Sentence Similarity Computation in Question Answering Robot

IOP Conf. Series: Journal of Physics: Conf. Series **1237** (2019) 022093 doi:10.1088/1742-6596/1237/2/022093

Sentence Similarity Computation in Question Answering Robot

Shijing Si, Weiguo Zheng[1], Liuyang Zhou, Mei Zhang

Zhiyan Technology (Shenzhen) Limited Shenzhen, China

Corresponding author is Weiguo Zheng.

shijingsi@webot.co; fringeily@webot.co; leo@webot.co; meizhang@webot.co

Abstract. Computing semantic similarity between sentences or texts is vital in many natural language processing (NLP) tasks such as search, query suggestion, and question answering (QA). Many methods have been developed, based on lexical matching, distributional semantics, etc. However, lexical features, like string matching, fail to capture semantic similarity. In this research, our focus lies on the implementation of distributional representations and how to tune parameters when obtaining representations of words with commonly used word embedding techniques, e.g., Word2Vec and GloVe. We conduct experiments in the setting of Chinese semantic sentence matching tasks on the finance-domain. We examine the goodness of word embedding by both the cosine similarity of semantically similar sentence pairs and semantically dissimilar pairs. Based on our experiments, Word2Vec performs better than GloVe in the sense that Chinese character embedding from Word2Vec yield larger disparity of cosine distances between similar sentence pairs and dissimilar pairs. Also we report the optimal parameters for Word2Vec continuous bag-of-word (CBOW) through our trials, with window size being 6 and embedding dimension being 400, which can be good initial values for other projects.

1. Introduction

Computing semantic similarity between two sentences or texts is to check if two pieces of text mean the same thing or how semantically similar they are [1]. Being able to do so greatly is helpful in many settings like intelligent search [2], query suggestion [3], text summarization [4] and QA [5].

Many approaches have been proposed for semantic matching by the employment of lexical matching, linguistic analysis and semantic features. Methods for lexical matching focus on whether the words in two short texts have similar spellings, e.g., by means of edit distance [6], lexical overlapping [7] or largest common substring [8]. Although they work well for trivial cases, they are not robust as they may cause simple mistakes. For example, by the edit distance, the US would be closer to the UK, than it would be to the States. Features based on linguistic analysis, like dependency parsers or syntactic trees, are often helpful for measuring text similarity [9, 10]. Most languages generally have linguistic tools such as parsers, though their quality varies significantly across languages. However, a significant number of texts are not parsable (e.g., some weibo bloggers) and high-quality parsers might be expensive to compute at run time.

For semantic features, many approaches utilize external sources of structured semantic knowledge such as Wikipedia [11] or WordNet [11,12]. These resources are treated as knowledge bases or

semantic networks. Wikipedia is constructed around entities, so it is primarily useful in settings where we focus on rather well-known persons, organizations and technical terminologies. One disadvantage of using dictionaries or WordNet is that high-quality resources like these are only available to a few languages, and proper names, domain-specific technical terms and slang tend to be underrepresented [13].

Section 2 introduces semantic representations of tokens and the two most commonly used tools to build word embedding: Word2Vec and GloVe; in this section, a simple description of each tool/method is explained and a comparison between the two is made in order to understand their main differences. In Section 3, we explain the setup of our experiment and the report our findings. Also, in Section 3, we compare the results obtained. Finally, in Section 4, we conclude the paper making some final conclusions and establishing some possibilities of future work.

2. Word embedding

A text consists of sentences and words that are plainly strings by themselves. How to mathematically represent tokens (words or characters) and semantics they conveyed has been a longstanding topic [14]. Here we review a few remarkable word representations or distributional semantic techniques.

Distributional semantic approaches rely on the assumption that words appearing in similar contexts tend to have similar meanings. The Latent Semantic Analysis algorithm (LSA) [15] incorporates this intuition by building a word-document co-occurrence matrix from large corpora and performing singular value decomposition (SVD) on it to get a lower-dimensional representation. Words are represented as vectors in this lower dimensional space. The distance between these word vectors (measured with the cosine function) can be used as a proxy for semantic similarity. The full co-occurrence matrix, however, can become quite substantial for a large corpus, in which case the SVD becomes memory-intensive and computationally expensive.

