Advances in Intelligent Systems Research, volume 134
2nd International Conference on Control, Automation, and Artificial Intelligence (CAAI 2017)

Text Classification Model Based on Document Matrix Convolutional Neural Networks

Xuemiao Zhang[1,*], Weizhong Qian[1], Zhaoyi Liu[2] and Xin He[1]
[1]School of information and software engineering University of Electronic Science and Technology of China Chengdu 6110054, P. R. China
[2]Chengdu Research Institute of UESTC University of Electronic Science and Technology of China Chengdu, Sichuan 610054, P. R. China
[*]Corresponding author

***Abstract*—This paper proposes a new text classification method named document matrix convolutional neural networks (DM-CNN). Unlike the original ways of converting texts into 1-dimensional vectors and processing each word as a pixel, this model is based on the n-dimensional word embeddings obtained in advance, taking each entry of the word embedding as a pixel, and converting the text into a 2-dimensional document matrix (DM) according to the method proposed in this paper, maintaining the order of the words in the original, so that the DM-CNN model can process the text as if it were an image. The model greatly maintains the information content of the more abstract words, as well as the structural information at all levels in the original. In the experiment, the DM-CNN text classification model is compared to the classifiers based on classical machine learning algorithms, and the results show the feasibility and superiority of DM-CNN.**

Keywords-natural language processing; convolutional neural networks; document matrix; text classification; word embedding

I. INTRODUCTION

In the field of natural language processing, text classification is a very important task, it is a behavior of dividing a set of input documents into two or more classes. After that each document belongs to one or more classes. In the information age, with the popularity and the rapid development of the Internet, information flow is undergoing tremendous growth. Finding the points of interest from massive texts becomes an attractive challenge, which facilitates the birth and development of automatic text categorization. Text classification technology is very close to us, for example, spam identification uses the technology, and automatically classifying large text collections into the corresponding topic categories, managing knowledge and Internet search engine are inseparable from the text classification technology.

Early natural language processing techniques were based on a number of linguistic rules and templates that linguists developed. It is true that these rule-based approaches had very noticeable effects at the time [1, 2]. But the inadequacies of these approaches are self-evident, and they are overly dependent on the subjective judgment of people (or domain experts) and can not be adapted to new areas. Later, the researchers applied statistical theory to natural language processing tasks, invented a series of effective statistics-based methods [3, 4, 5, 6]. Further, scholars have thought of integrating template-based methods and statistic-based methods together, so that the two can complement each other [7, 8]. It is undeniable that these technologies have achieved remarkable results. However, the methods based on statistical theory also have shortcomings. Often the later work of NLP tasks rely on the previous steps, such as word segmentation, Part-Of-Speech (POS) tagging, etc., so the final performance of the entire model, need to take into account the error passed from the previous.

The rise of the deep learning [9] provides a very different solution ideas to NLP tasks [10]. Due to the complex and powerful multi-layer network structure, deep learning has excellent fitting ability. Different from the traditional machine learning algorithm, the deep learning can do end-to-end learning directly [11]. There is no need to manually define rules and features, it can automatically learns more essential features that are usually more profound and more comprehensive, which makes it outstanding in many tasks [12, 13]. And it have achieved remarkable results in information retrieval, relationship extraction [14], QA systems and other tasks of NLP. At present, the deep learning framework commonly used in the field of natural language processing mainly includes convolutional neural networks [15, 16], recurrent neural networks (RNN) [17] and long short-term memory (LSTM) [18]. This paper mainly studies the application of convolution neural network in text processing.

Mentioning text classification task, we can naturally think of document emotional classification tasks [19]. So far, it seems difficult to find a solution which can handle document emotional classification task very well. There are always multiple paragraphs with multiple sentences in a document. Different paragraphs describe different objects, and express different emotional colors often. Even in the same paragraph, the first few sentences are in the expression of positive feelings, but the last sentence came a strong turn, expressing a negative feeling. What is more important is that the content behind the turning point is the core of the paragraph even of the entire document.

But on the common text classification task which does not care about emotional colors, DM-CNN models can usually perform very well. Because after appropriate training, it can easily find common feature sets of those documents classified

as the same category. For example, in a medical text, it is possible that each paragraph does introduce the different aspects of the drug separately, even describe the different viewpoints of the drug. However, from the perspective of text classification, no matter how many drugs the document involves, and no matter what viewpoints each paragraph expresses, the topic of the document is the same. So far, we have got an intuitive understanding of the feasibility of processing text classification tasks with DM-CNN model already.

