Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Syntactic word embedding based on dependency syntax and polysemous analysis*

Zhong-lin YE[1], Hai-xing ZHAO[†‡1,2]

[1]*School of Computer Science, Shaanxi Normal University, Xi'an 710119, China*

[2]*School of Computer, Qinghai Normal University, Xining 810800, China*

[†]E-mail: h.x.zhao@163.com

Received Dec. 21, 2016; Revision accepted Apr. 17, 2017; Crosschecked Apr. 12, 2018

Abstract: Most word embedding models have the following problems: (1) In the models based on bag-of-words contexts, the structural relations of sentences are completely neglected; (2) Each word uses a single embedding, which makes the model indiscriminative for polysemous words; (3) Word embedding easily tends to contextual structure similarity of sentences. To solve these problems, we propose an easy-to-use representation algorithm of syntactic word embedding (SWE). The main procedures are: (1) A polysemous tagging algorithm is used for polysemous representation by the latent Dirichlet allocation (LDA) algorithm; (2) Symbols '+' and '−' are adopted to indicate the directions of the dependency syntax; (3) Stopwords and their dependencies are deleted; (4) Dependency skip is applied to connect indirect dependencies; (5) Dependency-based contexts are inputted to a word2vec model. Experimental results show that our model generates desirable word embedding in similarity evaluation tasks. Besides, semantic and syntactic features can be captured from dependency-based syntactic contexts, exhibiting less topical and more syntactic similarity. We conclude that SWE outperforms single embedding learning models.

Key words: Dependency-based context; Polysemous word representation; Representation learning; Syntactic word embedding
https://doi.org/10.1631/FITEE.1601846 **CLC number:** TP391

1 Introduction

The processing of a natural language model should be transformed into a machine learning one, and the best method is representation learning (Hinton, 1986; Bengio et al., 2003). Word distribution in corpora is determined by multiple latent factors. The main purpose of representation learning is also to deconstruct the latent factors in language models. There are different types of continuous and discrete word representations, among which the main purpose of representation learning of words is to represent words as vectors.

There are mainly two methods for continuous word representation: distributional representation and distributed representation. In distributional representation, words are represented using a co-occurrence matrix based on the theory of distributional hypothesis. In distributed representation, words are represented by a compressed, low dimensional, and dense vector. Discrete word representation is another method that adopts a 'one-hot' representation form. Distributed word representation is the mainstream approach. Word2vec (Mikolov et al., 2013) is a kind of implementation of distributed representation.

One-hot representation is an original approach of word embedding. The vector length of one-hot is equal to the size of the directory in the corpus. The

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61663041 and 61763041), the Program for Changjiang Scholars and Innovative Research Team in Universities, China (No. IRT_15R40), the Research Fund for the Chunhui Program of Ministry of Education of China (No. Z2014022), the Natural Science Foundation of Qinghai Province, China (No. 2014-ZJ-721), and the Fundamental Research Funds for the Central Universities, China (No. 2017TS045)

ORCID: Zhong-lin YE, http://orcid.org/0000-0002-2429-3325

one-hot representation approach can result in semantic ambiguity between words. Distributional representation uses a co-occurrence matrix to represent the word embeddings. The co-occurrence matrix, using a high-frequency dictionary as a context window, is also a kind of high-dimensional and sparse representation (Baroni and Lenci, 2010; Turney and Pantel, 2010). Word2vec applies the local contexts, and contexts are captured mainly by a bag-of-words model, a linear model, or an *n*-skip-gram-bi-grams model. The above methods consider one or more words around the current word, but they have some fatal defects. For example, some words in the context are not meaningless for the current word, or some important words are out-of-reach within a small window. Syntactic analysis based on dependencies achieves good performance in a variety of tasks (Goldberg and Nivre, 2012, 2014). Meanwhile, dependency-based contexts outperform word-based contexts (Levy and Goldberg, 2014). Thus, we generalize the skip-gram algorithm by replacing the bag-of-words contexts with dependency-based contexts. The symbols '+' and '−' are introduced to indicate the directions of dependencies, and dependency skip is applied to denote indirect dependencies. Consequently, the word embeddings based on bag-of-words contexts tend to contextual structure similarity of sentences, and our dependency-based contexts tend to syntactic similarity of sentences.

Existing distributed representation approaches neglect the representation of polysemous words. Words with different contexts share some representations, which causes the loss of some information. We propose an approach to tag the polysemous words based on the combination of the latent Dirichlet allocation (LDA) and clustering algorithms. The LDA algorithm is first used to find the topic distribution. The voting strategies are then applied to obtain the sense of the polysemous words by topic distribution. Finally, the polysemous words are tagged by a clustering algorithm based on the context, and the clustering amount of the K-means algorithm (Krishna et al., 1999) is set to equal to the amount of sense.

In this paper, dependency-based contexts and polysemous language models are proposed to train the syntactic word embedding (SWE). The aim of SWE is to ensemble semantic and syntactic contexts into word embeddings. Regardless of the previous hypothesis, we propose a new hypothesis that words with similar semantic and syntactic contexts should have similar word embeddings. The dependency-based word embedding has less topical similarity and more syntactic similarity.

In this paper, our contributions are:

1. We propose a polysemous language model to generate multiple word embeddings for polysemous words.

2. We generalize the skip-gram model with optimized dependency-based contexts, and our model produces markedly different embeddings.

3. Our model can generate state-of-the-art word embeddings for word similarity tasks. The proposed SWE model is less topical and exhibits more functional similarity compared with other embedding models.

2 Related work

As a result of representation learning of words, word embedding is a suitable tool for natural language processing. Word embedding can be used for named entity recognition (NER), semantic role labeling (SRL), and part-of-speech tagging (Ren et al., 2016; Zhai et al., 2016).

