Two Improved Continuous Bag-of-Word Models

Qi Wang, Jungang Xu, Hong Chen, Ben He
School of Computer and Control Engineering
University of Chinese Academy of Sciences
Beijing, China
{wangqi615, chenhong113}@mails.ucas.ac.cn, {xujg, benhe}@ucas.ac.cn

Abstract—**Data representation is a fundamental task in machine learning, which affects the performance of the whole machine learning system. In the past few years, with the rapid development of deep learning, the models for word embedding based on neural networks have brought new inspiration to the research of natural language processing. In this paper, two kinds of schemes for improving the Continuous Bag-of-Words (CBOW) model are proposed. On one hand, the relative positions of adjacent words are taken as weights for the input layer of the model; on the other hand, the context is considered, and which can take part in the training course when the prediction of next target word is to be made. Experimental results show that our proposed models outperform the classical CBOW model.**

I. Introduction

Recently, with the rapid development of Web 2.0, the Internet data expand very fast. Based on statistics and prediction made by IDC (International Data Corporation), the amount of Internet data has reached 1.8ZB, and will increase by 50 times when the year 2020 comes. With the emergence of the large scale of unlabeled data, many researchers have proposed a number of schemes to automatically learn their patterns so as to obtain valuable information. In 2006, Hinton et al [1] proposed a fast learning algorithm and brought new inspiration into these problems. Since then, data representation techniques based on neural networks has reaped huge fruits in various fields, such as speech recognition, image processing and so on. What's more, the learning algorithms based on representation techniques have overwhelmed many traditional methods in performance.

However, in the domain of natural language processing, deep learning hasn't got such breakthroughs as that in speech and image applications. One important reason is that either image or speech consists of signal data, which infers that the similarity between different signals can be estimated simply via some distance metrics, such as Euclidean distance and etc. When it comes to text data, the issue becomes complex. For example, literal difference between two words may block the description of their relationship. People wish to find such a model that can directly estimate the similarities between text data while the model itself can be learned from large amount of unlabeled data automatically. In 1954, Harris proposed one distributed hypothesis that the words in similar context always have similar semantics, which provides theoretical basis for word representation obtained through neural networks. Based on this distributed hypothesis, various models aimed at obtaining word representation have been proposed. Additionally, the output of these models are typically known as distributed representation or word embedding. Note that the distributed representation is composed of a low-dimensional vector, where we can describe the similarities between words easily compared with the conventional BOW(Bag of Word) model. In general, the classical BOW model is also described in vector, but it is typically in some kind of high and sparse representation. In this paper, word embedding refers in particular to low-dimensional distributed representation, and which is a kind of semantic related metric and can be used to estimate the correlation between words as well as that between words and their context.Bengio et al proposed word embedding in 2003 and trained it together with model parameters in the natural language model [2].Thus, word embedding is considered as the basis of other tasks, such as text classification and part-of-speech tagging. Thus, word embedding in high quality can improve the performance of typical tasks in the domain of natural language processing.

II. Related Works

In 2003, Bengio et al proposed a language model based on feed forward neural network, known as FFNN. The model is mainly used to overcome the problem of curse of dimensionality [2], which has lied in all kinds of language models based on statistical method. In addition, Bengio et al obtained a copy of word embedding during the training course of their model. It is worth to note that the obtained word embeddings are merely coproduct of the neural network language model. Later, more and more researchers have realized the importance of word embedding and have proposed a multitude of models special for training word embedding. In 2007, M&H proposed the log-bilinear language model (LBL) [3], and then Mnih et al made a series of improvements on this model and proposed a hierarchy LBL model [4]. Later, Mikolov et al proposed the recurrent neural network based language model (RNNLM). Different from other neural network based models, the C&W model, proposed by Collobert and Weston in 2008, is the first model aiming at generating word embedding [5]. Much later, Mikolov et al proposed the Continuous Bag-of-Words (CBOW) and Skip-gram models [6], they designed these two models to train word embedding in a more efficient way, and made some simplifications among the NNLM model, the RNNLM model and the C&W model, where only most necessary parts are reserved. Mikolov et al compared their experimental results with other state-of-the-art models, and

