A Feature Extraction Method Based on Word Embedding for Word Similarity Computing

Weitai Zhang, Weiran Xu, Guang Chen, and Jun Guo

Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract. In this paper, we introduce a new NLP task similar to word expansion task or word similarity task, which can discover words sharing the same semantic components (feature sub-space) with seed words. We also propose a Feature Extraction method based on Word Embeddings for this problem. We train word embeddings using state-of-the-art methods like word2vec and models supplied by Stanford NLP Group. Prior Statistical Knowledge and Negative Sampling are proposed and utilized to help extract the Feature Sub-Space. We evaluate our model on WordNet synonym dictionary dataset and compare it to word2vec on synonymy mining and word similarity computing task, showing that our method outperforms other models or methods and can significantly help improve language understanding.

Keywords: word embeddings, feature sub-space, negative sampling, word similarity, prior statistical knowledge.

1 Introduction

Word similarity in NLP is a task of computing the similarity between two or more words by certain methods. In general, similarity always means the semantic similarity of words, for example, "apple" and "pear" have very close relationship, while "mike" and "class" are not relevant. Researchers improved the performance of word similarity task within methods like brown cluster, topic model, vector space model and so on[1][2]. Recently, word representation, mostly word embedding, is proved to be excellent at word similarity task [3].

In this paper, we focus on a new NLP task similar to word expansion task or word similarity task which could reveal words having the same semantic components with given seed words. The differentia between this task and word similarity task lies in that (1) it reveals words through more than one word and (2) the key point is how to represent the same semantic components of the seed words. We propose a method combining words' syntactic information gained with state-of-the-art methods and representation of the same semantic components of the seed words based on word embeddings.

Most words have more than one meaning, which leads to that certain aspects of some words may share the same semantic information. One specific example given here is that "Beijing", "Shanghai" and "Tokyo" have the same facet that they are all city names. Another example is that “sad”, “sorrow” and “low” also have a same meaning of sad or upset in mood. On condition of research needs and actual situations, we always need to reveal more words having the same semantic components with the given seed words which is exactly our problem. Like, given “Beijing”, “Shanghai” and “Tokyo”, we can find that “Guangzhou”, “Houston”, “Osaka” are also city names; given “sad”, “sorrow” and “low”, we may find that “upset”, “depressed” also have the meaning of sad or upset.

The structure of this paper is as follows. Section 2 describes Prior Statistic Knowledge proposed. Section 3 describes our method and gives structure of our model. Experimental results are presented in Section 4. Finally, conclusions are made in the last section.

2 Prior Statistical Knowledge

Researchers of deep learning in natural language processing believe word embeddings can represent syntactic information or semantic information [9]. However, current progress shows that word embeddings or neural language model may not outperform state-of-the-art methods in some tasks. For example, Brown Cluster is superior to the word embeddings on NER task [11] and there is only a small difference between Brown clusters and word embeddings on chunking task.

The effectiveness of word representation plays a significant role in our model. To more precisely represent a word, we propose Prior Statistical Knowledge of words to enrich the representation of words and compute using the published open source tools from Stanford NLP Group[10].

Table 1. Samples of labels and features in Prior Statistical Knowledge

Label name	No.	Feature name
POS	29	vb, cc, jjs, prp, in, nnp...
NER	3	person, location, organization
Parsing	94	cc_post, tmod_post, prt_pre, cop_pre ...

As showed in table 1, we assign each word with 3 most important labels: Named Entity Recognition, Part-Of-Speech tagging, Parsing Dependencies. According to Stanford CoreNLP tools and actual situation, we choose 29 features for POS and 3 features for NER. Particularly in Parsing Dependencies part, there are 47 kinds of ternary relation pair in total, so we extract 94 features for Parsing Dependencies, doubled because of the position each word exists. In Parsing Dependencies, features ended with ‘_pre’ indicate that words are in the front of ternary relations and ‘_post’ corresponds to the back of ternary relations. For example, in ternary relation "subj(undesirable-10,state-9)", the number of 9 and 10 is the position of words in the sentence; in our method, we assign feature "subj_pre" to "undesirable" and "subj_post" to "state".

