Improved Word Embeddings with Implicit Structure Information

Jie Shen **Cong Liu***
School of Data and Computer Science, Sun Yat-sen University
shenjie5@mail2.sysu.edu.cn, liucong3@mail.sysu.edu.cn

Abstract

Distributed word representation is an efficient method for capturing semantic and syntactic word relations. In this work, we introduce an extension to the *continuous bag-of-words* model for learning word representations efficiently by using implicit structure information. Instead of relying on a syntactic parser which might be noisy and slow to build, we compute weights representing probabilities of syntactic relations based on the Huffman softmax tree in an efficient heuristic. The constructed "implicit graphs" from these weights show that these weights contain useful implicit structure information. Extensive experiments performed on several word similarity and word analogy tasks show gains compared to the basic *continuous bag-of-words* model.

1 Introduction

Unsupervised word embeddings have been shown to improve many downstream NLP tasks, such as dependency parsing (Chen and Manning, 2014; Kong et al., 2014), part-of-speech tagging (Collobert et al., 2011), sentiment analysis (Socher et al., 2013) and machine translation (Devlin et al., 2014). Low-dimensional embeddings are generally learnt in a language model by maximizing the likelihood of a large corpus of raw text data. Word vectors that are close to each other are semantically related based on the *distributional hypothesis* (Harris, 1954), which states that words in similar contexts have similar meanings.

Based on the *distributional hypothesis*, many methods of building word vectors were explored. Examples of these are *SENNA* (Collobert and Weston, 2008), the *hierarchical log-bilinear* model (Mnih and Hinton, 2009), `Word2Vec` (Mikolov et al., 2013a; Mikolov et al., 2013b) and *GloVe* (Pennington et al., 2014). In this work, we introduce an extension to the *continuous bag-of-words* (CBOW) model (Mikolov et al., 2013b). The CBOW model is widely used for learning word embeddings from raw textual data, popularized via the `Word2Vec` tool. Not only does it build useful word embeddings, but it is also efficient for training and scales well to huge corpora.

In (Levy and Goldberg, 2014), based on the fact that nearby words are not necessarily syntactically related, word context is derived from dependency parse-trees relying on a syntactic parser instead of simply using the surrounding words. Only the words that have dependency relations with the center word are used as the context words, as illustrated in Figure 1.

However, syntactic parsing is a more difficult and time-consuming task than finding word embeddings. The challenge is to absorb the advantage of using syntactic information while avoiding the complexity of parsing. In this paper, we use a simple method to attach different weights to the context words to approximate the context obtained from explicit syntactic trees, such as dependency parse-trees. The method of obtaining contextual weights is based on the Huffman softmax tree in softmax period. Our method is as efficient as CBOW since we do not explicitly construct syntactic trees but only use the weights representing implicit syntactic structures.

*Cong Liu is the corresponding author.
This work is licensed under a Creative Commons Attribution 4.0 International Licence. Licence details: `http://creativecommons.org/licenses/by/4.0/`

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2408–2417, Osaka, Japan, December 11-17 2016.

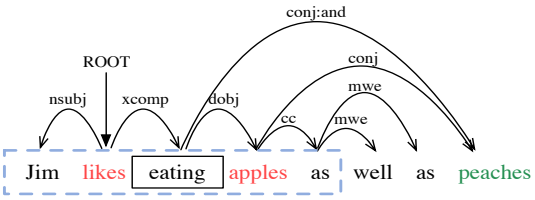

Figure 1: The enhanced dependency parse-graph for sentence "*Jim likes eating apples as well as peaches*". The blue dashed rectangle means the context window when predicting *eating* in the CBOW model.

In order to qualitatively inspect our implicit structure weights, we construct "implicit graphs" using our computed weights for different sentences. By comparing with their dependency parse-trees, these "implicit graphs" show that these weights contain useful implicit structure information.

To quantitatively evaluate the quality of the word embeddings generated from our proposed model, we experiment on several word similarity and word analogy tasks. Experiment results show gains using our method compared to the CBOW model.

The rest of the paper is organized as follows. Section 2 introduces related work on word representations. The CBOW model is briefly reviewed in Section 3. In Section 4, we present the proposed method. Evaluation and results are discussed in Section 5. Finally, Section 6 concludes the paper with future work.