Word embedding has recently seen an increasing interest as new ways of computing them efficiently have become available. In [16, 17] an algorithm called word2vec is proposed. It has two architectures: continuous bag-of-words (CBOW) and Skip-gram. Both are variations of the neural network language model (NNLM) [18]. Rather than predict a word conditioned on its predecessors, as in traditional language models, CBOW predicts a word from its surrounding words or context and Skip-gram predicts multiple surrounding words from one input word. To avoid computing a full softmax over the entire vocabulary, hierarchical softmax is applied on a Huffman tree representation of the vocabulary, which saves calculations, at the cost of some accuracy. An additional strategy to facilitate the computation is negative sampling, where, instead of only using the words observed next to one another in the training data as positive examples, random words are sampled from the corpus and presented to the network as negative examples.

Another ready-to-use way of getting word embedding, called GloVe, is proposed in [19]. Rather than being based on language models it is based on global matrix factorization. As such, it is closer to LSA, only a word-word co-occurrence matrix is used. GloVe avoids the large computational cost of, e.g., LSA by not building the full co-occurrence matrix, but training directly on the non-zero elements in it. As a cost function, the model uses a weighted least squares variant. The weighting function has two parameters, an exponent and a maximum cut-off value that influence the performance.

The recent trend of word representations lies on contextualized embedding, which is prompted by the fact that one word has multiple meanings in different circumstances. Perhaps the most notable ones are ELMo [20] and bi-directional encoder representation from transformers (BERT) [14].

In this paper, we focus on how to obtain the best possible word embedding from Word2Vec and GloVe models for the sentence matching task. Due to the lack of powerful computational resources like TPUs, we exclude dynamic embedding like ELMo and BERT. As both algorithms produce high-quality word embedding and their implementations are publicly available, we use them in our experiments. In Section 3 we report on the results and analyse the effect of different parameter settings for both methods.

3. Experiments

In this section, we conduct experiments to show the performances of Word2Vec and GloVe models, and examine their performances in the setting of sentence matching. The criteria of a good word embedding are: 1.) similar sentence pairs have large cosine similarity, close to 1; and 2.) dissimilar sentence pairs have small cosine similarity, close to 0.

The corpus we employed for word embedding is Chinese finance-domain knowledge and concepts from Baidu Baike, and it consists of approximately 13 million Chinese characters. To avoid the uncertainty caused by tokenization, we train character vectors with both the Word2Vec and GloVe models on the finance corpus, so we use the terms word embedding and character embedding interchangeably in this section.

When training the Chinese character vectors, we take multiple sets of hyper-parameters like the window size and embedding dimension. Specifically speaking, we set six choices for the embedding dimension, i.e., 50, 100, 200, 400, 500, 1000 and five values for window size, i.e., 3, 6, 9, 12, 15. Subsequently, we train character vectors on 6*5=30 combinations of hyper-parameters with two methods—Word2Vec and GloVe -- and we examine the character vectors with our test data. Our test data have two parts: the first part consisting of 1000 pairs of sentences that have similar meanings, i.e., the similar-sentence pairs, and the second part having 1000 dissimilar sentence pairs, i.e., the dissimilar-sentence pairs.

We evaluate word embeddings from the Word2Vec and GloVe models as follows. Firstly, we compute a sentence vector by taking the sum of all character embeddings in that sentence. Secondly, we use cosine similarity to represent the closeness between a pair of sentences. Thirdly, we take the average of cosine similarity of the set of similar sentence pairs, and that of the dissimilar sentence pairs. Also, we compute the difference by subtracting the average cosine similarity of the dissimilar pairs from that of the similar pairs. Lastly, we compare two indices – the average cosine similarity of similar pairs and difference of cosine similarity between two parts of test data -- under different combinations of hyper-parameters and different training methods. The good word embeddings should have a large average cosine similarity on the similar sentence pairs, and a small average cosine similarity on the dissimilar sentence pairs. Therefore the good one should also have a large difference between the average similarities of two parts of test data.

Fig.1 and Tab.1 present the performance of character embeddings from Word2Vec CBOW model. Similarly, Fig. 2 and Tab. 2 show that of GloVe model. Plots in Fig.1 correspond to tables in Tab.1 according to their positions. For instance, numbers in the top-left table in Tab.1 is presented in the top-left plot in Fig.1. The top-left plot shows the line graph of the average cosine similarities of similar-sentence pairs as a function of the hyper-parameter window size. Lines of different colors represent character embeddings of different dimensions. As the increase of window size from 3 to 15, character embeddings of different dimensions show different trends, some (dimension<=200) getting higher cosine similarities on the similar-sentence pairs, others with dimension >= 400 showing worse results – decreasing cosine similarity on the similar-sentence pairs. Another striking feature is that low-dimensional (dimension <= 200) embeddings yield significantly higher cosine similarities on the similar-sentence pairs than high-dimensional ones (below 0.86), regardless of the varying window size.

