Learning the Chinese Sentence Representation with LSTM Autoencoder

Mu-Yen Chen
Department of Information Management
National Taichung University of Science and Technology
Taiwan
mychen@nutc.edu.tw

Tien-Chi Huang
Department of Information Management
National Taichung University of Science and Technology
Taiwan
tchuang@nutc.edu.tw

Yu Shu
Department of Industrial Education and Technology
National Changhua University of Education
Taiwan
vera.yushu@gmail.com

Chia-Chen Chen
Department of Management Information Systems
National Chung Hsing University
Taiwan
emily@nchu.edu.tw

Tsung-Che Hsieh
Department of Information Management
National Taichung University of Science and Technology
Taiwan
s4851798@gmail.com

Neil Y. Yen
School of Computer Science and Engineering
University of Aizu
Japan
neilyyen@u-aizu.ac.jp

ABSTRACT

This study[1] retains the meanings of the original text using Autoencoder (AE) in this regard. This study uses the different loss (includes three types) to train the neural network model, hopes that after compressing sentence features, it can still decompress the original input sentences and classify the correct targets, such as positive or negative sentiment. In this way, it supposed to get the more relative features (compressing sentence features) in the sentences to classify the targets, rather than using the classification loss that may classify by the meaningless features (words). In the result, this study discovers that adding additional features for correction of errors does not interfere with the learning. Also, not all words are needed to be restored without distortion after applying the AE method.

KEYWORDS

Deep learning, Autoencoder, Long Short-Term Memory (LSTM), Chinese Sentence Representation, Sentiment classification

ACM Reference format:

M. Y. Chen, T. C. Huang, Y. Shu, C. C. Chen, T. C. Hsieh, and N. Y. Yen. 2018. SIG Proceedings Paper in word Format. In *The 2018 Web Conference Proceedings (WWW 2018), April 23-27, 2018, Lyon, France*, IW3C2, 6 pages.
DOI:https://doi.org/10.1145/3184558.3186355

[1] This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

1 INTRODUCTION

Sentiment analysis is mainly applied in sentimental data mining to determine the current status. Therefore, it is mostly used by hotels, restaurants, and online shopping platforms for collection of customer opinions [1]. To start with, one common problem such methods face is how to determine the sentimental implications of users' messages and comments. To solve this problem, machine should be trained to understand the meaning of sentences. The most traditional way to do this is by finding key feature words and categorized them. For example the widely known term frequency–inverse document frequency (TF-IDF). However, it does not take into account the complex relations between lexical words. In order to solve the implicit relations between the words [2], researchers attempted using singular value decomposition (SVD), principal component analysis (PCA), and other methods of reduction of dimension for feature extraction. By doing so, rather than simply analyzing the features of keywords, sentences are projected onto other vectors and their implicit features can be shown subsequently. This concept can be interpreted as researchers attempting to use continuous vectors to denote discrete sentences. Yet this method could not really show the implicit features of sentences. It merely targets the composition of lexical words shown in texts. Researches on the extraction of text sentence have continued to be conducted. In recent years, Google made its word2vec an open source tool [3], giving new meanings to the word vector definition, in that it creates an implicit influential relation between word vectors built upon sentence structures. The idea opens up a new path for research on text analysis and feature extraction.

In recent years, deep learning methods have become popular in extracting complex features from large amounts of data. It is also commonly used in research fields concerning natural language processing (NLP). Text features can be extracted from many aspects, such as keyword features, the influential relations between lexical words, or grammar structures of sentences. Using traditional statistical methods for extraction of the complex relations in texts under large data dimension, however, may require a staggering amount of research fund. In addition, the extracted features should be in line with the hypothetical situation proposed by the user. Deep learning methods, on the other hand, greatly simplify many complex relations, while adopting iterative methods to approximate undiscovered features within and beyond traditional statistical methods. In using deep learning methods to conduct text analysis, convolutional neural network (CNN) and recurrent neural network (RNN) are two commonly used structures. Kim [4] adopted CNN in his experiments, in which features of lexical words in a sentence were collected in accordance with the filter size of each window for categorization into positive or negative sentiments. Dos Santos & Gatti [5] used multiple CNN structures to perform efficiency experiment on short-text categorization. Li & Qian [6] used Long Short-Term Memory (LSTM) to conduct research on multiple categorizations of text sentiments It is worth noting that using deep learning methods for extraction of features may result in overfitting. Intuitively, it is not reasonable for a categorizer to correctly determine positive or negative sentiments based on meaningless lexical words. Therefore, this study retains the meanings of the original text using Autoencoder (AE) in this regard. It serves as an unsupervised neural model frequently used for compression. The study hopes that after compressing sentence features, it can still decompress and restore the original input sentences. This way, such compression feature could show the implicit meanings of a sentence, rather than a feature composition of meaningless lexical words.