Distributed and distributional representations adopt context information to generate embeddings. For the co-occurrence matrix, major improvements are achieved based on the co-occurrence and dimensionality reduction operations, such as the dimensionality reduction technique based on singular value decomposition (SVD) (Bullinaria and Levy, 2007), the LDA algorithm (Ritter et al., 2010), principal component analysis (PCA) (Lebret and Collobert, 2014), and the co-occurrence matrix based on probability (Lebret and Collobert, 2015). The distributed representation (Hinton, 1986), which has been widely used and improved, is usually called 'word representation' or 'word embedding'. It seems that the count-based distributed representation method eventually encounters a bottleneck. Accordingly, the language model based on neural networks (Xu and Rudnicky, 2000; Bengio et al., 2003; Mnih and Hinton, 2008) is proposed and it shows desirable performance on various natural language tasks (Nguyen et al., 2016; Wang et al., 2016).

Firth (1957) and Harris (1981) assumed that words in similar contexts have similar meaning. The basic idea of a language model based on neural networks is that word meaning is influenced and determined by its contexts. Thus, each word in the context is randomly initialized to an embedding and then trained on a large corpus. The idea of the former modeling is that similar words should have similar word embeddings in the semantic space. Word2Vec (Mikolov et al., 2013) is an implementation based on a probabilistic model using deep learning with a three-layer neural network. Word2Vec has been improved and various versions of Word2Vec have been generated, such as the global vectors for word representation (GloVe) (Pennington et al., 2014). GloVe is an unsupervised learning model based on a word-word co-occurrence matrix, rather than an entire sparse matrix or individual context windows in a large corpus. The performance of Word2Vec can be greatly improved when dependency-based contexts, rather than word-based contexts, are adopted (Levy and Goldberg, 2014). The language modeling of polysemous words is then presented, which assumes that many words are polysemous and that global contexts can also provide useful information for learning word embedding (Huang et al., 2012). Some other polysemous language models have been proposed based on a probability model (Tian et al., 2014) and topical word embedding (Liu et al., 2015). The above approaches of the polysemous language model consider the global contexts. However, the sense of polysemous words has weak connections with global contexts. We should concentrate on local contexts for comprehending polysemous words and training the polysemous language model.

3 Syntactic word embedding

3.1 Architecture overview

The dependency-based SWE presented in this study avoids the training defects of the polysemous language model. Capturing the semantic and syntactic contexts to word embeddings, our approach makes the embedding less topical and exhibits more functional similarity.

Fig. 1 depicts the main procedures of SWE. As the input of the training procedures of SWE, texts are used to generate bag-of-words contexts for each word. Then it feeds each word–context pair into the learning algorithm. Consequently, the real input of Word2Vec is a collection of word–context pairs. Therefore, we can replace the original word–context pairs with whatever content we want. We choose syntactic dependencies as the contexts of current words. Note that word–context pairs may have no connection with current words, which would bring noisy data into the model. Based on the original skip-gram model, our optimized approach adds a polysemy tagging layer, and the polysemous words are tagged by contexts and context topics. As the polysemous words in different contexts should have different senses and multiple embeddings, we use $w|t_i$ to denote the polysemous words and their context topics. More details are shown in Algorithm 1.

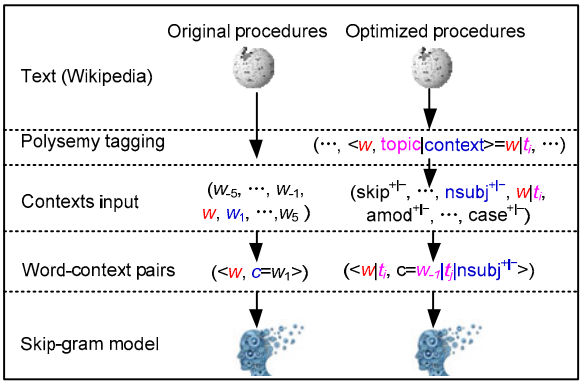

Fig. 1 Comparison of the original procedures and optimized procedures while training the syntactic word embedding

Algorithm 1 Syntactic word embedding generation

Input: entire corpus
Output: polysemous language model

```
corpus←Wikipedia text
for s←corpus do
    s←denoising(s)
    list.add(s)
end for
corpus←polysemytagging(corpus)
for sentence_i←corpus do
    for word_j←sentence_i do
        cxt←contextcapturing(word_j, sentence_i)
        map.put(word_j,i, cxt)
    end for
end for
polysemymodel←skip-gram(map)
```

In Algorithm 1, the SWE generating algorithm includes mainly three functions: polysemytagging, contextcapturing, and skip-gram. The procedures of the three functions are as follows.

3.2 Algorithm details

3.2.1 Polysemy reorganization and tagging

The polysemous representations are processed at the polysemy tagging layer. We assume that the word is a polysemous word if the word appears in several different topics and the number of topics must be greater than 1. Based on the assumption, we adopt the following procedures. First, the topic number of LDA is selected within the array [10, 20, 30, 50, 80, 130, 200, 500]. The appearance frequency is counted under different topic numbers. There are 433 158 words where the word frequency is greater than 1. Therefore, the word number under each topic is set to 433 158/topic_number. Then the appearance frequency under different topic numbers is voted to obtain the sense number of polysemous words. The smaller value is retained when the final voted results are equal. The sense number of polysemous words is calculated by the LDA algorithm and voting approach. The polysemous words need to be tagged using a clustering algorithm with bag-of-words contexts. The clustering number is set the same as the sense number. Through the above steps, the polysemous words are tagged by their senses and contexts. Finally, the corpus consists of polysemous and non-polysemous words. The details about polysemy tagging are shown in Algorithm 2.