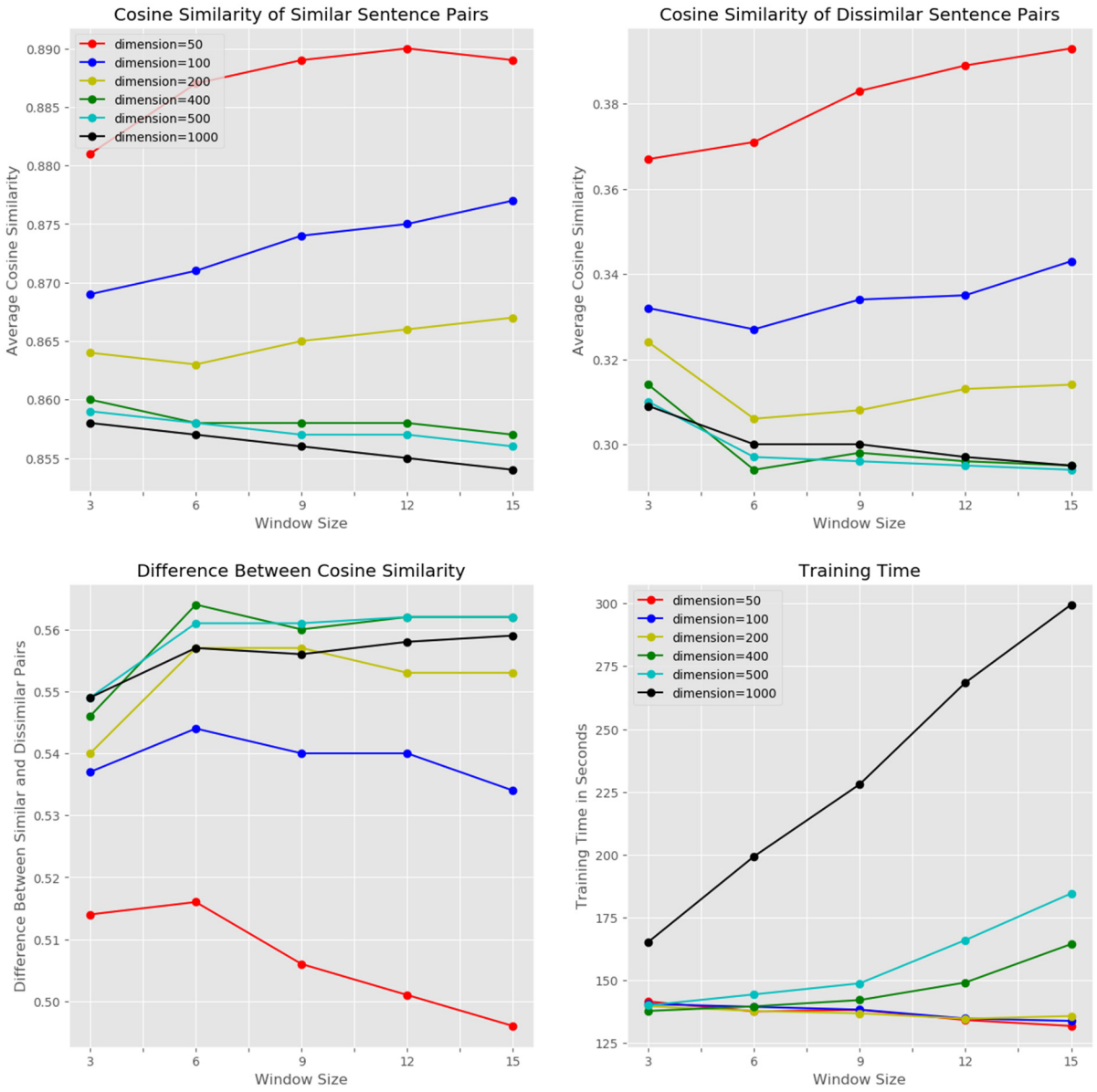

Fig.1 Results from Word2Vec CBOW model.

The top-left plot shows the line graph of the average cosine similarities of similar-sentence pairs. Lines of different colors represent embeddings of different dimensions. The x-axis is the window size during the training period, and y-axis is the average cosine similarity on the similar-sentence pairs. The rest three plots have the same structure as the top-left one, but showing other indices. The top-right plot shows the average cosine similarity computed on the dissimilar-sentence pairs, the bottom-left for differences of averages computed from the two parts of the test data and the bottom-right for the training time under different combinations of hyper-parameters.