Next, this section take a short essay on traditional Chinese medicine as an example to explain the contrast between DM-CNN model and traditional machine learning algorithms such as Bayesian classifier.

Table 1 shows a simple example on traditional Chinese medicine. When using the Bayesian classifier, it is possible that the text is incorrectly classified as a "plant" category instead of correctly classified as a "medicine" category, for the high frequency of plant-related words that appear in the text. However, the DM-CNN model based on word embeddings [20, 22] solves the problem from a different angle. It can detect that the main entities have higher similarity for word sense with word "medicine" than "plant". Similarity for word sense is just a shallow feature in the feature set that deep learning can find automatically. DM-CNN model can also effectively find deep features such as the association of entries from different word embeddings, and usually these features are can not be defined artificially. Therefore, the DM-CNN model has a greater probability of making the correct classification results.

TABLE I. SIMPLE EXAMPLE ON TRADITIONAL CHINESE MEDICINE

Liu- wei-di- huang bolus can be effective when the Tongue is dry, or the throat is a bit painful. The following is its main ingredient and its profile. ·Cornus, also known as taro meat, is the plant dogwood dry and mature pulp; ·Yam, its tubers are oblong, usually grown vertically and its cross section is white when it is fresh; ·Alisma, is the sticks of the diarrhea species of almondia; ·Durian is a dry root bark of Ranunculaceae plants Peony.

In view of that, this paper will explore a deep learning model based on classical CNN structure whose input is 2-dimension document matrices to improve the accuracy of text classification task.

The rest of the paper is organized as follows. In the section 2, we will introduce the CNN model proposed in this paper. We report the experiment and the results in section 3. In the section 4, we discuss some optimization problem of the model in the practical application, and conclude the paper in section 5.

II. DM-CNN FOR TEXT

The neural networks (shown in Figure 1) are good at learning the relationship between the characteristics of the data and some observed responses. DM-CNN is a special and effective multi-layer neural network model, and is outstanding in pattern recognition tasks, and do not require any particular feature extractor selection. The proposed architecture relies on several deep neural networks of alternating convolution layers and subsampling layers [21]. The features detected by the first layer in the DM-CNN can be identified and interpreted easily, and the later layers detected more abstract features.

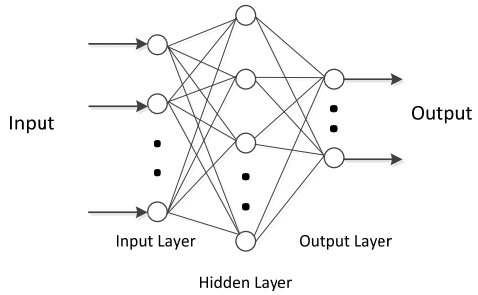

FIGURE I. GENERAL STRUCTURE OF COMMON NEURAL NETWORK

A. Basic Structure

First of all, we talk about the classic CNN architecture for image processing. The CNN structure shown in Figure 2 consists of a plurality of convolution layers, sub-sampling layers, fully-connected layers, and a softmax layer

Deep convolution neural networks in image processing, video processing, speech recognition have made breakthroughs, greatly enhance the accuracies of these tasks. Not surprisingly, more and more researchers applied it to text processing tasks and achieved excellent results. It is necessary to convert the text into a form that the model can handle firstly. Researchers usually choose the form of vector, and convert the entire text to the format the model requires in some ways. This paper obtains word embeddings trained by the word2vec tool provided by Google instead of traditional ways of obtaining word vectors, such as one-hot coding. And most of the existing CNN text processing models use 1-dimensional convolution kernels, but this paper chooses 2-dimensional convolution kernels to process the text.

B. Document Matrix

The model proposed in this paper needs to convert a text into a 2-dimensional document matrix $T_{m \times n}$, as shown in Figure 3, unlike the method of converting a text into a 1-dimensional vector with fixed dimensions, to meet the input of the model. We firstly use the word2vec tool to train each word into an n-dimensional word embedding (number n is selected by the operator). In this way, the processing tasks can be simplified as vector operations in the n-dimensional vector space. The open source tool uses the Continuous Bag-Of-Words (CBOW) and Skip-Gram models [22], and because of its superior performance, it is used to do a lot of NLP tasks, such as clustering, looking for synonyms and so on.