For each label, a word may have several features. For example, as shown in table 2, word "bush" has several features for each label, like "nnp", "nn", "jj", "vb" for POS. To compute the weight of each feature, we utilize the Wikipedia corpus crawled from Wikipedia website. The corpuses are all standard, clean, grammatical articles. We compute the occurrence times of each feature and assign a probability to each feature for each word concluded by normalization. For instance, specific to NER, we assign 0.9748 to "Person" feature because "bush" appears in our corpus with a 0.9748 probability of "Person" according to our statistic results.

We compute the probability of one word with the equation (1).

$$P_{f_i} = \frac{c_i}{\sum_i c_i} \tag{1}$$

where P_{f_i} stands for the weight of feature f_i and c_i stands for the occurrence of feature f_i.

Prior Statistical Knowledge firstly helps reduce calculation amount and improve general speed twice the original model by reducing words number and abandoning useless words of calculation. It can also enrich the representations of words by synthesizing word embeddings and state-of- the-art methods before.

Table 2. Example of a specific feature assignment and feature weight for word "bush"

Label name	Feature name & weight		
POS	nnp	nn	jj
	0.8919	0.0630	0.0445
NER	Person	Organization	Location
	0.9748	0.0190	0.0062
Parsing	pre_post	det_pre	amod_post
	0.2056	0.1772	0.1203

3 Feature Sub-Space Extraction Model

3.1 Feature Sub-Space

As stated before, a word embedding is always a vector associated with each word. Each dimension or several dimensions correspond to a feature and might even have a semantic or grammatical interpretation. Given several seed words that have one or more identical meanings, we believe several common dimensions of these vectors represent the common part of these words although we cannot point out what each dimension exactly represent. We believe this feature sub-space with a lower dimensionality will reveal and represent the common latent semantic component information.

On the condition that dimensions of vectors are assumed to be identical and independent distribution, we can assume that the values of all words in certain dimension to be Gaussian Distribution if the values are random variables which is always reasonable and effective as a assumption in natural language processing tasks. According to that, values of dimensions sharing the common meaning of the seed words should be Gaussian Distribution and around mean value while values of other dimensions should be very different and away from the mean value. Based on this theory, we propose the feature sub-space extraction model and reveal the common dimensions of seed words.

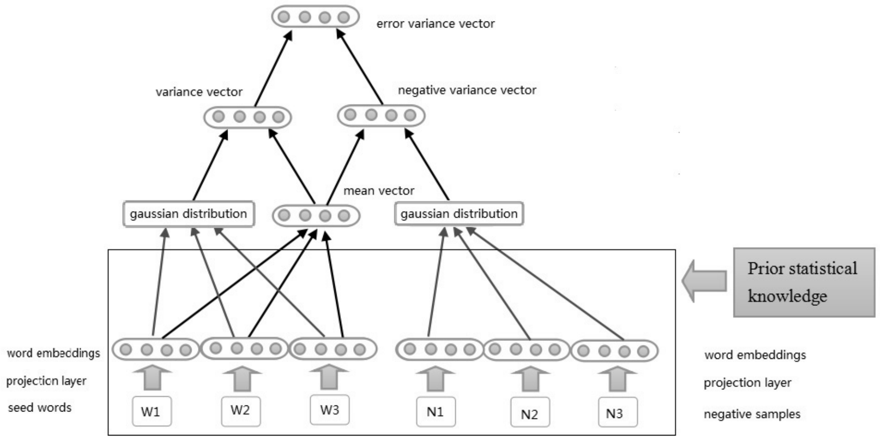

Fig. 1. Feature Sub-Space Extraction Model

3.2 Negative Sampling

Noise Contrastive Estimation (NCE), introduced in [11], posits that a good model should be able to differentiate data from noise by means of logistic regression. In our model, we concerned with extracting high-related sub-space, which means variances of words having the same sub-semantic information with the seed words are small, while those of negative samples are large. So we can simplify NCE as long as the feature sub-space retains its quality.

