A Word Embedding Transfer Model for Robust Text Categorization

Yiming Zhang[0000−0003−2516−0900], Jing Wang[0000−0001−9024−6458], Weijian Deng[0000−0003−1565−742X], and Yaojie Lu[0000−0002−5842−7715]

[1] Institute of Information Engineering, CAS zhangyiming@iie.ac.cn
[2] Beijing University of Posts and Telecommunications jingw@bupt.edu.cn
[3] Institute of Information Engineering, CAS dengwj16@gmail.com
[4] Institute of Software, CAS yaojie.lu@outlook.com

Abstract. It is common to fine-tune pre-trained word embeddings in text categorization. However, we find that fine-tuning does not guarantee improvement across text categorization datasets, while could introduce considerable parameters to model. In this paper, we study new transfer methods to solve the problems above, and propose "Robustness of OOVs" to provide a perspective to reduce memory consumption further. The experimental results show that the proposed method is proved to be a good alternative to fine-tuning method on large dataset.

Keywords: Word Embedding · Text Categorization · Transfer Learning.

1 Introduction

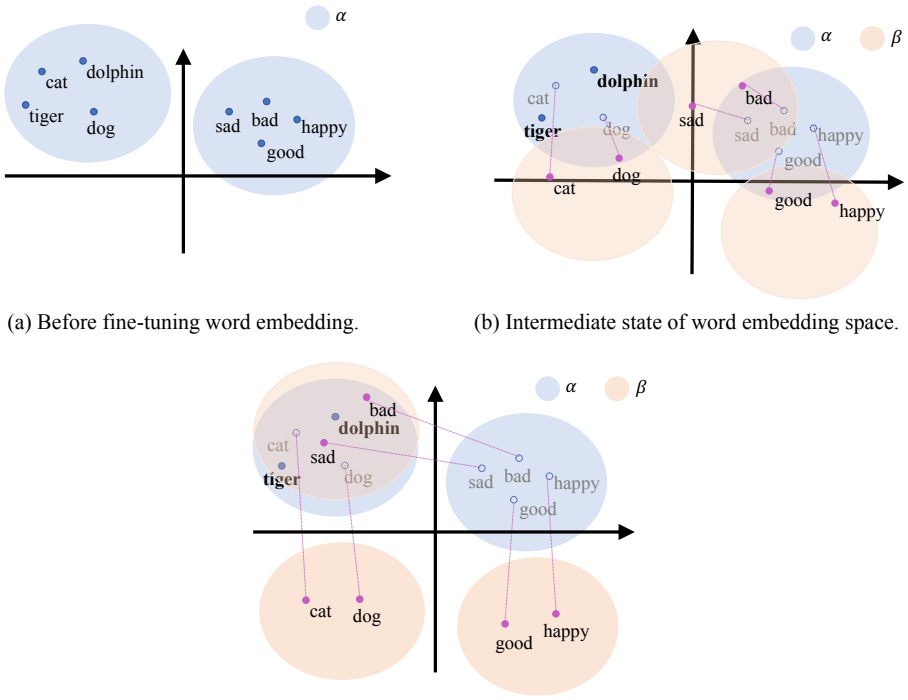
For many natural language processing (NLP) tasks such as text categorization, the word-level features of text usually derive from pre-trained word embeddings [1]. The word embeddings are learned by unsupervised learning algorithms such as Skip-gram, cBoW [15], Glove [18], and FastText [2] on very large-scale corpus.

For text categorization task, it is common to fine-tune pre-trained word embeddings (noted as "fine-tuning method" in this paper) to learn task-specific word-level features. Previous work [8, 24] report that this is effective on several datasets. And researchers usually attribute improvement to the learned task-specific word-level features [8, 21].

However, we argue that fine-tuning word embeddings is not always a good choice. To avoid confusion, we refer the words appearing in training set as in-vocabulary words (IVs), and words not appearing in training set as out-of-vocabulary words (OOVs) in the rest of this section.

As shown in Fig. 1(a), we denote the pre-trained word embedding space as α. Since the vocabulary of training set is limited, the fine-tuning model can only transform the IVs into task-specific word embedding space (denoted as β) and OOVs stay in space α. With the increasing number of training steps, the distribution of words in space α will become increasingly different from the distribution of words in space β (the knowledge in space α is general while the knowledge in space β is task-specific). As a result, there would exist two embedding space for words in test set to choose from.

