Text classification with Document Embeddings

Chaochao Huang, Xipeng Qiu, and Xuanjing Huang

[1] Shanghai Key Laboratory of Intelligent Information Processing
[2] School of Computer Science, Fudan University, Shanghai, China
`{chaochaohuang12, xpqiu,xjhuang}@fudan.edu.cn`

Abstract. Distributed representations have gained a lot of interests in natural language processing community. In this paper, we propose a method to learn document embedding with neural network architecture for text classification task. In our architecture, each document can be represented as a fine-grained representation of different meanings so that the classification can be done more accurately. The results of our experiments show that our method achieve better performances on two popular datasets.

1 Introduction

Text classification is a crucial and well-proven method for organizing the collection of large scale documents, which has been widely used in a lot of tasks in natural language processing or information retrieval, for instance, spam filtering[1, 3, 5],email routing[11] and sentiment analysis[21].

Currently, most of the state-of-the-art text classification methods are based on machine learning algorithms[25], such as decision tree[7, 22], Naive Bayes[15], support vector machine (SVM)[10], and so on.

Since the input of machine learning algorithms must be a fixed-length feature vector, the documents are often represented with vector space model (VSM) [24] (also called bag-of-words(BOW)). Each dimension corresponds to one word, and the dimensionality of the vector is the size of the vocabulary. However, this kind of representation has two disadvantages: (1) the represented vector is often high dimensional and very sparse, which brings a challenge for traditional machine learning algorithm; (2) it ignores the semantics of the words.

Although lots of feature selection methods [27] are proposed, these features are still sparse and not optimum. A great correlation and redundant information exist among these features.

Recently, word embeddings are becoming more and more popular and have shown excellent performance in various natural language processing tasks [17, 2, 6, 19, 9]. Each word is represented by a dense vector and words with similar meanings will be close to each other in the vector space. Distributed representations, which are originally designed for words[23], have also been used to represent phrases sentences[18, 16].

Although word embeddings have been applied to text classification [14, 13], there are two problems when utilizing the word embeddings on text classification.

1. It is still not clear to combine the word embeddings to represent the documents. The documents often have words with various lengths, we cannot use the word embeddings directly to train a classifier. A simple approach is using a weighted average of all the words in the document.
2. Traditional word embeddings are learned by probabilistic language model in a separate step, which are not optimal for text classification task.

In this paper, we propose a method to learn word embeddings directly in text classification task. A document is also represented by a vector, called *document embedding*. Document embeddings can be calculated by the vector representations of its containing words. Our method can handle all words contained in a document and need not reduce the dimensionality of input.

In our method, each document is represented as the combination of the word embeddings of its containing words. Since the word embedding can represent different meanings of each word, the document embedding can also represent different meanings of each document and documents close to each other in the vector space may be of the same topic. The experimental results also proves this hypothesis.

The remaining parts of the paper is arranged as follows: Section 2 surveys related works on text classification. Section 3 describes our architecture and learning algorithms. The experiments will be detailed described in section 4 and finally there will be a conclusion.

2 Related Works

Recently, deep neural networks are so popular and are widely used in lots of domains for the purpose of classification, including text classification. For instance, Restricted Boltzman Machines(RBMs) have been utilized to do the document and image classification[12].

Liu [14] used deep belief network (DBN) to obtain the high level abstraction of input data to model the semantic correlation among words of documents for text classification. However, since he used Restricted Boltzmann Machines (RBM) [8] to obtain the high level abstraction of input data, the dimensionality of the input data need be reduced in advance. Thus, a lot of information may be lost.

Le and Mikolov [13] proposed *Paragraph Vector*, an unsupervised framework that learns continuous distributed vector representations for pieces of texts. The texts can be of variable-length, ranging from sentences to documents. Although paragraph vector can be applied to variable-length pieces of document, it is learned separately before they are used in text classification.

Socher et al. [26] used a more sophisticated approach to combine the word vectors in an order given by a parse tree of a sentence, using matrix-vector operations. However, it has been shown to work for only sentences because it relies on parsing.

