Improving Concept Representations for Short Text Classification

Sijie Tao Tetsuya Sakai
Waseda University
tsjmailbox@ruri.waseda.jp tetsuyasakai@acm.org

Abstract

Short text classification is a challenging task in Natural Language Processing (NLP). Compared with documents, short texts are more sparse and ambiguous due to lack of context. In order to overcome this difficulty, recent work tends to take an approach of combining neural language models with external linguistic resources such as knowledge bases. In this kind of approaches, concepts obtained from a knowledge base are usually mapped to an implicit space and represented as a vector. However, how to effectively represent concepts in a neural language model is not well studied yet. Hence, in this study, we construct several formulae for concept embedding and compare them in a short text classification task. Our experimental results show that utilizing proper concept embeddings can slightly improve the performance of short text classification.

1 Introduction

With the remarkable development of the Internet, the amount of short texts is booming in many web application areas, such as short messages, Instant Messages, online Chat Logs, Bulletin Board System Titles, Web Logs Comments, Internet News Comments, SMS, Twitter etc[6]. Processing short texts becomes a crucial task in Natural Language Processing (NLP). For example, tweets related to influenza can be classified into positive or negative for disease surveillance[1]. However, short text classification is considered as a challenging task. The difficulty of this task originates from the sparsity and the ambiguity of short texts. Short texts contain less context information than documents or paragraphs. Moreover, short texts are more ambiguous as it is hard to identify the meaning of polysemes with limited context.

To overcome the characteristics of short texts, recent work [9][11][2] tends to tackle short text classification by combining neural language models with external linguistic resources such as knowledge bases. In this kind of approaches, a knowledge base obtains a set of concepts related to a word, and the concepts are mapped to a neural language model space and represented as a vector. The word embedding and the concept representation are then concatenated to enrich the text representation.

Looking into the results of previous work, it is obvious that combining a neural language model and a knowledge base can enrich the text representation and improve the performance of classification. However, as far as we are aware, how to effectively represent concepts in a neural language model is not well studied yet.

Thus in this paper, we construct several formulae for concept embedding and compare them in a short text classification task. By evaluating the performances of classification, our experimental results show that different types of concept representation can effect the performance of short text classification and utilizing proper concept embedddings can slightly improve the performance.

The remainder of this paper is organized as follows. Section 2 introduces related work. Section 3 explains the details of the proposed method. The information of experimental settings is provided in Section 4. Section 5 and 6 give the analysis of the experimental results and the conclusion of this study respectively.

2 Related Work

One of the effective methods to overcome the difficulty of short text classification is to improve text representations via associate short texts with external explicit language resources, such as a knowledge base. The knowledge retrieved from a knowledge base contains rich semantic information and has been used in several previous work.

Wang et al.[9] adopted a large-scale knowledge base Probase[10] to retrieve relevant concepts and associate them with short texts. Given a short text as an input, the conceptualization API of Probase can obtain a concept vector including relevant concepts and weights that show the relevance of the concepts related to the text. The concepts are

directly mapped to word embeddings from a pre-trained model and the embedding of the concepts is represented as a weighted average of the word embeddings.

In the work of Xu et al.[11], concepts obtained from Probase are first represented as the weighted average of all the words which belong each concept, rather than being directly transformed to word embeddings. Chen et al.[2] took full advantages of knowledge bases. Multiple knowledge bases such as YAGO[8], Probase, and CN-Probase[3] are applied in this work.

3 Proposed Method

In this section, we describe the details of proposed methods. First, given an input word x, the knowledge base returns a list C which contains the top-k concepts related to the word: $C = \{c_1 : w_1, c_2 : w_2, ..., c_k : w_k\}$, where $c_1, c_2, ..., c_k$ are the concepts and $w_1, w_2, ..., w_k$ are scores that represent the relavance of the concept with the given word. Second, every concept c is then transformed to vectors v_c using a formula of concept embedding. Next, the conceptual representation of word x is calculated as the weighted average using the concept embeddings and their weights. At last, the word embedding of x is concatenated with its conceptual representation as the final text representation for classification. The following sections presents the formulae of concept embedding.

In this study, we use Microsoft Concept Graph[10][1] to obtain concept knowledge and a pre-trained Word2Vec model[2] to obtain word embeddings. For out-of-vocabulary words, their embeddings are initialized with zero vectors.