However, as shown in Fig. 1(c), there could exist overlapping regions among α and β. In these overlapping regions, since the model generally cannot distinguish multiple



(a) Before fine-tuning word embedding.

(b) Intermediate state of word embedding space.

(c) Final state of the word embedding space.

Fig. 1: Illustration of a disadvantage of fine-tuning method. Blue areas represent pre-trained embedding space (denoted as space α). Orange areas represent generated task-specific embedding space (denoted as space β). From top to bottom: (a) the state before fine-tuning word embedding; (b) the beginning of fine-tuning where the updates on words generally occur within or between clusters; (c) the final state of fine-tuning where there are many overlapping regions among space α and space β, thus the meaning of words in α casts to the meaning of words in β.

word embedding spaces (can only work on space β), the meaning of the OOVs casts to the meaning of IVs automatically. However, these casting operations are usually unreasonable. For example, as shown in Fig. 1(c), suppose that "tiger" and "dolphin" are OOVs and the given task is sentiment classification. Because "tiger" and "dolphin" are unseen for model, a word cluster representing negative sentiment information is likely generated nearby. As a result, the "tiger" and "dolphin" will carry negative information if they appear in testing set, which could greatly mislead the decision of sentiment polarity for the corresponding samples.

Furthermore, our experimental results on AGNews and Yelp Reviews demonstrate that fine-tuning degrades model performance dramatically. On the other hand, the fine-tuning method on a large-scale dataset could introduce considerable parameters to the model, and consume hard-to-afford memory resources.

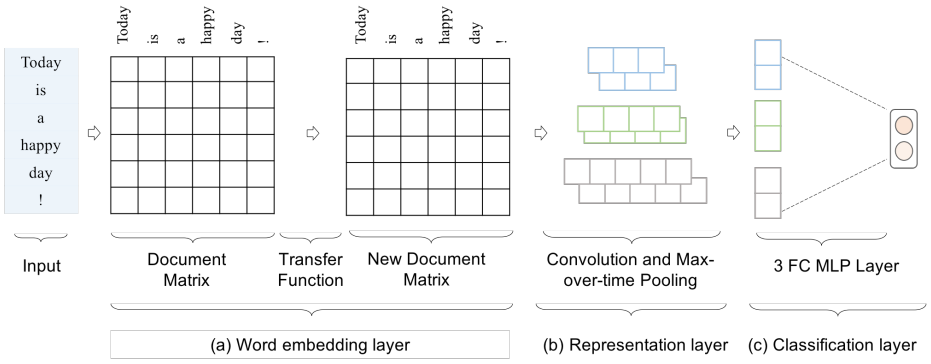


Fig. 2: The structure of our model. The model structure mainly consists of three parts (from input to output): (a) word embedding layer, (b) representation layer, and (c) classification layer. "FC" denotes Full-Connected. In this work, we focus on word embedding layer, and study the influence of transfer methods to model performance.

To solve the above issues of fine-tuning method, we propose another two new transfer methods based on the fixing method, i.e., from the simple method, noted as "scaling method" , to the complicated method "linear transformation method", noted as "lin-trans method". In addition, we also introduce "Robustness to OOVs" as a metric of memory consumption and we discuss it in detail in the Sec. 4.3.

2 Related Work

Distributed representation of words [1] which represent a word with a low-dimensional and dense embedding, alleviate the data sparsity problem and has been the basic methods in text categorization[14, 5, 6]. [7] and [8] first apply word-level convolutional neural network [12] to text categorization task and have achieved significant progress.

Recently, we have witnessed increasing efforts in utilizing the pre-trained word embedding[22], which introduce useful external knowledge to their model. [16] investigate the transferability of neural networks in NLP by Layer-by-Layer analysis, and found that word embeddings could be transferable to semantically different tasks.

Most of works [8, 7] suggest that fine-tuning word embedding in the training stage can get a better effect. On the contrary, almost all of the works on machine reading comprehension task[23, 19, 10, 25] are prone to fixing pre-trained word embeddings after transferred to MRC model, even [13] points out fine-tuning method can dramatically degrade model performance. Inspired by this, we report the disadvantages of fine-tuning method in Sec. 1, and propose two new transfer methods with the word embedding layer fixed.