3 Neural Network Architecture for Text Classification

According to our hypothesis, a document can be represented as the accumulation of all words it contains. Each word has an exact and unique meaning, which is represented by the different decimals in each element of the word embedding. Similarly, the document's meaning is also represented by each element of its *document embedding.*

3.1 Document Embeddings

The architecture we propose is described in figure 1.

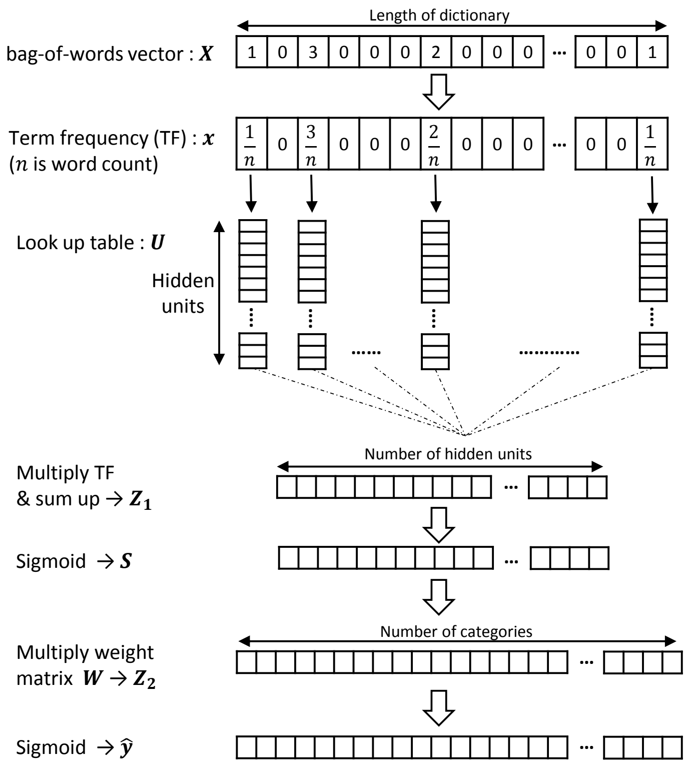

Fig. 1: Neural Network Architecture

A document is a bag-of-words representation $X = \{X_1, X_2, ..., X_n\}$, in which X_i means the i^{th} word shows up X_i times in the document. We first transform X into x by:

$$x_i = X_i / \sum_{X_i \in X} X_i \tag{1}$$

so that the following calculations will not be affected by the length of the document.
U is a look up table of the dictionary. Each column in U is a word embedding. As the description above, a document S is calculated as:

$$\begin{aligned} Z_1 &= Ux \quad (2) \\ S &= f(Z_1) \quad (3) \end{aligned}$$

where Z_1 is a document embedding.

Supposing that $\hat{y}$ is the output from the network, it is computed as follows:

$$\begin{aligned} Z_2 &= WS \quad (4) \\ \hat{y} &= f(Z_2) \quad (5) \end{aligned}$$

where W is the a weight matrix and $f(z)$ is sigmoid activation function:

$$f(z) = \frac{1}{1+e^{-z}} \quad (6)$$

3.2 Training phases

The training problem is to determine the parameters of the network $\theta = (U, W)$ from the training data.

The training is performed using Stochastic Gradient Descent (SGD). We go through all the training data iteratively, and update the weight matrices U and W online (after processing every document).
At each training phase, θ is updated with the standard backpropagation algorithm and the gradient of the error vector is computed using a cross entropy criterion:

$$error(x) = y(x) - \hat{y}(x) \quad (7)$$

Where $y(x)$ is the gold classification vector, using n-hot encoding and the cross entropy[20] we use here is:

$$E_m = -\sum_{i=1}^{m} [y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)] \quad (8)$$

where m is the category count.