3.1 Simple Average Embedding

Given a concept c, and the words $\{x_1, x_2, ..., x_n\}$ belong to it, the concept embedding v_c can be represented as the average of the word embeddings:

$$v_c = \frac{\sum_{i=1}^{n} e_{x_i}}{n} \tag{1}$$

where e_{x_n} is the word embedding of x_n obtained from the pre-trained word embedding model. We call this formula the Simple Average Embedding.

3.2 Weighted Average Embedding

In a knowledge base, knowledge is stored as (x, c, r), where r is a relation score shows the relavance of the word-concept pair (x, c). Based on the Simple Average formula, we can construct a formula to represent concept incorporating the weights. In other words, a concept can be represented as the weighted average of the word embeddings. Previous work [9][11] utilized this formula to represent concepts in experiments. In this study, this formula is named as the Weighted Average Embedding.

$$v_c = \frac{\sum_{i=1}^{n} r_i e_{x_i}}{\sum_{i=1}^{n} r_i} \tag{2}$$

3.3 Relation-aware Average Embedding

In this formula, we assume that when a concept c is mapped to a implicit space, its representation v_c should be similar to the word embedding of itself e_c. Based on this assumption, we can represent the relation between the concept and the words belong to it as:

$$v_{r_i} = e_c - e_{x_i} \tag{3}$$

Then the concept embedding is calculated by the sum of average of word embeddings and the weighted average of relation representations. We name this formula the Relation-aware Average Embedding.

$$v_c = \frac{\sum_{i=1}^{n} e_{x_i}}{n} + \frac{\sum_{i=1}^{n} r_i v_{r_i}}{\sum_{i=1}^{n} r_i} \tag{4}$$

4 Experiments

In order to evaluate the formulae of concept embedding, we establish a short text classification task and compare them. This section describes the experiment setup of this work. The environment of experiments is shown in Table 1.

Table 1: Environment of Experiments

CPU	Intel Core i7-6700K
Memory Size	32GB
GPU	Nvidia GeForce RTX 2080Ti
GPU Memory Size	11GB

4.1 Datasets

In our experiments, we adopt two widely known datasets: The Question Classification Dataset of Li and Roth, and AG's corpus of news articles.

[1] https://concept.research.microsoft.com/

[2] https://code.google.com/archive/p/word2vec/

Table 2: Hyper Parameters

Parameters	Values
kernel size	3, 4, 5, 6
embedding dimension	300
hidden layer dimension	100
dropout rate	0.5
learning rate	0.001

4.1.1 The Question Classification Dataset of Li and Roth

The Question Classification Dataset of Li and Roth[5][3] (hereafter reffered to as QCD) contains questions which are classified into six categories. The training set has 5,452 questions and the test set has 500 questions.

4.1.2 AG's corpus of news articles

AG's corpus of news articles[4] (hereafter reffered to as AG's news) is a dataset that includes title, description, category, etc. of over 1 million news articles. In order to construct a short text classification task, we only use the titles and the categories. The dataset used in this study is a subset of AG's news and it is adopted from the work of Zhang et al.[12]. The number of classes is four and there are 120,000 training samples and 7,600 test samples.

4.2 Text Classifier

After obtaining the text representations, we build a text classifier with Convolutional Neural Networks (CNN) to test the performances of different input embeddings. The classifier includes a convolution layer, a max-pooling layer and a hidden layer with dropout[7]. Inspired by previous work[13], we set multiple kernel sizes for the convolution layer to capture different scales of feature maps. The activation funcitons are set to ReLU and tanh for convolution layer and hidden layer respectively. The loss funciton is set to categorical cross entropy and we adopt Adagrad[4] to optimize the training process. Following the work of Wang et al.[9], the hyper parameters are set as shown in Table 2.

The training process is continued until the loss on the test set stops to change for two epochs. The highest accuracy on test set, which is calculated as the percentage of the samples are correctly predicted, is used for the evaluation of the model to make a comparison between the different embeddings.

