Using a Chinese Lexicon to Learn Sense Embeddings and Measure Semantic Similarity

Zhuo Zhen and Yuquan Chen(✉)

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, People's Republic of China
HHee@sjtu.edu.cn, chen-yq@cs.sjtu.edu.cn

Abstract. Word embeddings have recently been widely used to model words in Natural Language Processing (NLP) tasks including semantic similarity measurement. However, word embeddings are not able to capture polysemy, because a polysemous word is represented by a single vector. To address this problem, learning multiple embedding vectors for different senses of a word is necessary and intuitive. We present a novel approach based on a Chinese lexicon to learn sense embeddings. Every sense is represented by a vector that consists of semantic contributions made by senses explaining it. To make full use of the lexicon's advantages and address its drawbacks, we perform representation expansion to make sparse embedding vectors dense and disambiguate in gloss polysemous words by semantic contribution allocation. Thanks to the use of an intuitive way of noise filtering, we achieve noticeable improvement both in dimensionality reduction and semantic similarity measurement. We perform experiments on a translated version of Miller-Charles dataset and report state-of-the-art performance on semantic similarity measurement. We also apply our approach to SemEval-2012 Task4: Evaluating Chinese Word Similarity, which uses a translated version of wordsim353 as the standard dataset, and our approach also noticeably outperforms conventional approaches.

1 Introduction

To date, word embeddings [1] are widely used in NLP tasks including semantic similarity measurement and are proved to be effective. Most word embeddings, such as word2vec [2] and GloVe [3], are based on the distributional model that leverages neural networks to model expected contexts of words. However, word embeddings suffer from two major limitations. The first is that word embeddings are unable to capture polysemy since every word is represented by a single vector. The second is that most word embeddings are based on distributional statistics of corpora, which have no connection to semantic inventories, making it hard to learn reliable and accurate representations for infrequent words.

M. Sun et al. (Eds.): CCL 2018/NLP-NABD 2018, LNAI 11221, pp. 197–208, 2018.
https://doi.org/10.1007/978-3-030-01716-3_17

There are many research works seeking to address these problems by learning sense embeddings. [4–6] decompose word embeddings into multiple prototypes corresponding to distinct meanings of words. The main drawback of these approaches is that they have no connection to sense inventories. Thus they do not know which words are polysemous, how many senses they have. Besides, the mapping from sense embeddings to any sense inventory has to be conducted manually. [7,8] construct synsets using semantic resources like WordNet [9] and Wikipedia first and then leverage them to decompose word embeddings. Word Sense Disambiguation(WSD) algorithms or sense clustering algorithms are adopted to construct synsets. So the unsatisfactory accuracy and coverage of WSD algorithms and sense clustering algorithms become the bottlenecks. Instead of decomposing word embeddings, [10,11] learn sense embeddings from sense annotated corpus. Because there is no proper dataset with manual sense annotation, they utilize WSD algorithms or sense clustering algorithms to generate sense annotated corpus first. So they suffer from the non-optimal WSD algorithms and sense clustering algorithms too. Bilingual resources are leveraged by [12,13] for sense embedding learning. They hold an assumption that the translation of words and their contexts can at least partially be helpful for polysemy resolving in the original language. Establishing a connection to some sense inventory is hard for them too.

With the shortage of proper datasets with manual sense annotation, we realize lexicons have several advantages for sense embedding learning. First, lexicons are redacted by linguistic experts and are semantically authoritative. Besides, we can directly specify polysemous words' senses. What's more, infrequent words and senses are also explained clearly, which would significantly help to improve coverage and accuracy for embedding learning of infrequent senses.

In this article, we propose a novel approach based on a Chinese lexicon to learn sense embeddings. In Chinese, the combination of words usually accompanies with semantic coupling. So we assume words in glosses have semantic contributions to the words explained. Thus a word can be represented by the semantic contributions made by words explaining it. Similarly, a sense can be represented by a vector that consists of semantic contributions made by senses explaining it too. But there are some problems to be addressed if we want to use a lexicon as the only semantic resource. The first is that some glosses are so short that the embedding vectors learned are very sparse. The second is that we can not directly distinguish what a polysemous word in some gloss actually means. The third is that the amount of senses is large making learned embedding vectors high dimensional.