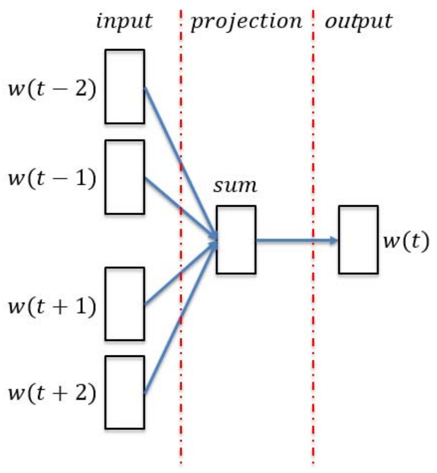

Fig. 1. Simulation results for the network.

concluded that the proposed CBOW and Skip-gram models have overwhelmed the state-of-the-art models with lower computation complexity and higher quality and accuracy. However, after plenty of deeper analysis, we find that the CBOW model has still much room for improvement.The CBOW model does not take the location information of each word into account, and only considers a fixed number of contextual information before and after the input word, which may lead to the loss of some of the important information related to the semantics.Based on the content about CBOW which can be improved,we propose two improved models.The first one is PE-CBOW model,and the second is Recurrent-CBOW model.We will show the details in Section IV.

III. The Classical Continuous Bag-of-Words Model

The classical CBOW model reserves the most necessary parts based on the NNLM model, the RNNLM model as well as the C&W model. For instance, the model has given up the method of jointing the head and the tail in turn to represent the context in the input layer, which has been widely exploited in other neural network based language models.

As shown in Figure 1, similar to the C&W model, the classical CBOW model also takes the middle word in a sequence of words as the target word. And then, the input of the model is just t words before and after the target word. As for the hidden layer, the CBOW model has made some modifications on the NNLM model, such as summing up the words in the input layer of the context as the input of the hidden layer, casting off the activation function in the hidden layer. The CBOW model exploits the hierarchy theory and some further optimizations are made as well. Prior research shows that any binary tree can be treated as the hierarchical representation of the output layer. while the CBOW model adopts the Huffman tree as its hierarchical representation, which is of the highest efficiency. Once the hidden layer x is obtained, the CBOW model makes predication of the target word according to (1).

$$p(w|c) = \frac{exp(e^{'}(w)^T x)}{\sum_{w' \in V} exp(e^{'}(w')^T x)} \tag{1}$$

Where w denotes the target word , c denotes the context of w and $e^{'}$ denotes the embedding of each word. The optimization function is defined as (2).

$$\sum_{(w,c)\in D} \log p(w|c) \tag{2}$$

IV. The Improved CBOW Models

In this section,two improved CBOW models are introduced in detail: On one hand, based on the method of summing up word embedding in the input layer, we augment weighted position encoding (PE) of each word for further operations. We call this improved model PE-CBOW model for simplicity, which uses the position encoding to measure the positional information of each word. On the other hand, based on the classical CBOW model, we enrich the context information of the target word in the hidden layer, which is circularly utilized. We call this improved model Recurrent-CBOW model for simplicity.

A. The PE-CBOW Model

Mikolov et al directly removed NNLM's hidden layer, where complicated activation functions are inside, and can obtain word embedding in a much more efficient approach. However, researches show that word order contains about 20% semantic information, while the rest part comes from what word the text picks up [7]. Considering that the word order plays an important role in understanding the inner meaning of the text, we point out that the simplification made by the CBOW model may lose some semantic related information, which is just what we want to make improvements. Based on the positional information in the context sequence of the target word, the architecture of PE-CBOW model is shown in Figure 2.

Compared with the classical CBOW model, PE-CBOW model remains to be three layers of neural network. In addition, based on word sequence $(w_{i-c}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+c})$, we make prediction of the target word w_i with the neural network. The input of the network is c words before and after w_i and positional encoding is also exploited in our model to include the word order. To be specific, each word embedding e of the input layer is multiplied by a factor l_i related with positional information instead of the prior method of solely summing up all these word embedding in the input layer. The word embedding in the hidden layer of the classical model can be computed with (3).

$$x = \sum_{w_i \in c} e(w_i) \tag{3}$$

Taken positional encoding into account, the above formula should be modified as (4).