The main difference between the Negative sampling and NCE is that NCE needs both samples and the numerical probabilities of the noise distribution, while Negative sampling uses only samples [12]. In this paper, we introduce negative sampling because (1) negative samples will help avoid acquiring uncorrelated dimensions where the variances are small enough by coincidence and (2) help enhance the noise immunity of dimensions that are really related to the sub-space.

Negative sampling plays a decisive role in the Feature Sub-Space Extraction Model. After calculating weight of dimensions of seed words and that of negative samples, we extract low-valued dimensions of error vector which is the difference value of $\vec{D}$ and $\vec{D}_{neg}$.

As showed in Figure I, our model consists of the following steps:

- Utilize Prior Statistical Knowledge and reduce the computing subset. We only consider words with the same features as the seed words in the following steps.
- Map the seed words into word embeddings using lookup table in the projection layer, each word is represented with a d-dimension real-valued vector.
- Calculate the center vector or mean vector of the seed words as followed:

$$\vec{v} = \frac{\sum_{i=1}^{k} \vec{v}_i}{k} \tag{2}$$

Given k seed words, we denote $\vec{v}_i$ as the word embedding of word i, and $\vec{v}$ as the mean vector of the seed words. Each dimension of $\vec{v}$ is the mathematical expectation of that dimension of all seed words.

- Calculate the variance vector as followed:

$$\vec{D} = (D^1, D^2, \dots, D^i, \dots) \tag{3}$$

$$D^i = \frac{\sum_{j=1}^{k} (\vec{v^i} - \vec{v_j^i})^2}{k} \tag{4}$$

We denote $\vec{D}$ as the variance vector, with each dimension D^i as a variance, representing the degree of deviation between the random variable and its mathematical expectation. We also denote $\vec{v^i}$ as the i^{th} dimension of mean vector $\vec{v}$ and $\vec{v_j^i}$ as the i^{th} dimension of vector of word j.

- Sample negative words stochastically in the lookup table, and calculate the negative variance vector $\vec{D}_{neg}$ as step 3.
- Extract feature sub-space with certain dimensions where D^i is small and D^i_{neg} is large. In our model, we gain the error variance vector through computing the difference of $\vec{D}$ and $\vec{D}_{neg}$ and reorder the error variance vector before extracting the first K small dimensions as sub-space representation vector.
- Re-compute the cosine similarity of mean vector and each word using their sub-space representation vector but not the full vector.

4 Experiments and Results

In our method, we actually choose Google's new published tool word2vec to train word embeddings because of its efficiency of learning high-quality distributed representations that capture a large number of syntactic and semantic word relationships. Word2vec's training is extremely efficient [11]: an optimized single machine implementation can train on more than 100 billion words in one day. We can finish training in several hours of processing 5.6G data after converting the tokens into lower case.

Following most researchers, we choose Wikipedia articles as the corpus to train the model and word embeddings because of their wide range of topics and word usages, and clean organization of document by topics [3]. We use the Wikipedia corpus with a total of 2 million articles and 990 million tokens.

After training the word embeddings, we choose the top 100,000 most frequent words in Wikipedia and implement our experiment based on these words. Each word embedding is a 200-dimension real-valued vector that supposed to represent a word's meaning and features.

In Prior Statistical Knowledge step, we assign 29 POS features to 99203 words, 3 NER features to 82795 words and 94 Parsing features to 64813 words. Some words may not have all three labels because of the statistical computing results.

We give (1) our model's performance in WordNet synonym dictionary in table 3 and (2) our model versus word2vec on the performance of this task in table 4.