To address the problem caused by short glosses, we perform representation expansion to make sparse embedding vectors dense. To guarantee and accelerate convergence, we use an attenuation parameter to control the process. Then, instead of telling what a polysemous word in some gloss actually means, we assign each of its senses a weight using softmax algorithm and allocate its semantic contributions to its senses, which we call semantic contribution allocation. Instead of matrix calculation, we perform dimensionality reduction in a more intuitive

way. If a sense makes few contributions to other senses, we treat its contributions as noise and filter them out. So vector dimension is reduced during noise filtering. Experimental results report that executing representation expansion and semantic contribution allocation in sequence iteratively improves the performance of similarity measurement rapidly. And a proper threshold for noise filtering is significantly helpful to performance improvement and dimensionality reduction.

Our approach has following advantages. The first is that a lexicon is the only semantic resource. Thus no more manual annotation effort is needed. The second is that it can learn accurate representations for infrequent senses. The third is that there are few parameters to tune with low training cost. The final one is the intuitive way of noise filtering successfully balances dimensionality reduction and performance improvement.

2 Algorithm

As introduced in the previous section, we assume that a word explaining another word in gloss has a semantic contribution to the one explained. So a word can be represented by the semantic contributions made by words explaining it. Similarly, a sense can be represented by a vector that consists of semantic contributions made by senses in sense gloss. Here we use a matrix $\boldsymbol{M} \in R^{N_{\text{sense}} \times N_{\text{sense}}}$ to denote all the semantic contributions, where N_{sense} is the number of sense gloss in the lexicon. And we want to learn an accurate estimation of all the entries in $\boldsymbol{M}$. More specifically, we set $\boldsymbol{M}_{i,j}$ as the semantic contribution to the i_{th} sense made by the j_{th} sense. Then $\boldsymbol{M}^i$, the i_{th} row of the matrix, is a vector consists of semantic contributions to the i_{th} sense made by all senses, which is the representation of the i_{th} sense. If we observe the matrix in another perspective, we can find $\boldsymbol{M}_j$, the j_{th} column of the matrix, consists of contributions made by the j_{th} sense to all senses.

A lexicon has several advantages for sense embedding learning but it also has some drawbacks. The first is that some glosses are so short that sense embedding vectors learned are very sparse. Besides, we can not directly tell what a polysemous word in some gloss actually means. What's more, the large amount of sense makes embedding vectors high dimensional.

To address the problem of sparse embeddings caused by short glosses, we do representation expansion to make sparse vectors dense. And instead of trying to figure out what a polysemous word actually means, we assign each of its senses a weight using softmax algorithm and allocate its semantic contributions to its senses, which we call semantic contribution allocation. It needs to be pointed out that every time $\boldsymbol{M}$ is modified by initialization or representation expansion, semantic contribution allocation should be carried out immediately. This is because changes in vectors induce changes in sense weight assignment. So we try to learn more accurate representations by executing representation expansion and semantic contribution allocation in sequence iteratively. Finally, dimensionality reduction is realized by filtering out the semantic contributions regarded as noise. The overview of the entire algorithm is shown in Algorithm 1.

Algorithm 1. Overview

1: Initialization
2: Semantic contribution allocation
3: **while (** consistency with standard dataset increases**):**
4: Representation expansion
5: Semantic contribution allocation
6: Dimensionality reduction

2.1 Initialization

We initialize $\boldsymbol{M}$ with zeroes at first. Then we scan sense glosses in the lexicon and compute initial semantic contributions made by words. Here we use $wordContribution_{i,k}$ to denote the semantic contribution to the i_{th} sense made by the k_{th} word. It is calculated like tf-idf [14]. Here $tf(sense_i, word_k)$ denotes the term frequency of the k_{th} word in the i_{th} sense's gloss and $ef(word_k)$ denotes the number of glosses that the k_{th} word appears in.

$$wordContribution_{i,k} = (1 + \log tf(sense_i, word_k)) \times \log \frac{N_{\text{sense}}}{ef(word_k)} \quad (1)$$

According to the principle of maximum entropy, we allocate a polysemous word's semantic contributions to its senses equally. Here N_{word_k} is the number of senses of the k_{th} word. For simplicity and without loss of generality, we suppose $sense_j \in word_k$, which means the j_{th} sense is from the k_{th} word.

