Copyright © 2019 Tech Science Press CMC, vol.60, no.1, pp.259-273, 2019

New Generation Model of Word Vector Representation Based on CBOW or Skip-Gram

Zeyu Xiong[1,*], Qiangqiang Shen[1], Yueshan Xiong[1], Yijie Wang[1] and Weizi Li[2]

Abstract: Word vector representation is widely used in natural language processing tasks. Most word vectors are generated based on probability model, its bag-of-words features have two major weaknesses: they lose the ordering of the words and they also ignore semantics of the words. Recently, neural-network language models CBOW and Skip-Gram are developed as continuous-space language models for words representation in high dimensional real-valued vectors. These vector representations have recently demonstrated promising results in various NLP tasks because of their superiority in capturing syntactic and contextual regularities in language. In this paper, we propose a new strategy based on optimization in contiguous subset of documents and regression method in combination of vectors, two of new models CBOW-OR and SkipGram-OR for word vector learning are established. Experimental results show that for some words-pair, the cosine distance obtained by the CBOW-OR (or SkipGram-OR) model is generally larger and is more reasonable than CBOW (or Skip-Gram), the vector space for Skip-Gram and SkipGram-OR keep the same structure property in Euclidean distance, and the model SkipGram-OR keeps higher performance for retrieval the relative words-pair as a whole. Both CBOW-OR and SkipGram-OR model are inherent parallel models and can be expected to apply in large-scale information processing.

Keywords: Distributed word vector, continuous-space language model, hierarchical softmax.

1 Introduction

A word vector representation is a mathematical processing object associated with each word. Generating word representations is an essential task of natural language processing (NLP) [Bengio, Ducharme and Vincent (2001);Collobert and Weston (2008)]. Many NLP tasks such as sentiment analysis,sentence or text classification and so on consider words as basic units. An important step is the introduction of continuous representations of words [Bengio, Ducharme, Vincent et al. (2003)]. When it comes to texts, one of the

[1] HPCL, School of Computer Science,National University of Defense Technology, Changsha, 410073, China.

[2] Department of Computer Science, University of North Carolina at Chapel Hill, 27599, USA.

[*] Corresponding Author: Zeyu Xiong. Email: xiongzeyu08@nudt.edu.cn.

most commonly used fixed-length features is bag-of-words. Traditionally, the default word representation regards a word as a one-hot vector, which shares the same size of the vocabulary. Despite its popularity, bag-of-words features have two major weaknesses: They lose the ordering of the words and they also ignore semantics of the words. In order to address these issues, Cao et al. use the histogram of the bag of words model (BOW) to determine the number of sub-images in the image that convey secret information for the purpose of improving the retrieval efficiency [Cao, Zhou, Sun et al. (2018)].

Continuous-space language models [Holger (2007); Bengio, Schwenk, Senécal et al. (2006)] are neural-network language models in which words are represented as high dimensional real-valued vectors. These vector representations have recently demonstrated promising results in various tasks [Collobert and Weston (2008); Bengio, Schwenk, Senécal et al. (2006)] due to their superiority in capturing syntactic and contextual regularities in language.

Recent works in learning vector representations of words use neural networks [Mnih and Hinton (2008); Turian, Ratinov and Bengio (2010); Mikolov, Sutskever, Chen et al. (2013)]. The outcome is that after the neural network model is trained, the word vectors are mapped into a vector space such that semantically similar words have similar vector representations. Distributed word representations draw more attention for better performance in a wider range of natural language processing tasks, ranging from speech tagging [Santos and Zadrozny (2014)], named entity recognition [Turian, Ratinov and Bengio (2010)], part-of-speech tagging, parsing [Socher, Lin, Manning et al. (2011)], semantic role labeling [Collobert, Weston, Bottou et al. (2011)], phrase recognition [Socher, Lin, Manning et al. (2011)], sentiment analysis [Socher, Pennington, Huang et al. (2011)], paraphrase detection [Socher, Huang, Pennin et al. (2011)], to machine translation [Cho, Merriënboer, Gulcehre et al. (2014)]. Han et al. [Kim, Kim and Cho (2017)] proposed a method to create concepts by clustering word vectors generated from word2vec, and used the frequencies of these concept clusters to represent document vectors.

The distributed word vector learning mainly depends on word in the vocabulary and corpus, corpus collected is generally according to time ordered in topic related or event related. In this paper, we divide documents into several subsets, in order to preserve accurate proximity information among subset, a combination model based on strategy of optimization and regression, as an extension of distributed word vector is constructed.

