Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 2, February 2019, pp. 667-675
DOI: 10.23940/ijpe.19.02.p31.667675

Short Text Classification based on Feature Extension using Information in Images

Shengjie Zhao[a,b,*] and Qianyun Jiang[a]

[a]*College of Electronic and Information Engineering, Tongji University, Shanghai, 200800, China*
[b]*School of Software Engineering, Tongji University, Shanghai, 200800, China*

Abstract

With the quick development and extensive application of the Internet, there is a growing desire for people to share their life or opinions on social networks, which produces a mass of short texts. Short texts are characterized by short length, sparse features, and a lack of contextual information. Thus, it is difficult for conventional methods to achieve high quality classification performance. To achieve a higher classification accuracy, this paper proposes a novel short text classification method based on feature extension by incorporating the information of the images. Specifically, we first generate a sentence that descripts the images by image caption technology, and then we combine the generated sentence with the text as the input of the classifier. Meanwhile, we introduce a similarity module in terms of the correlation between the image and the short text so as to determine whether the two sentences are combined or not. Simulation results show that our proposed model significantly outperforms the state-of-the-art methods in terms of classification accuracy.

Keywords: short text classification; image caption; feature extension; sentence similarity

(Submitted on November 10, 2018; Revised on December 12, 2018; Accepted on January 5, 2019)

1. Introduction

With the rapid development and widespread use of social networks, there are countless short text messages posted on Twitter every day. They are carriers of a great deal of information covering almost all areas, such as entertainment, education, sports, etc. In order to effectively sort out the content that we are interested in, it is fundamentally significant to organize these texts into different categories. However, the texts on Twitter are always very short. Thus, when dealing with these shorter text messages, many classic classification algorithms fail to achieve high quality classification performance as they would have achieved on longer texts. This is according to our intuition, since short texts have the following characteristics:

- Short texts lack context. This means that messages often express the feelings of an event at the time.
- Short texts do not always follow the grammar of language. As we often see on Twitter, the text often contains emoticons, like ☺ and T_T.
- The texts are short in length, generally less than 100 words. That is, these texts may fail to provide enough knowledge about the text itself, which leads to data sparseness problems.

To effectively tackle the challenges induced by such characteristics in short text classification, many well-developed methods have been devised to capture more information from short texts. The most commonly used methods are based on feature extension, which can be divided into two classes, namely based on rules and statistical approaches. The rule-based feature selection methods are proposed, such as 8F and TFIDF [1-2], to pick on the words that are shared by fewer texts. Nevertheless, using these approaches to build the feature space will lead to the sparseness problem as there are very few words shared by the original texts in the case of the short texts. The statistics-based method, from the perspective of machine learning, introduces external resources to expand the short text such as public online knowledge, e.g., WorldNet and HowNet [3-4]. However, feature extension by using searching engines or external resources is computationally

* Corresponding author.
E-mail address: shengjiezhao@tongji.edu.cn

expensive. Manual selection of features is a fully experienced work driven by experience, intuition, as well as domain knowledge.

Nowadays, due to the popularity of deep learning, an increasing number of machine leaning communities have begun to apply deep learning models to text classification. Multi-layer neural network has the capacity to combine low-level text information into a more abstract high-level text representation, thereby enhancing the performance of the classifier model. CNN (Convolutional Neural Networks), for example, is good at extracting features and can greatly reduce the necessity of manual feature extraction in classification tasks. Meanwhile, word embedding technology established on the neural network model is being applied to represent the semantic vector of short texts [5].

In this paper, borrowing the idea of the work of Lynks [6], we combine the text and an associated image that contains complementary information into a single classifier model. Namely, unlike the previous work, we not only analyze the text, but also use the information contained in the image. Recently, an increasing amount of works has exposed rich resources of semantic information conveyed by online user generated content: the texts and the associated images. This shows that people often attach an image when they use Twitter, and in most cases, this image is related to the content of the text. Thus, we propose a classifier model with related information drawn from the associated images that can deal with short and sparse text successfully. To our best knowledge, this is the first work that combines image and text for short text classification.

We generate image description by image caption technology and then combine the generated sentences with texts as the input of neural networks for text classification. Nevertheless, the image probably has nothing to do with the text, which gives rise to result that the information contained in the image is interference information. Therefore, we need to determine whether the information generated from the images is the available information by calculating the relevance of the image description and the text. Ultimately, we adopt a model that is a bit more complex but can yield significantly better performance compared to the most advanced approaches.