The study uses LSTM for extraction of implicit meanings of sentences. With a view to ensuring that the extracted features can provide better interpretations, the study incorporated the concepts of Autoencoder. The hypothesis of the study is that, when all categorization models achieve a certain level of accuracy, the difference between implicit meanings extracted by LSTM with AE and implicit meanings directly extracted by LSTM lie in the fact that the former has deeper and more intuitive meanings, and is thereby closer to the intended meanings.

2 LITERATURE REVIEW

2.1 Sentiment Analysis

Sentiment analysis is used to understand the sentiment of a text by extracting features from said text. Currently, there are two methods. One is by consulting sentiment dictionary to search from texts [7]. Searched lexical words from sentences or articles are weighted or scored. Determination of the type of sentiment is made according to the weighting or scores. Another is by conducting machine learning to perform feature learning and categorization. The study uses the later. Ortigosa et al. [8] mined the English lexical database of Facebook to map out a person's mood swing. The whole process includes data collection, data cleaning, and establishment of lexical database (to acquire feature lexical words). Lastly, Support Vector Machine (SVM), C4.5, and Naive Bayes were used for categorization of sentences. The input pattern, based on traditional vector representation methods, is shown by a comparison between sentences and feature lexical words shown and not shown in the lexical database. Kim [4] adopted CNN to train semantic models. Due to the fact that CNN would automatically collect features using slide windows, it mainly functions as a collector of neighboring lexical words at intervals for input. Li et al. [9] conducted semantic mining on Weibo, mainly collecting texts from disaster-related articles. First, word vectors of texts were pre-trained by Google's word2vec. Later, CNN was used for categorization and setting up models. The main purpose of his article is to mine features of lexical words. Methods similar to word vector compression were adopted to categorize sentences within set-up word vectors. Sentences were categorized into positive or negative sentiment. Filtering properties of CNN were used to observe the selected lexical words to pinpoint important feature lexical words in the 'positive' and 'negative' categories.

2.2 Deep Learning for Semantic Extraction

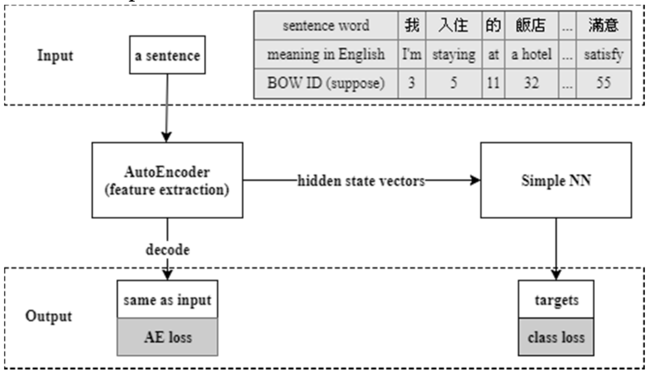
2.2.1 Semantic Vector. Semantic vector should first be viewed through small word vectors, which can be viewed as a denotation of the conversion from discrete lexical words to vectors with meanings. Among all traditional conversion methods, the one-hot representation of BOW (Bag of word) is the most frequently used. However, the meanings denoted by this method could only recognize different lexical words, while similar words may require the consultation of synonym dictionaries, making actual processing more complicated. Hence, researches that followed tend to use matrix factorization to extract the relation between lexical words and data. Word vectors should contain more meanings. Therefore, researches on semantics in recent years have begun to focus more in this regard. The word vector proposed by Mikolov et al. [10] refers to the use of slide windows to collect words and their neighboring words (can be viewed as target words and feature words next to target words) for conducting training for shallow neural networks. After the training, the hidden layer would represent the word vector of the input words (similar to the concept of compression). The concept of word vector and its granularity may increase, meaning that it may go beyond the relations between words and words. The concept may also be applied in sentences and articles for examining the relations between them.