3.2.2 Finding dependency-based contexts

In the Word2Vec model, the contexts are defined as the surrounding words of the current word within a window threshold. The context from the symmetric or asymmetric window that contains the current word is called the symmetric or asymmetric context window. The same numbers of words, before and after the current word, are added to the windows in the symmetric context window. The asymmetric situation is that the numbers of words before and after the current word are different.

How to capture the context words from the context window? The original implementation is based on the bag-of-words contexts, which are composed of linear bag-of-words contexts and *n*-skip-gram-bi-grams contexts using the *n*-gram method. The former regards consecutive several words as context. The latter captures the context words by skipping *n* words in the context window. Dependency-based syntactic contexts can capture more deeply semantic and syntactic information than bag-of-words approaches, as we take the sentence "American scientist discovers evidence for liquid water on Mars" as an example in Fig. 2.

Algorithm 2 Polysemy tagging

Input: the entire corpus segmented into separated sentences by full stop, comma, exclamation, and semicolon
Output: the corpus with the tagged polysemy

```
topic←[10, 20, 30, 50, 80, 130, 200, 500]
for i=0 to 7 do
    for j=1 to i do
        LDA[j]←topic and word distribution
        if LDA[j].contains(word_k+topic[i]) then
            map_topic.put(word_k+topic[i],
                map_topic(word_k+topic[i])+1)
        end if
    end for
end for
for map_t in map_topic do
    sensenumber←vote(map_topic)
    sensemap.put(map_t.getkey(), sensenumber)
end for
for map_s in sensemap do
    sensemap.put(map_s.getkey(),
        getrelatedsentence(map_s.getkey()))
    if ispolysemy(map_s.getkey()) then
        K←map_s.get(map_s.getkey())
        category_sentence_word←Kmeans(corpus,
            map_s.getkey(), K)
        clustermap.put(map_s.getkey(),
            category_sentence_word)
    end if
end for
for word_k in corpus do
    if ispolysemy(word_k) then
        category←getcategory(clustermap)
        sentence←tagging(word_k, category)
    end if
end for
```

In Fig. 2, the current word is 'discovers' and the size of window is two around the target word. Thus, there are four contexts (two words before and two words after the target word). Consequently, the contexts w_{-2}, w_{-1}, w_1, and w_2 are American, scientist, evidence, and for, which may result in the discovers

being a context of both ‘scientist’ and ‘evidence’, or may result in ‘scientist’ and ‘evidence’ ending up as neighboring words in the embedding space. Generally, the window size is set to five for capturing broad information. The optimized dependency-based syntactic contexts are presented and used in our model. We first gain the syntactic dependencies using the Stanford parser. The stopword and its dependencies with other words are deleted. We introduce the dependencies ‘+’ and ‘−’ to capture the directions of dependencies. For example, the word ‘for’ is a stopword; thus, it needs to be deleted and the dependency ‘case−’ is also deleted. Consequently, there are 889 stopwords, including prepositions, conjunctions, and link-verb.

We also introduce dependency ‘skip’ to link these words that are out-of-reach to a small window of bag-of-words. ‘Skip’ is a kind of indirect relation, which just consists of two direct dependencies. For example, the dependency ‘skip+’ (discovers, water) can be composed of the dependencies ‘nsubj’ (discovers, evidence) and ‘nmod’ (evidence, water), which denote a forward relation. All dependencies of the illustrated sentence are listed in Table 1.

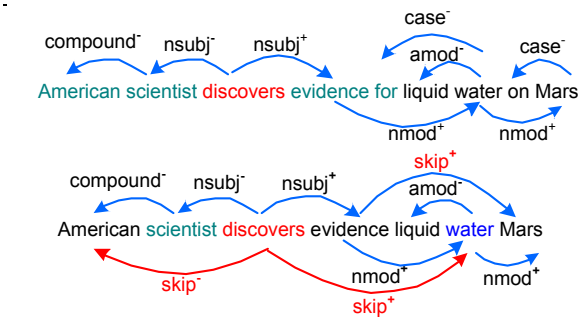

Fig. 2 Illustration of dependency-based syntactic contexts
The first sentence is the original result of syntactic dependency analysis. The second sentence is the result of optimized syntactic dependency. The red line denotes skip dependency. The symbol ‘+’ or ‘−’ denotes the direction of dependency. References to color refer to the online version of this figure

In Table 1, the dependency-based syntactic contexts can capture the semantic and syntactic information. The similar results can be found in the semantics literature. The syntactic contexts can also filter out coincidental contexts, which are within the window but indirectly related to the target word. For example, the context of the word ‘liquid’ is just the word ‘water’ in Fig. 2. Details about capturing the syntactic context are shown in Algorithm 3.

Table 1 Examples of dependency-based syntactic contexts

Word	Context
American	scientist/compound^{-}, discovers/skip^{-}
Scientist	American/compound^{+}, discovers/nsubj^{-}
Discovers	American/skip^{-}, scientist/nsubj^{-}, evidence/nsubj^{+}, water/skip^{+}
Evidence	discovers/nsubj^{+}, water/nmod^{+}, Mars/skip^{+}
Liquid	water/amod^{-}
Water	liquid/amod^{-}, evidence/amod^{+}, discovers/skip^{+}, Mars/nmod^{+}
Mars	water/nmod^{+}, evidence/skip^{+}

The example sentence is analyzed using the Stanford parser

Algorithm 3 Context capturing

Input: word and sentence
Output: word contexts

```
list.add(dep(word_i, word_j)←Parser(sentence))
for lst←list do
    if isstopword(word_i) then
        list←delete(word_i, dep(word_i, word_j))
    end if
    if dep(word_i, word_j)&&dep(word_j, word_k) then
        generating dep(word_i, word_k)
    end if
end for
for lst←list do
    if i≤j then
        dep(word_i, word_j)←dep+(word_i, word_j)
    else
        dep(word_i, word_j)←dep−(word_i, word_j)
    end if
    map.put(word_i, dep(word_i, word_j))
end for
```

