Combining Word Embedding and Semantic Lexicon for Chinese Word Similarity Computation

Jiahuan Pei[1], Cong Zhang[1], Degen Huang[1(✉)], and Jianjun Ma[2]

[1] School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, Liaoning, China
{p_sunrise,cccaaag}@mail.dlut.edu.cn, huangdg@dlut.edu.cn
[2] School of Foreign Languages, Dalian University of Technology, Dalian 116024, Liaoning, China
majian@dlut.edu.cn

Abstract. Large corpus-based embedding methods have received increasing attention for their flexibility and effectiveness in many NLP tasks including Word Similarity (WS). However, these approaches rely on high-quality corpora and neglect the human's intelligence contained in semantic resources such as Tongyici Cilin and Hownet. This paper proposes a novel framework for measuring the Chinese word similarity by combining word embedding and Tongyici Cilin. We also utilize retrieval techniques to extend the contexts of word pairs and calculate the similarity scores to weakly supervise the selection of a better result. In the Chinese Lexical Similarity Computation (CLSC) shared task, we rank No. 2 with the result of 0.457/0.455 of Spearman/Pearson rank correlation coefficient. After the submission, we boost the embedding model by merging an English model into the Chinese one and learning the co-occurrence sequence via LSTM networks. Our final results are 0.541/0.514, which outperform the state-of-the-art performance to the best of our knowledge.

Keywords: Chinese word similarity · Word embedding · Semantic lexicon · LSTM networks

1 Introduction

Word similarity is a task of measuring the lexical similarity degree between word pairs, which has attracted much attention as a fundamental research in many NLP tasks. To date, numerous approaches have been proposed for computing lexical similarity, which can be briefly categorized into thesaurus-based methods [1], traditional corpus-based methods [2] and corpus-based embedding methods [3–6].

Typically, thesaurus-based strategies mainly rely on manual semantic resources and define the degree of similarities based on the distance between

C.-Y. Lin et al. (Eds.): NLPCC-ICCPOL 2016, LNAI 10102, pp. 766–777, 2016.
DOI: 10.1007/978-3-319-50496-4_69

the two items in the structural semantic thesaurus, including Tongyici Cilin [1,7], Hownet [8,9] and Wordnet [10,11]. These semantic measures, while interpretable and effective, have the disadvantage that the computation only affects the pairs when both members are presented in the lexicons. Therefore, corpus-based methods tend to be more attractive to predict unknown words, especially the hot embedding approaches, such as skip-gram model [3–5] and GloVe model [6], which use the degree of replaceability between words to measure the similarity and prevent the learning process from *Data Sparsity* problem in the traditional corpus-based means.

However, limited to the *distributional hypothesis*, which assumes that similar words occur in similar context [12], basic embedding methods generally have three drawbacks in nature. Firstly, they can hardly distinguish *semantic similarity* from *conceptual association* [13]. For instance, the words in the pair (残疾/disability, 死亡/death, score = 2.8) may be regarded as similar by mistake because they can both occur in the X position of contexts like "An illness resulted in his X". Second, solely embedding technique cannot capture the differences between synonyms and antonyms [22]. For example, the words in (积极/positive, 消极/negative, score = 4.1) have a cosine similarity of approximately 0.76 in the *pre-train word2vec model* released by Mikolov et al. [3]. At last, context-dependent embedding methods are unable to differentiate distinct senses of a word [14]. One example of polysemy phenomenon is the pair (包袱/baggage; jokes in the crosstalk, 段子/joke, score = 2.6). To be specific, if the word "包袱" is assigned as the most direct meaning of "baggage", the two words are most probably unsimilar, but the other meaning "jokes in the crosstalk" draws the distance closer. In addition, for Chinese language, the challenge is also due to the lack of contexts for the single-character word in such pair as (面/face; noddles, 首/head, score = 4.7), because it is a rare utterance in general texts. Therefore, the drawbacks arouse people's recent interest in integrating lexicons into word embeddings to capture multiple semantics [15–19,23].