Our contributions in this paper are as follows:

- We propose a novel feature extension approach by utilizing the information contained in images for short text classification. We employ the image caption technology to generate the description of the image.
- We also take the correlation between image and text into account by introducing the similarity module.
- Our experiments demonstrate that the model we proposed is indeed effective. Furthermore, the classifier we proposed can obtain nearly 10% accuracy improvement.

We organize the rest of paper as follows: firstly, we give a brief overview of works related to image caption generator and short text categorization in Section 2. Then, we describe each part of our model in Section 3. Experiments and results are presented in Section 4. Lastly, we summarize the contributions of this paper.

2. Related Works

In the last few years, researchers in related fields have introduced some new approaches to address short text classification problems [7]. A large number of works have managed to reduce the spatial dimension by integrating additional features. The most widely used feature is to represent text as a vector, e.g., Continuous Bag-of-Words (CBOW), continuous skip-gram model, and glove model [8-9]. Latent topic models like LDA (Latent Dirichlet Allocation) are often used for document classification. Blei et al. [10] developed the LDA algorithm to extract topic information from short text, which can be represented as a multinomial distribution of words in the vocabulary. Then, these potential latent topics can generate the document. However, these approaches require sufficient information of word occurrences. Some methods rely on human selected features to address the short text sparse problem. The characteristics of twitter text were fully exploited by Sriram et al. [1] , and they put forward to present the short texts by using a small set of domain-specific features (such as the existence of shortened words and slangs, opinions, reference to another user, and so on). Some studies have also introduced methods of short texts expansion by using searching engines. It is easy to see that querying these data sources online can cause long-term problems, and snapshots using these data sources can lead to outdated information.

Due to the remarkable performance of deep neural networks, many papers have been devoted to experiments on sentence-level classification tasks with the CNN model trained on pre-trained word vectors [11]. Johnson et al. [12] directly used the one-hot vector as input to the CNN to reduce the number of learning parameters of the model. Kim [13] further improved the performance by using multi-channel embedding. Santos et al. [14] combined the word-level and sentence-level features learned from the short text sequence to improve the accuracy of short text classification. These showed that a

simple CNN with few hyper-parameters tuning can achieve good performance in a range of tasks. Inspired by these works, in this paper, we settled on a CNN model that is flexible but still easy enough.

As mentioned before, the information contained in the image can be used as a feature of the short text classification. However, understanding the content of images is hard, especially when the images are blurred. The goal of the image caption technology is to generate grammatical and consistent sentences of pictures. There are several models for generating image descriptions. Kiros et al. [15] first addressed the caption generation by introducing a neural network structure. Recently, many models based on RNN (Recurrent Neural Networks) have been proposed, which adopt encoder-decoder framework wildly used in machine translation [16]. Using this framework allows us to "translate" an image into a sentence. LSTM (Long Short Term Memory) [17] units are utilized to "decode" the feature vectors "encoded" by CNN. A generation model based on the deep recursive structure was proposed in [18-19]. By combining the deep convolutional network in computer vision with the recursive network, the natural statement describing the image was generated. In our work, we utilize a deep learning model under probabilistic framework to generate descriptions from images.

We can combine the sentences generated by image caption with the original texts for text categorization. However, as mentioned in Section 1, the information contained in the image may be interference. Thus, we need to evaluate the correlation between the generated sentence and the text by mapping them into a joint feature space. Several complex lexical, syntactic, and semantic features were used to encode the input text pairs [20], and then various similarity measures between the representations could be obtained. Meanwhile, [13, 21] showed that convolutional neural networks can learn the low-dimensional vector from the input sentences and retain important syntactic and semantic aspects of the input sentence, which makes the results of many NLP tasks the most advanced. The distributed sentence model based on CNN (Section 3.1) is the main building block of the model in this paper. There are two potential sentence models working in parallel, projecting the generated sentence and text into vectors, which are then used to obtain the similarity score between them [22].

3. Overall Framework

Our goal is to classify the short texts based on feature extension using information in images. Thus, we decided to combine texts and images into a single machine learning model, since they contain complementary information. In order to combine original texts and image contents together as a new input of CNN text classifiers, they must be projected to the same vector space. The overall framework of the model is shown in Figure 1. Furthermore, the general idea of our model is this: we extract image features through a CNN image embedder, followed by a LSTM model to generate the corresponding sentence. Then, we project the generated sentence and the original text into the intermediate feature representation, represented by the red grids in the picture. Then, the similarity score is calculated between the generated sentence and the original text to determine whether or not the two sentences are combined. Finally, the concatenated text is sorted by a CNN classification network. Below, we will describe each part of the function and module in detail.