3.2.3 Skip-gram model

Word2vec offers two models, CBOW and skip-gram, which can be optimized by hierarchical softmax and negative sampling (NEG) (Mikolov et al., 2013). The hierarchical softmax approach builds a Huffman tree for all words using the word frequency. The shortcoming of this method is that the training procedure is time-consuming. Negative sampling, a simplification of noise contrastive estimation, is used to improve the training efficiency and the quality of embedding. Therefore, the skip-gram with NEG is applied in our model. In this study, we generalize and rebuild the skip-gram model from the linear bag-of-

words contexts to the dependency-based syntactic contexts.

The training procedure of the skip-gram model is as follows. For context(w) of word w, the negative subset is neg(u) and neg(u)≠∅. For all $u\in D$, the label is defined as

$$L^w(u)=\begin{cases}1, & u=w,\\ 0, & u\neq w,\end{cases} \tag{1}$$

where $L^w(u)$ is the label of word u. The labels of positive and negative sampling are 1 and 0, respectively.

For a word and its contexts (w, context(w)), the objective function of skip-gram is to maximize the probability

$$g(w)=\prod_{t\in\text{context}(w)}\prod_{u\in\{w\}\cup\text{neg}^t(w)}p(u\,|\,t), \tag{2}$$

where

$$p(u\,|\,t)=\begin{cases}\sigma(v^{\mathrm{T}}(t)\theta^u), & L^w(u)=1,\\ 1-\sigma(v^{\mathrm{T}}(t)\theta^u), & L^w(u)=0,\end{cases} \tag{3}$$

$\sigma(\cdot)$ is the sigmoid function, $v(t)$ is the embedding of word t, u is one of the words in the context, θ^u is the embedding of word, and $\text{neg}^t(w)$ denotes the subset of the negative sample of word t. For a given corpus C, the optimized target function is defined as

$$G=\prod_{w\in C}g(w). \tag{4}$$

For the optimized target function, the logarithm computing processes are as follows:

$$\begin{aligned}\mathcal{L}&=\log G=\log\prod_{w\in C}g(w)\\ &=\sum_{w\in C}\log g(w)\\ &=\sum_{w\in C}\log\prod_{t\in\text{context}(w)}\prod_{u\in\{w\}\cup\text{neg}^t(w)}\Big\{[\sigma(v^{\mathrm{T}}(t)\theta^u)]^{L^w(u)}\\ &\quad\cdot[1-\sigma(v^{\mathrm{T}}(t)\theta^u)]^{1-L^w(u)}\Big\}\\ &=\sum_{w\in C}\sum_{u\in\text{context}(w)}\sum_{u\in\{w\}\cup\text{neg}^t(w)}\Big\{L^w(u)\cdot\log[\sigma(v^{\mathrm{T}}(t)\theta^u)]\\ &\quad+(1-L^w(u))\cdot\log[1-\sigma(v^{\mathrm{T}}(t)\theta^u)]\Big\}.\end{aligned} \tag{5}$$

Then Eq. (5) can be simplified as follows:

$$\begin{aligned}L(w,t,u)=\Big\{&L^w(u)\cdot\log[\sigma(v^{\mathrm{T}}(t)\theta^u)]\\ &+(1-L^w(u))\cdot\log[1-\sigma(v^{\mathrm{T}}(t)\theta^u)]\Big\}.\end{aligned} \tag{6}$$

In this study, problem (6) is optimized using the stochastic gradient ascent algorithm. The updating rules of θ^u and $v(t)$ are as follows:

$$\theta^u:=\theta^u+\eta[L^w(u)-\sigma(v^{\mathrm{T}}(t)\theta^u)]v(t), \tag{7}$$

$$v(t):=v(t)+\mu\sum_{u\in\{w\}\cup\text{neg}^t(w)}\frac{\partial L(w,t,u)}{\partial v(t)}. \tag{8}$$

The details about how to train a model and generate the word embedding have been presented. Algorithm 4 gives the skip-gram algorithm and updating procedures of word embedding.

Algorithm 4 Skip-gram

Input: word and its contexts
Output: polysemous language model

```
for t←context(w) do
    e=0
    for u←{w}∪neg^t(w) do
        q←σ(v^T(t)θ^t)
        g←η(L^w(u)−q)
        e:←e+gθ^t
        θ^t:←θ^t+gv(t)
    end for
    v(t):←v(t)+e
end for
```

Mikolov et al. (2013) adopted the Word2Vec model, and the best performance is achieved when the embedding length is 300. Then the Word2Vec model is continually optimized. Consequently, hierarchical softmax and negative sampling are applied to optimize the presentation model. The efficiency of the model reaches a peak when the negative sampling number is 15 and the length of embedding is 300. The best accuracy is achieved when the window size is 10 and the embedding length is 300 in word representation for the GloVe approach, and the accuracy is gradually changed. In our model, we set the window size as 10, embedding length as 300, and negative sampling (the number of negative contexts to sample for each correct context word) as 15.

4 Experiments and evaluation

4.1 Word similarity dataset

Word embedding is usually evaluated and measured by word similarity tasks. We also used this method in our experiment. The similarity datasets include Rubenstein and Goodenough (RG) (Rubenstein and Goodenough, 1965), WordSimilarity-353 (WS) (Finkelstein et al., 2002), Sentential Context Word Similarity (SCWS) (Huang et al., 2012), Rare Word (RW) (Luong et al., 2013), and SimLex-999 (SimLex) (Hill et al., 2015). Each dataset is described in Table 2. The datasets consist of word pairs along with human-assigned similarity judgments. The judgments are made by at least 10 linguists. Each dataset contains different numbers and categories of subjects. The types of word pairs include noun–noun, verb–verb, adjective–adjective, verb–noun, noun–adjective, and verb–adjective.