In this paper, we present a framework that combines word embedding and semantic lexicon for Chinese word similarity computation by simple but meaningful linear combination, which initially comes from the basic question that how does a person evaluate the similarity between a pair of words: He may search words in his knowledge base and compute a similarity score based on the distance between the two lexical items. Simultaneously, he will retrieve the words in search engines to find out the sentences containing these words and then estimate the score via the similarity of the contexts. At last, he may give the final result after balancing the previous two results. To demonstrate the performance, we participated in the Chinese Lexical Similarity Computation (CLSC) shared task [20], which provides a benchmark dataset to evaluate and compare different lexical similarity methods.

2 Methodology

Figure 1 briefly illustrates the general architecture of our Chinese word similarity computation system.

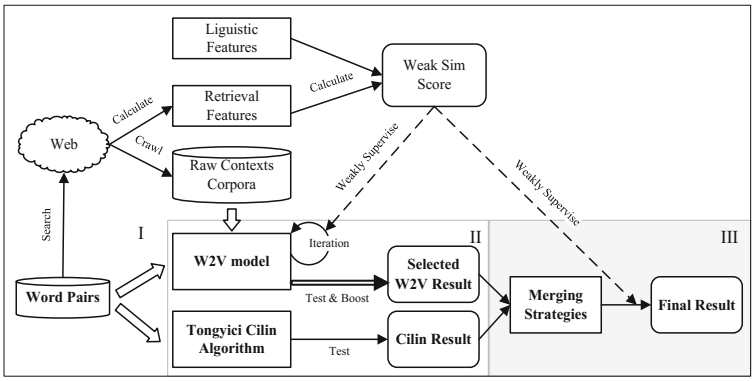

Fig. 1. The general architecture of Chinese word similarity computation

2.1 Similarity Computation Based on Tongyici Cilin

The *Cilin* model utilized in this paper is developed from the algorithm proposed in reference [1], which fully exploits the coding and structure information to estimate both similarity and relevance between the words.

The cilin dictionary is organized by a 5-layer hierarchical structure. Correspondingly, it supplies 5-layer patterns to generate coding for a group of words, which are joined by relationships like "synonym" or "relevance". For instance, the words in the pair (紫禁城/the Forbidden City, 故宫/the Imperial Palace, score = 10) are coded in the items "Bn23A03# 正殿...金銮殿 紫禁城" and "Bn23A02# 行宫 东宫...爱丽舍宫 故宫" respectively. Figure 2 shows the two example words represented in cilin's hierarchical structure.

Given two words w_a and w_b, let set $C_a = \{c_a^1, c_a^2, \ldots, c_a^A\}$ and $C_b = \{c_b^1, c_b^2, \ldots, c_b^B\}$ be concepts of w_a and w_b respectively. Set $\Lambda = \{\lambda_1, \lambda_2 \ldots, \lambda_6\}$ = {0.65, 0.8, 0.9, 0.96, 0.5, 0.1} denotes the parameters in the computation. The similarity between w_a and w_b can be computed as maximum value of every "sense" in set C_a and C_b respectively, which is denoted as:

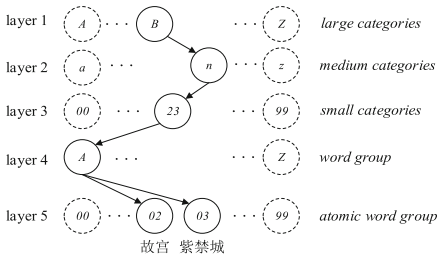

Fig. 2. The words "紫禁城" and "故宫" in the 5-layer hierarchical structure

$$SIM_{cilin}(w_a, w_b) = \max_{1 \leqslant i \leqslant A; 1 \leqslant j \leqslant B} \{sim(c_a^i, c_b^j)\} \tag{1}$$

where function $sim(c_a^i, c_b^j)$ defines the concept similarity between sense c_a^i and c_b^j as:

$$sim(c_a^i, c_b^j) = \begin{cases} 1.0, \ if \ code(c_a^i) = code(c_b^j) \ \& \ end \ with \ \text{“=”} \\ \lambda_5, \ elif \ code(c_a^i) = code(c_b^j) \ \& \ end \ with \ \text{“\#”} \\ \lambda_6, \ elif \ c_a^i, c_b^j \ not \ in \ the \ same \ tree \\ \lambda_{l-1} \times cos(n_{l-1} \times \frac{\pi}{180}) \times (\frac{n_{l-1}-k+1}{n_{l-1}}), \ else \end{cases} \tag{2}$$

where $l = 2, 3, 4, 5$ and it represents the number of layer in a hierarchical tree, n_{l-1} denotes the number of nodes in the branch layer l, k is the distance between two branches.