$$\boldsymbol{M}_{i,j} = \frac{wordContribution_{i,k}}{N_{word_k}} \quad (2)$$

2.2 Semantic Contribution Allocation

Since we can not directly tell what a polysemous word in some gloss actually means, we choose to allocate its semantic contributions to its senses. Then what we should consider is weight assignment. We assume that if one sense is semantically closer to the explained sense, it deserves higher weight. Here we use $weight(i, j)$ to denote the weight of the j_{th} sense explaining the i_{th} sense. We try to measure the semantic distance in two different ways. $distanceE(i, j)$ is the Euclidean distance between the i_{th} sense and j_{th} sense, i.e. $\boldsymbol{M}^i$ and $\boldsymbol{M}^j$. And $distanceC(i, j)$ is the cosine of the angle of $\boldsymbol{M}^i$ and $\boldsymbol{M}^j$. For simplicity and without loss of generality, we suppose $sense_j \in word_k$.

When using $distanceE(i, j)$ to measure the semantic distance, we assign weights in the following way. To make sure the weights lie in a reasonable interval, we use μ to control the process.

$$weight(i, j) = \frac{e^{-\frac{distanceE(i,j)^2}{\mu}}}{\sum_{sense_l \in word_k} e^{-\frac{distanceE(i,l)^2}{\mu}}} \quad (3)$$

When using $distanceC(i,j)$ to measure distance, the weights are assigned in a different way. And we use ϵ for smoothing.

$$weight(i,j) = \frac{e^{distanceC(i,j)} - 1 + \epsilon}{\sum_{sense_l \in word_k}(e^{distanceC(i,l)} - 1 + \epsilon)} \tag{4}$$

After the weights are determined, senses' semantic contributions are updated in the following way.

$$\boldsymbol{M}_{i,j}^{\text{new}} = (\sum_{sense_l \in word_k} \boldsymbol{M}_{i,l}) \times weight(i,j) \tag{5}$$

2.3 Representation Expansion

Suppose we have learned embedding vectors for all senses in the lexicon but they are sparse. How can we make them dense? Since every sense is represented by a vector that consists of semantic contributions, we can expand a sense's sparse embedding vector by adding a weighted sum of learned embedding vectors. An intuitive explanation is that if sense B has a direct semantic contribution to sense A, then senses that have direct semantic contributions to sense B have indirect contributions to sense A. And it can be implemented by adding embedding vector of sense B with a weight to the one of sense A.

Intuitively, a sense's semantic contribution plays a key role in representation expansion. For example, suppose the embedding vector of sense A is sparse, and the entries in it corresponding to sense B and sense C are positive, which means sense B and sense C have semantic contributions to sense A. If sense B has a larger semantic contribution than sense C, the vector of sense B deserves a larger weight than the one of sense C in the expansion.

Besides, we assume that a sense's embedding vector is not semantically equal to the sense itself. While we are expanding representations, we are losing semantic accuracy. And the more times we do representation expansion, the more accuracy we lose. Here n denotes how many times we do representation expansion and η is the attenuation parameter to denote accuracy loss. We express their relationship in the following way, while α is the initial value we need to figure out.

$$\eta(n) = \alpha^n \tag{6}$$

Then the expansion of a sparse embedding vector is realized by adding all senses' embedding vectors multiplied by corresponding semantic contributions and η. The process can be described as follows, where $\boldsymbol{M}_{\text{new}}^{i}$ is the updated representation of the i_{th} sense.

$$\boldsymbol{M}_{\text{new}}^{i} = \boldsymbol{M}^{i} + \eta \times \sum_{j=1}^{N_{\text{sense}}} \boldsymbol{M}_{i,j} \cdot \boldsymbol{M}^{j} \tag{7}$$

After integration, we can directly update $\boldsymbol{M}$,where $\boldsymbol{M}_{\text{new}}$ denotes the updated $\boldsymbol{M}$.

$$\boldsymbol{M}_{\text{new}} = \boldsymbol{M} + \eta \times \boldsymbol{M}^2 \tag{8}$$

2.4 Dimensionality Reduction

As illustrated above, $\boldsymbol{M}_j$ is the j_{th} column of $\boldsymbol{M}$ and consists of the semantic contributions to all senses made by the j_{th} sense. Here we use $contribution(i)$ to denote the sum of semantic contributions made by the i_{th} sense.

$$contribution(i) = \sum_{j=1}^{N_{\text{sense}}} \boldsymbol{M}_{j,i} \tag{9}$$

We assume if $contribution(i)$ is very small, it is very likely to be noise and has a negative effect on semantic similarity measurement. More specifically, if $contribution(i)$ is small, the i_{th} column would be removed from $\boldsymbol{M}$. Thus we set a threshold to filter noise out, which at the same time reduces the dimension of embedding vectors.