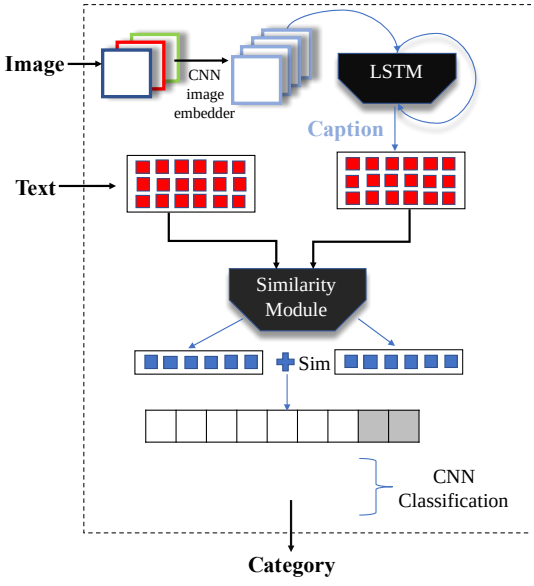

Figure 1. The overall framework of the model

3.1. Convolution Sentence Model

Since the distributional sentence model is involved in both the sentences similarity module and the sentences classification module, we first propose a convolutional architecture for sentence modeling. The goal of this sentence model is to convert input sentence into an intermediate feature representation, which is then used for computing the semantic similarity or performing various sentence classification tasks.

As shown in Figure 2, this convolutional sentence model consists of a convolution layer with multiple filter sizes followed by a simple *max* pooling layer. It takes as input fixed-length embedding vectors (pre-trained word embedding vectors), through convolution layer and max pooling layer, until a fixed length representation is reached on the last level. Like most convolutional neural network models [13, 23], we adopt a convolution filter taking on the characteristic of " shared weights".

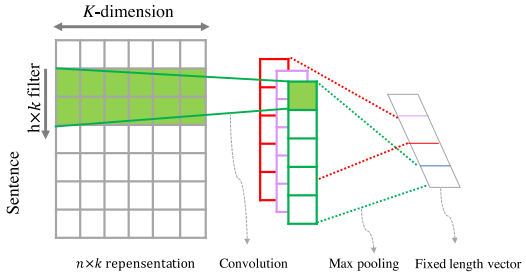

Figure 2. A convolutional layer with multiple filter widths and feature maps followed by a simple max pooling layer

We denote $\mathbf{S} \in \mathbb{R}^{n \times k}$ as the input of the network, where n represents the length of the sentence (padded where necessary) and k represents the dimension of word embedding. The function of convolution layers is to extract different level features from the input matrix.

Given a convolution filter $\boldsymbol{\omega} \in \mathbb{R}^{h \times k}$, a feature $\boldsymbol{F}_i$ is generated from a window of h words $[w_i : w_{i+h-1}]$ by:

$$\boldsymbol{F}_i = f(\boldsymbol{\omega} \cdot [w_i : w_{i+h-1}] + \boldsymbol{b}). \tag{1}$$

Here, $\mathbf{b} \in \mathbb{R}^{n \times k}$ represents the bias term. f is the non-linear activation function. In this work, we take ReLU as the activation function of the convolution layers. The filter slides on the sentence S to produce a series of *feature map* $\boldsymbol{F} \in \mathbb{R}^{n-h+1}$. This process can be repeated for various filters with different sizes to increase the feature coverage of the model.

The function of the *max* pooling layer is to further abstract the features generated from convolution layer. The idea is to choose the maximal value $\widehat{\boldsymbol{F}}$=max($\boldsymbol{F}$) to capture the most important feature. With the pooling layer, we can induce a fixed-length vector, which can be used as the input of the other modules.

3.2. Extracting Information from Image

The image caption problem can be defined as a binary *(I*, **S***)* form, where *I* is an image and **S** is a sequence of target words. The overview of the image caption module can be seen in Figure 3.

