

The training hyperparameters were a learning rate of 0.001, an embedding size of 256, a dropout rate of 0.5, and a batch size and epoch size of 30 with adaptive moment estimation. The system used for the training was a dual Intel i7-7820X CPU, clocked at 3.6 GHz, with 256 GB of RAM, equipped with an Nvidia Titan Xp GPU with 11 GB of dedicated VRAM. The training phase required about 2 h for each model.

4.4 Evaluation metrics

The evaluation method in this study is the macro average. Because the problem is a multilabel problem, the whole evaluation process is divided into an individual evaluation of each label, and then all labels are averaged. The label-based metric calculation starts from the precision P_i. R_i and the F1 score F_i for each class c_i are calculated as follows:

$$P_i = \frac{tp_{c_i}}{tp_{c_i} + fp_{c_i}}, \tag{1}$$

$$R_i = \frac{tp_{c_i}}{tp_{c_i} + fn_{c_i}}, \tag{2}$$

$$F_i = \frac{2 \times P_i \times R_i}{P_i + R_i}, \tag{3}$$

where tp_{ci} , fp_{ci}, and fn_{ci} are the number of true positives, false positives, and false negatives for class c_i, respectively.

5. Results and Discussion

We repeated the experiment 20 times to calculate the average F1 score. The results in Table 1 show that the deep learning algorithms such as fastText and the MLTCNN yielded far better results than the other algorithms. The result of the SVM is not included because of its excessively long running time. The deep learning algorithms had stronger modeling abilities than the machine learning algorithms. The algorithms improved the precision of legal document classification. The MLTCNN had the best precision, recall, and F1 values.

Table 2 shows that the algorithms in this study took less time than the machine learning ones. The proposed algorithms included data preprocessing and a deep learning model. It took about 2 h to segment and create a glossary from 600000 legal documents. After getting the embedding vector of the word index in the previous step, 30 cycles using the MLTCNN algorithm took about 1.5 h.

Table 1
Results of different models.

Model		Precision	Recall	F1
Binary relevance	LR	0.6132	0.6035	0.6083
	KNN	0.5821	0.5704	0.5762
	GaussianNB	0.4914	0.4954	0.4934
Classifier chains	LR	0.6573	0.6641	0.6607
	KNN	0.6132	0.6039	0.6085
	GaussianNB	0.6398	0.6232	0.6314
Label powerset	LR	0.6291	0.6001	0.6143
	KNN	0.6131	0.5917	0.6022
	GaussianNB	0.4254	0.4396	0.4324
fastText		0.843	0.8434	0.8432
MLKNN		0.6087	0.6025	0.6056
MLTCNN		0.9807	0.9961	0.9884

Table 2
Time comparison of different algorithms.

Model	Time (h)
Machine learning methods	20 (on average)
MLKNN	17
MLTCNN	3

We compared the results from a deep learning algorithm (the MLTCNN) with other algorithms (binary relevance, classifier chains, label powerset, MLKNN, fastText) to demonstrate the superiority of the deep learning algorithms. The MLTCNN outperformed the other methods. The results show that TextCNN is powerful for text classifications. The MLTCNN had better precision and recall than the other algorithms. In addition, it took much less time for processing. The MLTCNN, a deep learning method, can be trained quickly to obtain the model file, which can be saved for actual use. It can classify types of charges for new legal documents effectively. Thus, the MLTCNN is applicable to any multilabel problem as it creates a dictionary from the first step of data preprocessing. This converts the case description in texts directly into index vectors with strong applicability. The whole process of the MLTCNN takes much less time than other methods but gives better results.

6. Conclusions

In this study, we scanned and converted legal documents to text data for processing with a computer. Then, we applied the MLTCNN. The MLTCNN algorithm showed better performance than other algorithms, including binary relevance, classifier chains, the label powerset, the MLKNN, and fastText. We expect the MLTCNN to provide an efficient way to find tendencies, habitual deviations, and errors in similar legal cases from different courts. As charges and trials in legal processes may be arbitrary and varied, the use of the MLTCNN is expected to improve the quality of the legal service and promote fairer law enforcement. On the basis of the results in this study, we will refine the MLTCNN method further by adding more semantic information, which will give faster and more precise results.

Acknowledgments

This work was supported by the National Key R&D Project (No. 2017YFC1703303), Natural Science Foundation of Fujian Province of China (Nos. 2019J01846, 2018J01555, and 2017J01773), External Cooperation Project of Fujian Province, China (No. 2019I0001), and Science and Technology Guiding Project of Fujian Province, China (No. 2019Y0046).

References

1 F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens: IEEE Trans. Med. Imaging **16** (1997) 187. https://doi.org/10.1109/42.563664.

2 J. Lafferty, A. McCallum, and F. Pereira: Proc. 18th Int. Conf. Machine Learning (2001) 282–289. https://doi.org/10.1109/ICIP.2012.6466940.

3 Z. S. Harris: Papers on Syntax. Syntheses Language Library (Text and Studies in Linguistics and Philosophy) eds. H. Hiż. Distributional Structure. Vol. 14. (Springer, Dordrecht, 1970). https://doi.org/10.1007/978-94-009-8467-7_1.

4 G. Salton and M. Lesk: J. ACM **15** (1968) 8. https://doi.org/10.1145/321439.321441.

5 D. M. Blei, A.Y. Ng, and M. I. Jordan: J. Mach. Learn. Res. **3** (2003) 993. https://doi.org/10.1162/jmlr.2003.3.4-5.993.