Table 3. Our model's performance in WordNet synonym dictionary

Seed words	Our FSS model / word(rank)	WordNet
distressing, sad, pitiful	dreadful(2), miserable(5), painful(6), deplorable	Deplorable, sad, distressing, miserable, pitiful, sorry, painful, dreadful
animal, creature	monster(2), giant(4), cat, alien, ape	animal, creature, monster, giant
acquire, win, gain	obtain(1), make(3) reach(4), achieve(7), retain, get	Acquire, win, gain, attain, obtain, reach, achieve, make, earn

Table 4. Our model versus word2vec on the performance of this task

Seed words	Our model	Word2vec
Beijing, Shanghai, Nanjing	Harbin, Suzhou, Taiwan	Chinese, China, Taiwan
son, woman, lady	widow, lover, man	man, cousin, father
sorry, sad, upset	terrible, ironically, hurt	obviously, feeling, hurt

Table 3 shows that our mode can reveal new words having the same meaning as the seed words. The number in the parentheses after each word is the rank of the word among the expand words. Words without a parentheses means they have some common semantic components with the seed words but not appear in the WordNet synonym dictionary.

Table 4 shows that our model can reveal synonymous words of seed words that word2vec cannot and improve the rank of nearest words. More importantly, Word2vec can only find words with high co-occurrence of seed words, but our model can reveal words with the same feature sub-space.

Table 5. Performance of our model versus word2vec on WordNet synonym dictionary

Models	Precision@5	Recall@5	F1@5
Our model	77.20%	80.24%	78.69%
Word2vec	79.45%	73.40%	71.36%
Models	Precision@10	Recall@10	F1@10
Our model	90.40%	92.20%	91.29%
Word2vec	86.30%	87.35%	86.82%

We also give a comparison between the performance of our model and word2vec on WordNet synonym dictionary in table 5.

Our model expands words with several seed words while Word2vec expand words with only one word. To compare the two methods more precisely, we reorder the results of Word2vec on each seed word by averaging each expanded word's similarity score. In the new rank, words with a better place more likely similar to all seed words. For example, given seed words "Beijing", "Shanghai", "Guangzhou", "capital" is expanded through "Beijing" only, we will divide its weight by 3. While "Chinese" is expanded through all three seed words, we will average the three scores as the new score.

We choose 235 word groups extracted from WordNet synonym dictionary and utilize all words but one of each word group as seed words and the left one as evaluation word. We evaluate two methods on top 5 and top 10 of rank lists. For example, Precision@5 means that evaluation word is in the top 5 of expanded rank list. Table 5 shows that our model outperforms Word2vec on revealing latent common semantic components and expanding synonymy.

5 Conclusion

We introduced a new NLP task similar to word expansion task or word similarity task and proposed a Feature Sub-Space Extraction Model with Prior Statistical Knowledge and Negative Sampling based on word embeddings. Our problem and methods are significant in information retrieval and natural language processing. It can be used to reveal new similar words, synonymous words and explore the common meaning part of seed words. Our model has been proved to be proper, effective and outperform original word embeddings like word2vec.

References

1. Dagan, I.: Contextual word similarity. Handbook of Natural Language Processing, 459-475 (2000)
2. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. arXiv preprint cmp-lg/9511007 (1995)
3. Huang, E.H., et al.: 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, Association for Computational Linguistics (2012)
4. Budanitsky, A.: Lexical Semantic Relatedness and Its Application in Natural Language Processing. Technical Report CSRG-390,Dept. of Computer Science, Univ. of Toronto (August 1999)
5. Lin, D.: An Information-Theoretic Definition of Similarity. In: Proc. Int'l Conf. Machine Learning (July 1998)
6. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics (2010)
7. Bengio, Y., et al.: Neural probabilistic language models. In: Innovations in Machine Learning, pp. 137–186. Springer, Heidelberg (2006)
8. Mikolov, T., et al.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
9. Collobert, R., et al.: Natural language processing (almost) from scratch. The Journal of Machine Learning Research 12, 2493–2537 (2011)
10. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics, vol. 13. Association for Computational Linguistics (2000)
11. Gutmann, M.U., Hyävrinen, A.: Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. The Journal of Machine Learning Research 13, 307–361 (2012)
12. Mikolov, T., et al.: Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems (2013)