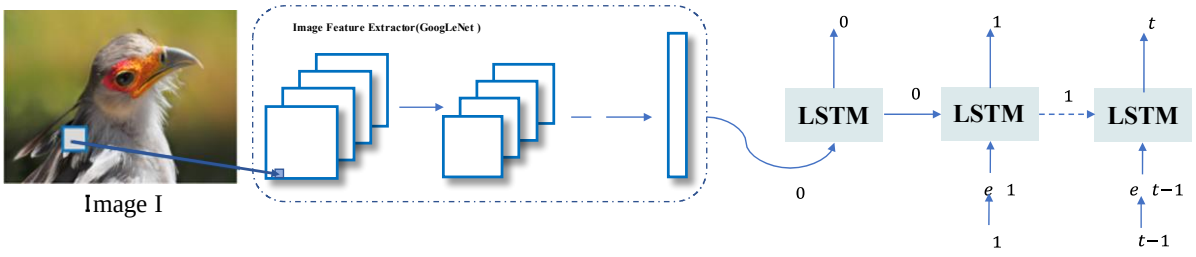

Figure 3. The overview of the image caption module

The module includes two parts: "encoder" and "decoder". The encoding part makes use of GoogLeNet [24], which "encodes" the given image into a fixed dimensional feature vector. The decoding part adopts the classical LSTM model to

"decode" the feature vector to the desired output sentences. The LSTM consists of a series memory cells (see Figure 4). Each cell performs the same task so that they can share some parameters. The output of a cell depends on the current input $\boldsymbol{x}_t$ and "memory" state $\boldsymbol{h}_{t-1}$. As we can see from the Figure 4, there are three gates in a cell (input gate i, output gate o, and forget gate f). The function of the forget gate is to decide what information we want to discard from the state of the cell. It puts the input $\boldsymbol{x}_t$ and the last hidden state $\boldsymbol{h}_{t-1}$ into a sigmoid layer and decides how to process the previous information of unit $\boldsymbol{c}_{t-1}$. Input gate also takes as input $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$, but it has a different function to update the memory cell. This step is to decide what new information we intend to "remember" in the cell state. With the function of these two gates, we can change the state of the memory unit. Finally, the LSTM unit uses an output gate, which takes the same input as the other two gates, and gets the result according to the state of the unit.

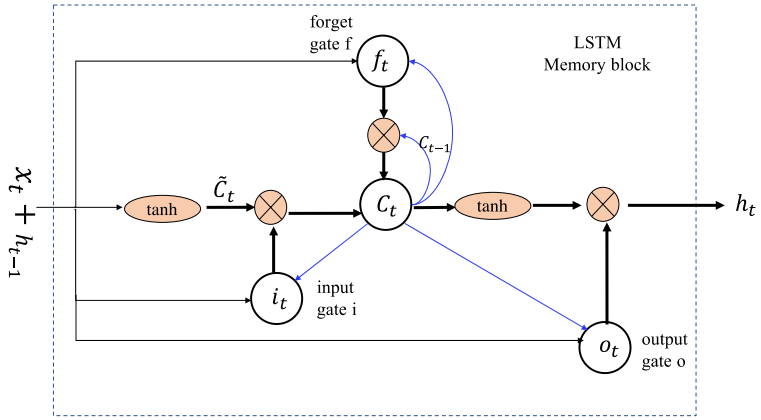

Figure 4. ⊗ represents the product with a gate value

Therefore, the LSTM model can be trained to predict word $\mathbf{s}_t$ of the description in the case of known image I as well as all predicted words $[\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_{t-1}]$, which is defined by $P(\mathbf{s}_t | I, \mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_{t-1})$. Our implementation of LSTMs closely follows the one used in [25].

$$\boldsymbol{x}_t = \boldsymbol{w}_e \mathbf{s}_t \tag{2}$$

$$f_t = \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f) \tag{3}$$

$$i_t = \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i), \tag{4}$$

$$\tilde{C}_t = tanh(\boldsymbol{W}_c \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c) \tag{5}$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t, \tag{6}$$

$$o_t = \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \tag{7}$$

$$\boldsymbol{h}_t = o_t * \tanh(C_t) \tag{8}$$

$$P(s_t | I, \mathbf{s}_0, \cdots, \mathbf{s}_{t-1}) = Softmax(\boldsymbol{W}_p \boldsymbol{h}_t) \tag{9}$$

Where the various W_* and b_* are trained parameters, $\boldsymbol{w}_e$ are word embeddings, and σ is the logic sigmoid function. We project each word into a one-hot vector $\mathbf{s}_i$, and the dimensions of these vectors are equal to the vocabulary size. These gates make it possible to properly handle the exploding and vanishing gradients issues that the traditional RNN suffered [26]. $\boldsymbol{h}_t$ in Equation (8) is the input of the Softmax. Equation (9) can obtain the probability distribution of all possible sentence words. The image I is put only at the very beginning when t = 0, to notify the LSTM model of the image content.