In order to be able to meet the input of DM-CNN model, and to ensure that the input of the document matrices are always the size of $m \times n$, we define the following rules when constructing document matrices:

a) If $wordnum < m$ *, then:*

Construct a document matrix, to fill the vacancies in the matrix with $(m - wordnum) \times n$ zeroes, and classify the same

class as the original.

b) If $wordnum > m$ *and* $wordnum\%m \neq 0$ *, then:*

Construct $\left\lfloor \frac{wordnum}{m} \right\rfloor + 1$ document matrices, and the last document matrix is made of these words embeddings in $wordlist[wordnum-m:wordnum]$, and all the document matrices will be classified as a class;

c) If $wordnum > m$ *and* $wordnum\%m == 0$ *, then:*

Construct $\frac{wordnum}{m}$ document matrices, and all the document matrices will be classified as a class;

Where *wordnum* is the number of the text's words and *wordlist* is the list of the text's words after data cleaning.

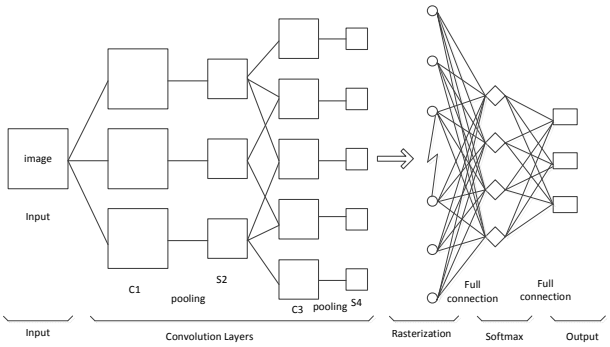

FIGURE II. THE STRUCTURE OF CLASSICAL CNN MODEL

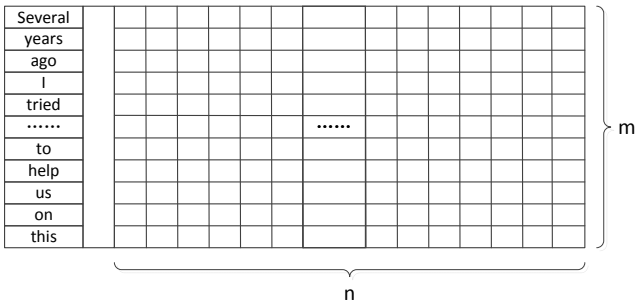

FIGURE III. A SCHEMATIC DIAGRAM OF THE DOCUMENT MATRIX MODEL

C. DM-CNN Classifier

Now we consider the application of DM-CNN model in the text. We can get the document matrices $T_{m\times n}$ of all texts according to the conversion rule of the previous section. Each document matrix $T_{m\times n}$ maintains the word order of the original. Unlike the preprocessing method that treats each word directly as a pixel, we treat each entry of each word embedding, that is, each entry of the document matrix as a pixel, so that DM-CNN model can handle text like dealing with common image, as shown in Figure 4, finally get classification result of the text.

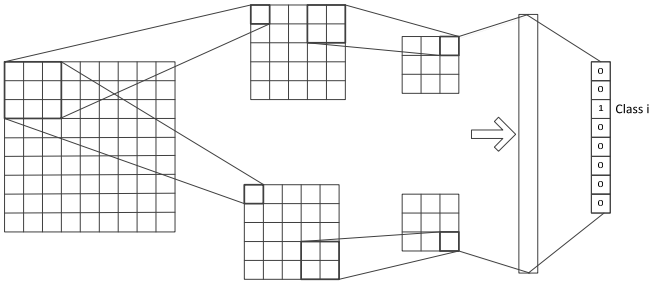

FIGURE IV. THE STRUCTURE OF DM-CNN MODEL FOR PROCESSING TEXT

a) Convolution layer

Assume that the current layer is an i-th layer, The $i-1$ layer has N feature maps as input and the size of the convolution kernel is $K_x \times K_y$ (usually square), then the output feature map M_j^i of the j-th convolution kernel of the i-th convolution layer can be calculated as follows:

$$M_j^i = \phi(\sum_{n=1}^{N} M_n^{i-1} * K_n^i + b_j^i) \tag{1}$$

Where b_j^i represents a bias that matches the corresponding convolution kernel, $\phi(x)$ represents a non-linear squashing function.