The rest of this paper is organized as follows: Section 2 introduces prior research related to n-gram, CBOW model and Skip-gram model. Section 3 formally presents our approach in the integrated extension model for word vector representation. Two novel models CBOW-OR and SkipGram-OR are proposed. Section 4 describes the experimental settings and experimental results. At last, we conclude the paper and discuss some future work in Section 5.

2 Related works

2.1 n-gram model

The goal of statistical language modeling [Bengio, Ducharme, Vincent et al. (2003)] is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the curse of dimensionality: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training.

Curse of dimensionality: For example, if one wants to model the joint distribution of 10 consecutive words in a natural language with a vocabulary V of size 100,000, there are potentially 100000^{10}-1 = 10^{50}-1 free parameters.

A statistical model of language can be represented by the conditional probability of the next word given all the previous ones, since

$$P\left(w_1^T\right) = \prod_{t=1}^{T} P\left(w_t \left| w_1^{t-1}\right.\right) \tag{1}$$

where w_t is the t-*th* word, $w_i^j = (w_i, w_{i+1}, \cdots w_{j-1}, w_j)$

Such statistical language models have already been found useful in many technological applications involving natural language, such as speech recognition, language translation, and information retrieval.

Following the above mentioned models, n-gram models construct tables of conditional probabilities for the next word and for each one of a large number of contexts, i.e., combinations of the last n-1 words:

$$P\left(w_t \left| w_1^t\right.\right) \approx P\left(w_t \left| w_{t-n+1}^{t-1}\right.\right) \tag{2}$$

Bengio et al. [Bengio, Ducharme, Vincent et al. (2003)] proposed a neural network model to calculate formula (2), Feature vectors of words are learned based on their probability of co-occurring in the same documents.

The training set is a sequence $w_1, w_2, \cdots, w_T$ of words belong to V, where the vocabulary V is a large but finite set. The objective is to learn a good model $f\left(w_t, \cdots w_{t-n+1}\right) = P\left(w_t \left| w_1^{t-1}\right.\right)$ that can give high out-of-sample likelihood. The model is decomposed into the following two parts:

1) A mapping C from any element i of V to a real vector $C(i) \in \mathrm{R}^m$. It represents the *distributed feature vectors* associated with each word in a vocabulary. In practice, C is represented by a $|V| \times m$ matrix of free parameters.

2) A function g maps an input sequence of feature vectors of words in context

$$\left(C\left(w_{t-n+1}\right), \cdots, C\left(w_{t-1}\right)\right)$$

to a conditional probability distribution over words in V for the next word w_t. The output of g is a vector whose i-th element estimates the probabilit $P\left(w_t = i \left| w_1^{t-1}\right.\right)$ shown in Fig.1.

$$f\left(i, w_t, \cdots w_{t-n+1}\right) = g\left(i, C\left(w_{t-1}\right), \cdots C\left(w_{t-n+1}\right)\right) \tag{3}$$

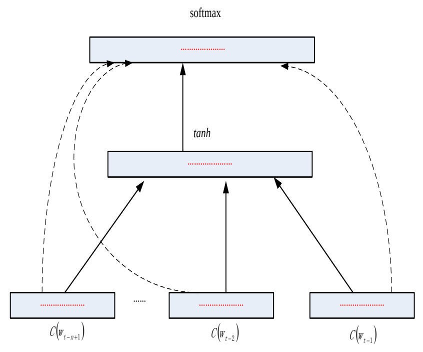


Figure 1: Neural architecture $f(i, w_t, \cdots, w_{t-n+1}) = g(i, C(W_{t-1}), \cdots, C(W_{t-n+1}))$

Training result is achieved by finding θ that maximizes a training corpus penalized log-likelihood:

$$L = \frac{1}{T}\sum_t \log f\left(w_t, w_{t-1}, \cdots, w_{t-n+1}; \theta\right) + R\left(\theta\right) \tag{4}$$

where $R\left(\theta\right)$ is a regularization term. R is a weight decay penalty applied to the weights of a neural network and to the matrix C.

The softmax output layer is calculated as follows:

$$P\left(w_t \left| w_{t-1}, \cdots, w_{t-n+1}\right.\right) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{5}$$

y_i is the unnormalized log-probability for each output word i, computed as follows, with parameters b, W, U, d and H:

$$y = b + Wx + U\tanh\left(d + Hx\right) \tag{6}$$

where the hyperbolic tangent *tanh* is applied element by element, W can be optionally set to zero (no direct connections), and x is the word features layer activation vector, which is the concatenation of the input word features from the matrix C:

$$x = (C(w_{t-1}), C(w_{t-2}), \cdots C(w_{t-n+1})) \tag{7}$$

$$\theta = (b, d, w, U, H, C) \tag{8}$$

$$\theta \leftarrow \theta + \varepsilon \frac{\partial \log P\left(w_t \left| w_{t-1}, \cdots w_{t-n+1}\right.\right)}{\partial\theta} \tag{9}$$

ε is the “learning rate”.