To minimize E_m, we calculate the derivative of W and U. The process is as follows:

$$\begin{aligned} \frac{\partial E_m}{\partial Z_{2j}} &= \frac{\partial E_m}{\partial \hat{y}_j} \frac{\partial \hat{y}_i}{\partial Z_{2j}} \\ &= (-\frac{y_j}{\hat{y}_j} + \frac{1-y_j}{1-\hat{y}_j})\hat{y}_j(1-\hat{y}_j) \\ &= \hat{y}_j - y_j \quad (9) \end{aligned}$$

We assume

$$e_{0j} = \frac{\partial E_m}{\partial Z_{2j}} \tag{10}$$

$$e_{1j} = \frac{\partial E_m}{\partial Z_{1j}} \tag{11}$$

Now that through back propagation we get

$$\begin{aligned}\frac{\partial E_m}{\partial Z_{1i}} &= \sum_{k=1}^{m} \frac{\partial E_m}{\partial Z_{2k}} \frac{\partial Z_{2k}}{\partial S_i} \frac{\partial S_i}{\partial Z_{1i}} \\ &= \sum_{k=1}^{m} e_{0k} W_{ki} S_i (1 - S_i)\end{aligned} \tag{12}$$

Then using chain derivation rule we get

$$\begin{aligned}\frac{\partial E_m}{\partial W_{ij}} &= e_{0i} \frac{\partial Z_{2i}}{\partial W_{ij}} \\ &= (\hat{y}_i - y_i) S_j\end{aligned} \tag{13}$$

$$\begin{aligned}\frac{\partial E_m}{\partial U_{ij}} &= e_{1i} \frac{\partial Z_{1i}}{\partial U_{ij}} \\ &= [\sum_{k=1}^{m} e_{0k} W_{ki} S_i (1 - S_i)] x_j \\ &= [\sum_{k=1}^{m} (\hat{y}_k - y_k) W_{ki} S_i (1 - S_i)] x_j\end{aligned} \tag{14}$$

Now we can easily find that the weight matrix W between the document vector $d(x)$ and the output layer $y^*(x)$ can be updated as:

$$W^* = W + \alpha \frac{\partial E_m}{\partial W} \tag{15}$$

and the dictionary look up table matrix U can be updated following:

$$U^* = U + \alpha \frac{\partial E_m}{\partial U} \tag{16}$$

where α is the learning rate.

4 Experiments

We evaluate the performance our method by comparing the BOW representation.

4.1 Datasets

We setup our experiments on two datasets: LSHTC and Sogou datasets.

LSHTC This dataset is from the 4th Large Scale Hierarchical Text Classification (LSHTC) Challenge[3]. The challenge is based on a large dataset created

[3] http://www.kaggle.com/c/lshtc

from Wikipedia and the document set is multi-class, multi-label and hierarchical though we do not utilize any hierarchical information. Documents here is high dimensional(roughly 1,620,000) and very sparse on each category. The format of each document is like:

12370,306783 1:1 45:3 1982:1 ... 32600:1

which means the document belongs to category 12370 and 30678 at the same time, and the remaining part is the sparse representation in bag-of-words.

In consideration of time efficiency, we choose 100 categories from the dataset that most frequently appear. We only choose the documents which only contain one category in the top 100 categories.

For each category, we choose less than 150 samples for training and less than 10 samples for testing. Finally we get 13113 training documents and 800 testing documents, every document is single-labeled. The categories we choose are shown in table 1.

Table 1: LSHTC Chosen Categories

24177, 285613, 98808, 264962, 167593, 242532, 52954, 300558,
444502, 78249, 237290, 220514, 10721, 337728, 174545, 73518, 24016,
327590, 154064, 374771, 366417, 87241, 73092, 115838, 334220,
169902, 59758, 347803, 364106, 178462, 287120, 14843, 260304,
73462, 23611, 322170, 174425, 167844, 29462, 158599, 299629, 34161,
390974, 228232, 150636, 341276, 36224, 289559, 418360, 323972,
352578, 284433, 383600, 300073, 231746, 60639, 251484, 2830,
183203, 234578, 283823, 161537, 286264, 304661, 93718, 348488,
139391, 397350, 244711, 186125, 419276, 1508, 398319, 428719,
290537, 403132, 395447, 351111, 324660, 13252, 131804, 430081,
24052, 244616, 86836, 393137, 374859, 111772, 206933, 109127,
96443, 228238, 269785, 2903, 272741, 213350, 225356, 174595,
414726, 429208

Sogou Dataset This dataset[4] is all of website news in Chinese and the corpus mainly come from Sohu.com. All the documents' categories are manually labeled and the category count is 10.