We recommend maximizing the probability of a correct description of the given image directly by using the following Equation (10). Thus, we can obtain the probability of a sentence by multiplying the probability of each word in the image and the probability of words before the current time.

$$P(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_m) = \prod_{i=0}^{m} P(\mathbf{s}_i | I, \mathbf{s}_0, \ldots, \mathbf{s}_{i-1}) \tag{10}$$

The log likelihood sum of the correct words in a sentence sequence is defined as the loss of our model. We can optimize Equation (11) by adopting the stochastic gradient descent method throughout the training.

$$\mathrm{L}(I, \mathbf{S}) = \sum_{i=0}^{m} \log(P(\mathbf{s}_i | I, \mathbf{s}_0, \ldots, \mathbf{s}_{i-1})) \tag{11}$$

For all parameters of LSTM as well as word embeddings $\boldsymbol{w}_e$, our goal is to minimize the above loss.

3.3. Similarity Score of Sentences

The architecture of sentences similarity module is presented in Figure 5. The inputs of the model are the output of the previous layers, which can be utilized to compute their similarity. We use the texts and images of tweets where the topic is related as positive samples, otherwise as negative samples. Here, we are going to describe our similarity module in detail:

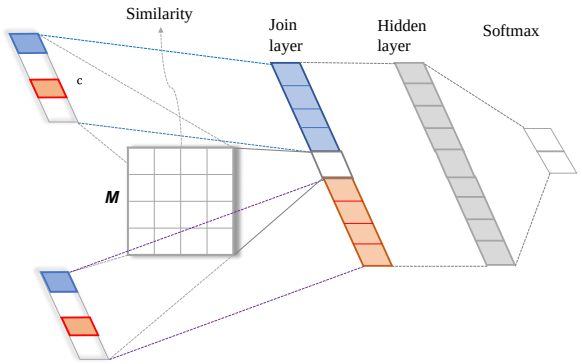

Figure 5. The architecture of similarity module, where **M** represents the similarity matrix

A distributional representation can be obtained by feeding the generated sentence and the text into our sentence models (Section 3.1), which can be denoted as $\boldsymbol{x}_c \in \mathbb{R}^d$ and $\boldsymbol{x}_o \in \mathbb{R}^d$ respectively. The similarity score is then calculated using the similarity matrix **M** according to Equation (12). The similarity between the $\boldsymbol{x}_c$ and $\boldsymbol{x}_o$ vectors can be defined as follows [27]:

$$\boldsymbol{x}_{sim}(\boldsymbol{x}_c , \boldsymbol{x}_o) = \boldsymbol{x}_c^T \mathbf{M} \boldsymbol{x}_o \tag{12}$$

Where $\mathbf{M} \in \mathbb{R}^{d \times d}$ represents the similarity matrix, which is also the parameter of our model and is optimized through the training process (Section 4.2).

The single result $\boldsymbol{x}_{\boldsymbol{sim}}$ is a measure of the similarity (including syntactic and semantic) of two sentences. Therefore, according to Equation (13), we concatenate two intermediate vectors and the similarity index $\boldsymbol{x}_{sim}$ into a single vector $\boldsymbol{x}_{join}$ in the joint layer [27]:

$$\boldsymbol{x}_{join} = [\boldsymbol{x}_c^T; \boldsymbol{x}_{sim}; \boldsymbol{x}_o^T] \tag{13}$$

This vector is passed to next hidden layer, which is fully connected. This layer can model the interactions among the concatenated representation vector. This following transformation is conducted by the hidden layer:

$$\alpha(\mathbf{w}_h \cdot \boldsymbol{x}_{join} + \mathbf{b}) \tag{14}$$

Where $\mathbf{w}_{\boldsymbol{h}}$ is hidden layer's weight vector and α represents the non-linear function.