2.2.2 Deep Learning for Text Analysis. Cheng et al. [11] trained lexical data collected from Weibo and adopted RNN for categorization of text features based on input between each interval. Zhou et al. [12] conducted research on QA(Question Answer) systems, with an eye to making clear the implicit relations between questions and answers. Their data were provided by YAHOO API. This study hopes to approximate questions and their best answers after projecting them onto a certain vector, therefore it uses Autoencoder to compress semantic meanings. Finally, SVMRank is used for sequencing.

The study hopes that by doing so, best answers can be recommended first. Chen et al. [13] also conducted research on QA system. It used data collected from WIKI for training and the RNN structure as the model for extraction of implicit features between questions and answers. Araque et al. [14] used six open source data sets, all of which sourced from, Twitter and movie reviews. The data sets were all labeled either 'positive' or 'negative'. The data was pre-processed; punctuation marks, URLs, numbers, user names and other irrelevant words were deleted before the data was input. Convolutional vectors were used to extract in-depth features for sentimental polarity prediction. During the prediction, single-logic features in new data could be extracted. Each surface feature would be put into one category, which served as its complete feature. Then, features of the target fields were defined for new input data to determine target fields based upon their trainings for feature recognition. In the experimental result, the accuracy of six data sets was evaluated based on different indicators. Gui et al. [15] adopted linear equations to match the statistical features of product reviews and words of previous posts by product users. Then, they reconstructed a new feature vector, which is categorized according to CNN. The results showed that this conversion is valid. Park et al. [16] collected 611,590 news articles from 2010 to 2014. During a baseball game, every movement of player is recorded for statistical purposes. Their data was collected from the official score sheet provided by Korea Baseball Organization (KBO). The study proposed a player evaluation model based on deep learning. It is combined with quantitative statistical analysis for sports and qualitative analysis for news articles. Sports statistical data was used to categorize and label players' performance as 'positive' or 'negative'. The labels were used as target fields of articles related to rated players. Labeled articles were used in Deep Neural Network (DNN) categorizer for training of sentence-by-sentence recognition. Based on the results of QA system, their study proved that it is reasonable to use polarity scores in predicting the result of a baseball game. Ronnqvist & Sarlin [17] proposed the use of deep learning methods in examining relevant discussions of texts. They also extract natural language descriptive features by using DNN. The model is based on the unsupervised learning mechanism denoted by semantic vectors of extensive text data and validated the applicability of news articles in financial risk research. Furthermore, they gave examples to explain how texts and constantly-updating, and extensively applicable descriptive data may serve as a helpful supplementary sources for financial and system analysis. Kraus & Feuerriegel [18] used RNN to extract features from time sequence and found highly nonlinear relations. They used different lexical databases to predict the stock trend, and proved that the accuracy compared with traditional machine learning methods is higher. Therefore, it is helpful to reveal the commercial value of deep learning.

3 LSTM WITH AUTOENCODER CLASSIFIER

This chapter explains the structure of this study. First, the study extracts features of sentences before restoring them based on said features, to ensure that said features hold deeper meanings, rather than simply modifying the weighting. The errors taken into account in this study include: errors arising from word restoration and errors arising from categorization, as shown in the following formula: $\text{Loss}_{total} = Loss_{AE} + Loss_{class}$. $Loss_{AE}$ is the loss of Autoencoder. $Loss_{class}$ is the loss of the targets, including positive and negative sentiment. In theory, if the error of each word so serious as to affect the efficiency of categorization, no deeper meanings can be extracted from words in sentences being compressed, or the extracted semantic meanings are completely irrelevant to targets set for categorization. This means that the hypothesis of this study is not valid in regards to machine learning. In contrast, if the errors, after being aggregated, do not have too much effect on categorization, the extracted semantic meanings hold a certain level of interpretation for abstract ideas.