Also for time efficiency, we use the mini version of the dataset, which contains 17910 documents on 9 categories. The detail on each category is shown in table 2 We firstly do word segmentation on the whole set and get about 270,000 words in our dictionary. Then we transfer each document into bag-of-words representation just like the above dataset. Finally we got 17,014 training vectors and 896 testing vectors, all are single-labeled.

The overall description of datasets is in table 3.

[4] http://www.sogou.com/labs/dl/c.html

Table 2: Summary of Sogou Mini Set

Category Number	Category	Train	Test
C000008	Finance	1890	100
C000014	Sports	1891	99
C000024	Military	1890	100
C000023	Culture	1891	99
C000022	Recruitment	1890	100
C000020	Education	1891	99
C000016	Travel	1890	100
C000013	Health	1891	99
C000010	Vehicle	1890	100

Table 3: Overall Description of Datasets

Dataset	Category Count	Train Count	Test Count	Vocabulary Size
LSHTC	100	13113	800	161899
Sogou Mini	9	17014	896	270000

4.2 Experimental settings

Firstly, we initialize U and W (described in section 3) with random decimals between -1 and 1. We found that the larger the hidden units count is, the higher the classification accuracy will be, however the memory and time cost will also grow up. So hidden units count is set to 30, balancing the memory cost and the outcome accuracy. Learning rate is dynamic and initialized to 1.0. When $error(x)$ grows up in 10 consecutive training documents, learning rate will decrease by 0.01, vice versa.

Just as we demonstrate in section 3, the training is performed using Stochastic Gradient Descent (SGD). We go through all the training data iteratively, and update the weight matrices U and W online (after processing every document) until convergence appears. Here we define convergence as $error(x)$ is smaller than 10^{-5} in 10 consecutive training documents.

We evaluate our model in two ways. First, we use the output $\hat{y}$ of our neural network directly as the classification vector and use the index of the largest element as the category number. Second, we replace each point in bag-of-words vector with the corresponding column in U(which is in fact a kind of word embedding), multiplying the TF of the word. Then we use a linear SVM classifier to train and test the new document vectors.

During all the process, we use AMD GPU and its Aparapi[5] to do parallel computing and accelerate the whole process.

[5] http://developer.amd.com/tools-and-sdks/opencl-zone/opencl-libraries/aparapi/

4.3 Results

To compare the performance of our representations with BOW, we use the popular LIBSVM[4] as the final classifier. Table 4 and 5 show the results. *NN-15* represents that the dimensionality of document embedding is 15, while *NN-30* represents that the dimensionality of document embedding is 30.

Since our NN architecture can output the class label directly, we also give the results without combining SVM. The evaluations without SVM have a similar accuracy with BOW+SVM when a document is represented by 30 dimensional vector. With combining SVM, our method outperforms BOW+SVM a lot on both the datasets.

Table 4: Comparative results on LSHTC dataset.

Method	Micro-P	Macro-P	Macro-R	Macro-F
BOW + SVM	64.50	61.46	64.50	62.95
NN-15	52.12	52.13	47.12	49.50
NN-15 + SVM	62.12	62.12	61.71	61.91
NN-30	53.00	50.39	53.00	51.66
NN-30 + SVM	68.00	62.76	68.00	65.28

Table 5: Comparative results on Sogou dataset

Method	Micro-P	Macro-P	Macro-R	Macro-F
BOW + SVM	91.07	91.17	91.07	91.12
NN-15	90.18	90.17	90.42	90.29
NN-15 + SVM	90.85	90.84	90.94	90.89
NN-30	91.07	91.15	91.06	91.11
NN-30 + SVM	91.52	91.58	91.51	91.54

From the results, we can see that our architecture achieve better performances on both the datasets. This is mainly because we represent each document in a more detailed way comparing with pure bag-of-words representations. And the detailed representations are highly related to the topic of documents. Thus making the classification has a higher accuracy.

We can also find that the experimental results over the two test datasets are quite different. Performance on Sogou is better than that on LSHTC, on all the five methods. That is perhaps due to different data density of the two datasets. We believe that our architecture is more powerful on dense dataset, which has more average documents on each category. Therefore, our method which can generate document embedding to represent a document and do the document classification task is efficient and useful.