2 Related Work

Word embedding is a key component in many downstream NLP tasks. Prior works explore effective and efficient methods to learn word embeddings. Bengio et al. (2006) proposed a *Neural Network Language Model* (NNLM) which predicts the distribution of the center word through several previous words. Mnih and Hinton (2007) proposed the *Log-Bilinear Language* (LBL) model which has been later accelerated by using hierarchical softmax (Mnih and Hinton, 2009) to exponentially reduce the computational complexity. Pennington et al. (2014) introduced *GloVe* which combines global matrix factorization and local context window together. Mikolov et al. (2013a; 2013b) proposed `Word2Vec` which contains two models: CBOW and Skip-Gram.

While `Word2Vec` is not sensitive to word order, many models have been proposed to deal with this problem. Ling et al. (2015) present two simple modified models: "Structured Skip-n-gram" and "Continuous Window", solving syntax-based problems. From another perspective, Levy and Goldberg (2014) use another type of context which uses word contexts derived from dependency parse-trees.

3 Continuous Bag-of-Words (CBOW)

Our departure point is the *continuous bag-of-words* (CBOW) model introduced in (Mikolov et al., 2013b). The CBOW model predicts the center word w_t given the representations of the surrounding words $w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}$, where c is a hyper-parameter defining the window size of context. The objective function is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}^{t+c}),$$

where T is the size of a sequence of words. The basic CBOW defines the probability of predicting the center word w_t using the softmax function:

$$p(w_t|w_{t-c}^{t+c}) = \frac{\exp({\mathbf{v}_t'}^{\top}\mathbf{v}_{t-c}^{t+c})}{\sum_{i=1}^{V}\exp({\mathbf{v}_i'}^{\top}\mathbf{v}_{t-c}^{t+c})},$$

where $\mathbf{v}_t^{'}$ is the word embeddings of w_t and $\mathbf{v}_{t-c}^{t+c}$ is the context representation. V is vocabulary size. $\mathbf{v}_{t-c}^{t+c}$ is calculated by summing the word representations of context words:

$$\mathbf{v}_{t-c}^{t+c} = \sum_{-c \leq j \leq c, j \neq 0} \mathbf{v}_{t+j},$$

where $\mathbf{v}_{t+j}$ is the word representation of word w_{t+j}.

In order to improve the computation speed of the full softmax layer, several efficient extensions are proposed including hierarchical softmax and negative sampling (Mikolov et al., 2013b).

4 CBOW with Implicit Structure Information

We will discuss the proposed CBOW-CW model in three parts. Section 4.1 explains the advantages of adding implicit structure information to CBOW. In Section 4.2, we introduce a more advanced model using implicit structure information to offer theoretical soundness of the later CBOW-CW model. Then we derive a simplified method CBOW-CW what we actually have done in this paper in Section 4.3.

4.1 Motivation

While the window size c is a hyper-parameter of the CBOW model which is fixed before the training starts, the model samples the actual window size between 1 and c uniformly for each token in actual implementation. This scheme is equivalent to weighting according to distances from the center word divided by the window size (Levy et al., 2015).

However, this might not be reasonable, since a word from a large distance away can also be informative. For instance, in Figure 1, the words *apples* and *peaches* are equally important for predicting the word *eating*, while their relative distances to *eating* are different.

To solve this problem, Levy and Goldberg (2014) use word contexts derived from the dependency parse-trees. Figure 1 shows an enhanced dependency parse-graph which is generated using the Stanford parser (Chen and Manning, 2014). In the CBOW model, when predicting the word *eating*, the context words are *Jim*, *likes*, *apples* and *as* for a size-2 window. However, in the model of Levy and Goldberg (2014), the context words are words that have dependency relations with the center word *eating*: *likes*, *apples* and *peaches*. Note that the word *peaches*, one of the modifiers of *eating* that is not a context word in CBOW model, is now taken into the set of context words.

However, this method relies on syntactic parsing which is time-consuming. The challenge is how to absorb the advantages of a syntax-based context while avoiding the complexity of parsing. In the following, we first introduce a more advanced model using implicit structure information. Then we derive a simplified method CBOW-CW to attach different weights to the context words to approximate the effect of different context words obtained from syntactic trees.