2.2 The word vector model

Mikolov et al. [Mikolov, Sutskever, Chen et al. (2013)] introduced the CBOW and Skip-Gram model. Both models include three levels: input, projection and output (Fig. 2 and Fig. 3). The training objective is to learn word vector representations that can predict the nearby words well.

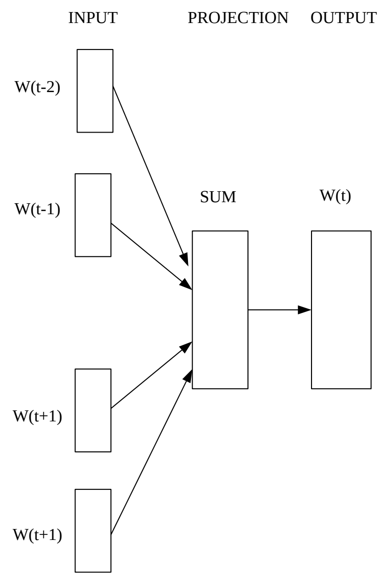


Figure 2: The CBOW model

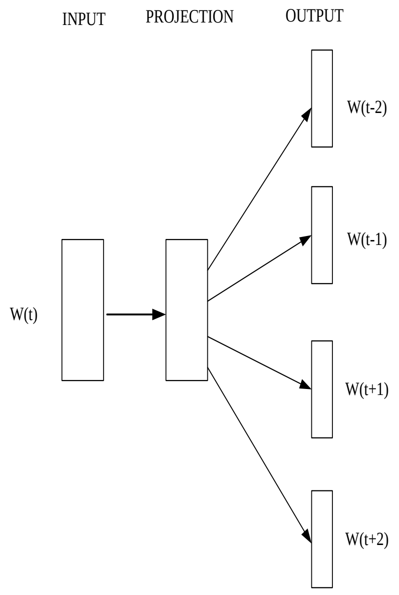


Figure 3: The Skip-gram model

Given a sequence of training words $w_1, w_2, \cdots, w_T$, the CBOW model is asked to maximize the following average log probability,

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p\left(w_t\left|w_{t+j}\right.\right) \tag{10}$$

but the Skip-Gram model is asked to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p\left(w_{t+j}\left|w_t\right.\right) \tag{11}$$

where c is the size of the training context. The basic Skip-Gram formulation of $p\left(w_{t+j}\left|w_t\right.\right)$ is defined using softmax function as follows:

$$p(w_{t+j}\left|w_t\right.) = \frac{\exp((v'_{w_{t+j}})^T v_{w_t})}{\sum\limits_{i=1}^{W}\exp((v'_{w_i})^T v_{w_t})} \tag{12}$$

where v_{w_t} is input vector representation of word w_t, and $v'_{w_{t+j}}, v'_{w_i}$ are output vector representations of words w_{t+j}, w_i. W is the number of words in a vocabulary.

There are many methods for visualizing the relationship between words vector representation. Fig. 4 shows one way for terms' relevancy: two-dimensional PCA projection of the 1000-dimensional Skip-Gram vectors of countries and their capital cities [Mikolov, Sutskever, Chen et al. (2013)]. It illustrates the ability of the model [Mikolov, Sutskever, Chen et al. (2013)] to automatically organize concepts and learn implicitly. Without any supervised information about what a capital city means during training are given before the mining relationships between them are obtained.

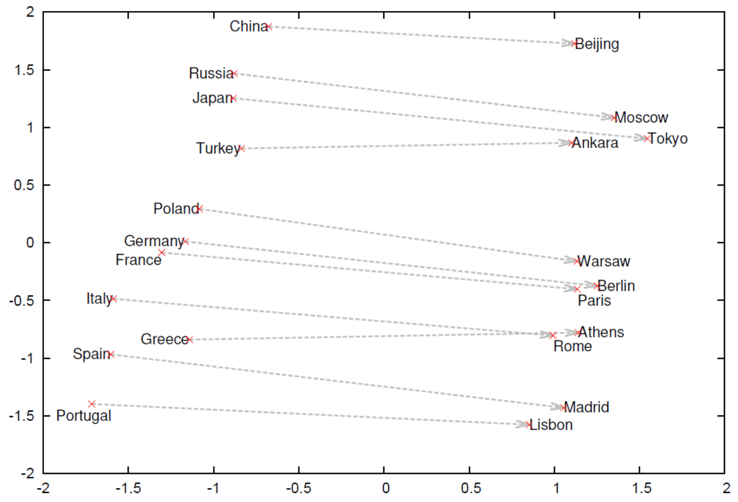


Figure 4: Country and Capital Vectors Projected by PCA [Kim, Kim and Cho (2017)]