The similarity score between the two sentences is approximately equal to the correlation between the text and the image. Therefore, as described earlier, we can determine whether to consider the information contained in the image or not when classifying sentences according to Equation (15).

$$\mathbf{P} = \begin{cases} \mathbf{S} \oplus \mathbf{Y}, sim = 1 \\ \mathbf{Y}, sim = 0 \end{cases} \tag{15}$$

Where **S** is the sentence generated by LSTM model and **Y** is the original text. ⊕ denotes concatenation, and we

concatenate them as $\mathbf{P} = (r_1, r_2, \cdots, r_l)$, where l is the length of the sentences (padded where necessary) and is a fixed value. Namely, if the two sentences are related, then the image caption and the original text concatenate together; otherwise, they are padded with a special character.

3.4. CNN Model

The last step of our model is a fully-connected deep learning model to process the text. The last layer of our model is simply the combination of both parts of the model, and it is used to produce the final classification. Readers who are interested in this subject can find more detailed discussions in [12-13,17].

4. Experiments and Results

4.1. Dataset

The dataset we perform experiments on is MS COCO (the Microsoft Common Objects in Context), which was carefully created to provide resources for automatic image caption generation tasks [28]. Currently, the MS COCO 2014 dataset contains one million captions and more than 160,000 images. Compared to the existing image caption dataset, for example, flickr8k or flickr30k, the COCO dataset is advantageous because it has more images and annotations for training and testing. Corresponding to the problem that we need to solve, we regard MS COCO as the positive samples and add some texts and images with irrelevant content as the negative samples.

4.2. Training Details

When we train models on the MS COCO dataset, we inevitably encounter over-fitting problems. In fact, supervised machine learning methods need a mass of data for training, while the dataset we used has less than 100000 images, which is far from enough. Thus, we explore several techniques to handle overfitting. In the image caption generation module, we decide to make $\boldsymbol{w}_e$, the word embeddings randomly initialized. Meanwhile, for the Convolution Sentence Model (Sec 3.1), we initialized the embedding vectors with pre-trained word embedding vectors. Due to the complexity of our model, we implemented some model-level techniques for avoiding over-adaptation. We tried the drop and ensemble models and explored the size of the model by weighing the number and depth of hidden units.

The parameters of the SGD (stochastic gradient descent) network are optimized by backpropagation algorithm. We randomly initialized all weights except for the weights of the CNN model. We set the dimension size as 512 for the embeddings as well as the size of the LSTM memory.

4.3. Generation Results

The first experiment performs in the absence of the similarity module. The experiments are carried out on binary datasets. The combined model we proposed is larger and a bit more complex than ones that focus only on text or images. In order to understand and justify the performance gains of the model, it makes sense to look at each component separately and compare them to the final model:

The experimental result is quite dramatic as the performance of only analysing text is slightly better than analysing only images, while the computational cost is much lower. Moreover, it is striking that the images alone can also achieve good classification performance. We can draw a conclusion from Table 1 that the CNN model outperforms the other two traditional text classification models (SVM and LDA). It is intuitive that the combination of both the images and the text should lead to the best performance since they presumably pick up on different things. Indeed, when we focus on the performance of the model, we see a significant increase in precision rate. Figure 6 shows the ROC curve (receiver operating characteristic curve). This curve plots two parameters: True Positive Rate and False Positive Rate. The pre-trained GoogLeNet used here is trained on 1000 classes of images in Image-Net.

Table 1. The comparation of three models: classifier only using text or images and the classifier using both (not add the similarity module)

Method	ResNet (Images Only)	SVM (Text Only)	LDA (Text Only)	CNN (Text Only)	Images +Text
Precision Rate	0.8496	0.8412	0.8210	0.8651	0.9550

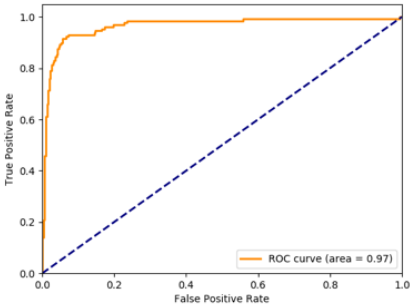

Figure 6. The receiver operating characteristic curve

The second experiment is to introduce the similarity module on the basis of the experiment one. As mentioned in the introduction, in real life, the image we sent sometimes has nothing to do with the text. In order to simulate this phenomenon as much as possible, we add some negative samples (the images and texts are irrelevant) to the COCO dataset. Experiment with our model on this dataset. The experimental results are as follows:

From the experimental results in Table 2, we can reach the conclusion that the accuracy of the classification decreases after adding the irrelevant data. With the introduction of the similarity module, the accuracy of the classification increases by nearly 10%, which also proves that our idea is correct.