sentence word	我	入住	的	飯店	...	滿意
meaning in English	I'm	staying	at	a hotel	...	satisfy
BOW ID (suppose)	3	5	11	32	...	55

Figure 1. LSTM with Autoencoder classifier. Left: Feature extraction layer based on Autoencoder, performed through LSTM. Right: Target for categorization (positive or negative sentiment); output layers all linked to softmax layer.

First, this study uses LSTM for extraction of sentence concepts and features. The reason is that using traditional RNN often leads to a serious problem of vanishing gradient. Chung et al. [19] also proposed the same, thereby assuming the importance of words sequence in every sentence. And owing to the fact that humans tend to understand the meaning of a sentence from a single perspective, sentences are input from the beginning of the sentence into LSTM. The image below shows the step-by-step process of LSTM modification of weighting. Concepts of Autoencoder (AE) are incorporated; therefore its final purpose is to input itself in hopes of restoring itself after being compressed. The compressed semantic vectors are stored in 256 neurons. The number of target node point being output at last shall equal to words in the dictionary. The formula for calculating errors of LSTM and AE is Cross Entropy, shown as $Loss_{AE} = \sum_{w=0}^{|S|} \left(- \sum_{i=1}^{|BOW|} (real_i^w \log(predict_i^w)) \right)$. $|S|$ is the length of a sentence; w is the index for the current word in a sentence; i is the BOW index of words; 0 is the reserved value, therefore the calculation starts from 1; |BOW| is the total amount of words in the dictionary; real is the actual target value; predict is the predicted value. In all, the formula aggregates the errors of all words in one sentence.

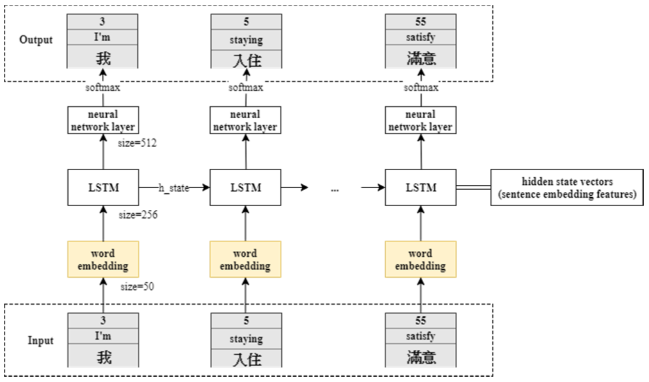


Figure 2. LSTM with Autoencoder. One sentence input at a time for compression in LSTM. Sentences are later decompressed error feedbacks are given. During the prediction, the targets are the most likely candidates among all words in the dictionary.

Based on the above method, reconstructed features can be extracted from sentences. Vectors of the size of 256 dimensions are used. This study uses this feature to perform categorization of positive and negative sentiments, proposes the following formula for calculating errors: $Loss_{class} = -\sum_{i=0}^{|class|}(real_i \log(predict_i))$. |class| is the total number of target categories, which means positive and negative sentiments.

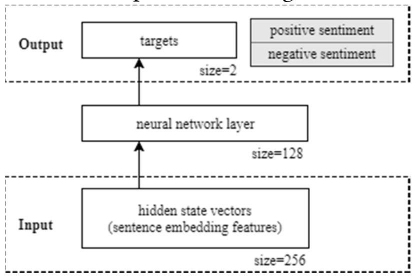


Figure 3. Simple Neural Model. Input: implicit meanings compressed by LSTM.