On the top-right plot of Fig.1, we discover that low-dimensional embeddings also generate higher cosine similarities on the dissimilar-sentence pairs than their high-dimensional counterparts, with the red line well above 0.36, blue line above 0.32. By contrast, the high-dimensional embeddings produce cosine similarities below 0.30 on the dissimilar-sentence pairs as the window size increases to 6. Combining the findings from the top-left and top-right plots, we conclude that low dimensions like 50 and 100 only capture less of the semantics than high-dimensional ones. That is why the 50 dimensional embeddings (red lines) yield highest cosine similarities on both the similar-sentence pairs

and the dissimilar-sentence pairs. The bottom-left plot presents the difference of average cosine similarities between the semantically similar-sentence pairs and the dissimilar-sentence pairs, which measures the discrimination power of character embeddings. The large difference means that the character embeddings can easily discriminate whether two sentences are semantically similar or not. This discrimination power varies over the embedding dimension and window size and the largest value is 0.564 (from the bottom-left table in Tab. 1), with optimal dimension and window size being 400 and 6, respectively. In other words, on average, the difference of cosine similarity between a semantically similar pair and a dissimilar one is 0.564 by using the best embedding. The bottom-right plot illustrates the training time cost to produce character embeddings with CBOW over different combinations of hyper-parameters. In general, the training time increases with the increment of window size or dimension.

Cosine Similarity of Similar Sentence Pairs

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	0.881	0.887	0.889	0.89	0.889
Dim=100	0.869	0.871	0.874	0.875	0.877
Dim=200	0.864	0.863	0.865	0.866	0.867
Dim=400	0.86	0.858	0.858	0.858	0.857
Dim=500	0.859	0.858	0.857	0.857	0.856
Dim=1000	0.858	0.857	0.856	0.855	0.854

Cosine Similarity of Dissimilar Sentence Pairs

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	0.367	0.371	0.383	0.389	0.393
Dim=100	0.332	0.327	0.334	0.335	0.343
Dim=200	0.324	0.306	0.308	0.313	0.314
Dim=400	0.314	0.294	0.298	0.296	0.295
Dim=500	0.31	0.297	0.296	0.295	0.294
Dim=1000	0.309	0.3	0.3	0.297	0.295

Difference Between Cosine Similarity

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	0.514	0.516	0.506	0.501	0.496
Dim=100	0.537	0.544	0.54	0.54	0.534
Dim=200	0.54	0.557	0.557	0.553	0.553
Dim=400	0.546	0.564	0.56	0.562	0.562
Dim=500	0.549	0.561	0.561	0.562	0.562
Dim=1000	0.549	0.557	0.556	0.558	0.559

Training Time

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	141.553	137.585	138.189	134.138	131.778
Dim=100	140.497	139.415	138.291	134.742	133.804
Dim=200	139.649	137.779	136.818	134.68	135.739
Dim=400	137.768	139.593	142.1	149.099	164.375
Dim=500	140.011	144.345	148.761	165.98	184.541
Dim=1000	165.208	199.277	228.007	268.487	299.53

Tab. 1 Four tables of results from word embeddings produced by Word2Vec model.

These four tables corresponds to four plots in Fig. 1 by position. The top-left table presents the average cosine similarity computed on the similar-sentence pairs over different combinations of dimension and window size.Each row shows the average similarities on the similar-sentence pairs with embeddings of the same dimension and each column represents results from the same window size. The rest three tables share the same structure as the top-left one. The top-right illustrates the average similarity computed on the dissimilar-sentence pairs, the bottom-left for difference between average cosine similarities from two parts of the test data, and the bottom-right for the required training time.