[3] https://cogcomp.seas.upenn.edu/Data/QA/QC/

[4] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

Table 3: Accuracy on the test sets

Formula	QCD	AG's news
Simple	91.40	83.80
Weighted	92.00	84.54
Relation-aware	**93.20**	**85.13**

Figure 1: Cosine similarity distribution of the Simple Average Embeddings

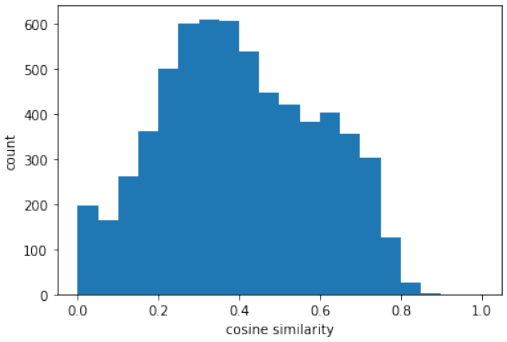


Figure 2: Cosine similarity distribution of the Weighted Average Embeddings

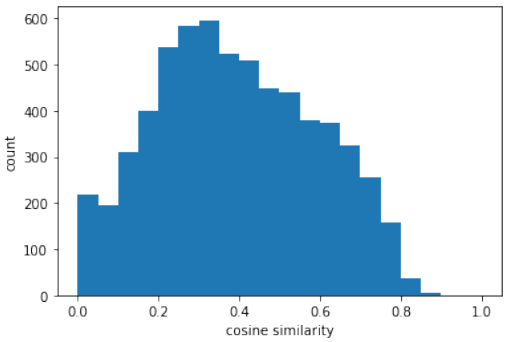
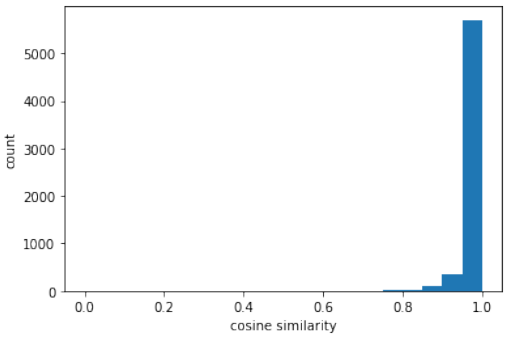


Figure 3: Cosine similarity distribution of the Relation-aware Average Embeddings



5 Results and Discussions

The experimental results on both datasets are shown in Table 3. We can see that the performance of the model using the Relation-aware Average Embedding outperforms those with the Simple Average Em-

bedding and the Weighted Average Embedding by 1.8% and 1.2% on QCD, 1.13% and 0.59% on AG's news, which indicates that the Relation-aware Average Embedding can slightly improve the performance of text classification on these datasets. The reason is that the Relation-aware Average Embeddings can help acquire features from the concepts effectively by adopting the representation of the relations between concepts and words. As a result, the Relation-aware Average Embedding is able to enrich the representation of short texts better than the other formulae.

In order to figure out the difference between the three concept embedding formulae, we calculated the cosine similarity between the concept embeddings and the word embeddings of the concept words. Except the words that do not have an embedding in the pre-trained Word2Vec model (Out-of-Vocabulary words), 6,337 pairs of concept embeddings and word embeddings are collected and their distributions of cosine similarity are presented in Figure 1, 2 and 3. From the graphs, we can clearly find that the Relation-aware Average Embeddings have higher similarity with the word embeddings while the other two types do not. It also indicates that the Relation-aware Average Embeddings meet the assumption that the concept embeddings are similar to their word embeddings.

6 Conclusion

In this paper, we construct three different formulae of concept embedding and compare them in a short text classification task. The experimental results show that the Relation-aware Average Embedding formula can slightly improve the performance of text classification because it can effectively represent concepts by considering relations between concepts and words. For future work, we would like to test the formulae on more datasets, with more classifiers in different architectures.

References

[1] Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. Twitter catches the flu: Detecting influenza epidemics using twitter. In *EMNLP*, 2011.

[2] Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. Deep short text classification with knowledge powered attention. In *AAAI*, 2019.

[3] Jindong Chen, Ao Wang, J. Bradley Chen, Yanghua Xiao, Zhendong Chu, Jingping Liu, Jiaqing Liang, and Wei Wang. Cn-probase: A data-driven approach for large-scale chinese taxonomy construction. *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1706–1709, 2019.

[4] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2010.

[5] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[6] Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. Short text classification: A survey. *Journal of Multimedia*, 9:635–643, 2014.

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929 - 1958, January 2014.

[8] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th International Conference on the World Wide Web*, pages 697–706, 2007.

[9] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, 2017.

[10] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 481 - 492, New York, NY, USA, 2012. Association for Computing Machinery.

[11] Jingyun Xu and Yi Cai. Incorporating context-relevant knowledge into convolutional neural networks for short text classification. In *AAAI*, 2019.

[12] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[13] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *IJCNLP*, 2015.