4 EXPERIMENT DESIGN & PROCESS

4.1 Experiment environment

This chapter introduces the experiment environment of this study, which include the hardware part and the software part. Regarding hardware, the operating system is Ubuntu (64-bit), which comes with 4 CPUs and a 12G RAM. Regarding software, the study uses Python 3.6 for development. Software used for development kit includes Jieba and Pytorch. Jieba kit is an open source software from github for segmentation of Chinese characters. It is also compatible with a wide array of programming languages such as PHP, Python and R. The study uses Jieba 0.39 for segmentation and performs random manual inspection to check whether the segmented sentences make sense. Pytorch is a deep learning framework that evolved from the deep learning software developed by Torch7 for Lua. Later it was transferred to Python, hence named Pytorch. In recent years, tech giants such as Facebook, twitter and others have relied on Pytorch for software development, therefore it has become widely known. The study uses Pytorch because of its user-friendly framework design.

4.2 Experiment Process

This section briefly explains the process of the experiment, which includes four stages: data collection, data pre-processing, categorization and modeling experiment, and model evaluation. Details of each stage are given in the followings.

4.2.1 Data Collection. The Chinese lexical data is collected from datatang. After deleting repeated ones, there are 16680 sets of data, among which positive sentiments for 8680 and negative sentiments for 8000. It can be inferred that data amount of both categories are on the same level. In addition, because they have yet been processed, every sentence is a complete comment and can be put into either one of the category. It is worth noting, however, that every sentence varies in length, and the sentimental lexical words in a comment may conflict with the category of the comment. The reason is that the labeling of each comment is based on the narrative of the whole sentence. Therefore, while the comment goes on, it may mix positive words with negative ones. Use traditional methods and view the weighting of each word as equal may result in great errors in constructing the modeling learning. Thankfully, models like LSTM would adjust the weighting of each word sequence, making learning easier.

4.2.2 Data Pre-Processing. This study uses dictionaries provided by Jieba for segmentation. Punctuation marks are all deleted. The study also makes the length of all sentences uniform, in that although sentences of different lengths can be input into LSTM for training, sentences that are too lengthy get higher weighting. Therefore, to solve the problem, the study makes all sentence lengths equal. If a sentence exceeds the maximum length, the maximum length would be extracted; insufficient parts are made up by padding. Furthermore, because this method may seem unfair in certain circumstances, the study adopts the try-and-error method to figure out the threshold for said batch of data.

4.2.3 Categorization and Modeling Experiment. Modeling is performed in this stage for input of processed sentences. Here, sentences can be categorized into positive sentiment or negative sentiment based on the model. This stage is mainly divided into two parts: feature extraction and sentiment categorization. Feature extraction is performed by one-way feature matching. Autoencoder is also used for addition of extracted features modified by compressed and restored errors. The positive and negative sentiment categorization is performed after determining the semantic features. The study compares proposed models and LSTM without Autoencoder. In addition, regarding the definition of loss function, the study proposes 2 hypotheses. One is that the weighting of all words equals that of the final sentiment targets; therefore, errors in $Loss_{AE}$ are directly aggregated. Another is that the weighting of all sentences equals that of the final sentiment targets, while the weighting of each word in every sentence are the same. The study slightly modifies the original formula into: $\frac{1}{|S|}\sum_{w=0}^{|S|}\left(-\sum_{i=1}^{|BOW|}(real_i^w \log(predict_i^w))\right)$. The purpose is to simply examine whether the modifications made according to the errors arising from newly added words would positively or negatively affect the targets for categorization.

4.2.4 Model Evaluation. Owing to the fact that the balanced data set used merely shows whether the returned errors would positively or negatively affect the categorization into positive sentiment or negative sentiment, the study adopts overall accuracy for examination. Loss values come from each batch of data, but for clearer presentation they are shown in the form of average error value of each example.

5 EXPERIMENT RESULTS

5.1 Data Description

The original data [20] distribution of the 20000-plus sets of open lexical data used in this study is shown below. After examining the data, the study deleted the repeated and invalid sentences, leaving only 16680 sets of data, among which those categorized as positive sentiment for 8680 and negative sentiment for 8000. The average length of sentence is 54.79 words. Data sequence is also randomized. Data for training purposes is total of 14440. The remaining becomes data for testing is 2240. The ratio of training and testing is approximately 9:1.