3 Experiment and Evaluation

In this section, we first perform experiments on a translated version of Mill-Charles dataset [15]. With attenuation parameter set to 0.7, which is randomly chosen, we try to figure out whether a stop word list is helpful to semantic similarity measurement and using which distance metric is better for semantic contribution allocation. Then, we try to find out the relationship between the attenuation parameter and the correlation coefficient with the dataset. With an ideal attenuation parameter, we try noise filtering, which is also a way of dimensionality reduction, and we want to find a threshold that balances dimensionality reduction and semantic similarity measurement performance. After all these are finished, we can determine attenuation parameter for representation expansion, distance metric for semantic contribution allocation and the threshold for dimensionality reduction. Then, we compare with some conventional approaches. Finally, we apply our approach to SemEval-2012 Task4, Evaluating Chinese Word Similarity, for another evaluation.

3.1 Preparation

We use Modern Chinese Dictionary [16] as our corpus. Before applying it to sense embedding learning, some preprocessing is necessary. First, we filter out meaningless symbols like phonetic notations. Then we filter out some sentences that redirect pages. Finally, we filter out some descriptions about pragmatics.

The cosine of the angle of two embeddings is computed as semantic similarity. When we compute the semantic similarity between two words, especially polysemous words, we choose one sense from each word and try all possible combinations. For example, the similarity between the m_{th} word and the n_{th} word, $wordSimilarity(m, n)$, is computed as follows.

$$wordSimilarity(m, n) = \max_{sense_i \in word_m, sense_j \in word_n} \frac{\boldsymbol{M}^i \cdot \boldsymbol{M}^j}{||\boldsymbol{M}^i|| \times ||\boldsymbol{M}^j||} \tag{10}$$

We first perform experiments on a dataset based on Miller and Charles dataset [17]. Because the original dataset is in English, we choose a translated version [15]. Some words that are not written as an entry in the lexicon are removed. The correlation between the dataset and our result is evaluated by Pearson correlation coefficient.

We then switch to the standard dataset of SemEval-2012 Task4: Evaluating Chinese Word Similarity, which is a translated version of wordsim353 [18]. And the task uses Kendall correlation coefficient as the evaluation metric.

3.2 Experiment and Result

Whether to Use a Stop Word List and Which Distance Metric Is Better. Setting attenuation parameter to 0.7, we perform semantic contribution allocation based on the two distance metrics mentioned above, with or without a stop word list. As is shown in Table 1, results show us that an improper stop word list really does harm to semantic similarity measurement. And using cosine to allocate semantic contribution is better. To achieve the best performance, we should allocate semantic contribution using cosine as distance metric without a stop word list.

Table 1. Correlation under different conditions

With Stop Word List or Not	Yes	Not
Euclidean	0.7690	0.8170
Cosine	0.8005	0.8227

Relationship Between Attenuation Parameter and Correlation. Without a stop word list, we try to figure out the relationship between the attenuation parameter and correlation with the dataset. Allocating semantic contribution based on cosine, we get the curve shown in Fig. 1. Bigger attenuation parameter brings better performance. But as the attenuation parameter increases, the curve tends to be flat. So if we go on increasing attenuation parameter, the correlation would only increase slightly. To achieve the best performance, we fix the attenuation parameter to 0.9 for following experiments and evaluations.

Dimensionality Reduction Threshold's Effect on Correlation. As illustrated in previous sections, we treat a column of $\boldsymbol{M}$ as the semantic contributions made by corresponding sense. And we assume if a sense makes very few semantic contributions, its contributions are very likely to be noise and should be removed. It may be helpful to both dimensionality reduction and performance improvement. Then we need to figure out a proper threshold. In this part, we try different thresholds and want to figure out the relationship between threshold

and correlation. And before dimensionality reduction, the highest correlation we achieve is 82.55%.