4.2 Advanced Implicit Syntax-based Model

In this section, we present a more advanced implicit-syntax-based model. The implicit-syntax-based model assigns a weight α_{t+j} to each context word w_{t+j} given a center word w_t. If we could rely on a syntactic parser, we could compute α_{t+j} by summing up the number of syntactic relations between w_{t+j} and w_t on syntactic trees. Alternatively, in the implicit-syntax-based model, we assume a neural network to predict the weights $\boldsymbol{\alpha}$ using the center word and its context words, without relying on explicit syntactic trees.

To summarize, the implicit-syntax-based model contains two components as illustrated in Figure 2. The first component sums up the word vectors of the context words with weights, and then passes the sum to the softmax layer to predict the probability of the center word. The second component is a neural network that predicts these weights. To train them, we can use the alternative optimization method which optimizes one component at a time assuming the other component is optimal. To simplify discussion, we temporarily assume that a full softmax model is used. The first CBOW-with-weights component predicts the center word by:

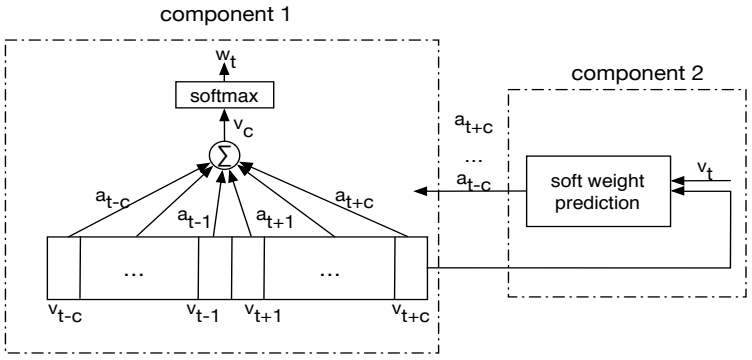

Figure 2: The two components in the implicit syntax-based model.

$$p(w_t|w_{t-c},\cdots,w_{t-1},w_{t+1}\cdots,w_{t+c}) = softmax(U \sum_{-c\leq j\leq c, j\neq 0} \alpha_{t+j}\mathbf{v}_{t+j}) = softmax(U\boldsymbol{\alpha}_t V_t^\intercal),$$

where U is the parameter in the full softmax layer, and the i-th column in U represents the expected context vector $\mathbf{u}_i$ of each word w_i in the vocabulary, $\boldsymbol{\alpha}_t$ is a vector consists of weights $\alpha_{t-c},\cdots,\alpha_{t-1},\alpha_{t+1},\cdots,\alpha_{t+c}$ and $V_t^\intercal$ is the transpose of the matrix concatenating the word embeddings of the context words, i.e. $\mathbf{v}_{t-c},\cdots,\mathbf{v}_{t-1},\mathbf{v}_{t+1}\cdots,\mathbf{v}_{t+c}$. During an iteration in the alternative optimization, the "soft weights" predictor component is optimized in order to predict a weight α_t for each center word w_t, such that the weighted sum $\mathbf{v}_c$ of the context word vectors approximates the expected context vector $\mathbf{u}_t$ of the center word. However, we can use a simple method to approximate the "soft weights" predictor component, since the optimal $\boldsymbol{\alpha}_t$ such that $\boldsymbol{\alpha}_t V_t^\intercal = \mathbf{u}_t$ can be solved analytically, assuming that the CBOW-with-weights is optimal, by $\boldsymbol{\alpha}_t = \mathbf{u}_t(V_t^\intercal)^{-1}$.

However, the actual CBOW model uses a Huffman softmax tree instead of a full softmax tree for improving performance, where context vectors $\mathbf{u}_t$ cannot be obtained directly. Therefore, we design a simple heuristic to calculate $\boldsymbol{\alpha}_t$ as described in Section 4.3. To sum up, our simple CBOW-CW model is an efficient approximation to the more advanced implicit-syntax-based model described above.

4.3 CBOW with Context Weights (CBOW-CW)

Now that the advanced implicit-syntax-based model offers the theoretical soundness of our idea, we can design our model to be simple, so that it is efficient to run and incremental to implement based on CBOW and, hopefully, other word embedding algorithms. We introduce the CBOW-CW model which is actually what we have done in this paper.