$$x = \sum_{w_i \in c} l_i \cdot e(w_i) \tag{4}$$

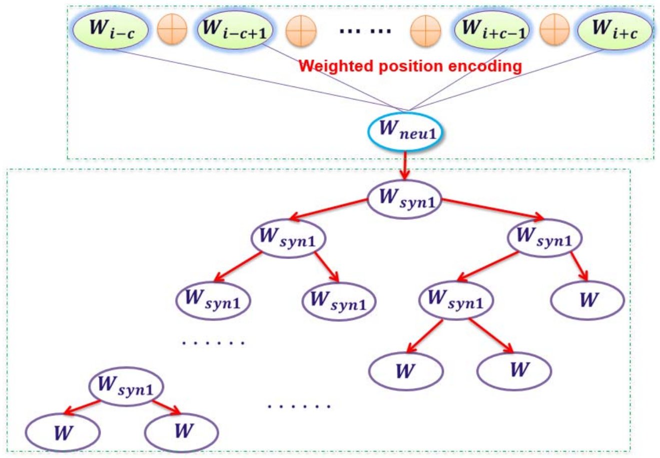

Fig. 2. The architecture of the PE-CBOW model.

Here, $\cdot$ denotes an inner product while l_i is a column vector with the same dimension as that of the word embedding. The equation can be organized as (5).

$$l_{ji} = (1 - i/I) - (j/d)(1 - 2i/I) \tag{5}$$

Where I denotes the number of words in the input layer and $i(i = 1, \cdots, I)$ denotes positional indexes of the input word sequence. Additionally, d denotes the dimension of the word embedding and $j(j = 1, \cdots, d)$ is dimensional indexes of

```
Algorithm 1.  The pseudo codes of the PE-CBOW model.
input:
    $window - the default window size
    $global_dict - global dictionary
output:
    word embeddings of the hidden layer

main steps:
1. initialize the context window $b, where $b ranges
   from 0 to $window;
2. initialize a counter $cw := 0, denoting the indexes
   of the context;
3. count the number of words in the context:
    $wnum := 2*($window-b);
4. for each word $w in the context
5.      if $w is not the target word
6.        compute the index $sent_ind in the whole
          sentence;
7.          if $sent_ind < 0 || $sent_ind > length of
            the whole sentence
8.              continue;
9.          else
10.             compute the index $dict_ind in
                $global_dict;
11.             if $dict_ind is not in $global_dict
12.                 continue;
13.             end if
14.             update the index of $w under the
                context;
15.             for each dimension $h
16.                 compute the weighted factor
                    $weight of each dimension of $w;
17.                 update the value of corresponding
                    dimension under the hidden layer
                    to: $h*$w;
18.             end for
19.         end if
20.     end if
21. end for
22. return embeddings of the hidden layer;
```

each word embedding. With (4) and (5), the obtained embeddings in the hidden layer just contains the information of word orders. The pseudo codes of the above scheme is listed in Algorithm 1.

Note that, the classical CBOW model just simply adds up all the word embeddings of the input nodes when dealing with the hidden layer, and all the positional weights of each word embedding are equivalent. By contrast, the proposed PE-CBOW model transforms the information of word orders into weighted factors of each word embedding via a novel positional computation, which is more rational.

B. The Recurrent-CBOW Model

As for large-scale corpus, the three-gram model is the most widely used model for its ideal balance between performance and computation complexity. Afterwards, NNLMs based on polynomial-fitting also make some simplifications upon the pick-up strategy of the context. For instance, the input layer of NNLMs consists of $n - 1$ words before the target word while the input layer of the CBOW model consists of c words before and after the target word. This simplification strategy upon $Context_i$ throws away parts of semantic information more or less. By contrast, RNNLM can make full use of the whole context before the target word and its overall performance outperforms the traditional N-gram models as a result. However, various problems, such that the model itself is complicated and inefficient during the training course, still lie in the classical RNNLM. Inspired by RNNLM, we propose an improved CBOW model called Recurrent-CBOW model that can make cyclic utilization of the hidden layer vector containing the context information. To be specific, we make back propagation of the hidden layer of the classical CBOW model and take the returned results as part of input, which can provide richer context information. The architecture of the Recurrent-CBOW model is shown in Figure 3.