First, we set the threshold small and increase it slowly. We find that correlation coefficient fluctuates in a very narrow interval and there is no significant change. The result is shown in Fig. 2. Then we keep on increasing the threshold. And we find that although the correlation coefficient is fluctuating, it is obviously higher than the former best result. As is shown in Fig. 3, when the threshold is set to 11000, the correlation coefficient reaches its peak to 90.52%. Keeping on increasing the threshold, we find that larger threshold makes larger information loss. And the correlation declines rapidly, as is shown in Fig. 4.

Setting the threshold to 11000, we want to check out how many dimensions are filtered out. Result reports that 89.66% of dimensions are ignored. So the intuitive noise filtering method filters out nearly 90% of features and brings nearly 8% performance improvement from 82.55% to 90.52%.

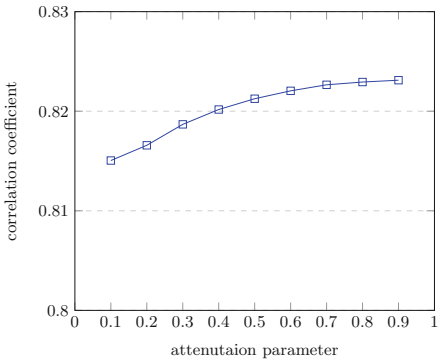

Fig. 1. When the attenuation parameter increases, the correlation coefficient increases but the curve tends to be flat.

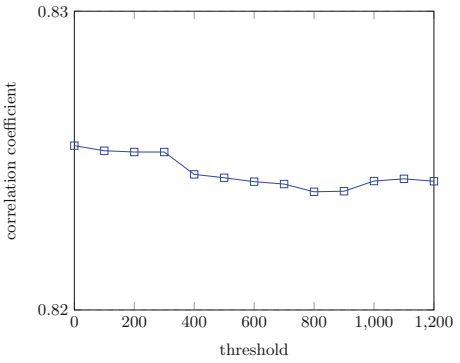

Fig. 2. When a small threshold increases slowly, the correlation coefficient fluctuates in a very narrow interval. This means a small threshold has no significant effect on correlation coefficient.

3.3 Evaluation

Evaluation on Translated Miller-Charles Dataset. With the attenuation parameter set to 0.9, we use cosine to allocate semantic contribution and do not use a stop word list. After dimensionality reduction, the Person correlation coefficient we get is 0.9052. We compared this result with some conventional approaches. *How* leverages the taxonomy and attribute knowledge from a Chinese knowledge base, HowNet, which is similar to WordNet, to calculate semantic similarity [19]. *Dc* utilizes snippets of query results from Google for similarity measurement. If a word appears in the snippets of the query of another word and

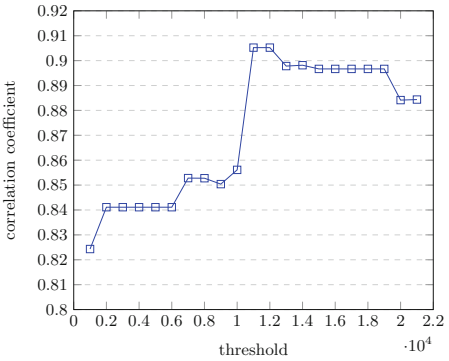

Fig. 3. Using a moderate threshold for dimensionality reduction, the correlation coefficient obviously exceeds the former best one.

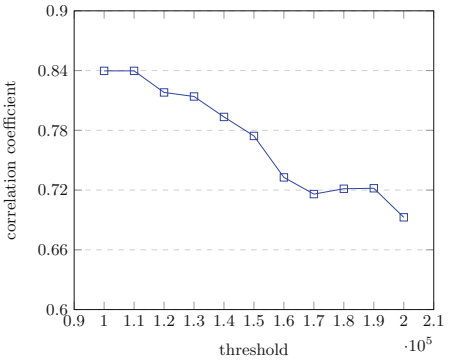

Fig. 4. When a large threshold increases, the correlation coefficient declines rapidly. This is because the threshold is so large that it induces serious information loss.