This work generalizes the CBOW model by attaching different weights α's to different context words. We denote $\mathbf{v}_{t-c}^{t+c}$ as the context representation of a center word w_t. It is the weighted sum of the word vectors $\mathbf{v}_{t+j}$ of context words of w_t, and is defined below:

$$\mathbf{v}_{t-c}^{t+c} = \sum_{-c\leq j\leq c, j\neq 0} \alpha_{t+j}\mathbf{v}_{t+j}.$$

In Levy and Goldberg (2014), a dependency parse-tree is used to determine the context of each word. For a center word, its neighbor nodes, including its head word and its modifier words in the dependency parse-tree are assigned a weight 1, and the other words are assigned a weight 0 in a sense. Instead of using a single best dependency parse-tree and assigning context words with "hard weights", i.e. 0's and 1's, a natural alternative is to ask a dependency parser to return a number of best dependency parse-trees and their probabilities, and then assign context words with "soft weights". In CBOW-CW, we define "soft weights" in the same sense, but we compute them using a simple and efficient method that does not explicitly rely on dependency parse-trees. The concrete method we use to compute the "soft weights" is described in the following part.

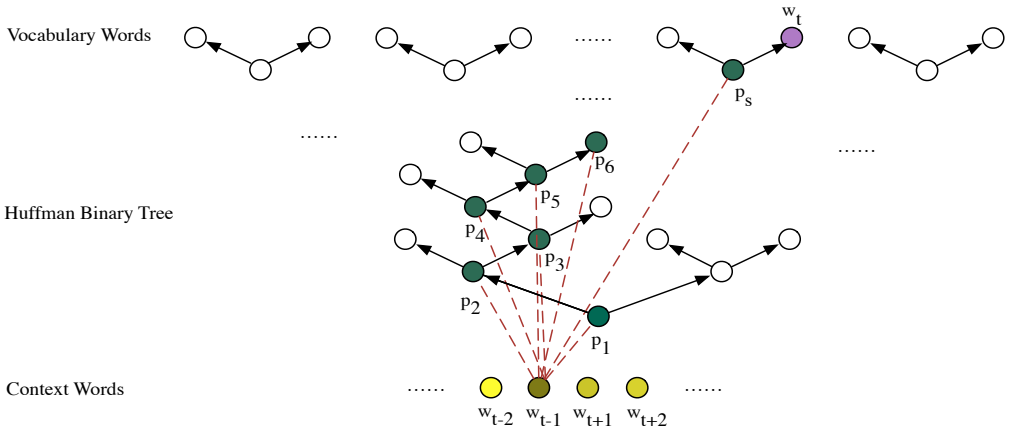

Figure 3: Computing the "soft weight" for each word w_{t+j} in the Huffman softmax tree of CBOW. Nodes on the top are the words in the vocabulary, which are leaf nodes of the binary softmax tree. The $P_t = \{p_1, p_2, \ldots, p_s\}_t, w_t$ is a path from the root node to the center word w_t in the Huffman softmax tree, where p_1 is the root node. The red dashed lines indicate the dot product of a context word vector and an vector associated with the nodes in the softmax tree.

Computation of the "Soft Weights" Before discussing the details of CBOW-CW, we need to talk about the Huffman softmax tree in CBOW. CBOW uses a Huffman softmax tree to accelerate its softmax layer. As shown in Figure 3, the leaf nodes of the Huffman softmax tree are the words to predict in the vocabulary, and each internal node is associated with a vector. To predict a word given its context vector $\mathbf{v}_c$, a path $P = \{p_1, p_2, \cdots, p_s, w_t\}$ is found from the root node p_1 of the tree to the predicted leaf node w_t. This path is determined hop-by-hop: the next node p_{k+1} of the partially computed path $\{p_1, p_2, \cdots, p_k\}$ is determined by the sign of the inner-product $\mathbf{v}_c \cdot \mathbf{v}_k$, where the vector $\mathbf{v}_k$ is associated with p_k. If the inner product is positive, p_{k+1} is the left child of p_k, otherwise the right child of p_k.