5.2 Experiment Results and Analysis

In the study, the epoch is set at 10. To shorten the time required for the experiment, batch sizes are uniformly set at 32, to which data at the beginning are distributed. It is worth noting that if the batch size is too large, a few features may be distributed to the same batch and the training efficiency will be reduced; too small, calculation time may increase, and the algorithm may become more sensitive to few features. These types of neural model include LSTM. The input dimension of the parameter of LSTM in this study is set at 50 to correspond with the dimension of word embeddings. Only one hidden layer is on the inner part. The number of node is 256. Dropout size is set at 0.3. The adopted optimizer is Adam, primarily because traditional gradient descent methods may result in local optimum; therefore the more advanced algorithm Adam is used. Regarding loss function, the study selected the Cross Entropy Loss method mainly because it provides better calculation for errors of discrete categories. The effect of batch learning is better when the learning rate is set at 0.001. The learning rate cannot be converged when it is too high.

The table below shows the experiment results. It is worth noting that the study uses random seeds to anchor random functions; therefore, each epoch is provided with the same training data to perform back-propagation to modify neural work network models. The same data is used for model testing. Using these data equally would not result in back-propagation or modification of parameters. The results are shown in Table 1. Among them, average loss is the aggregation of all batch loss values divided by the number of samples. It can be inferred from the table that after the 10th epoch, the overall accuracy of each experiment method converges above 85%, with the average error rate decreasing. Epoch seems to have yet achieved its optimal convergence. But apparently, although the correction performed for errors arising from word compression affect the speed of convergence, it is conducive to the categorization. After the 10th epoch, the error rate of LSTM with AE decreases significantly. Even if the weighting of each word equals that of both 'positive' and 'negative' categories, errors can be largely decreased without interfering the learning. After being weighted, LSTM with AE seems to be no different than LSTM. This is probably because of the insufficient reduction of weighting of each word, and the word weighting of the whole sentence equaling that of the target weighting of the sentiment categories, causing the neural model to benefit more in the categorization of sentiments. Therefore, it is also important to set up proper weightings for each word, which may be able to strike a balance between training efficiency and concept interpretation.

Table 1: Each Epoch of Categorization Models

Epochs	LSTM with AE		LSTM with AE (weight loss)		LSTM	
-	acc	avg loss	acc	avg loss	acc	avg loss
1	51.79	13.23	64.78	2.19	53.44	2.13
2	51.79	11.64	70.09	2.00	71.74	2.03
3	51.79	11.04	78.30	1.53	84.46	1.53
4	52.99	9.81	84.46	1.15	84.91	1.10
5	67.14	7.15	83.62	0.96	85.27	0.89
6	81.79	4.91	84.42	0.81	85.31	0.74
7	80.63	3.46	86.43	0.69	83.93	0.64
8	81.78	2.67	85.71	0.59	86.29	0.55
9	86.16	2.19	87.54	0.52	87.14	0.48
10	86.96	1.82	87.68	0.46	87.28	0.41

Unit: percentage (%)

This study used the above models that underwent 10 times of epoch to examine their output in testing. The following is one of the batches that includes 32 samples. The accuracy of this batch is 75%. Among them, one sample A was mistakenly predicted as negative when in fact it should be positive. Another sample B was correctly predicted as positive, as it should be. After being compressed by AE, sample A was distorted. This is normal. Because the test data wasn't used for training. Therefore, a combination following a small paragraph (or words) should appear frequently in data training. From sample A, it is known that distorted words do not affect human categorization of sentiments. But for machine, they slightly do. However, judging from the fact that a sentence contains 100 words, they do not have significant influence over machine categorization, and is not likely to be labeled as both 'positive' and 'negative'. Hence, the study infers that this is because the AE has not picked up the weighting of stressed tones. Although sample B was correctly predicted, its distortion rate is higher than that of sample A. The study reckons that each word has different influence over the weighting of targets. Therefore, decompression without distortion isn't required. As long as important word features and sentence structures (e.g. stressed tone) can be compressed, positive and negative sentimental categorization can be performed based on abstract semantic concepts.