Fig. 5 shows another visualization method for clustering same word association, t-sne clustering method is used. "t-SNE" is a technique which visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map. The technique is a variation of Stochastic Neighbor Embedding [Georey and Roweis (2002)] that is much easier to optimize, and produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map. Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities [Kim, Kim and Cho (2017)]. In t-SNE [Maaten and Hinton (2008)], it employs a Student t-distribution with one degree of freedom as the heavy-tailed distribution in the low-dimensional map.

As Fig. 5 shown that words are represented in a continuous embedded space is very important. Various conventional machine learning and data mining techniques can be applied in this space to solve various text mining tasks [Cui, Shi and Chen (2016); Bansal, Gimpel and Livescu (2014); Xue, Fu and Zhan (2014); Cao and Wang (2015); Ren, Kiros and Zemel (2015)]. Fig. 5 also shows an example of such embedded space visualized by t-sne [Cui, Shi and Chen (2016)]. The embedded words located in one circle represent the names of baseball players, the names of soccer players and the names of countries are separated in different clustering circle, and words similar meanings are located close. The words with different meanings are located far away.

2.2.1 Hierarchical softmax

The formula (12) is a full softmax model which is impractical because the cost of computing is proportional to W, which is often large (10^5–10^7 terms). The hierarchical softmax is a computationally efficient approximation of the full softmax. The main advantage of

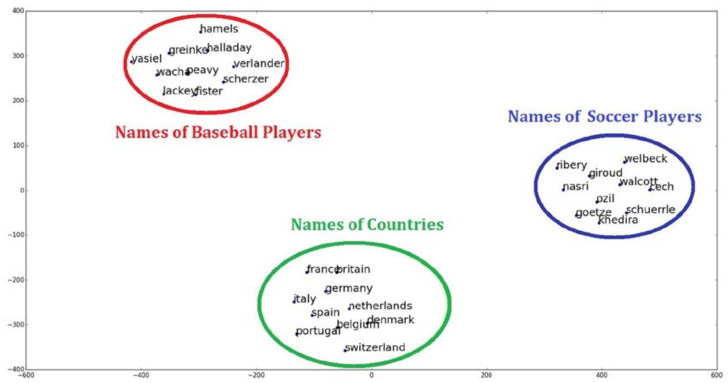


Figure 5: Embedded space using t-sne [Maaten and Hinton (2008);Kim, Kim and Cho (2017)]

hierarchical softmax is that instead of evaluating W output nodes in the neural network to obtain the probability distribution, it evaluates only $\log_2(w)$ nodes.

Hierarchical probabilistic neural network language model was first proposed by Morin [Morin and Bengio (2005)], Mnih and Hinton [Mnih and Hinton (2008)] explored a number of methods for constructing a tree structure and ameliorated the effect on both the training time and the resulting model accuracy. Mikol et al. [Mikolov, Sutskever, Chen et al. (2013); Mikolov (2012)] used a binary Huffman tree, as it assigns short codes to the frequent words which results in fast training.

The hierarchical softmax uses a binary tree representation of the output layer with the W words as its leaves. For each word w located at the leaf, let $n(w, j)$ be the j-th node on the path from the root to w, and let $Len(w)$ be the length of this path, so $n(w, 1) = root$ and $n(w, Len(w)) = w$. let $child(n)$ be an arbitrary fixed child of inner node n, and let $\langle x \rangle$ be 1 if x is true and -1 otherwise, then the hierarchical softmax is defined as follows:

$$p(w|w_I) = \prod_{j=1}^{Len(w)} \sigma\left(\langle n(w, j+1) = child(n(w, j))\rangle . (v'_{n(w,j)})^T v_{w_I}\right). \tag{13}$$

where $\sigma(x) = 1/(1 + \exp(-x))$, the cost of computing $\log p(w_O\, |w_I\,)$ and $\nabla \log p(w_O\, |w_I\,)$ is proportional to $Len(w_O)$, which on average is no greater than $\log W$.

3 New learning model of word vector representation

We first divide training document into several relative subsets of document, the relative property may be considered as document collection of contextual feature and semantic feature.

Let $Cp_1, \cdots, Cp_n$ be n subsets, $V_1(w), \cdots, V_n(w)$ be corresponding distributed word vectors for word w generated from CBOW or Skip-Gram model. Let $SAMT$ be a sampling set in the vocabulary for topic words, in order to preserve accurate proximity information among subsets, we consider a regression model as an extension of distributed word vectors. The new learning model for distributed word vector representation is described as the following optimization problem and regression strategy.