In the actual programming, we encapsulate (1) into the following easy-to-understand expression:

$$u_j^i = \sum_{j=1}^{N} conv2(M_j^{i-1}, K_j^i, 'valid') + b_j^i .$$
$$M_j^i = \phi(u_j^i)$$

Where the $conv2(\square)$ function encapsulates the convolution operation.

b) Sub-sampling layer

As most models' selections, DM-CNN also uses max-pooling. Defines a down-sampling function $down(\square)$ based on max-pooling, assuming that the number of strides for window movement is 2, the size of the pooling window is 2×2, then to a 4×4 matrix, the conversion go as the following:

$$down(\begin{bmatrix} a_1 & a_2 & b_1 & b_2 \\ a_3 & a_4 & b_3 & b_4 \\ c_1 & c_2 & d_1 & d_2 \\ c_3 & c_4 & d_3 & d_4 \end{bmatrix}) = \begin{bmatrix} \max\{a_1,...,a_4\} & \max\{b_1,...,b_4\} \\ \max\{c_1,,...,c_4\} & \max\{d_1,,...,d_4\} \end{bmatrix}.$$

Thus, the output of each Feature Map S_j^{l-1} of the layer $l-1$ after the sub-sampling operation can be calculated as follows:

$$S_j^l = \phi(\beta_j^l down(S_j^{l-1}) + b_j^i) .$$

Where β_j^l represents the multiplicative bias corresponding to the pooling operation.

c) Full connection layer and softmax regression

The final full connection layer and softmax regression play the role of classification. After the previous down-sampling operation, a series of feature maps are obtained. In order to meet the input of the multi-layer sensors, these feature maps need to be rasterized into a vector. Specifically, for the feature maps $S_1, S_2, \cdots, S_j$, assuming that the size of each feature graph is $p \times q$, the vector $\vec{o}$ obtained after rasterization is as follows:

$$\vec{o} = [t_{111}, t_{112}, \cdots, t_{11q}, t_{121}, t_{122}, \cdots, t_{12q}, \cdots, t_{1pq}, \cdots, t_{2pq}, \cdots, t_{jpq}]^T$$

Unlike the logistic regression to solve the two classification problem, this model needs to solve the multi-classification problem, that is, the class y can take k different values. Therefore, this model uses softmax regression. Specifically, we assume that the function $h_\theta(x^i)$ is as follows:

$$h_\theta(x^i) = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^i}} \begin{bmatrix} e^{\theta_1^T x^i} \\ e^{\theta_2^T x^i} \\ \cdots \\ e^{\theta_k^T x^i} \end{bmatrix}.$$

Where $\theta_1, \theta_2, \cdots, \theta_k \in R^{n+1}$ is the parameter of the model, and $\frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^i}}$ is the normalization of the probability distribution so that the sum of all the probabilities is 1.

III. EXPERIMENTS

A. The Dataset

We introduce the corpus firstly. We used the 20 newsgroups dataset. The 20 Newsgroups data set is a collection of approximately 20,000 documents that are evenly distributed (almost) in 20 different newsgroups.

The data set is divided into 20 different newsgroups, each of which corresponds to a different theme. Some newsgroups are closely related to each other (eg. comp. sys. ibm. pc. hardware / comp. sys. mac. hardware), while other newsgroups are highly irrelevant (eg. misc. forsale/ soc. religion. christian).

TABLE II. THE INFORMATION OF THE FIVE CLASSIFIED DATASETS

category	number of files	total size/MB
comp. graphics	1000	2.07
comp.sys. ibm. pc. hardware	1000	1.58
comp.os.ms- windows. misc	1000	2.75
comp. windows. x	1000	2.30
comp. sys. mac. hardware	1000	1.50

Considering some reasons, such as the similarity between the various categories, the performance of the experimental machine, we conducted experiments based on these data sets of the following five categories: "comp. sys. ibm. pc. hardware", "comp. graphics", "comp.os.ms-windows. misc", "comp. windows. x", "comp. sys. mac. hardware". And the information of the five classified datasets is shown in the Table 2.