Table 2 Word similarity datasets

Dataset*	Pair number	Subject number	Linguist number	Word number
WS	353	29	13–16	4924
SimLex	999	3	0–10	
SCWS	2003	7	0–10	
RW	2034	4	0–10	

* The four datasets were used to evaluate the performance of our method and other methods. All the attribute contents of each dataset are shown. The last column is the total number of words without repetition in the distinct datasets

4.2 Data corpus preprocessing and denoising

Our English corpus was composed of the entire Wikipedia. The following procedures are conducted with the corpus: (1) Text information was extracted from the original English Wikipedia pages. (2) The appearance frequency of each word in the similarity datasets was counted in the corpus. We deleted those sentences that did not contain words in the similarity datasets. Their appearance frequency was more than 20 000. (3) The noisy symbols and characters were filtered. Additionally, the appearance frequency of each word was counted in the corpus. 722 960 words were deleted because the word frequency was 1. Consequently, there were 433 158 non-repeating words and 3 776 346 sentences in the corpus. (4) All texts were segmented by full stop, comma, exclamation, and semicolon. Each sentence in the corpus was saved in a single row. (5) All the uppercase characters were converted to lowercase ones. All quantifiers were replaced with the string 'NUMBER'.

4.3 Polysemy and visualization

Polysemous word recognition and training are the main tasks of our language model. Traditional word embedding is a single low-dimensional vector, ignoring the implied multiple word meanings. A perfect model should train multiple embeddings for polysemous words. We adopted the LDA algorithm to find the polysemous words and count the sense amount by voting strategies. For example, the apple, star, left, and cell are polysemous words, which have at least two senses and represent different senses in different contexts. For these words, the appearance frequency of polysemous words in different topics was calculated and shown in Table 3.

Table 3 The number of topics

Word	Topic number								Vote
	10	20	30	50	80	130	200	500	
Apple	4	3	5	3	0	2	2	2	2
Star	4	3	4	6	3	6	3	4	3
Left	3	5	6	6	6	19	3	3	3
Cell	7	4	5	2	2	2	2	3	2

Take 'apple, star, left, and cell' as an example to explain how to obtain the sense amount, which is used for the cluster algorithm. The appearance frequency of each word in different topic numbers is voted to obtain the possible number. Generally, polysemous words occur in multiple topic distribution spaces

In Table 3, we counted the appearance frequency under different topic amounts. We assumed that a word is a polysemous word if it occurs in different topic distributions. Accordingly, the word sense number can be gained based on the appearance frequency of polysemous words in different topics. The polysemous words were tagged and divided into multiple words. For example, the word sense number of stars is three. Therefore, the word star is tagged into $star_1$, $star_2$, and $star_3$. Another question is how to place the three words into the corpus using contexts. A *K*-means clustering algorithm was adopted in this procedure. The *K* value of the clustering algorithm was set to the sense number of polysemous words. The polysemous words were tagged using their contexts with the clustering algorithm, and then they

were added to the corpus for training to obtain the language model containing polysemous words. Using the model, the nearest words to the target word are shown in Table 4.

Table 4 The examples of polysemous and nearest words

Word	Tagging	Topic	Nearest words
Apple	$apple_1$	e-products	powerpc, amstrad, hp, xeon
	$apple_2$	fruits	mamey, avocade, raspberries, brassicas, remoulade
Cell	$cell_1$	e-products	fax, apple1, phone, back-office, controller
	$cell_2$	biology	hepatocyte, translocates, hepatocytes, leukocytes, fibroblasts
Left	$left_1$	direction	bottom, right, hand, top, outerdoors
	$left_2$	party	wing, democracy, coalition, venstre, fsln
	$left_3$	depart	star1, cage, airplane, minutes, NUMBER
Star	$star_1$	entertain-ment	movie, hollywood, arneric, brando, lelo
	$star_2$	ranking	NUMBER, company, apartment, hotel, marinetta
	$star_3$	galaxy	starburst, glint, galaxy, neula, sagittarius

The polysemy is tagged by its context topics using the cluster algorithm. The number of senses can be calculated using Algorithm 2 (e.g., word 'apple' has two senses and 'star' has three senses). We give a name for each tagged polysemy according to the nearest words. The nearest words given here are part of those in the experiment

In Table 4, different topical contexts can generate different word embeddings for polysemous words. Additionally, we show the five nearest words as above. The first column is the polysemous word. The second column is the result of polysemy tagging. The third column is topic category. The last column is the nearest word. For example, the polysemy cell is tagged into two words $cell_1$ and $cell_2$. The former and the latter are the vocabularies related to the electronic product and biology, respectively. The polysemous word 'apple' has two topical contexts, which can also be tagged into two words $apple_1$ and $apple_2$. The former and the latter are the vocabularies about fruits and technology corporations, respectively. Therefore, $cell_1$ and $apple_1$ are closer in the semantic model cyberspace. If the polysemous words are represented by a single embedding, which would lose the other multiple senses within various contexts, this is a compromised approach. Based on this approach, the similarity is less than the actual similarity in similarity computing tasks.

As described above, we generate five nearest words for polysemous words. To observe the space relative distribution between polysemous and neighboring words, we need to generate more neighboring words and their word embedding. If there is a close relationship among these words, they should be nearer in the language model cyberspace. Fig. 3 shows the similarity results of SWE embedding using the T-SNE algorithm (Mnih and Hinton, 2008) in the original higher-dimensional space.