Step 1. finding parameters $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$, $\sum_{i=1}^{n} \alpha_i = 1$, which subjects to the following problem:

$$\alpha^* = \arg\min_{\alpha} \sum_{w \in SAMT} \sum_{i=1}^{n-1} \|\alpha_i V_i(w) - \alpha_{i+1} V_{i+1}(w)\|^2 \tag{14}$$

Step 2. reconstruct word vector for each word w

$$V(w) = \sum_{i=1}^{n} \alpha_i^* V_i(w) \tag{15}$$

Fig. 6 shows the integrated extension model for distributed word vector representation.

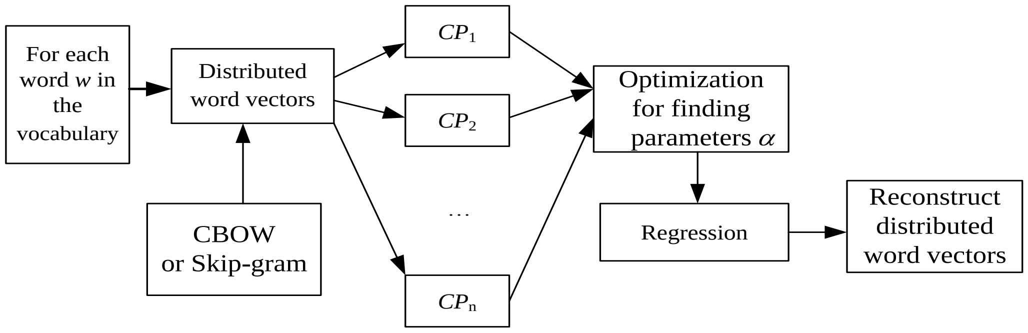

For each word w in the vocabulary
Distributed word vectors
CBOW or Skip-gram
CP1
CP2
...
CPn
Optimization for finding parameters α
Regression
Reconstruct distributed word vectors

Figure 6: The new generation model of word vector representation

4 Experiments

4.1 Task description

Our task is to develop a new method for generating word vectors and verify its efficiency on actual document dataset. The new generation model of word vector includes two submodels, one is based on the combination of CBOW [Mikolov (2012)] and our regression with optimization strategy, we denote CBOW-OR model, and the other is based on the combination of Skip-Gram [Mikolov (2012)] and our regression with optimization strategy, denote SkipGram-OR model. In the following experiments, we divide documents into three sub-documents, and on the premise of sharing same vocabulary, same word vectors are trained respectively on three subsets with same dimension. The optimization and regression methods are used to integrate the three vectors into a vector, which is regarded as the word vector of the word.

4.2 Dataset description

The dataset we are using is text8, which is download using Google word2vector. The size of the corpus is 100 MB, vocabulary size is 71291 in documents, some auxiliary words have been removed, for example, a, the, is, and so on. And some rare words are removed. In addition, the whole documents contains 4406976 words. Using method, we divide the documents into three parts in size: 36.6 MB, 36.6 MB, 26.8 MB. We name these three sub-documents as text8_1, text8_2, and text8_3. The same word is trained separately on the 3 sub-documents to obtain 3 vectors, and then a word vector is obtained by using the optimization and regression mentioned above.

4.3 Evaluation mode

In order to test the effect of our method, we design two sets of comparative experiments, one is to measure Euclidean distance of the vector in two different vector spaces with same dimension, the other is to measure cosine distance of the two word vectors in each vector space.

Three groups of experiment are conducted according to different *SAMT* in Eq. (14), for Tabs. 1 and 3, *SAMT*={cat, China, computer, dog, exam, hospital, Japan, nurse, school, software}, for Tabs. 4 and 6, *SAMT*={car, children, country, driver, hospital, nation, nurse, parent, school, students}, for Tabs. 7 and 9, *SAMT*={army, chef, friend, fruit, gentlemen, ladies, partner, restaurant, soldier, vegetables}. Four kinds of word vector space are respectively generated by CBOW, Skip-Gram, CBOW-OR and SkipGram-OR. The last two models is proposed in this paper.

In Tabs. 1, 2, 4, 5, 7, 8, we compare the Euclidean distance of the vector learned for the same word under different vector spaces. We know that a vector can be represented as a point in vector space. Since the dimension for each words of vector space is same, we can compare the Euclidean distance of a word in different spaces. We can test the relation between the Euclidean distance of multiple key words in any two different vector spaces, in order to test the structure consistency of different vector spaces.