B. DM-CNN Model Training

The experimental steps of The DM-CNN model is shown in Figure 5, we logically divided into three parts to describe.

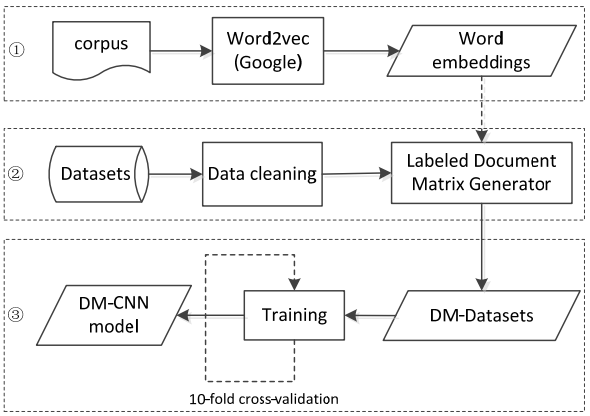

FIGURE V. EXPERIMENTAL STEPS FOR THE DM-CNN MODEL

1) Word embeddings training

We used the word2vec tool provided by Google to train the corpus (text8) to get the word embeddings of the words in the corpus, and specified the dimension of the word embedding to 200 dimensions. Text8 derived from enwiki8 is the default corpus used by word2vec tool, and nearly 100MB. The final output file, such as "vectors. bin", contains almost all the needed words and word embeddings.

2) Document matrix constructing

It is necessary to cleaning the datasets before using them. We define a cleaning rule which filters out all the words that do not appear in the "vectors. bin" file after the text segmentation task, and maintain the relative word order of the original. We convert each text according to the construction rules into one or more document matrices. We specified that the number of word embeddings in the document matrix is 200, so that the document matrix is a square of 200×200. And assigns a 5-dimensional classification label vector $\vec{l} = [a_i], a_i = 0,1 \; and \; i = 0,1,2,3,4$ for each document matrix. Finally we got the DM-datasets including all the labeled document matrices derived from the 5000 texts.

3) Model training

The experiment used the deep learning framework - TensorFlow which is Google's open source for machine learning and deep learning framework to implement DM-CNN model, so we set the parameters of the model according to the TensorFlow specification.

In practical training process, usually a large part of the time is used to tune the hyper-parameters, which largely determines the model performance, the convergence rate and so on. And

after some attempts, we finally chose the structure including a convolution layer, a max-pooling layer, and selected ReLU [26] as an activation function, and use the softmax function to get the confidence of each category. Specifically, the size of the convolution kernel was set to 5×5, the number of strides of the convolution kernel sliding both were 4 in the length and width direction, and the number of convolution kernel was set to 512; in the sub-sampling layer, set the window size to 5, and the window sliding strides were both set to 5 in the length and width direction. Naturally, the number of output classification is 5.

And DM-CNN model used the dropout method proposed by paper [28] to reduce overfitting. Dropout is a powerful regular method. Because the part of the weights have not been updated to reduce the overfitting, and the network models used for each training are different from each other, so the final model is equivalent to the results which comes from the one mixed a number of models. Finally the generalization of the hybrid models are often relatively strong. We use dropout for the full connection layer, since the full connection layers are easy to over-fit, and set the value of dropout rate to 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially many dropout networks [27].

We used 10-fold cross-validation to train the hyper-parameters of DM-CNN, and use the average of 10 accuracies as the final accuracy of DM-CNN. After enough iterations, experiment obtains a high average accuracy.

C. Baseline Methods

In order to compare with the above model, we have done this experiment on the same experimental corpus, selecting three mainstream text classification models, including Bayesian classifier, support vector method (SVM), K-Nearest Neighbor (KNN). In this experiment, we chose to use sklearn module to implement these three algorithms.

Sklearn is a commonly used python third party module for machine learning. The sklearn module encapsulates some of the commonly used machine learning algorithms. We just use the SVM classifier, the KNN classifier, the Bayesian classifier from sklearn modules in it. We also use the 10-fold cross-validation for these model, and select the average accuracy as the final accuracy

D. Classification Results

After the experiment, we got four classifiers. We tested each classifier on the corresponding test sets, and finally got the average accuracies of all classifiers, as shown in Table 3.