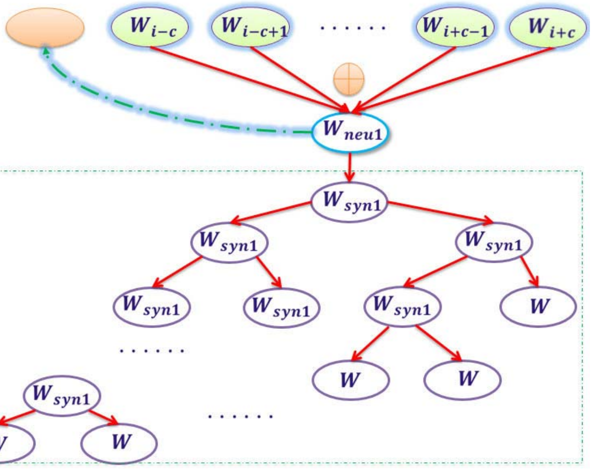

Fig. 3. The architecture of the Recurrent-CBOW model.

Compared with the original CBOW model, the Recurrent-CBOW model still includes three layers of neural network.

What make these two models different is that the Recurrent-CBOW model makes cyclic utilization of the hidden layer vector, which contains the context information. Consequently, the Recurrent-CBOW model can provide more context information for the prediction of the target word. In Figure 3, we make propagation of the hidden layer into the input layer, as shown in the yellow node. During the subsequent process, we treat the yellow node as $context_{recurrent}$ and initialize it as a zero vector. Within the training course, the window of the input layer slides forward constantly and $context_{recurrent}$ is updated to the hidden layer of the last iteration. Actually, $context_{recurrent}$ contains all the context information before the current word. Thus, we can obtain word embedding x in the hidden layer with (6).

$$x = \sum_{w_i \in c} e(w_i) + e(context_{recurrent}) \tag{6}$$

In addition, The pseudo codes of the above scheme is listed in Algorithm 2.

```
Algorithm 2.  The pseudo codes of the Recurrent-CBOW
model.
input:
    $window - the default window size
    $global_dict - global dictionary
    $context_recurrent - the preceding information of
    the target word
output:
    embeddings of the hidden layer

main steps:
1. initialize all the preceding information of the
     target word: $context_recurrent := 0;
2. initialize a counter $cw := 0, denoting the number
     of already processed words in the context;
3. for each word $w in the context
4.       if $w is not the target word
5.             compute the index $sent_ind in the whole
               sentence;
6.             if $sent_ind < 0 || $sent_ind > length of
               the whole sentence
7.                   continue;
8.             else
9.                   compute the index $dict_ind in
                     $global_dict;
10.                  if $dict_ind is not in $global_dict
11.                        continue;
12.                  end if
13.                  update the index $w in the context;
14.                  for each dimension $h in the hidden
                     layer
15.                        add all the preceding information
                           with $context_recurrent of the
                           target word to $h;
16.                  end for
17.                  update the number of processed words
                     $cw;
18.            end if
19.      end if
20. end for
21. if the number of words in the context is greater
     than 0
22.      for each dimension $h of the hidden layer
23.            do normalization for $h;
24.            update each dimension in
               $context_recurrent with $h;
25.      end for
26. end if
27. return embeddings of the hidden layer;
```

In the pseudo codes listed above, the Recurrent-CBOW model treats $context_{recurrent}$ as part of input when make prediction for the current units in the hidden layer. What's more, Line 24 denotes that the model does update for $context_{recurrent}$ with units in the hidden layer obtained in the current iteration, which provides extra information for prediction of the next target word. As a result, the Recurrent-CBOW model includes all the preceding information of the target word indeed, although a small amount of redundant information does exist.

V. Experiments

A. Experimental Design

The corpus used in our experiments is the text8 document set downloaded from Wikipedia, which is the validation set of the Word2Vec toolkit as well. In addition, we do all the experiments on a IBM server(Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz) with 2 physical CPUs and 24 virtual logical cores.