Since the cosine distance of the two similar word vectors that are trained should be relatively large, the semantic relationship between words is similar in an article. In other words, the probability of simultaneous occurrence of two words should be large, such as cats and dogs, hospitals and nurses, school and students, etc. So in the Tabs. 3, 6, 9, we compare the cosine distance between the vector pairs of the same set of word pairs, which are obtained under three different learning mechanisms respectively, as a criterion for evaluating the vector of words.

Tab. 1 to Tab. 9 show some interesting properties: Tabs. 1, 2, Tabs. 4, 5, Tabs. 7, 8 show that the words vector space for Skip-Gram and SkipGram-OR keep the same structure property in Euclidean distance. Tab. 3 shows that there exists more accurate cosine distance between two words using SkipGram-OR model than other models of CBOW, CBOW-OR and Skip-Gram, meanwhile, Tabs. 6, 9 show that there exist more accurate cosine distance

Table 1: Euclidean distance of the vector learned from the same word under different methods

	cat	China	computer	dog	example
CBOW vs Skip-Gram	22.494564	23.070270	22.998427	23.701213	20.271844
CBOW-OR vs Skip-Gram	14.716238	13.883066	12.857164	15.436527	8.545754
CBOW vs CBOW-OR	25.066553	23.412062	23.468847	26.279802	19.599713
Skip-Gram vs SkipGram-OR	**5.1814342**	**3.9224142**	**3.7186657**	**4.7696899**	**5.6128070**
CBOW vs SkipGram-OR	22.727962	23.257147	23.103810	23.922204	19.735225

Table 2: Euclidean distance of the vector learned from the same word under different methods

	hospital	Japan	nurse	school	software
CBOW vs Skip-Gram	24.376002	20.603936	18.884017	26.130630	23.791694
CBOW-OR vs Skip-Gram	15.119891	13.874292	7.953869	14.609852	14.428911
CBOW vs CBOW-OR	24.336653	23.291979	18.729539	27.662022	24.371412
Skip-Gram vs SkipGram-OR	**4.651420**	**4.1706605**	**4.941486**	**3.7302991**	**4.1395899**
CBOW vs SkipGram-OR	24.4143274	21.396625	18.407546	25.747152	24.288571

Table 3: The cosine distance of the two word vectors

	China-Japan	computer-software	cat-dog	school-exam	hospital-nurse
CBOW	0.633185	0.562796	0.482052	0.457408	0.490181
CBOW-OR	0.556097	0.523432	0.462266	0.384344	0.571792
Skip-Gram	0.553206	0.692142	0.671334	0.457154	0.509307
SkipGram-OR	0.374356	0.654409	0.687029	0.530261	0.521168

Table 4: Euclidean distance of the vector learned from the same word under different methods

	car	children	country	drivee	hospital
CBOW vs Skip-Gram	21.497266	22.361323	18.389801	24.983802	23.996126
CBOW-OR vs Skip-Gram	14.714419	12.609392	10.146121	14.270934	14.618502
CBOW vs CBOW-OR	24.504684	22.155388	18.503288	27.343691	22.858498
Skip-Gram vs SkipGram-OR	**4.3383199**	**3.6450409**	**3.4383442**	**4.8058831**	**4.6308525**
CBOW vs SkipGram-OR	22.030651	22.331462	18.267592	25.393571	23.854217

Table 5: Euclidean distance of the vector learned from the same word under different methods

	nation	nurse	parent	school	students
CBOW vs Skip-Gram	19.372371	17.867234	20.406933	25.895824	22.387821
CBOW-OR vs Skip-Gram	13.055126	7.963500	12.252224	14.181254	15.032777
CBOW vs CBOW-OR	21.542782	18.514933	22.544448	26.205820	24.907860
Skip-Gram vs SkipGram-OR	**3.4048577**	**5.0936379**	**4.797587**	**3.680170**	**3.8840114**
CBOW vs SkipGram-OR	19.254489	18.401096	21.260329	26.273944	22.888385

Table 6: The cosine distance of the two word vectors

	car-driver	children-parent	country-nation	hospital-nurse	school-students
CBOW	0.601893	0.315147	0.578614	0.520923	0.510829
CBOW-OR	0.645621	0.363168	0.458637	0.564505	0.538678
Skip-Gram	0.679108	0.418876	0.630063	0.492796	0.694628
SkipGram-OR	0.622663	0.418477	0.594700	0.514006	0.718959