6 T. K. Landauer: Encyclopedia of Cognitive Science. Latent Semantic Analysis (Springer, Berlin, 2006). https://doi.org/10.1002/0470018860.s00561.

7 T. Hofmann: Proc. 22nd Annu. Int. ACM SIGIR Conf. Research and Development in Information Retrieval (ACM, 1999) 50–57. https://doi.org/10.1145/312624.312649

8 G. E. Hinton: Proc. 1986 COGSCI 8th Annu. Conf. Cognitive Science Society (COGSCI, 1986) 1–12.

9 Yoshua, D. Rejean, V. Vincent, and J. Jauvin: JMLR **3** (2003). https://doi.org/10.1007/3-540-33486-6_6.

10 T. Mikolov, K. Chen, G. Corrado, and J. Dean: Proc. Int. Joint Conf. Neural Networks (ICLR, 2013) 248. https://arxiv.org/abs/1301.3781

11 T. Mikolov, I. Sutskever, and K. Chen: Proc. 23th Neural Information Processing Systems (NeurlPS, Nevada, 2013) 3111. https://arxiv.org/abs/1310.4546.

12 M. L. Zhang and Z.H. Zhou: IEEE Trans. Knowl. Data Eng. **26** (2014). https://doi.org/10.1109/TKDE.2013.39.

13 C. Saunders, M.O. Stitson, and J. Weston: Support Vector Machine **1** (2002). https://doi.org/10.1007/978-3-642-27733-7_299-3.

14 J. R. Quinlan: Machine Learning **1** (1986). https://doi.org/10.1007/BF00116251.

15 M. E. Maron: J. ACM **8** (1961). https://doi.org/10.1145/321075.321084.

16 T. Chen and T. He: xgboost: eXtreme Gradient Boosting (2020) http://cran.fhcrc.org/web/packages/xgboost/vignettes/xgboost.pdf (accessed April, 2020).

17 Y. Kim: Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP, Doha, 2014) 1747. https://doi.org/10.3115/v1/D14-1181.

18 P. Liu, X. Qiu, and X. Huang: Proc. 2016 Int. Joint Conf. Artificial Intelligence (IJCAI, New York, 2016) 2874. https://arxiv.org/abs/1605.05101v1

19 G. Chen, D. Ye, and Z. Xing: Proc. 2017 Int. Joint Conf. Neural Networks (IEEE, Alaska, 2017). https://doi.org/10.1109/IJCNN.2017.7966144.

20 G. Tsoumakas, I. Katakis, and I. Vlahavas: IEEE Trans. Knowl. Data Eng. **23** (2011). https://doi.org/10.1109/TKDE.2010.164.

21 M. Zhang and Z. Zhou: Pattern Recognit. **40** (2007). https://doi.org/10.1016/j.patcog.2006.12.019.

22 T. Joachims: Proc. 12th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (ACM SIGKDD, 2006). https://doi.org/10.1145/1150402.1150429

23 Standford University: https://cs224d.stanford.edu/reports/BergerMark.pdf (accessed April, 2020).

24 J. Mullenbach, S. Wiegreffe, and J. Duke: ACL Anthology (2018). https://doi.org/10.18653/v1/N18-1100.

25 Github: https://github.com/fxsjy/jieba (accessed April, 2020).

26 M. P. Lavalley: Circulation **117** (2018) 2395. https://doi.org/10.1161/CIRCULATIONAHA.106.682658.

27 G. Gongde, W. Hui, and B. David: Lecture Notes Comput. Sci. **2888** (2003) 96. https://doi.org/10.1007/978-3-540-39964-3_62

28 V. Zolotov and D. Kung: https://arxiv.org/abs/1702.05531v1 (accessed April, 2020).

About the Authors

Ming Qiu received his B.S. degree from Hunan University, China, in 1994 and Ph.D. degree from Zhengjiang University in 2006. Then, he has been working as an assistant professor at the School of Informatics, Xiamen University, since 2007. His research interests include software engineering, especially machine learning, and intelligent information processing technology.

Yiru Zhang was born in Anhui province, China in 1995. He has been studying at Xiamen University since 2017, where he is currently pursuing a master's degree in software engineering. His current research interests include big data, machine learning, and recommendation systems.
(24320171152280@stu.xmu.edu.cn)

Tianqi Ma was born in Anhui province, China in 1995. She has been studying at Xiamen University since 2017, where she is currently pursuing a master's degree in computer science. Her current research interests include machine learning, medical image processing, and deep learning.
(23020171153046@stu.xmu.edu.cn)

Qingfeng Wu received his B.S. degree from Xiamen University in 2000, his M.S. degree from Harbin University of Science and Technology in 2003, and his Ph.D. degree from Xiamen University in 2007. Then, he was an assistant professor at the Software School, Xiamen University. From 2009 to 2015, he was an associate professor. Since 2016, he has been a professor. His research interests include digital media technology, intelligent information processing technology, computer graphics, and bioinformatics. (qfwu@xmu.edu.cn)

Fanzhu Jin received his Ph.D. degree from Seoul National University, Republic of Korea, in 2001. He then started working at Gdansk Medical University as an assistant professor. He is affiliated with the College of Mathematics and Information Engineering, Longyan University, Fujian, China. His research interests are in IoT and its applications.
(dennisbskim@hotmail.com)