Moreover, two proposed models are designed to be evaluated with the analogy tasks of English words, with precision rates as the evaluation metric [6]. And test documents used in Word2Vec toolkit are taken as our test documents as well, where 19544 items are included and it can be divided into two general categories: the semantic analogy task items (8869 items) and the grammar analogy task items (10675 items). And the semantic analogy task items can be further divided into five subcategories:

- capital-common-countries, such as ”Athens Greece Baghdad Iraq”, 506 items;
- capital-world, such as ”Abuja Nigeria Accra Ghana”, 4524 items;
- currency, such as ”Algeria dinar Angola kwanza”, 866 items;
- city-in-state, such as ”Miami Florida Amarillo Texas”, 2467 items;
- family, such as ”boy girl grandfather grandmother”, 506 items.

Similarly, the grammar analogy task items can be further divided into nine subcategories:

- gram1-adjective-to-adverb, such as ”amazing amazingly apparent apparently”, 992 items;
- gram2-opposite, such as ”acceptable unacceptable comfortable uncomfortable”, 812 items;
- gram3-comparative, such as ”bad worse bright brighter”, 1332 items;
- gram4-superlative, such as ”bad worst bright brightest”, 1122 items;
- gram5-present-participle, such as ”dance dancing discover discovering”, 1056 items;
- gram6-nationality-adjective, such as ”Albania Albanian Argentina Argentinean”, 1599 items;
- gram7-past-tense, such as ”decreasing decreased describing described”, 1560 items;
- gram8-plural, such as ”banana bananas pineapple pineapples”, 1332 items;

TABLE I
EXPERIMENTAL RESULTS ON SEMANTIC ANALOGY TASKS (%)

Semantic analogy task	CBOW	PE-CBOW	Recurrent-CBOW
capital-common-countries	82.16	82.02	79.37
capital-world	67.42	70.07	67.08
currency	63.70	66.04	62.51
city-in-state	56.83	58.97	56.96
family	58.35	60.91	57.47

TABLE II
EXPERIMENTAL RESULTS ON GRAMMAR ANALOGY TASKS (%)

Grammar analogy task	CBOW	PE-CBOW	Recurrent-CBOW
gram1-adjective-to-adverb	20.63	20.37	22.75
gram2-opposite	22.41	22.69	23.45
gram3-comparative	45.00	46.73	47.37
gram4-superlative	44.66	45.40	46.92
gram5-present-participle	43.74	43.32	45.71
gram6-nationality-adjective	54.59	54.81	56.27
gram7-past-tense	51.76	52.03	53.03
gram8-plural	53.81	53.80	54.78
gram9-plural-verbs	51.79	52.32	53.53

- gram9-plural-verbs, such as ”decrease decreases implement implements”, 870 items.

Here,we use the method Mikolov et al proposed [6], which exploits addition and subtraction of word embeddings to implement the analogy task of words. For example, we can compute the cosine distance between word embeddings first: ”biggest” - ”big” + ”small”, and then the most similar word, denoted by the result embedding, should be the word ”smallest”. In our experiments, the text8 document set is trained with the two improved CBOW models first, and then the obtained word embeddings are applied to the above 14 document subcategories, and finally the precision rates of each analogy task are calculated. The test procedure consists of the following steps:

1) For each item in the test document, obtain embeddings of the first three words via retrieving word table obtained in document training phase;
2) A simple weighting operation is applied to the above embeddings and then the word closest to the the result is retrieved via retrieving word table obtained in document training phase;
3) Check whether each result word identifies with the provided answer in the test document;
4) Calculate the precision rate of each analogy task of words.

B. Experimental Results and Analysis

We set the same experimental environment as well as the same parameter setting in [6]. At each turn, the original CBOW model, PE-CBOW model and Recurrent-CBOW model are executed. Table I and Figure 4 show the contrastive results of these models in terms of precision rate on semantic analogy tasks, while Table II and Figure 5 show that on grammar analogy tasks.

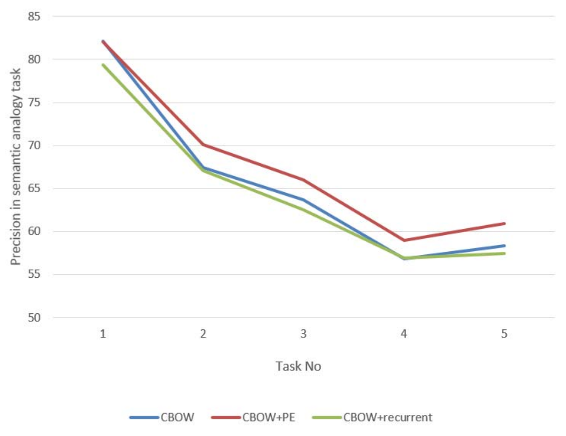

Fig. 4. The precision rates of semantic analogy task.