Table 2. The comparation of whether to add similarity module

Strategy	Similarity module absent	Add similarity module
Classifier accuracy (%)	72.08	82.14

5. Conclusions

This paper introduces a feature extension method that takes advantage of information in images for short text classification. We enrich the features of short texts with the help of image caption technology to capture the complementary information in images. The method we proposed for feature extension does improve short text classification. Meanwhile, we also take the relevance of images into account by adding the similarity module, and the accuracy is further improved. Then, we compare the proposed method with other existing classification models. A series of experimental results show that our proposed method is better in classifying the short text.

References

1. B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas, "Short Text Classification in Twitter to Improve Information Filtering," in *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 841-842, 2010
2. M. Wang, L. Lin, and F. Wang, "Improving Short Text Classification through Better Feature Space Selection," in *Proceedings of International Conference on Computational Intelligence and Security*, pp. 120-124, 2014
3. D. Bollegala, M. Ishizuka, and Y. Matsuo, "Measuring Semantic Similarity Between Words using Web Search Engines," in *Proceedings of International Conference on World Wide Web*, pp. 757-766, Banff, Alberta, Canada, May 2007
4. X. Hu, N. Sun, C. Zhang, and T. S. Chua, "Exploiting Internal and External Semantics for the Clustering of Short Texts using World Knowledge," in *Proceedings of ACM Conference on Information and Knowledge Management*, pp. 919-928, 2009
5. A. Paccanaro and G. E. Hinton, "Learning Distributed Representations of Concepts using Linear Relational Embedding," *IEEE Transactions on Knowledge & Data Engineering*, Vol. 13, No. 2, pp. 232-244, 2002
6. X. Zhang and B. Wu, "Short Text Classification based on Feature Extension using the n-Gram Model," in *Proceedings of 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery* (*FSKD*), pp.710-716, IEEE, 2015
7. Christopher Bonnett, "Classifying E-Commerce Products based on Images and Text," (http://cbonnett.github.io/Insight.html)
8. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Proceedings of International Conference on Neural Information Processing Systems*, pp. 3111-3119, 2013
9. J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 1532-1543, 2014
10. D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *J Machine Learning Research Archive*, Vol. 3, pp. 993-1022, 2003
11. A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning Structured Embeddings of Knowledge Bases," *AAAI*, Vol. 6. No. 1, 2011

12. R. Johnson and T. Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks," *arXiv preprint arXiv*:*1412.1058*, 2014
13. Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv preprint arXiv*:*1408.5882*, 2014
14. C. N. D. Santos and M. Gattit, "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts," in *Proceedings of International Conference on Computational Linguistics*, 2014
15. R. Kiros, R. Salakhutdinov, and R. Zemel, "Multimodal Neural Language Models," in *Proceedings of International Conference on Machine Learning*, pp. 595-603, 2014
16. K. Cho, B. Van Merri ënboer, C Gulcehre, et al., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv*:*1406.1078*, 2014
17. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, pp. 1735-1780, 1997
18. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156-3164, 2015
19. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. "Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 652-663, 2016
20. M. Surdeanu, M. Ciaramita, and H. Zaragoza, "Learning to Rank Answers to Non-Factoid Questions from Web Collections," *Computational Linguistics*, Vol. 37, No. 2, pp. 351-383, 2011
21. N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," *arXiv preprint arXiv*:*1404.2188*, 2014
22. A. Severyn and A. Moschitti. "Modeling Relational Information in Question-Answer Pairs with Convolutional Neural Networks," *arXiv preprint arXiv*:*1604.01178*, 2016
23. O. Abdel-Hamid, A. R. Mohamed, H. Jiang, and G. Penn, "Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4277-4280, 2012
24. C. Szegedy, L. Wei, J. Yangqing, et al., "Going Deeper with Convolutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9, 2015
25. W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *Eprint Arxiv*, 2014
26. L. M. Surhone, M. T. Tennoe, and S. F. Henssonow, "Long Short Term Memory," *Betascript Publishing*, 2010
27. A. Severyn and A. Moschitti, "Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks," in *Proceedings of the International ACM SIGIR Conference*, pp. 373-382, 2015
28. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, et al., "Microsoft coco: Common Objects in Context," in *Proceedings of European Conference on Computer Vision*, Springer, pp. 740-755, 2014