In CBOW-CW, we compute the "soft weight" α_{t+j} of context word w_{t+j} given the center word w_t as follows. Let $P_t = \{p_1, p_2, \cdots, p_s, w_t\}$ be the path from the root p_1 to the center word w_t in the Huffman softmax tree. We know that for a particular context word, once a "hop" is wrong, the rest of the path will be wrong. The weight α_{t+j} is vaguely defined as its "contribution" in finding the path P_t. This "contribution" is here defined as the number of correct next hops on path P_t that are computed, if the context vector $\mathbf{v}_c$ was replaced by $\mathbf{v}_{t+j}$. That is, the number of correct hops on the path that can be predicted, if we use context word w_{t+j} alone instead of the complete set of context words of w_t. The weight is a measure of similarity between a context word vector and the sum of context vectors. Concretely, the weight α_{t+j} for the context word w_{t+j} is defined as:

$$\alpha_{t+j} = \frac{\sum_{1 \le k \le s} 1\{(\mathbf{v}_k \cdot \mathbf{v}_{t+j}) \cdot (\mathbf{v}_k \cdot \mathbf{v}_c) > 0\}}{Z},$$

where s is the length of the path P_t. The indicator function $1\{x\}$ returns 1 iff x is $true$. $\mathbf{v}_k$ is the vector associated with node p_k on path P_t. $\mathbf{v}_{t+j}$ is the word vector of context word w_{t+j}. Z is the normalization factor: $Z = \sum_{-c \le j \le c, j \ne 0} \alpha_{t+j}$.

5 Experiments

To qualitatively analysis our method of attaching implicit syntactic weights, we make comparisons of CBOW-CW and CBOW, and we also visualize "implicit graph" for several sentences to compare with dependency parse-trees. To quantitatively evaluate the quality of the word embeddings generated from the proposed model, we experiment on several word similarity and word analogy tasks.

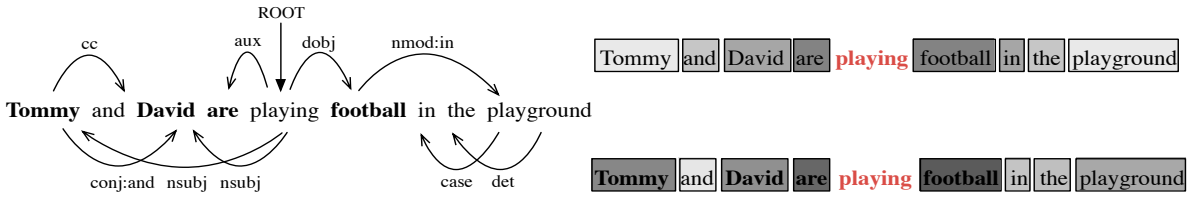

Figure 4: A sentence "*Tommy and David are playing football in the playground* " and its extended dependency parse-tree (top). The context weights in CBOW model (middle). The context weights in CBOW-CW (bottom). The center word is "playing". Darker colors indicate larger weights.

5.1 Comparison of CBOW-CW and CBOW

As illustrated in Figure 4, we show the difference between CBOW-CW and the CBOW using the sentence "*Tommy and David are playing football in the playground* " with the center word *playing* and the context window size 4. The dependency parse-graph is generated using the Stanford parser (Chen and Manning, 2014). In the CBOW model, the context words are *Tommy*, *and*, *David*, *are*, *football*, *in*, *the* and *playground*. The weights of the context words become smaller as their distances to the center word increase. However, one can see that *Tommy*, *David*, *are* and *football* are modifiers of *playing* so that they should be more important for predicting *playing*. In contrast, the proposed CBOW-CW model is able to attach more reasonable weights to the context words: the weights of *Tommy*, *David*, *are* and *football* are higher than most of the other words in the sentence.

5.2 Visualization of Implicit Syntactic Graph

In order to inspect our implicit syntactic weights, we visualize the "implicit graph" constructed using the "soft weights" for several sentences as illustrated in Figure 5. These sentences have different grammatical structures and they are chosen randomly. We draw undirected edges between two words if the inner-product of their word vectors exceed a threshold. The constructed graphs are not identical to the dependency parse-graphs. The dependency parse-graphs are built on human defined syntactic relations. There are other relations between words in reality, and therefore our "soft weights" are less restricted. Nevertheless, from the comparison of the "implicit graphs" and the enhanced dependency parse-graphs, we can find that our efficiently computed "implicit graphs" implicitly contain some dependency relations. This shows that our "soft weights" contain useful implicit structure information.

5.3 Word embeddings

We experiment with a large number of hyper-parameters. The space of hyper-parameters explored in this work is shown in Table 1. `win`, `neg`, `dim` and `ite` denote the window size, the number of negative examples, the dimension of word vectors and training iterations, respectively. The model will discard words that appear less than `min` times.