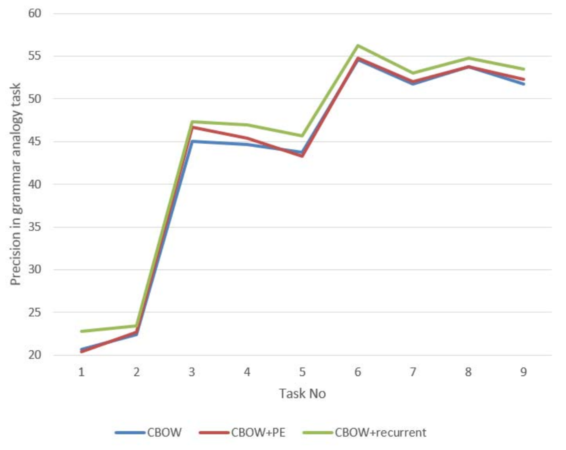

Fig. 5. The precision rates of grammar analogy task.

Intuitively, the two proposed models, PE-CBOW and Recurrent-CBOW models, overwhelm the original CBOW model in the whole performance. Moreover, we can further conclude that the main advantage of the PE-CBOW model is embodied in the semantic analogy tasks with more stable results and smaller fluctuation caused by the number of iterations or dimensions of the word embedding. As for the Recurrent-CBOW model, the experimental results reveal that it performs better on grammar analogy tasks and outperforms the original CBOW model in all aspects. Last but not the least, neither the PE-CBOW model nor the Recurrent-CBOW model introduces extra computational complexity, and which has consistent time cost with the original CBOW model.

VI. CONCLUSIONS

In this paper, we proposed two improved CBOW models. On one hand, we introduce the position encoding to supplement the missing word order information. The head-tail strategy for word embedding of the input layer, exploited in the original CBOW model, typically results in the entire loss of the word order information, which has been proved to contain 20% of the semantic information. Unlike the above strategy, we have designed much more reasonable expressions for encoding the order information into weighted factors. On the other hand, we make better use of the preceding text of the target word, which makes up the semantic loss caused by the simplification made by the original CBOW model. Inspired by RNNLM, we make back propagation of the hidden layer and take them into the input layer for cyclic utilization. Experimental results show that the two improved models outperform the original CBOW model in various aspects without introducing inefficient computations. In addition, we found that the PE-CBOW model is more suitable for processing semantic analogy tasks,because the same word may have different meanings in different positions, the addition of order information of the words will improve the semantics representation of the words. We also found that the Recurrent-CBOW model is suitable for processing grammar analogy tasks because it can use more context information. Our future work will concentrate on how to combine these two improvements together and extend them to other applications.

VII. Acknowledgments

This work is supported in part by the Beijing Natural Science Foundation under Grant No. 4162067, Beijing Science and Technology Program under Grant No. Z171100000117021 and the National Science Foundation of China under Grant No. 61472391.

References

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504-507, 2015.

[2] Y. Bengio, H. Schwenk, J. S. Senecal, F. MOrin, Gauvain and J. L., "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. 6, pp. 1137-1155, 2003.

[3] A. Mnih and G. E. Hinton, "Three new graphical models for statistical language modelling," *In Proceedings of the 24th International Conference on Machine Learning*, ACM, pp. 641-648, 2013.

[4] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," *In Advances on Neural Information Processing Systems*, MIT Press, vol. 21, pp. 1081-1088, 2008

[5] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," *In Proceedings of the 25th International Conference on Machine Learning*, ACM, pp. 160-167, 2008.

[6] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, http://arxiv.org/abs/1301.3781, 2013.

[7] Landauer and T. K., "On the computational basis of learning and cognition: Arguments from LSA," *Psychology of Learning and Motivation*, vol. 41, no. 41, pp. 43-84, 2002.