From Table 3, we can observe that the text classification model based on DM-CNN has better performance than the text classifiers based on the traditional machine learning algorithms.

The text classifiers based on the traditional machine learning algorithms have indeed achieved a good classification performance (about 0.8). However, in the experiment, we find that the improvement of model performance encountered a bottleneck, although many methods were used. But to the DM-CNN, we find that we can always get a superior model as long as we are patient enough to take a long time to tune the parameters (if there is not a good machine). We realize that this accuracy is still not the bottleneck of the classifier, and if we do some appropriate optimization, the performance of the classifier can be improved.

TABLE III. THE CLASSIFICATION ACCURACY OF EACH CLASSIFIER

Classification model	**Average Accuracy**
K-Nearest Neighbor	0.781259
Support Vector Machine	0.777765
Naïve Bayes	0.819091
DM-CNN	0.902512

IV. DISCUSSION

In this section, we focus on the improvement of DM-CNN model performance.

A. Depth Of Networks

We notice that the CNN model applied to text processing is relatively shallow compared to the CNN model applied to image processing. The CNN models used to process images usually have multiple layers and sub-sampling layers [21, 23, 24], and most of the CNN models that deal with the text can use very few layers and sub-sampling layers to achieve good performance [25], as long as there are required word embeddings. This is probably because the pixel of a image is a low-level representation, and the word embedding itself is a more advanced abstract representation, the amount of information carried is much larger than the pixel. But this is not to say that the CNN model for processing text can not be used with multiple layers and sub-sampling layers. On the contrary, this often leads to better performance, but usually we will make sacrifices in other ways, such as convergence speed. And if the data set is very small, it is likely that the deeper networks is more difficult to convergence for the deeper level means more parameters.

B. Selecting Activation Function

This model uses the ReLU activation function. ReLU makes the network itself introduce sparseness. The data provided by the paper [26] show that the ReLU activation function leads other activation functions without pre-training. ReLU still has room for improvement after pre-training, of course. From this perspective, ReLU narrows the gap between unsupervised learning and supervising learning and gains faster training speed, and greatly shortened the learning cycle [27]. Comprehensive rate and efficiency, with ReLU as the activation function is a good choice in deep learning tasks.

V. CONCLUSION AND FUTURE

This paper shows that, different from the traditional 1-dimensional input CNN model, by constructing 2-dimensional document matrices, DM-CNN can effectively complete the text classification task, and in the training process, the model with a layer of convolution also performs well, and better hyper-

parameters have a great influence on the performance and convergence speed of the model.

Future work will involve studying the relationship between the value of the word embedding dimension and the performance of the model, and how the model will behave as the number of categories increases further.

ACKNOWLEDGMENT

X. Zhang, W. Qian, Z. Liu and X. He are supported in part by National Natural Science Foundation of China 61502082.

REFERENCES

[1] J. Zhang, and N. El-Gohary, Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking, Reston: Journal of Computing in Civil Engineering, vol. 30. Mar 2016.

[2] N. Kang, B. Singh, and Z. Afzal, Using Rule-based Natural Language Processing to Improve Disease Normalization in Biomedical Text, Elsevier: Journal of the American Medical Informatics Association, vol. 20. October 2012, pp. 876-881.

[3] W. Kenlin, Z. Yifei, and W. Cheng, Adaptive Normalized Weighted KNN Text Classification Based on PSO, Dnipropetrovsk: Scientific Bulletin of National Mining University, January 2016.

[4] A. Guo, and T. Yang, Based on Rough Sets and the Associated Analysis of KNN Text Classification Research, Tokyo: International Symposium on Distributed Computing & Applications for Business Engineering & Science, 2015, pp. 485-488.

[5] M. Haddoud, A. Mokhtari, T. Lecroq, and S. Abdeddaïm, Combining Supervised Term-weighting Metrics for SVM Text Classification with Extended Term Representation, England: Knowledge Information Systems, vol. 49. 2016, pp. 1-23.

[6] L. Jiang, C. Li, S. Wang, and L. Zhang, Deep Feature Weighting for Naive Bayes and Its Application to Text Classification, England: Engineering Applications of Artificial Intelligence, vol. 52. 2016, pp. 26-39.