Training Corpora We built word embeddings using the original CBOW implementation[1] and our modified model on an English Wikipedia dump[2] containing about 1,989 million words, pre-processed by removing non-textual elements, sentence splitting and tokenization. The default value of the hyper-parameter `min` is 5, which means filtering out words with less than 5 instances and resulting in a vocabulary of 1,953,057 words. We also use the text8 corpus provided in `Word2Vec` package as a smaller dataset. The text8 corpus is the first 10^8 bytes of fil9, which is a 715 MB file filtered from the 1 GB file enwiki9.

Training Details For both the original CBOW and our CBOW-CW, we set the learning rate to the default value 0.05 and decrease it as the training process (Mikolov et al., 2013a). The settings of other hyper-parameters are showed in Table 1.

[1]https://code.google.com/p/word2vec/
[2]Collected in November of 2015.

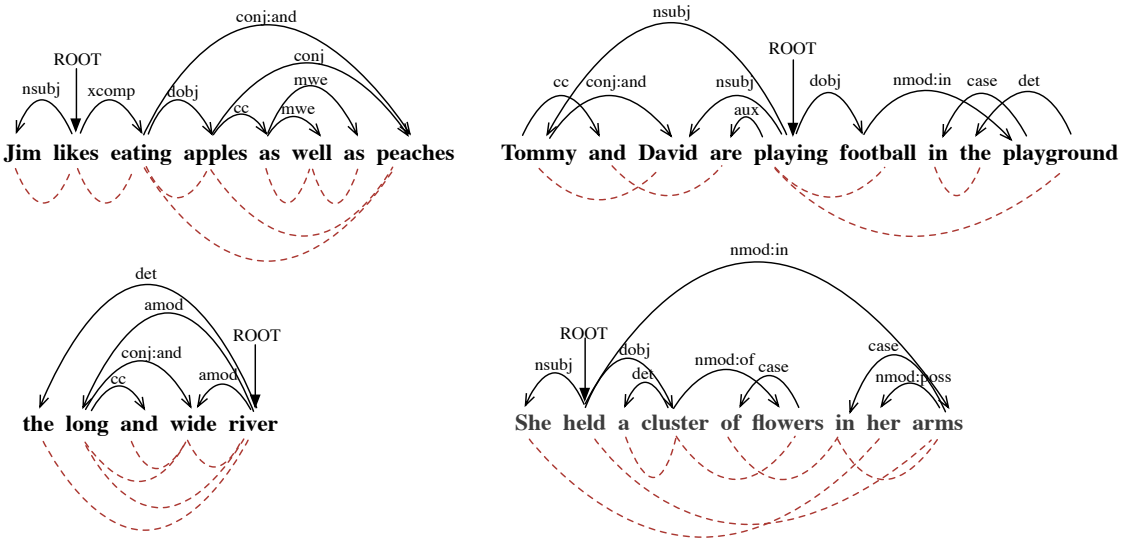

Figure 5: Constructed "implicit graphs" for several sentences. The enhanced dependency parse-graphs (shown in dark solid curves) above the sentences and the visualization of "implicit graphs" using our implicit syntactic weights (shown in red dashed curves) below the sentences.

Hyper-params	values	enwiki	text8
`win`	3, 5, 10, 20	10	10
`neg`	0, 8	8	8
`dim`	50, 100, 200, 300	300	200
`min`	5, 20, 50, 400	5	5
`ite`	5, 10	10	10

Table 1: The space of hyper-parameters explored in this work (the second column). The setting of hyper-parameters at which we report the results when using the enwiki corpus (the third column). The setting of hyper-parameters at which we report the results when using the text8 corpus (the forth column).

5.4 Test Datasets

We evaluate the word representations on several word similarity and word analogy tasks.

Word Similarity The word similarity computation is a traditional task for evaluating word embeddings. We used four datasets to evaluate word similarity including *WordSim353* (ws) (Finkelstein et al., 2001) partitioned into two datasets, *WordSim Similarity* (wss)and *WordSim Relatedness* (wsr) (Zesch et al., 2008; Agirre et al., 2009); Bruni et al.'s (2012) *MEN* dataset; Radinsky et al.'s (2011) *Mechanical Turk* (MTurk) dataset; the *TOEFL* dataset (Landauer and Dumais, 1997).

The first three datasets contain word pairs together with human-annotated similarity scores. The word embeddings are evaluated by ranking according to their cosine similarities and calculate the Spearman's rank correlation(Spearman's ρ) with the human-assigned scores. For the *TOEFL* set, we choose the nearest neighbor of the question word from the 4 candidates based on the cosine distance and use the accuracy to measure the performance.