SIM_{cilin} is represented as a real number in domain **[0,1]**. Therefore, we transform original domain **[0,1]** into **[1,10]** via a simple linear function $f(x) = 9x + 1$.

2.2 Similarity Computation Based on Embedding Vectors

We introduce a general approach for improving word embeddings by weakly supervising the learning process with the *weak similarity score* (WSS), which is generated from some similarity-related retrieval statistics and linguistic features. We begin with reviewing the basic skip-gram model and then present our boosting method.

The Skip-Gram Model. The skip-gram model is a learning framework to learn continuous word vectors from text corpora [3,4]. It maps each word in the vocabulary into a continuous vector space by the method of looking up the embedding matrix $W^{(1)}$. $W^{(1)}$ is learned through maximizing the prediction probability of its neighbouring words within a context window, and the prediction probability is calculated using another embedding matrix $W^{(2)}$. For a sequence of training data: $w_1, w_2, \ldots, w_N$, this model aims at maximizing the following objective function:

$$Q = \frac{1}{N} \sum_{n=1}^{N} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{n+j} | w_n) \tag{3}$$

where N represents the number of words, c is the size of context windows, w_n denotes the input central word and w_{n+j} stands for its neighbouring word and the conditional probability $p(w_{n+j}|w_n)$ is defined as:

$$p(w_{n+j} | w_n) = \frac{\exp(\mathbf{w}_{n+j}^{(2)} \cdot \mathbf{w}_t^{(1)})}{\sum_{k=1}^{V} \exp(\mathbf{w}_k^{(2)} \cdot \mathbf{w}_t^{(1)})} \tag{4}$$

where $\mathbf{w}_n^{(1)}$ and $\mathbf{w}_k^{(2)}$ denote row vectors in matrices $W^{(1)}$ and $W^{(2)}$, corresponding to word w_n and w_k respectively.

Our Improved Model. The basic unsupervised skip-gram model can produce impressive results depending on the large contexts corpora. However, unsupervised learning may not be suitable for the task of interest as Yu and Dredze [21] stated. Therefore, they incorporate prior knowledge by supervised learning. But supervised learning highly relies on tagged resources. To avoid this limitation, we use weak supervising method in this paper, by which the automatically computed WSS can reflect the similarity between the word pair to some degree with the hypothesis – Not only the contexts of the words, but also similarity-related retrieval statistics and linguistic features can reflect the degree of the word similarity. So far, our objective function can be denoted as:

$$J = \underset{1 \leqslant iter \leqslant I}{max} \{\Phi_{iter}(\overrightarrow{S_{pred}}, \overrightarrow{S_{wss}})\} \tag{5}$$

where $\overrightarrow{S_{pred}}$ is the sequence of prediction similarity scores of the word pairs, $\overrightarrow{S_{wss}}$ is the sequence of WSS scores, Φ is a task specific function like Spearman or Pearson index detailed in Sect. 4.1, and I is the maximum number of iterations. In each iteration, a new W2V model is trained and evaluated by the function J. And the model with highest performance is selected to be used in the merging stage.

As cosine similarity, which is used to measure the similarity degree in W2V, between vectors ranges from -1 to 1, we transform original domain **[0, 1]** into **[1, 10]** via a simple function $g(x)$ as:

$$g(x) = \begin{cases} 1, & x \leqslant 0 \\ 9x + 1, & x > 0 \end{cases} \tag{6}$$

Computation of WSS. We use 4 kinds of web features [24], including web-jaccard, web-overlap, web-dice, web-pmi and 3 kinds of linguistic features, pinyin-similarity, sequence-similarity and pattern-similarity to compute the value of WSS.