vice versa, these two words are semantically similar [20]. *Attr* extracts attribute sets from the Internet and measures semantic similarity by measuring the overlapping extent of two words' attribute sets [15]. *word2vec* is the famous word embedding algorithm and we leverage its Skip-Gram model for this evaluation [2]. *Dic* learns word embeddings using Modern Chinese Standardized Dictionary as the corpus [21]. *Multi-sense* relies on both monolingual and bilingual information to learn sense embeddings. This model consists of an encoder to choose senses for given words, and a decoder to predict expected context words for chosen senses. When the autoencoder's training is finished, sense embeddings are learned [12]. *We* is our result. As is shown in Table 2, our approach outperforms others.

Table 2. Comparison with some conventional approaches on translated M&C dataset

Approach	Dc	How	Attr	word2vec	Dic	Multi-sense	We
Pearson	0.5027	0.6394	0.7169	0.7876	0.7922	0.8049	**0.9052**

Evaluation on Translated WordSim353 Dataset. After experiments on the M&C dataset, we carry out an evaluation on a translated version of wordsim353 [18], which is the standard for SemEval-2012 Task4: Evaluating Chinese Word Similarity [22]. *We* is our novel approach. *word2vec* represents word embeddings learned by word2vec with Skip-Gram model [2]. *Multi-sense* learns sense embeddings by an autoencoder based on both monolingual and bilingual information [12]. *MIXCC*, *MIXCD*, *GUO-ngram and* GUO-words are systems that participated in the task [22]. As is shown in Table 3, our novel approach outperforms other approaches.

Table 3. Comparison with some conventional approaches on translated wordsim353 dataset

Approach	Guo-words	Guo-ngram	MIXCD	MIXCC	Multi-sense	word2vec	We
Kendall	-0.011	0.007	0.040	0.050	0.130	0.295	**0.305**

4 Related Work

Decomposing word embeddings into multiple prototypes corresponding to distinct meanings of words is conducted by [4–6]. These methods have no connection to sense inventories. Thus they can not specify which words are polysemous and how many senses they have. And the mapping from sense embeddings to some sense inventory has to be carried out manually. [7] tries to combine word embeddings and synsets constructed from WordNet [9] to learn more accurate sense embeddings. [8] uses WordNet and Wikipedia as semantic resources. On the one hand, it extracts contextual information to learn word-based representations for words, on the other hand, it constructs synsets to learn synset-based representations for senses. Both these two kinds of representations are utilized to measure semantic similarity. [23] learns word embeddings first and initializes a polysemous word's sense embeddings by averaging semantically similar words in sense glosses. Then it uses two WSD algorithms to obtain senses' relevant occurrences that are used to learn more accurate sense embeddings. There are still some researchers using the sense-annotated corpus to exploit semantic knowledge. But they have to generate a sense-annotated corpus first. [10] uses a WSD system to annotate corpus while [11] chooses a sense clustering algorithm, i.e. k-means, for polysemous word annotation. However, the non-optimal WSD techniques and sense clustering algorithms seriously limit the annotation accuracy, which makes it hard to learn accurate and high-coverage sense embeddings. Bilingual resources are also leveraged with the assumption that "polysemy in one language can be at least partially resolved by looking at the translation of the word and its context in another language" [12,13]. [24] proposes a probabilistic model for sense embedding. The model takes the dependency between sense choices of neighboring words into account. Then it uses an algorithm similar to hard-EM to optimize a max-margin objective. It uses a knowledge base to help model training, but the knowledge base is only used to determine the numbers of polysemous words' senses, which makes it hard to map from sense embeddings to sense inventories.

5 Conclusion

In this article, we present a novel method that learns sense embeddings using a Chinese lexicon as the only corpus. Using a lexicon to learn sense embeddings has several advantages. A lexicon is redacted by linguistic experts and is semantically authoritative. It specifies polysemous words' senses and is significantly helpful to learn embeddings for infrequent senses. But it also has some drawbacks. Short

glosses tend to make sense embeddings vectors sparse, polysemous words in sense glosses induce semantic ambiguity and the sense embedding vectors are high dimensional before dimensionality reduction. First, to address the problem caused by short glosses, we conduct representation expansion to make sparse vectors dense and use an attenuation parameter to accelerate convergence. Then, instead of determining what a polysemous word in some gloss actually means, we assign each of its senses a possibility by softmax algorithm. Finally, if a sense makes few semantic contributions to other senses, we treat its contributions as noise and filter them out. Filtering out noise by a proper threshold is significantly helpful to dimensionality reduction and semantic similarity measurement.