Table 7: Euclidean distance of the vector learned from the same word under different methods

	army	chef	friend	fruit	gentlemen
CBOW vs Skip-Gram	26.168615	19.233047	23.523675	23.388722	18.979354
CBOW-OR vs Skip-Gram	14.177083	7.7553412	14.745648	13.936408	7.063437
CBOW vs CBOW-OR	26.907111	19.194172	25.834257	24.922290	19.091539
Skip-Gram vs SkipGram-OR	**3.4792353**	**4.9905233**	**4.0097756**	**4.8250793**	**5.45685725**
CBOW vs SkipGram-OR	26.520399	19.503424	24.481720	23.137643	19.7534149

Table 8: Euclidean distance of the vector learned from the same word under different methods

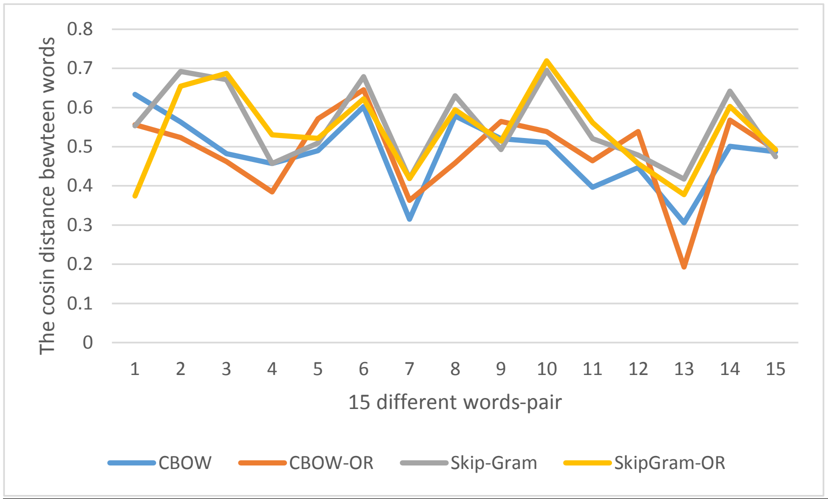
	ladies	partner	restaurant	soldier	vegetables
CBOW vs Skip-Gram	19.75417	21.064983	23.074531	20.532218	22.398374
CBOW-OR vs Skip-Gram	8.933997	11.632876	12.532995	11.880395	12.304071
CBOW vs CBOW-OR	19.118866	22.784325	21.823904	21.828869	22.398122
Skip-Gram vs SkipGram-OR	**5.0244142**	**4.5168005**	**5.0830350**	**3.9900769**	**5.3432736**
CBOW vs SkipGram-OR	20.169336	21.470190	22.823085	20.832961	22.543230

Table 9: The cosine distance of the two word vectors

	army-soldier	chef-restaurant	friend-partner	fruit-vegetables	gentlemen-ladies
CBOW	0.396233	0.446979	0.305729	0.500937	0.486808
CBOW-OR	0.464227	0.538376	0.193126	0.568196	0.492330
Skip-Gram	0.520646	0.479046	0.417140	0.641901	0.474538
SkipGram-OR	0.561824	0.457037	0.377557	0.602662	0.492939

between two words by using CBOW-OR model than other models of CBOW, SkipGram-OR and Skip-Gram.

By combining Tabs. 3, 6, 9, we get the synopsis for 15 different words towards four different models. Fig. 7 shows that the model SkipGram-OR keeps higher performance for retrieval the relative words-pair as a whole.



1-china-japan	2-computer-software	3-cat-dog	4-school-exam	5-hospital-nurse
6-car-driver	7-children-parent	8-country-nation	9-hospital-nurse	10-school-students
11-army-soldier	12-chef-restaurant	13-friend-partner	14-fruit-vegetables	gentlemen-ladies

Figure 7: the comparative result of four models for 15 different words-pair

5 Conclusions

We develop two kinds of models for generating words of vector: CBOW-OR and SkipGram-OR. The key strategy for these two models is using optimization in contiguous training documents and regression method in combination of vectors. CBOW-OR and SkipGram-OR can be performed in parallel. Experimental results show that for some words pair, the cosine distance obtained by the CBOW-OR or SkipGram-OR model is generally larger and is more reasonable than CBOW and Skip-Gram.

We also achieved exciting results. The Euclidean distance between the vectors of the same word learned under different mechanisms is nearby. It can be seen that the vector space obtained by different models has some consistency. That is, the Euclidean distance of different word vectors in any two vector spaces is approximately the same. Especially, we also find that the vector space for Skip-Gram and SkipGram-OR keep the same structure property in Euclidean distance.

Based on the inherent parallel in generating words of vector and semantic validity in words pair, the proposed models in this paper can be expected to apply in large-scale information processing.

Acknowledgement: The authors would like to thank all anonymous reviewers for their suggestions and feedback. This work Supported by the National Natural Science Foundation of China (No. 61379103,61379052), the National Key Research and Development Program (2016YFB1000101) the Natural Science Foundation for Distinguished Young Scholars of Hunan Province (Grant No. 14JJ1026), Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20124307110015).