We use the notation P and Q to denote the 2 words in a word pair, $H(P)$ to denote the result counts for the query P in a search engine, $P \cap Q$ to denote the conjunction query P and Q. The retrieval statistics web-jaccard, web-overlap, web-dice and web-pmi can be defined as Eqs. 7–10:

$$web - jaccard(P \cap Q) = \begin{cases} 0, & H(P \cap Q) < c \\ \frac{H(P \cap Q)}{H(P) + H(Q) - H(P \cap Q)}, & H(P \cap Q) \geq c \end{cases} \tag{7}$$

$$web - overlap(P \cap Q) = \begin{cases} 0, & H(P \cap Q) < c \\ \frac{H(P \cap Q)}{min(H(P), H(Q)}, & H(P \cap Q) \geq c \end{cases} \tag{8}$$

$$web - dice(P \cap Q) = \begin{cases} 0, & 2H(P \cap Q) < c \\ \frac{H(P \cap Q)}{H(P) + H(Q)}, & H(P \cap Q) \geq c \end{cases} \tag{9}$$

$$web-pmi(P\cap Q)=\begin{cases}0, & H(P\cap Q)<c\\ \log(\frac{\frac{H(P\cap Q)}{N}}{\frac{H(p)}{N}\frac{H(Q)}{N}}, & H(P\cap Q)\geq c\end{cases} \tag{10}$$

where N stands for the number of documents indexed by the search engine. In the present work, we set $N=10^{16}$.

Pinyin similarity is a feature which measures the similarity between Pinyin representations of 2 words, and it is defined as:

$$S_{py}=\frac{2n_{ps}}{L_{pp}+L_{pq}} \tag{11}$$

where n_{ps} stands for the count of same character combinations in the representations of P and Q in Pinyin. L_{pp} and L_{pq} denote the length of the 2 Pinyin sequence. For example, the Pinyin representations of Chinese words (必须, 必需, score = 7.3) are (bi xu, bi xu), so the pinyin similarity between them is $(2*2)/(2+2)=1.0$, which infers a high similarity in terms of pronunciation.

Sequence-similarity is a feature which measures the similarity between 2 Chinese words in the inspect of Chinese character sequences, and it is defined as:

$$S_s=\frac{2n_{sa}}{L_p+L_q} \tag{12}$$

where n_{sa} stands for the count of same Chinese character at the same relative position in P and Q. L_p and L_q denote the length of the 2 words. For instance, the sequence similarity between "阿拉伯" and "阿拉伯人" is $(2*3)/(3+4)=0.857$.

Pattern-similarity is a feature which measures the similarity between 2 Chinese words in the aspect of similar Chinese characters, and it is defined as:

$$S_{pa}=\frac{2n_{si}}{L_p+L_q} \tag{13}$$

where n_{si} stands for the count of similar Chinese character in P and Q. Similar characters are judged by a similar-characters-dictionary. L_p and L_q denote the length of the 2 words. Taking the words "生命" and "性命" as an example, the Chinese character "生" and "性" are similar to each other, so the pattern similarity is $(2*2)/(2+2)=1.0$, which indicates a high similarity from the perspective of word types.

As all the 7 features range in domain **[0, 1]**, we map each of them into **[1, 10]** and compute their weighted average as the weak similarity score.

2.3 Combination Strategies

We utilize 6 strategies to merge the best results of Cilin and W2V model for the submission to CLSC task. For each similarity score in the two results, we calculate a merged score according to the merging strategy. We use the notation S_c and S_v to denote the scores in the two results respectively, and S_m to denote the merged score. The 6 merging strategies are defined as:

– Max:

$$S_m = max\{S_c, S_v\} \tag{14}$$

– Min:

$$S_m = min\{S_c, S_v\} \tag{15}$$

– Replace 1:

$$S_m = \begin{cases} S_c, & S_c \neq 1 \\ S_v, & S_c = 1 \end{cases} \tag{16}$$

– Replace 1 and 10:

$$S_m = \begin{cases} S_c, & S_c \neq 1, S_c \neq 10 \\ S_v, & S_c = 1 \ or \ S_c = 10 \end{cases} \tag{17}$$

– Arithmetic Mean:

$$S_m = \frac{S_c + S_v}{2} \tag{18}$$

– Geometric Mean:

$$S_m = \sqrt{S_c * S_v} \tag{19}$$

2.4 A-Posteriori Improvements

Boosting Embedding Model by Machine Translation. The *pre-train word2vec model*, which is trained on part of Google News dataset with 300-dimensional vectors for 3 million words and phrases, is adopted to improve our W2V model. Firstly, the Chinese words are translated into English words or phrases via Google Translation. Then, spelling checking and length filtering are utilized to guarantee the correctness of the translated texts. Finally, the embedding vectors of the remaining translated words, which are searched in the English model, are adopted to replace the corresponding vectors in the Chinese model.