[7] R. Marta, and R. Banchs, Automatic Normalization of Short Texts by Combining Statistical and Rule-based Techniques, Netherlands: Language Resources and Evaluation, vol. 47. 2013, pp. 179-193.

[8] T. Ehara, Machine Translation System for Patent Documents Combining Rule-based Translation and Statistical Post-editing Applied to the PatentMT Task, Japan: Proceedings of Ntcir, June 2016, pp. 335-338.

[9] Y. Lecun, and Y. Bengio, and G. Hinton, Deep Learning, England: Nature, vol. 521. 2015, pp. 436-444.

[10] R. Socher, Y. Bengio, and C. Manning, Deep Learning for NLP, Tutorial Abstracts of Acl, 2012, pp. 5-5

[11] T. Wang, D. Wu, A. Coates, and A. Ng, End-to-end Text Recognition with Convolutional Neural Networks, Japan: International Conference on Pattern Recognition, Nov 2012, pp.3304-3308.

[12] Z. Xu, Y. Yang, and A. Hauptmann, A Discriminative CNN Video Representation for Event Detection, Boston: Computer Vision & Pattern Recognition, June 2015, pp.1798-1807.

[13] H. Su, C. Qi, Y. Li, and L. Guibas, Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views, Chile: IEEE International Conference on Computer Vision, Dec 2015, pp.2686-2694.

[14] T. Nguyen, and R. Grishman, Relation Extraction: Perspective from Convolutional Neural Networks, Denver: Workshop on Vector Space Modeling for Natural Language Processing, 2015, pp.39-78

[15] F. Yang, L. Gan, A. Li, D. Huang, and X Chou, Combining Deep Learning with Information Retrieval for Question Answering, Germany: Springer International Publishing, Dec 2016, pp.917-925.

[16] M. Iyyer, J. Boyd-Graber, L. Claudino, and R. Socher, A Neural Network for Factoid Question Answering over Paragraphs, Doha: Conference on Empirical Methods in Natural Language Processing, Oct 2014, pp. 633-644.

[17] T. Mikolov, M. Karafiát, L. Burget, J Cernocký, and S. Khudanpur, Recurrent Neural Network Based Language Model, Japan: Conference of the International Speech Communication Association, Sep 2010, pp. 1045-1048.

[18] M. Sundermeyer, R. Schlüter, and H. Ney, LSTM Neural Networks for Language Modeling, Portland: Interspeech, vol. 31. 2012, pp. 601-608.

[19] V. Basile, A. Bolioli, M. Nissim, V. Patti, and P. Rosso, Overview of the Evalita 2016 SENTIment POLarity Classification Task, International Workshop Evalita, 2016

[20] D. Tang, F. Wei, N. Yang, M. Zhou, and T. Liu, Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification, Baltimore: Meeting of the Association for Computational Linguistics, June 2014, pp. 1555-1565.

[21] P. Buyssens, A. Elmoataz, and O. Lézoray, Multiscale Convolutional Neural Networks for Vision–Based Classification of Cells, Germany: Springer Berlin Heidelberg, 2012, pp. 342-352

[22] Y. Liu, Z. Liu, T. Chua, and M. Sun, Topical Word Embeddings, Texas: Twenty-ninth Aaai Conference on Artificial Intelligence, Jan 2015, pp. 2418-2424.

[23] J. Bontar, and Y. Lecun, Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches, USA: Journal of Machine Learning Research, vol. 17. 2015, pp. 2287-2318.

[24] P. Seo, and B. Han, Image Question Answering Using Convolutional Neural Network with Dynamic Parameter Prediction, Poland: Computer Science, 2015, pp. 30-38.

[25] Y. Kim, Convolutional Neural Networks for Sentence Classification, Doha: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Oct 2014, pp. 1746–1751.

[26] X. Glorot, A. Bordes, and Y. Bengio, Deep Sparse Rectifier Neural Networks, USA: Journal of Machine Learning Research, 2011, pp. 315-323.

[27] A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Nevada: International Conference on Neural Information Processing Systems, vol.25. Dec 2012, pp. 1097-1105.

[28] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Improving Neural Networks by Preventing Co-adaptation of Feature Detectors, USA: Compute Science, vol. 3. 2012, pp. 212-223.