References

Bansal, M.; Gimpel, K.; Livescu, K. (2014): Tailoring continuous word representations for dependency parsing. *Proceedings of the Annual Meeting of the Association for Computational Linguistics ACL*, pp. 809-815.

Bengio, Y.; Ducharme, R.; Vincent, P. (2001): A neural probabilistic language model. *NIPS'01*, vol. 13, pp. 932-938, Vancouver, British Columbia, Canada.

Bengio, Y.; Ducharme, R.; Vincent, P.; Jauvin, C. (2003): A neural probabilistic language model. *Journal of Machine Learning Research*, vol. 3, no. 2003, pp. 1137-1155.

Bengio, Y.; Schwenk, H.; Senécal, J.-S.; Morin, F.; Gauvain, J.-L. (2006): Neural probabilistic language models. *Innovations in Machine Learning*, pp. 137-186.

Cao, L.; Wang, F. (2015): Robust latent semantic exploration for image retrieval in social media. *Neurocomputing*, vol. 2015, no. 169, pp. 180-184.

Cao, R.; Zhou, Z.; Sun, X.; Cao, C. (2018): Coverless information hiding based onthe molecular structure images of material. *Computers, Materials & Continua*, vol. 54, no. 2, pp. 197-207.

Cho, K.; Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F. et al. (2014): Learning phrase representations using rnn encoder-de- coder for statis-tical machine translation. *EMNLP*, pp. 1724-1734.

Collobert, R.; Weston, J. (2008): A unified architecture for natural language processing: deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning*, pp. 160-167.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M. (2011): Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, vol. 12, pp. 2493-2537.

Cui, Z.; Shi, X.; Chen, Y. (2016): Sentiment analysis via integrating distributed representations of variable-length word sequence. *Neurocomputing*, vol. 2016, no. 187, pp. 126-132.

Georey, E.; Roweis, S. (2002): Stochastic neighbor embedding. *NIPS'02 Proceedings of the 15th International Conference on Neural Information Processing Systems*, vol. 15, pp. 857-864.

Holger, S. (2007): Continuous space language models. *Computer Speech & Language*, vol. 21, no. 3, pp. 492-518.

Kim, H. K.; Kim, H.; Cho, S. (2017): Bag-of-concepts: comprehending document representation through clustering words in distributed representation. *Neuro Computing*, vol. 2017, no. 266, pp. 336-352.

Maaten, L. V. d.; Hinton, G. (2008): Visualizing data using t-sne. *Journal of Machine Learning Research*, vol. 2008, no. 9, pp. 2579-2605.

Mikolov, T. (2012): *Statistical Language Models Based on Neural Networks. (Ph.D. thesis).* Brno University of Technology.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. (2013): Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems 26*, pp. 3111-3119.

Mnih, A.; Hinton, G. E. (2008): A scalable hierarchical distributed language model. *Advances in Neural Information Processing Systems*, pp. 1081-1088.

Morin, F.; Bengio, Y. (2005): Hierarchical probabilistic neural network language model. *Proceedings of the international workshop on artificial intelligence and statistics*, pp. 246-252.

Ren, V.; Kiros, R.; Zemel, R. (2015): Exploring models and data for image question answering. *Proceedings of the Advances in Neural Information Processing Systems*, pp. 2935-2943.

Santos, C.; Zadrozny, B. (2014): Learning character-level representations for part-of-speech tagging. *Proceedings of the 31st International Conference on Machine Learning*, pp. 1818-1826.

Socher, R.; Huang, E.; Pennin, J.; Manning, C.; Ng, A. (2011): Dynamic pooling and unfolding recursive auto encoders for paraphrase detection. *Advances in Neural Information Processing Systems*, pp. 801-809.

Socher, R.; Lin, C.; Manning, C.; Ng, A. (2011): Parsing natural scenes and natural language with recursive neural networks. *Proceedings of the 28th International Conference on Machine Learning*, pp. 129-136.

Socher, R.; Pennington, J.; Huang, E.; Ng, A.; Manning, C. (2011): Semi-supervised recursive auto encoders for predicting sentiment distributions. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 151-161.

Turian, J.; Ratinov, L.; Bengio, Y. (2010): Word representations: a simple and general method for semi-supervised learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384-394, Stroudsburg, PA, USA.

Xue, B.; Fu, C.; Zhan, S. (2014): A study on sentiment computing and classification of sina weibo with word2vec. *Proceedings of the IEEE International Congress on the Big Data (BigData Congress)*, pp. 358-363.