In Fig. 3, the blue marks represent polysemous words, and other marks denote the neighboring words. Different neighboring words use different color marks. Apple and cell have two kinds of topical contexts. Left and star have three kinds of topical contexts. Based on the T-SNE algorithm, the high-dimensional embedding can be projected into a two-dimensional space. In Fig. 3, there exists the clustering phenomenon between tagged polysemous words and neighbor words. For example, in Fig. 3a, $apple_1$ is the context about electronic products, and the neighbor words of $apple_1$ are sony, xbox, hp, ipod, and so on. They are all e-products, which belong to a common cluster. The other neighboring words of $apple_2$ are in another cluster. There is a clear split and bound between the two clusters in the two-dimensional space. Using multiple topical contexts, the polysemous words are trained for multiple embeddings, which avoids the problem that the polysemous words share common word embeddings.

4.4 Evaluation and discussion

The CBOW and skip-gram models in Word2Vec aim to obtain the word embeddings based on the statistical model. Using deep learning with a three-layer neural network, the words in the context are converted to low-dimensional embeddings in the model cyberspace. The operation of the word is also converted to that of embedding. According to the hypothesis that words in similar contexts should have similar meaning, the skip-gram model advocates predicting the embedding of contexts through the current words. However, it neglects the syntactic relevance between context words and current words.

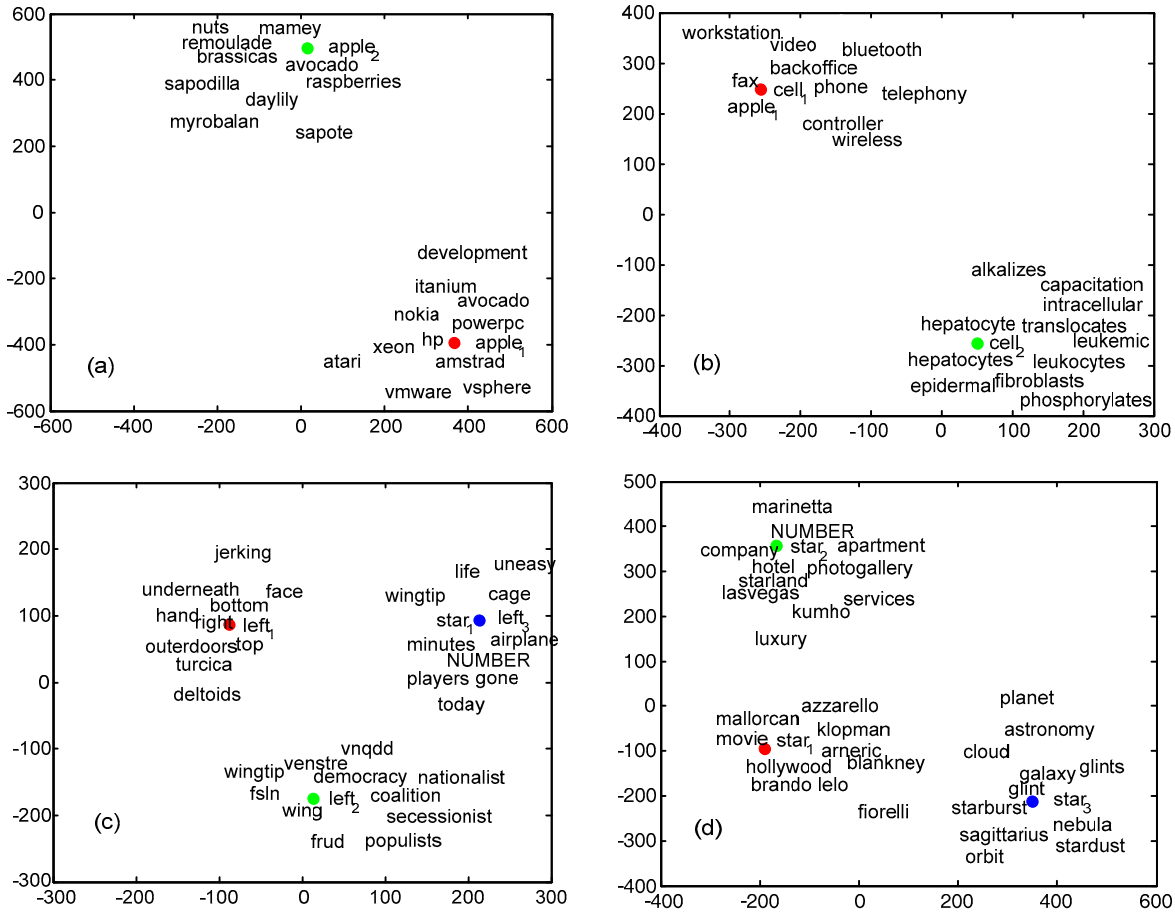

Fig. 3 The neighboring words and 2D visualization: (a) results of polysemy tagging and 2D visualization for apple; (b) results of polysemy tagging and 2D visualization for cell; (c) results of polysemy tagging and 2D visualization for left; (d) results of polysemy tagging and 2D visualization for star

Polysemy has multiple senses; therefore, it exists in multiple communities. Different communities show a same clustering phenomenon. The *x*-axis and *y*-axis represent the length of the embedding in the 2D space. References to color refer to the online version of this figure

The words have a strong association with neighboring words in the syntactic structures. In this study, the dependency-based syntactic context is proposed, which can easily find the relationship between them. The syntactic contexts have some benefits that, they can filter out uncorrelated words and retain associated syntactic dependencies.

The skip-gram model is based on local contexts. The local contexts of the original method are context words, whereas the local contexts of our approach are based on dependencies between current words and context words. The embeddings generated by our approach are termed SWE. The GloVe approach and the approach proposed by Huang et al. (2012) are statistical models based on global information. We apply different methods to obtain the five nearest words of words king and Tsinghua. The results are shown in Table 5.