6 CONCLUSIONS

After conducting the experiment, the study discovers that adding additional features for correction of errors does not interfere with the learning. Also, not all words are needed to be restored without distortion after applying the AE method. What matters most is that important combinations of word features (meanings) can be learned. The study merely used 10 epochs to perform experiments, but the results are worth delving deeper into, especially the parts of interpretative capabilities. In the future development, to prove that the semantic compression of autoencoder is related to the training of generalized features, the study may further adopt attention-based models for visualized interpretations. In addition, to speed up the improvement in accuracy of these types of models, the study will consider using external pre-trained word vectors for initialization.

ACKNOWLEDGMENTS

The authors wish to thank the Ministry of Science and Technology of the Republic of China for financially supporting this research under Contract Grants No. MOST106-2634-F-025-001, MOST 106-2511-S-025-003-MY3, MOST105-2410-H-025-015-MY2, MOST105-2511-S-005-001-MY3, and MOST104-2511-S-005-003.

REFERENCES

[1] Q. Gan, B. H. Ferns, Y. Yu, & L. Jin. 2017. A text mining and multidimensional sentiment analysis of online restaurant reviews. *Journal of Quality Assurance in Hospitality & Tourism*, 18, 465-492.
[2] S. M. Patel, V. K. Dabhi, & H. B. Prajapati. 2017. Extractive Based Automatic Text Summarization. *JCP*, 12, 550-563.
[3] Q. Le, & T. Mikolov. 2014. Distributed representations of sentences and documents. *In Proceedings of the 31st International Conference on Machine Learning*, 1188-1196.
[4] Y. Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
[5] C. Dos Santos, & M. Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. *In COLING*, 69-78.
[6] D. Li, & J. Qian. 2016. Text sentiment analysis based on long short-term memory. *In Computer Communication and the Internet (ICCCI)*, 2016 IEEE International Conference on, 471-475.
[7] S. Zhang, Z. Wei, Y. Wang, & T. Liao. 2018. Sentiment analysis of Chinese micro-blog text based on extended sentiment dictionary. *Future Generation Computer Systems*, 81, 395-403.
[8] A. Ortigosa, J. M. Martín, & R. M. Carro. 2014. Sentiment analysis in Facebook and its application to e-learning. *Computers in Human Behavior*, 31, 527-541.
[9] Q. Li, Z. Jin, C. Wang, & D. D. Zeng. 2016. Mining opinion summarizations using convolutional neural networks in Chinese microblogging systems. *Knowledge-Based Systems*, 107, 289-300.
[10] T. Mikolov, K. Chen, G. Corrado, & J. Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[11] J. Cheng, P. Li, Z. Ding, S. Zhang, & H. Wang. 2016. Sentiment Classification of Chinese Microblogging Texts with Global RNN. *In Data Science in Cyberspace (DSC), IEEE International Conference on*, 653-657.
[12] G. Zhou, Y. Zhou, T. He, & W. Wu. 2016. Learning semantic representation with neural networks for community question answering retrieval. *Knowledge-Based Systems*, 93, 75-83.
[13] D. Chen, A. Fisch, J. Weston, & A. Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. arXiv preprint arXiv:1704.00051.
[14] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, & C. A. Iglesias. 2017. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77, 236-246.
[15] L. Gui, Y. Zhou, R. Xu, Y. He, & Q. Lu. 2017. Learning representations from heterogeneous network for sentiment classification of product reviews. *Knowledge-Based Systems*, 124, 34-45.
[16] Y. J. Park, H. S. Kim, H. Lee, D. Kim, S. B. Kim, & P. Kang. 2017. A deep learning-based sports player evaluation model based on game statistics and news articles. *Knowledge-Based Systems*, 138, 15-26.
[17] S. Rönnqvist, & P. Sarlin. 2017. Bank distress in the news: Describing events through deep learning. *Neurocomputing*, 264, 57-70
[18] M. Kraus, & S. Feuerriegel. 2017. Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, 104, 38-48.
[19] J. Chung, C. Gulcehre, K. Cho, & Y. Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.
[20] DATATANG. http://www.datatang.com/index.html (2017)