3 Model

3.1 Preliminaries

Given a document $S = w_1 w_2 ... w_i ... w_l$, where l is the document length and w_i is the i-th token. Each token w_i is first transformed to v_i by looking up the $V * D$ word embedding table E, where V is vocab size and D is the feature dimension of each word vector. Then, S can be represented by the output matrix $S_{mat} = [v_1, v_2, ..., v_l]$. Given a "transfer function" $F_{tr}(v)$ to represented different transfer methods, each element v_i in S_{mat} is fed to $F(v)$ then S_{mat} can be represented by the new output matrix $S_{tr\_mat} = [v_{T1}, v_{T2}, ..., v_{Tl}]$, where $v_{Ti} = F_{tr}(v_i)$.

As introduced in Sec.1, there are two choices for the "transfer function" : scaling method and lin-trans method.

3.2 Scaling method

Assume v is a D-dimensional vector, then in scaling method, the transfer function $F_{tr}(v)$ is defined as

$$F_{tr}(v) = v \otimes u \tag{1}$$

where $\otimes$ means element-wise multiplication and U is a trainable real vector, $u = [u_1, u_2, ..., u_D]$. Each element u_i in u denotes the scaling degree of the i-th element of input vector v.

3.3 Lin-trans method

Assume v is a *D*-dimensional vector, then in lin-trans method, the transfer function $F_{tr}(v)$ is defined as

$$F_{tr}(v) = U \cdot v + b \tag{2}$$

where U is a $D * D$ real matrix, b is bias vector $b = [b_1, b_2, ..., b_D]$, and "$\cdot$" denotes matrix multiplication. U and b are trainable parameters. After this transfer function, pre-trained embedding space is linearly transformed to a new embedding space.

Compare with fine-tuning method. The proposed two transfer methods are all applied on the whole embedding feature space rather than directly fine-tuning word embedding layer. Therefore, the proposed two methods would not suffer from unaffordable memory problem on large-scale datasets.

3.4 Model Structure

As demonstrated in Fig.2, the model structure mainly contains three parts: word embedding layer, representation layer and classification layer.

We adopt shallow-and-wide convolution structure [8, 11] as our representation layer. We adopt MLP with two hidden layers as our classification layer. The activation functions used in convolution layer and classification layer are all rectified linear unit (ReLU) [17]. And the softmax activation function is used in the output layer to normalize the output logits to get the predictive probabilities for all target labels. In addition, Dropout[20] is applied on the result of representation layer to improve the robustness of model.

3.5 Model Training

Given all of training instances and their labels from dataset D, the objective function $J(\theta)$ is as follows:

$$J(\theta) = \sum_{(x_i, y_i \in D)} \log p(y_i|x_i, \theta), \tag{3}$$

where θ is the parameters of model; x_i is a instance from dataset D and y_i is its label.

We maximize the log likelihood $J(\theta)$ through stochastic gradient descent over shuffled mini-batches method. We adopt Adam[9] as optimizer in this paper.

4 Experiment

Dataset	#Train	#Test	#Classes	Vocab Size	#OOVs & Proportion
AGNews	120K	7.6K	4	90549	8946 (9.8%)
Yelp2	560K	38K	2	291541	143708 (49.2%)
Yelp5	650K	50K	5	314526	161224 (51.2%)
Yahoo	1,400K	60K	10	1156111	848237 (73.3%)

Table 1: Statistics of four text categorization datasets used in this paper.

4.1 Datasets

We select four available datasets, from a relative small scale (120K samples in training set) to a relative large scale (1400K samples in training set) on text categorization for experiment. They are AGNews, Yelp Reviews Polarity (noted as "Yelp2"), Yelp Reviews Full (noted as "Yelp5") and Yahoo Answers (noted as "Yahoo"). These four datasets introduced by previous work[26] are summarized in Table 1. To avoid confusion, the OOVs in this paper is referred to the words appear in training or testing set while not appear in pre-trained word embeddings. The vocab size in this paper refers to words in both training and testing set.

4.2 Implementation Details

Before experiment, we tokenize all datasets using *WordpunktTokenizer* provided in NLTK and pad five tokens "<pad>" to each side of document. Public available *word2vec* [15] is used as pre-trained word embeddings in this paper. The out-of-vocabulary words are initialized randomly in reset settings. We fix random seed to 1 for all experiments to ensure fairness of comparisons.