Word Analogy The analogy dataset presents questions in the form of "a is to a^* as b is to b^*" in which b^* needs to be guessed from the entire vocabulary. Mikolov et al. (2013c) found that the learned word representations capture meaningful syntactic and semantic regularities referred to as *linguistic regularities*. We apply the Google's analogy dataset (Mikolov et al., 2013a) to evaluate word analogy performance. It contains 19,544 questions divided into two categories: 8,869 semantic questions and 10,675 syntactic questions. Such questions are answered through finding the word vector v_{b^*} which has the maximum cosine distance to the vector "$v_{a^\star} - v_a + v_b$" (Levy et al., 2014): $\arg\max_{b^\star \in V}(\cos(b^\star, b + a^\star - a)$. The

Results on the enwiki corpus (%)									
Method	similarity						analogy		
	ws	wss	wsr	MEN	MTurk	TOEFL	tot	sem	syn
CBOW	67.64	73.72	61.91	71.55	64.19	**80.00**	**68.41**	77.10	**61.20**
CBOW-CW	**69.00**	**73.77**	**65.06**	**72.84**	**67.17**	**80.00**	67.98	**79.03**	58.80
Results on the text8 corpus (%)									
Method	similarity						analogy		
	ws	wss	wsr	MEN	MTurk	TOEFL	tot	sem	syn
CBOW	69.02	70.22	66.40	64.06	60.35	70.00	30.86	30.10	31.40
CBOW-CW	**72.03**	**73.36**	**70.84**	**65.59**	**63.55**	**73.75**	**37.28**	**42.04**	**33.89**

Table 2: Performance of CBOW and CBOW-CW (our work) on different word similarity and word analogy tasks. The best result for each dataset is highlighted in **bold**.

evaluation metric for word analogy is the percentage of questions for which the $\arg\max$ result is the correct answer.

5.5 Results and Analysis

We compare the proposed CBOW-CW model with the original CBOW model and report the results using the settings in Table 1 for the text8 and enwiki corpus, respectively. Other settings of hyper-parameters can also obtain similar results.

Word Similarity The left of Table 2 shows the results for several word similarity tasks. The best result for each dataset is highlighted in **bold**. We found that in almost all of these datasets, except for the *TOEFL* dataset when training on enwiki corpus, the proposed model performs better than the original CBOW model. The results indicate that the method of appending different weights to different context words is effective and the heuristic for computing the "implicit weights" is helpful for building word embeddings. When training on the enwiki corpus, the CBOW-CW model can obtain similar results compared to the CBOW model on the *TOEFL* dataset. We suspect this maybe due to the fact that the *TOEFL* dataset is composed of low-frequency words and our model favors only high-frequency words since the Huffman softmax tree is built according to word frequency.

Word Analogy Table 2 also shows the results of the word analogy tasks. The CBOW-CW model does not perform well as it does on the word similarity tasks compared with the CBOW. When training on the large enwiki corpus, the proposed model performs better in terms of semantic accuracy but not so well in terms of syntactic accuracy. When training on the text8 corpus, CBOW-CW achieves pretty well improvement, especially on semantic accuracy where it increases nearly 12 percent compared with CBOW. Those results show that CBOW-CW is better in catching semantic information than syntactic information.

By comparing the results training on enwiki and text8, we interestingly find that our model provides a large accuracy gain over CBOW with small training data. This provides an advantage for small training corpora and indicates that our model has higher convergence rate than the original CBOW model. We are still exploring the reason of this, and we suspect that maybe due to that the method of computing weights is biased by the depth of the word in the Huffman tree. The shallow of the word in the Huffman tree, the higher accuracy of computing the weights.

Regarding the computation speed, the CBOW-CW model runs at a rate around a quarter of the rate of CBOW due to the additional computation for context weights. Even so, our model is capable of training word embeddings on the largest corpora in one day.

6 Conclusions

We proposed an extension to the CBOW model by adding implicit structure information. Our method absorbed the advantage of using a dependency tree-based context while avoiding the complexity of it. In future work, we will conduct more evaluations with other weighting schemes.

Acknowledgements

This work was funded in part by the National Science Foundation of China (grant 61472459).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *In Proc. of EMNLP*. Citeseer.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *AAAI*, volume 8, pages 861–866.