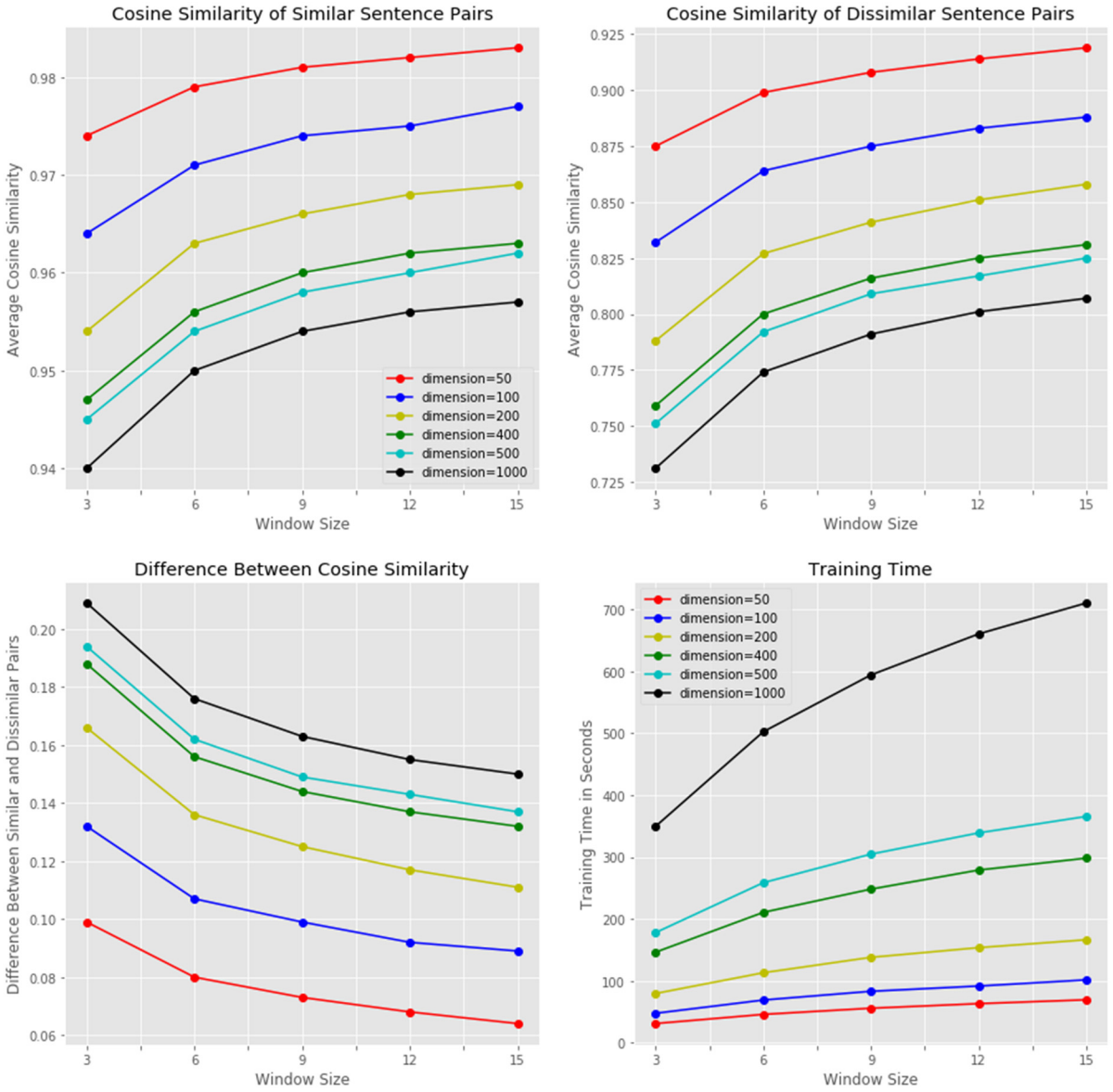

Fig.2 Results from GloVe model. This figure is the same to Fig.1 except the word embedding model.

Please refer to Fig.1 for description.Similar to Fig.1 and Tab.1, Fig.2 and Tab.2 present the results of character embeddings trained with the GloVe model. Plots in Fig.2 corresponds to tables in Tab. 2 by position. From the top-left and right plots, we observe that character embeddings of the same dimension produce increasing cosine similarities on both the similar-sentence and dissimilar-sentence pairs as the rise of window size.However, the bottom-left plot shows that the difference between the average cosine similarities on the similar-sentence and dissimilar-sentence pairs. The largest value of the difference is just over 0.20, which means that, on average, the cosine similarity between a semantically similar pair and a dissimilar one is 0.20. The optimal hyper-parameters to produce this embedding is 1000 and 3 for dimension and window size, respectively. By comparing the training models: Word2Vec CBOW and GloVe, we observe that embeddings from CBOW tends to produce larger differences (>= 0.5) between similar-sentence pairs and dissimilar sentence pairs than GloVe (mostly <= 0.2). This means that word embeddings from CBOW have better discrimination ability than the ones from GloVe and they can distinguish whether a pair of sentences are semantically similar or not.