In all experiments, we keep all hyper-parameters the same. We set batch size 32. We use Adam[9] as optimizer and set learning rate 0.001. The convolution layer has

four filter groups with $2, 3, 4, 5$ sizes, respectively, and each filter group contains 1500 filters. We use Xavier [3] method to initialize the parameters of filters. The dropout rate is set to 0.5. Each hidden layer in MLP has 300 hidden units and initialized using [4]. In scaling method, the parameters in transfer function F_{tr} are randomly initialized with Gaussian distribution with a mean of 1.0 and a standard deviation of 0.1. In lin-trans method, the W in transfer function F_{tr} is initialized with a unit diagonal matrix and Gaussian noise with a mean of 0.0 and a standard deviation of 0.1 is added, and b in F_{tr} in initialized to 0.

4.3 Evaluation

The evaluation results of the four transfer methods are shown in Table 2. To test the effectiveness of our re-implemented model, we extra give a baseline (fine-tune.CNN) [11], which is consist of a embedding layer using fine-tuning method, a representation layer with the same structure as ours, and a linear classification layer.

Method	AGNews	Yelp2	Yelp5	Yahoo
fine-tune.CNN	92.2	95.9	64.9	73.0
fixing	93.75	96.27	65.57	74.40
fine-tuning	93.11	96.27	65.31	74.90
scaling (Ours)	**93.82**	96.35	**65.96**	**75.09**
lin-trans (Ours)	93.73	**96.42**	65.73	74.68

Table 2: The accuracy of various methods on four text categorization datasets.

Fine-tuning method is unaffordable on large dataset. There are more than 1,000,000 words in Yahoo. When using the Adam optimizer, the word embedding layer can not even be loaded into a single graphics card (e.g., Titan Xp with 12GB GPU memory). As a result, the word embedding layer has to be loaded into CPU memory and updated by CPU, which greatly slows down the training speed. The training hours of different transfer methods (with the same training steps across methods for a dataset) are shown in Table 3. Noted that the fine-tuning method needs training for 208 hours on Yahoo, 6X slower than fixing method.

Fine-tuning method does not guarantee improvement. Noted that fine-tuning method degrades the performance on AGNews, Yelp2 and Yelp5. We guess that the

Method	AGNews	Yelp2	Yelp5	Yahoo
fixing	0.4	9	9.7	33
fine-tuning	0.5	9.5	10.5	208
scaling	0.6	11	12	37
lin-trans	1.5	13	14.5	39

Table 3: The training hours of different transfer methods

Method	AGNews	Yelp2	Yelp5	Yahoo
fixing	0.16	0.01	0.24	0.11
fine-tuning	0.26	0.05	**0.07**	0.94
scaling	**-0.11**	**0.00**	0.15	**0.10**
lin-trans	0.06	**0.00**	0.11	0.12

Table 4: Accuracy decrement of four transfer methods.

model overfits these not very large datasets due to the introduced considerable parameters by fine-tuning method. This phenomenon is not consistent with that in [8], we guess the reason is datasets in [8] are too small that model could gain significant improvement from task-specific word-level knowledge. Only if a dataset is relative large, the limited task-specific word-level knowledge could not easy to cancel out the bad effect of overfitting.

Scaling method is a good solution on a large-scale dataset. It is noted that the performance of scaling method is significant in relative large datasets such as Yelp5 (further improve 0.4% than fixing method) and Yahoo (further improve 0.69% than fixing method), even surpasses fine-tuning method on Yahoo. Noted that training with scaling method only need 37 hours while fine-tuning method need 208 hours on Yahoo. Besides, the lin-trans method is inferior to scaling method both in testing performance and training time. Therefore, scaling method is better in large dataset under the considerations above.

Robustness to out-of-vocabulary words (OOVs). As summarized in Table 1, OOVs occupy a very large proportion in datasets (surpass 50% of vocabulary in a relative large dataset). It can further reduce the memory consumption by replacing OOVs to some appointed symbol because the size of embedding layer is O(N) space complexity to vocab size N. The reduction of memory resources provides conditions for designing larger and more complex presentation and categorization layers. However, this reduction can only be safely conducted when the transfer method is robust enough to the replacement of OOVs.

We again conduct experiments with the reset settings, except for the replacement of all OOVs to "< unk >". This could leads to performance decrement because the information about the distinction of OOVs is discarded. Then we regard performance decrement (reduced accuracy compared to Table 2) as a simple metric of robustness to OOVs. We say a transfer method is more robust to OOVs if its performance decrement is smaller (if the decrement is less than 0, it means replacing OOVs could get extra improvement). The experimental results are shown in Table 4.