Our novel method has the following advantages. A Chinese lexicon is all we need and no annotation effort is needed. The coverage and accuracy of infrequent sense embeddings are guaranteed. Few parameters need to be tuned and the learning cost is low. The intuitive and efficient way of noise filtering is remarkably helpful to both dimensionality reduction and semantic similarity measurement.

We carry out an evaluation on a translated version of Miller-Charles dataset and the Pearson correlation coefficient is 0.9052, which is state-of-the-art. We also apply our method to SemEval-2012 Task 4: Evaluating Chinese Word Similarity, and report noticeably better results compared with conventional approaches.

References

1. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**(Feb), 1137–1155 (2003)
2. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
3. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
4. Reisinger, J., Mooney, R.J.: Multi-prototype vector-space models of word meaning. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 109–117. Association for Computational Linguistics (2010)
5. 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, vol. 1, pp. 873–882. Association for Computational Linguistics (2012)
6. Tian, F., et al.: A probabilistic model for learning multi-prototype word embeddings. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 151–160 (2014)
7. Pilehvar, M.T., Collier, N.: De-conflated semantic representations. arXiv preprint arXiv:1608.01961 (2016)
8. Camacho-Collados, J., Pilehvar, M.T., Navigli, R.: Nasari: a novel approach to a semantically-aware representation of items. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 567–577 (2015)

9. Miller, G.A.: Wordnet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
10. Iacobacci, I., Pilehvar, M.T., Navigli, R.: Sensembed: learning sense embeddings for word and relational similarity. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), vol. 1, pp. 95–105 (2015)
11. Zhou, H., Jia, C., Yang, Y., Ning, S., Lin, Y., Huang, D.: Combining large-scale unlabeled corpus and lexicon for Chinese polysemous word similarity computation. In: Wen, J., Nie, J., Ruan, T., Liu, Y., Qian, T. (eds.) CCIR 2017. LNCS, vol. 10390, pp. 198–210. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68699-8_16
12. Šuster, S., Titov, I., van Noord, G.: Bilingual learning of multi-sense embeddings with discrete autoencoders. arXiv preprint arXiv:1603.09128 (2016)
13. Guo, J., Che, W., Wang, H., Liu, T.: Learning sense-specific word embeddings by exploiting bilingual resources. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 497–507 (2014)
14. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval, vol. 463. ACM Press, New York (1999)
15. Zhao, J., Liu, H., Lu, R.: Attribute-base computing of word similarity. In: The 11th China Conference on Machine Learning (2008)
16. Lv, S., Ding, S.: Chinese Modern Dictionary. The Commercial Press, Beijing (2005)
17. Miller, G.A., Charles, W.G.: Contextual correlates of semantic similarity. Lang. Cogn. Process. **6**(1), 1–28 (1991)
18. Evgeniy, L.F., Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G.: Placing search in context: the concept revisited. ACM Trans. Inf. Syst. **20**(1), 116–131 (2002)
19. Liu, Q.: Word similarity computing based on hownet. Comput. Linguist. Chin. Lang. Process. **7**(2), 59–76 (2002)
20. Chen, H.H., Lin, M.S., Wei, Y.C.: Novel association measures using web search with double checking. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, pp. 1009–1016. Association for Computational Linguistics (2006)
21. Liu, H., Zhao, J., Lu, R.: Computing semantic similarities based on machine-readable dictionaries. In: IEEE International Workshop on Semantic Computing and Systems, WSCS 2008, pp. 8–14. IEEE (2008)
22. Jin, P., Wu, Y.: Semeval-2012 task 4: evaluating Chinese word similarity. In: Joint Conference on Lexical and Computational Semantics, pp. 374–377 (2012)
23. Chen, X., Liu, Z., Sun, M.: A unified model for word sense representation and disambiguation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1025–1035 (2014)
24. Qiu, L., Tu, K., Yu, Y.: Context-dependent sense embedding. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 183–191 (2016)