5 Related Works

Liu [14] used deep belief network (DBN) for text classification. However, since he used Restricted Boltzmann Machines (RBM) [8] to obtain the high level abstraction of input data, the dimensionality of the input data needs to be reduced in advance. Thus a lot of information may be lost.

Le and Mikolov [13] proposed *Paragraph Vector*, an unsupervised framework that learns continuous distributed vector representations for pieces of texts. The texts can be of variable-length, ranging from sentences to documents. Although paragraph vector can be applied to variable-length pieces of document, it is learned separately before they are used in text classification.

Socher et al. [26] used a more sophisticated approach to combine the word vectors in an order given by a parse tree of a sentence, using matrix-vector operations. However, it has been shown to work for only sentences because it relies on parsing.

6 Conclusion

In this paper, we propose a neural network architecture for text classification. In our architecture, each document is represented by a low dimensional embedding that is similar to word embedding. Experiments show that our embeddings have a higher classification accuracy than BOW vectors.

In future, we will use our method to do the multi-label text classification task. Besides, we will also investigate whether it can increase the performance by increasing the network layers.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was funded by NSFC (No.61003091) and Science and Technology Commission of Shanghai Municipality (14ZR1403200).

References

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropoulos, C.D.: An evaluation of naive bayesian anti-spam filtering. arXiv preprint cs/0006013 (2000)

[2] Bengio, Y., Schwenk, H., Senécal, J.S., Morin, F., Gauvain, J.L.: Neural probabilistic language models. In: Innovations in Machine Learning, pp. 137–186. Springer (2006)

[3] Carvalho, V.R., Cohen, W.W.: On the collective classification of email speech acts. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 345–352. ACM (2005)

[4] Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), 27 (2011)
[5] Cohen, W.W.: Learning rules that classify e-mail. In: AAAI spring symposium on machine learning in information access. vol. 18, p. 25. California (1996)
[6] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. The Journal of Machine Learning Research 12, 2493–2537 (2011)
[7] Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In: Proceedings of the seventh international conference on Information and knowledge management. pp. 148–155. ACM (1998)
[8] Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
[9] Huang, E.H., Socher, R., Manning, C.D., Ng, A.Y.: Improving word representations via global context and multiple word prototypes. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. pp. 873–882. Association for Computational Linguistics (2012)
[10] Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. Springer (1998)
[11] Khosravi, H., Wilks, Y.: Routing email automatically by purpose not topic. Natural Language Engineering 5(3), 237–250 (1999)
[12] Larochelle, H., Bengio, Y.: Classification using discriminative restricted boltzmann machines. In: Proceedings of the 25th international conference on Machine learning. pp. 536–543. ACM (2008)
[13] Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053 (2014)
[14] Liu, T.: A novel text classification approach based on deep belief network. In: Neural Information Processing. Theory and Algorithms, pp. 314–321. Springer (2010)
[15] McCallum, A., Nigam, K., et al.: A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization. vol. 752, pp. 41–48. Citeseer (1998)
[16] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. pp. 3111–3119 (2013)
[17] Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of NAACL-HLT. pp. 746–751 (2013)
[18] Mitchell, J., Lapata, M.: Composition in distributional models of semantics. Cognitive science 34(8), 1388–1429 (2010)
[19] Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. In: NIPS. pp. 1081–1088 (2008)
[20] Nasr, G.E., Badr, E., Joun, C.: Cross entropy error function in neural networks: Forecasting gasoline demand. In: FLAIRS Conference. pp. 381–384 (2002)

[21] Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and trends in information retrieval 2(1-2), 1–135 (2008)

[22] Quinlan, J.R.: Induction of decision trees. Machine learning 1(1), 81–106 (1986)

[23] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. MIT Press, Cambridge, MA, USA (1988)

[24] Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613–620 (1975)

[25] Sebastiani, F.: Machine learning in automated text categorization. ACM computing surveys 34(1), 1–47 (2002)

[26] Socher, R., Lin, C.C., Ng, A.Y., Manning, C.D.: Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In: Proceedings of the 26th International Conference on Machine Learning (ICML) (2011)

[27] Yang, Y., Pedersen, J.: A comparative study on feature selection in text categorization. In: Proc. of Int. Conf. on Mach. Learn. (ICML). vol. 97 (1997)