In Table 5, we use six different approaches to capture neighboring words as listed in the first column. The skip-gram model is the original method. The CBOW model is another method proposed in the Word2Vec model. CBOW predicts the embedding of the current word by context words, while the skip-gram predicts the embeddings of context words by the current word. To verify the impact of global information, the term frequency-inverse document frequency (TF-IDF) value is calculated and then added to each dimension in SWE embeddings, which can add global statistical information to embeddings.

In Table 5, for the word king, each method can obtain the words prince, queen, and other near words. The only difference is the uncommon words, such as tzar and constantine. These uncommon words are also relevant to the word king. Different approaches show different neighboring words. The difference is that different methods give different weights for neighboring words when the model is trained. For the current word Tsinghua, its neighboring words are fudan, heriot-watt, makerere, thammasat, and so on. As these words have similar structures in syntactic dependencies, all the neighboring words are the names of universities. To measure the differences of the tested algorithms, we evaluated different models in various word similarity tasks. First, we calculated the similarity between word pairs using cosine similarity. Then we compared the results with human-assigned similarity. The absolute value of the Hellinger distance was used to measure the distances between them. The performances of different algorithms are shown in Table 6.

Table 6 shows the absolute values of the Hellinger distance between human-assigned similarity and calculated similarity. Smaller distance indicates that the calculation results are closer to human-assigned similarity. In other words, the results are more accurate. As shown in Table 6, the performance of the SWE method is better than that of the other methods. To evaluate the influence of global parameters on local contexts, TF-IDF values were calculated for each word in the corpus. Then the TF-IDF weights were added to each dimension in word embeddings. The SWE+TF-IDF method works better than the skip-gram and CBOW models. However, it performs slightly worse than SWE, which denotes that the global TF-IDF works negatively on the overall result. Compared with other word embedding methods, SWE has a favorable effect on word embedding. We compare our embedding with those of other existing approaches in Table 7.

As shown in Table 7, compared with other methods, the improved rate of the SWE method is more prominent in the WS dataset. It roughly increases the performance by at least 25%. For the RG dataset, the performance of GloVe is the poorest. The performances of other methods are almost the same. For the datasets of RG, RW, and SCWS, the performances of SWE and Huang et al. (2012)'s model are basically the same. For all datasets, the performance of the model proposed by Huang et al. (2012) is superior to that of GloVe. Compared with the skip-gram, CBOW, Huang et al. (2012)'s model, and GloVe, the

Table 5 Related words of target words

Algorithm	King	Tsinghua
Skip-gram	prince, emperor, throne, sultan	belfer, whidden, makerere, islamia
CBOW	prince, constantine, sancho, throne	aligarh, louvain, brandeis, georgetown
Huang et al. (2012)	lord, prince, emperor, queen	broxbourne, uia, liberman, krewe
GloVe	queen, prince, monarch, kingdom	qinghua, fudan, peking, renmin
SWE	queen, monarch, tzar, prince	fudan, heriot-watt, makerere, thammasat
SWE+TF-IDF	norodom, queen, bhumibol, monarch	heriot-watt, carnegie-mellon, osmania, waseda

SWE embedding is generated by the optimized dependency-based context and tagged polysemous words

Table 6 The performance of different algorithms on five datasets

Algorithm	Absolute value of the Hellinger distance				
	RG	RW	SCWS	SimLex	WS
Skip-gram	3.996	32.986	24.453	16.677	10.692
CBOW	4.092	32.831	25.000	16.601	11.069
Huang et al. (2012)	4.161	21.487	18.710	21.290	11.394
GloVe	5.087	34.735	21.654	14.665	10.249
SWE	**3.769**	**19.993**	**17.435**	**14.417**	**7.687**
SWE+TF-IDF	3.919	24.930	21.806	17.641	9.901

RG: Rubenstein and Goodenough; RW: rare word; SCWS: sentential context word similarity; SimLex: SimLex-999; WS: word similarity-353

Table 7 The improved rate of the SWE approach on five datasets

Algorithm	Improved rate					Average
	RG	RW	SCWS	SimLex	WS	
SWE-Skip-gram	0.06	0.39	0.29	0.14	0.28	0.23
SWE-CBOW	0.08	0.39	0.30	0.13	0.31	0.24
SWE-Huang	0.09	0.07	0.07	0.32	0.33	0.18
SWE-GloVe	0.26	0.42	0.19	0.02	0.25	0.23

Different datasets are designed by different linguists. Each dataset is used to verify a different aspect of the algorithm. Each algorithm is used to find some aspects considered as important factors. RG: Rubenstein and Goodenough; RW: rare word; SCWS: sentential context word similarity; SimLex: SimLex-999; WS: word similarity-353

performance is better by 21% on average and 18% at least, which suggests that SWE embedding is more accurate and closer to human assessments. Why is the SWE model superior to other methods in our experiment? The reason is that dependency-based syntactic contexts play an important role in denoising, which can filter out uncorrelated context words. Besides, the optimized strategies proposed for dependencies in our model play an important role. We can conclude that similar structure is better than similar context in similarity tasks.