It is clear that fine-tuning method has the worst robustness to OOVs. Even on large dataset Yahoo, the performance of fine-tuning method will drop to a lower level than that of the fixing method. So it is unwise to replace OOVs when using fine-tuning method. Conversely, replacing OOVs in proposed scaling method and lin-trans method only slightly degrades performance across all datasets in our experiments. Thus, the memory consumption could be further reduced by the proposed method.

Selection of transfer methods. Although in previous works [8], fine-tuning has been proved to be effective on very small datasets, e.g., MR, SST and TREC, we report that it is unwise to use fine-tuning method on larger datasets, e.g., from AGNews with 120K samples to Yahoo with 1400K samples, either because of poor testing performance or unaffordable memory consumption. In contrast, the proposed scaling method as an improved fixing method is proved to be effective and memory friendly on relative large datasets in this paper. Therefore, using fine-tuning method on very small datasets and using the scaling method on a larger dataset is recommendable.

5 Conclusion

In this work, we report fine-tuning word embedding could suffer from poor testing performance and unaffordable memory consumption problems on relatively large scale text categorization datasets. To alleviate those problems, we propose new transfer methods based on the fixing method, and introduce "Robustness of OOVs" to provide a perspective to further reduce memory consumption. The experimental results demonstrate that the proposed scaling method improves the accuracy of fixing method, while only introduces minor parameters and has the better robustness to OOVs. Thus, the scaling method is an effective and robust alternative for text categorization.

References

1. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. Journal of Machine Learning Research pp. 1137–1155 (2003)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**(0), 135–146 (2017)
3. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (13–15 May 2010), http://proceedings.mlr.press/v9/glorot10a.html
4. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. CoRR **abs/1502.01852** (2015), http://arxiv.org/abs/1502.01852
5. Ji, Y., Smith, N.A.: Neural discourse structure for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 996–1005. Association for Computational Linguistics, Vancouver, Canada (July 2017), http://aclweb.org/anthology/P17-1092
6. Johnson, R., Zhang, T.: Deep pyramid convolutional neural networks for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 562–570. Association for Computational Linguistics, Vancouver, Canada (July 2017), http://aclweb.org/anthology/P17-1052
7. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 655–665. Association for Computational Linguistics, Baltimore, Maryland (June 2014), http://www.aclweb.org/anthology/P14-1062

8. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar (October 2014), http://www.aclweb.org/anthology/D14-1181
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014), http://arxiv.org/abs/1412.6980
10. Krause, B., Murray, I., Renals, S., Lu, L.: Multiplicative lstm for sequence modelling. arXiv: Neural and Evolutionary Computing (2017)
11. Le, H.T., Cerisara, C., Denis, A.: Do convolutional networks need to be deep for text classification (2017)
12. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp. 2278–2324 (1998)
13. Li, P., Li, W., He, Z., Wang, X., Cao, Y., Zhou, J., Xu, W.: Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. arXiv preprint arXiv:1607.06275 (2016)
14. Liu, P., Qiu, X., Huang, X.: Adversarial multi-task learning for text classification. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1–10. Association for Computational Linguistics, Vancouver, Canada (July 2017), http://aclweb.org/anthology/P17-1001
15. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
16. Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., Jin, Z.: How transferable are neural networks in nlp applications? In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 479–489. Association for Computational Linguistics, Austin, Texas (November 2016), https://aclweb.org/anthology/D16-1046
17. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning. pp. 807–814 (2010)
18. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
19. Seo, M.J., Kembhavi, A., Farhadi, A., Hajishirzi, H.: Bidirectional attention flow for machine comprehension. international conference on learning representations (2017)
20. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
21. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks pp. 1556–1566 (July 2015), http://www.aclweb.org/anthology/P15-1150
22. Wang, P., Xu, J., Xu, B., Liu, C., Zhang, H., Wang, F., Hao, H.: Semantic clustering and convolutional neural network for short text categorization pp. 352–357 (2015)
23. Wang, W., Yang, N., Wei, F., Chang, B., Zhou, M.: Gated self-matching networks for reading comprehension and question answering. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 189–198 (2017)
24. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A.J., Hovy, E.H.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1480–1489 (2016)
25. Yu, A.W., Dohan, D., Luong, M.T., Zhao, R., Chen, K., Norouzi, M., Le, Q.V.: Qanet: Combining local convolution with global self-attention for reading comprehension. arXiv preprint arXiv:1804.09541 (2018)

26. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. pp. 649–657. NIPS'15 (2015)