Refitting by Sequence Learning via LSTM Networks. In the previous work, we train the W2V model with the paragraphs where the words occur. However, the contexts where the pair of words co-occur may reflect their relationship as well. For instance, the pair of words (孤单, 寂寞, score = 8.1) occur in the coordinate structure of texts like "寂寞和孤单的区别是什么" are more likely to be comparable. To refit W2V model based on this association, Long Short Term Memory (LSTM) networks are employed to perform similarity analysis for sentences that target words co-occurrence. For each sample sequence, the vector of each word is jointed by a 300-dimension word vector and a 300-dimension distance vector, which represents the distance between the current word and the 2 target words, the input tag is the round number of the word-pair's similarity and the final similarity score is the *mathematical expectation* of all probable prediction scores by the LSTM model.

3 Experiment Settings

3.1 Data Set

The proposed approach is evaluated on the dataset released by NLPCC-ICCPOL 2016 CLSC shared task [20]. The dataset contains 40 sample data and 500 test word pairs with their similarity scores, which are properly balanced in terms of the different factors including *Domain*, *Frequency*, *POS Tags*, *Word length* and *Senses*. The similarity score between the two words ranges from 1 to 10, and the higher the score is, the more similar the 2 words are.

3.2 Evaluation

The performance in our experiments is evaluated by the Spearman (ρ) and Pearson (r) rank correlation coefficient, which are widely used to test the consistency between automatic predicting results and the golden human labelled data. The Spearman correlation coefficient (ρ) is defined as:

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(R_{Xi} - R_{Yi})^2}{n(n^2-1)} \tag{20}$$

where R_{X_i} and R_{Y_i} are the standard deviations of the rank variables, which are converted from the raw scores X_i and Y_i, and n is the size of observations.

The Pearson correlation coefficient (r) is shown as:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{21}$$

where X_i and Y_i are the raw score, $\bar{X}$ and $\bar{Y}$ are the mean value respectively, and n is the number of sample data.

In our experiment, we regard the Spearman ρ as the main index and the Pearson r as the second important index for evaluation. Limitation of space, all the resource utilized in this paper are listed at https://github.com/JiahuanPei/NLPCC-2016-CLSC.

4 Results and Analysis

4.1 Results of Submission

Table 1 shows the results of W2V models trained with 8 groups of different corpora, where ρ represents the Spearman ρ between the result and the golden score, and ρ' denotes that between the result and WSS, which is the main index used for weak supervision; r is the Pearson r between the result and the golden score, and r' stands for that between the result and WSS. For each W2V result, we train the W2V model under the weak supervision of WSS and choose the best one within 50 iterations.

Table 1. Results of W2V models based on different corpora

No.	Corpora	ρ	r	ρ'	r'
1	Xieso (62M)	0.205	0.203	0.249	0.230
2	Datatang (199M)	0.267	0.272	0.337	0.343
3	News (381M)	0.311	0.305	0.317	0.277
4	News + Xieso	**0.311**	**0.311**	**0.359**	**0.310**
5	Wiki (1.1G)	0.211	0.213	0.324	0.343
6	News + Xieso + Wiki	0.178	0.197	0.221	0.220
7	News + Xieso + Datatang	0.174	0.190	0.211	0.207
8	News + Xieso + DataTang + Wiki	0.214	0.239	0.314	0.308

As is shown in Table 1, both quantity and quality of corpora have a significant effect on performance of W2V model. Comparing the results No. 1–4 of corpora with different scale, we can see that larger quantity of corpora may enrich more contexts to improve the performance. To be specific, No. 4 achieves a ρ value of 0.311 and r value of 0.311, which performs 0.106 and 0.108 higher than No. 1. However, the larger scale does not absolutely mean the higher performance (See No. 5–8), we infer that the quality of the corpora leads to these phenomena. Specifically, there are some paragraphs offending against the rules of grammar in the Datatang and Wiki corpus. Finally, the approach No. 4 is selected as the best W2V model in this step.