5 Conclusions

We have proposed a kind of generalization approach of word embedding based on the skip-gram. The linear bag-of-words contexts are replaced with dependency-based syntactic contexts. To generate multiple embeddings for polysemous words, the LDA algorithm is first used to capture the sense number of polysemous words. Then polysemous words are tagged into multiple words using a clustering algorithm based on context topics. The word embeddings tend to the contextual structure similarity of sentences in the original language model. To solve the problem, syntactic contexts of the current word are first gained by the dependency parser. The syntactic contexts are then optimized as follows: (1) Symbols '+' and '−' are used to denote directions of syntactic dependencies; (2) Dependency skip is adopted to capture the indirect relations. A stopword in a sentence usually plays a complementary role without a practical meaning. Thus, stopwords and their dependencies on other words are removed in contexts. This procedure could reduce the noisy data in contexts. Therefore, SWE embedding is closer to syntactic structure similarity. The performance of our model is better than those of existing popular approaches by about 20% in similarity tasks. In the future, we would like to explore incremental learning and denoising.

References

Baroni M, Lenci A, 2010. Distributional memory: a general framework for corpus-based semantics. *Comput Ling*, 36(4):673-721. https://doi.org/10.1162/coli_a_00016

Bengio Y, Ducharme R, Vincent P, et al., 2003. A neural probabilistic language model. *J Mach Learn Res*, 3(6): 1137-1155. https://doi.org/10.1007/3-540-33486-6_6

Bullinaria JA, Levy JP, 2007. Extracting semantic representations from word co-occurrence statistics: a computational study. *Behav Res Methods*, 39(3):510-526. https://doi.org/10.3758/BF03193020

Finkelstein L, Gabrilovich E, Matias Y, et al., 2002. Placing search in context: the concept revisited. *ACM Trans Inform Syst*, 20(1):116-131. https://doi.org/10.1145/503104.503110

Firth JR, 1957. A synopsis of linguistic theory. *Stud Ling Anal*, 41(4):1-32.

Goldberg Y, Nivre J, 2012. A dynamic oracle for arc-eager dependency parsing. Proc Coling, p.959-976.

Goldberg Y, Nivre J, 2014. Training deterministic parsers with non-deterministic oracles. *Trans Assoc Comput Ling*, p.403-414.

Harris ZS, 1981. Distributional structure. *Word*, 10(2-3):146-162. https://doi.org/10.1007/978-94-017-6059-1_36

Hill F, Reichart R, Korhonen A, 2015. SimLex-999: evaluating semantic models with (genuine) similarity estimation. *Comput Ling*, 41(2):665-695. https://doi.org/10.1162/COLI_a_00237

Hinton GE, 1986. Learning distributed representations of concepts. Proc 8th Annual Conf of the Cognitive Science Society, p.1-12.

Huang EH, Socher R, Manning CD, et al., 2012. Improving word representations via global context and multiple word prototypes. Proc 50th Annual Meeting of Association for Computational Linguistics, p.873-882.

Krishna K, Murty MN, 1999. Genetic K-means algorithm. *IEEE Trans Syst Man Cybern Part B*, 29(3):433-439.

https://doi.org/10.1109/3477.764879

Lebret R, Collobert R, 2014. Word embeddings through Hellinger PCA. Proc 14th Conf on European Chapter of the Association for Computational Linguistics, p.482-490.

Lebret R, Collobert R, 2015. Rehabilitation of count-based models for word vector representations. Int Conf on Intelligent Text Processing and Computational Linguistics, p.417-429. https://doi.org/10.1007/978-3-319-18111-0_31

Levy O, Goldberg Y, 2014. Dependency-based word embeddings. Proc 52nd Annual Meeting of Association for Computational Linguistics, p.302-308. https://doi.org/10.3115/v1/P14-2050

Liu Y, Liu ZY, Chua TS, et al., 2015. Topical word embeddings. Proc 29th AAAI Conf on Artificial Intelligence, p.2418-2424.

Luong MT, Socher R, Manning CD, 2013. Better word representations with recursive neural networks for morphology. Proc 17th Conf on Computational Natural Language Learning, p.104-113.

Mikolov T, Sutskever I, Chen K, et al., 2013. Distributed representations of words and phrases and their compositionality. Int Conf on Neural Information Processing Systems, p.3111-3119.

Mnih A, Hinton GE, 2008. A scalable hierarchical distributed language model. Proc 21st Int Conf on Neural Information Processing System, p.1081-1088.

Nguyen KA, Walde SSI, Vu NT, 2016. Neural-based noise filtering from word embeddings. Proc 26th Int Conf on Computational Linguistics, p.2699-2707.

Pennington J, Socher R, Manning CD, 2014. Glove: global vectors for word representation. Proc Conf on Empirical Methods in Natural Language Processing, p.1532-1543.

Ren YF, Wang RM, Ji DH, 2016. A topic-enhanced word embedding for Twitter sentiment classification. *Inform Sci*, 369:188-198. https://doi.org/10.1016/j.ins.2016.06.040

Ritter A, Mausam, Etzioni O, 2010. A latent Dirichlet allocation method for selectional preferences. Proc 48th Annual Meeting of Association for Computational Linguistics, p.424-434.

Rubenstein H, Goodenough JB, 1965. Contextual correlates of synonymy. *Commun ACM*, 8(10):627-633. https://doi.org/10.1145/365628.365657

Tian F, Dai HJ, Bian J, et al., 2014. A probabilistic model for learning multi-prototype word embeddings. Proc 25th Int Conf on Computational Linguistics, p.151-160.

Turney PD, Pantel P, 2010. From frequency to meaning: vector space models of semantics. *J Artif Intell Res*, 37(1):141-188. https://doi.org/10.1613/jair.2934

Wang P, Xu B, Xu JM, et al., 2016. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174(B):806-814. https://doi.org/10.1016/j.neucom.2015.09.096

Xu W, Rudnicky AI, 2000. Can artificial neural networks learn language models? Proc 6th Int Conf on Spoken Language Processing, p.202-205.

Zhai M, Tan J, Choi DJ, 2016. Intrinsic and extrinsic evaluations of word embeddings. Proc 30th AAAI Conf on Artificial Intelligence, p.4282-4283.