Cosine Similarity of Similar Sentence Pairs

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	0.974	0.979	0.981	0.982	0.983
Dim=100	0.964	0.971	0.974	0.975	0.977
Dim=200	0.954	0.963	0.966	0.968	0.969
Dim=400	0.947	0.956	0.96	0.962	0.963
Dim=500	0.945	0.954	0.958	0.96	0.962
Dim=1000	0.94	0.95	0.954	0.956	0.957

Cosine Similarity of Dissimilar Sentence Pairs

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	0.875	0.899	0.908	0.914	0.919
Dim=100	0.832	0.864	0.875	0.883	0.888
Dim=200	0.788	0.827	0.841	0.851	0.858
Dim=400	0.759	0.8	0.816	0.825	0.831
Dim=500	0.751	0.792	0.809	0.817	0.825
Dim=1000	0.731	0.774	0.791	0.801	0.807

Difference Between Cosine Similarity

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	0.099	0.08	0.073	0.068	0.064
Dim=100	0.132	0.107	0.099	0.092	0.089
Dim=200	0.166	0.136	0.125	0.117	0.111
Dim=400	0.188	0.156	0.144	0.137	0.132
Dim=500	0.194	0.162	0.149	0.143	0.137
Dim=1000	0.209	0.176	0.163	0.155	0.15

Training Time

Dim./Wind.	Size=3	Size=6	Size=9	Size=12	Size=15
Dim=50	31.148	45.924	55.886	63.31	69.491
Dim=100	47.669	69.103	83.264	91.698	101.71
Dim=200	79.656	113.184	138.096	153.713	166.499
Dim=400	146.327	210.703	248.266	279.039	298.328
Dim=500	177.97	258.605	304.806	339.063	365.682
Dim=1000	349.481	502.409	594.207	660.491	710.47

Tab.2 Four tables of results for word embeddings from GloVe model. This table is the same to Tab.1 except the word embedding model. Please refer to Tab.1 for detailed description.

4. Conclusion and Future Work

Here are some conclusions from our experiments, which trains Chinese character embeddings with Word2Vec CBOW and GloVe models on a medium size (ten of millions tokens) corpus. Character embeddings from CBOW perform better than those from GloVe in the sense that it has stronger discrimination ability, i.e., the word embeddings produced by CBOW are better at distinguishing whether a pair of sentences are semantically similar or not.

In the future, we will take contextualized word embeddings into account and check how much they can improve the sentence matching task. But for now, we have not enough computing resources, so we can only investigate Word2Vec and GloVe models. Also, we will work on scientific and comprehensive approaches to evaluate the quality of word embeddings.

Acknowledgments

This work is financially supported by the Shenzhen Government's Chuang Ke Chuang Ye Foundation (CKCY201705081212). The findings in this research are very useful to the development of chatbots in Zhiyan Technology (Shenzhen) Limited.

References

[1] Kenter, T. and De Rijke, M., 2015, October. Short text similarity with word embeddings. In Proceedings of the 24th ACM international on conference on information and knowledge management (pp. 1411-1420). ACM.

[2] H. Li and J. Xu. Semantic matching in search. Foundations and Trends in Information Retrieval, 7(5):343–469, 2014.

[3] D. Metzler, S. Dumais, and C. Meek. Similarity measures for short segments of text. In ECIR 2007, 2007.

[4] R. M. Aliguliyev. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. Expert Systems with Applications, 2009.
[5] Li, S., Zhang, J., Huang, X., Bai, S. and Liu, Q., 2002. Semantic computation in a Chinese question-answering system. Journal of Computer Science and Technology, 17(6), pp.933-939.
[6] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In AAAI, 2006.
[7] V. Jijkoun and M. de Rijke. Recognizing textual entailment using lexical similarity. In Proceedings Pascal 2005 Textual Entailment Challenge Workshop, 2005.
[8] A. Islam and D. Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. TKDD, 2008.
[9] L. Han, A. Kashyap, T. Finin, J. Mayfield, and J. Weese. UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *SEM-2013, 2013.
[10] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In NIPS 2011, 2011.
[11] D. Bär, C. Biemann, I. Gurevych, and T. Zesch. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In SemEval 2012, 2012.
[12] S. Fernando and M. Stevenson. A semantic similarity approach to paraphrase detection. CLUK 2008, 2008.
[13] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo. sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *SEM 2013, 2013.
[14] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
[15] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6):391–407, 1990.
[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. ArXiv preprint arXiv:1301.3781, 2013.
[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In NIPS 2013, 2013.
[18] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In Innovations in Machine Learning. Springer, 2006.
[19] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. EMNLP 2014, 2014.
[20] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.