Table 2 shows the comparison between the original and weakly supervised W2V models. The original model could be any W2V model which is generated randomly at one iteration. The weakly supervised model is the best one in all 50 iterations, which also illustrated in Table 1.

Table 2. Comparison between the original and weakly supervised W2V models

No.	Strategy	ρ	r	ρ'	r'
1	Original W2V	0.296	0.241	0.330	0.262
2	Weakly supervised W2V	**0.311**	**0.311**	**0.359**	**0.310**

Table 3 shows the results of 6 merging strategies, where the symbols ρ, ρ', r, r' share the same meanings with those of Table 1. According to Table 3, different merging strategies can greatly affect the final result. Given No. 5 achieves the best ρ' value of 0.335 and r' value of 0.306, which outperforms other results by weak supervision, No. 5 is selected with the 0.457/0.455 value of ρ and r. Although the results No. 4 and No. 6 perform better than No. 5 in terms of ρ and r, the results No. 1–3 can be remarkably differentiated from the results No. 4–6, which indicates that the weak supervision method can distinguish the good results from the bad ones.

Table 3. Merging results based on 6 strategies

No.	Strategy	ρ	r	ρ'	r'
1	Replace 1 and 10	0.104	0.090	0.096	0.074
2	Replace 1	0.457	0.446	0.258	0.223
3	Min	0.301	0.314	0.288	0.296
4	Max	0.469	0.464	0.290	0.254
5	Arithmetic mean	0.457	0.455	**0.335**	**0.306**
6	Geometric mean	**0.478**	**0.468**	0.326	0.285

Table 4 shows the comparison between the 2 single models and the best merging model. The result No. 3 achieves the 0.457/0.455 value of ρ/r, which performs 0.146 (47%) and 0.144 (46%) higher than No. 2. It illustrates the effectiveness of the merging approaches and is submitted as our final result to CLSC task.

Table 4. Comparison between single and merging models

No.	Strategy	ρ	r
1	Cilin	0.405	0.393
2	W2V	0.311	0.311
3	Cilin + W2V	**0.457**	**0.455**

4.2 Result of Improvement

Table 5 indicates the effectiveness of machine translation and sequence learning via LSTMs network, where the result of the best merging model is regarded as the baseline. Specifically, the result No. 3 achieves the 0.541/0.514 value of ρ/r, which performs 0.146 (47%) and 0.144 (46%) higher than baseline.

Table 5. Result of improvement by translation and LSTM networks

No.	Strategy	ρ	r
1	Baseline	0.457	0.455
2	Baseline + Translation	0.531	0.476
3	Baseline + Translation + LSTM	**0.541**	**0.514**

5 Conclusion

This paper proposes a novel framework for the CLSC task. In the final framework, our boosting tricks are as follows: (1) Utilizing Tongyici Cilin to compute the dictionary-based similarity scores and extend the corpora for corpus-based word embedding. (2) Using semantic similarity scores generated from retrieval statics and manual features to weakly supervise the model selection process. (3) Leveraging translation technique to improve the Chinese embedding model with an English model, which also reduce the effect of contexts shortage for Chinese single-character word. (4) Adopting similarity calculated by retrieval information to weakly supervise the skip-gram model. (5) Applying LSTM networks to build language model for the words co-occurrence sentences and return the embedding vectors in this process. The experiments on the CLSC shared task have demonstrated the effectiveness of our approach: we rank No. 2 with the result of 0.457/0.455 of Spearman ρ/Pearson r by merging the result of Cilin model and W2V model, where the W2V model is weakly supervised using scores generated from retrieval information and manual features. In our posteriori improvements after the submission, we strengthen the W2V model by merging an English model and refitting the word vectors through LSTM networks. Finally, we get a final result of 0.541/0.514 of Spearman ρ/Pearson r, which outperforms the state-of-the-art performance to the best of our knowledge.

Acknowledgements. This research is supported by National Natural Science Foundation of China (Nos. 61672127, 61173100) and National Social Science Foundation of China (No. 15BYY175). We also wish to thank NVIDIA Corporation for their donation of Tesla K40c GPU device.

References

1. Tian, J.L., Zhao, W.: Words similarity algorithm based on Tongyici Cilin in semantic web adaptive learning system. J. Jilin Univ. (Inf. Sci. Edn.) **28**(6), 602–608 (2010)
2. Zhao, J., Hu, S.Z., Fan, X.H.: Word similarity computation based on word link distribution. J. Chongqing Univ. Posts Telecommun. (Nat. Sci. Edn.) **4**, 021 (2009)
3. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of Workshop at ICLR (2013a)
4. Mikolov, T., Sutskever, I., et al.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS, pp. 3111–3119 (2013b)
5. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems, pp. 2177–2185 (2014)
6. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of EMNLP 2014, pp. 1532–1543 (2014)
7. Mei, J.J., Zhu, Y.M., et al.: Tongyici Cilin. Shanghai Lexicon Publishing Company, Shanghai (1983)
8. Dong, Z., Dong, Q.: HowNet and the Computation of Meaning, pp. 85–95. World Scientific, Singapore (2006)
9. Liu, Q., Li, S.: Word similarity computing based on How-Net. Comput. Linguist. Chin. Lang. Process. **7**(2), 59–76 (2002)

10. Wu, S.Y., Wu, Y.Y.: Chinese and English word similarity measure based on Chinese WordNet. J. Zhengzhou Univ. (Nat. Sci. Edn.) **42**(2), 66–69 (2010)
11. Ahsaee, M.G., Naghibzadeh, M., Naeini, S.E.Y.: Semantic similarity assessment of words using weighted WordNet. Int. J. Mach. Learn. Cybernet. **5**(3), 479–490 (2014)
12. Turney, P.D., Pantel, P.: From frequency to meaning: vector space models of semantics. J. Artif. Intell. Res. **37**(1), 141–188 (2010)
13. Hill, F., Reichart, R., Korhonen, A.: Simlex-999: evaluating semantic models with (genuine) similarity estimation. Comput. Linguist. **41**(4), 665–695 (2015)
14. Iacobacci, I., Pilehvar, M.T., Navigli, R.: SensEmbed: learning sense embeddings for word and relational similarity. In: Proceedings of ACL, pp. 95–105 (2015)
15. Mrkšić, N., Séaghdha, D.Ó, et al.: Counter-fitting word vectors to linguistic constraints (2016). arXiv preprint arXiv:1603.00892
16. Nguyen, K.A., Walde, S.S.I., Vu, N.T.: Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction (2016). arXiv preprint arXiv:1605.07766
17. Chen, Z., Lin, W., et al.: Revisiting word embedding for contrasting meaning. In: Proceedings of ACL, pp. 106–115 (2015)
18. Rothe, S., Schütze, H.: AutoExtend: extending word embeddings to embeddings for synsets and lexemes. In: Proceedings of ACL-IJNLP, pp. 1793–1803 (2015)
19. Faruqui, M., Dodge, J., et al.: Retrofitting word vectors to semantic lexicons. In: Proceedings of NAACL (2015)
20. Wu, Y.F., Li, W.: NLPCC-ICCPOL 2016 shared task 3: Chinese word similarity measurement. In: Proceddings of NLPCC 2016 (2016)
21. Yu, M., Dredze, M.: Improving lexical embeddings with semantic knowledge. In: Proceedings of ACL, pp. 545–550 (2014)
22. Ono, M., Miwa, M., Sasaki, Y.: Word embedding-based antonym detection using thesauri and distributional information. In: Proceedings of NAACL (2015)
23. Liu, Q., Jiang, H., et al.: Learning semantic word embeddings based on ordinal knowledge constraints. In: Proceedings of ACL-IJCNLP, pp. 1501–1511 (2015)
24. Bollegala, D., Matsuo, Y., Ishizuka, M.: Measuring semantic similarity between words using web search engines. In: WWW, vol. 7, pp. 757–766 (2007)