Co-learning of Word Representations and Morpheme Representations

Siyu Qiu
Nankai University
Tianjin, 300071, China
ppqq2356@gmail.com

Qing Cui
Tsinghua University
Beijing, 100084, China
cuiq12@mails.tsinghua.edu.cn

Jiang Bian
Microsoft Research
Beijing, 100080, China
jibian@microsoft.com

Bin Gao
Microsoft Research
Beijing, 100080, China
bingao@microsoft.com

Tie-Yan Liu
Microsoft Research
Beijing, 100080, China
tyliu@microsoft.com

Abstract

The techniques of using neural networks to learn distributed word representations (i.e., word embeddings) have been used to solve a variety of natural language processing tasks. The recently proposed methods, such as CBOW and Skip-gram, have demonstrated their effectiveness in learning word embeddings based on context information such that the obtained word embeddings can capture both semantic and syntactic relationships between words. However, it is quite challenging to produce high-quality word representations for rare or unknown words due to their insufficient context information. In this paper, we propose to leverage morphological knowledge to address this problem. Particularly, we introduce the morphological knowledge as both additional input representation and auxiliary supervision to the neural network framework. As a result, beyond word representations, the proposed neural network model will produce morpheme representations, which can be further employed to infer the representations of rare or unknown words based on their morphological structure. Experiments on an analogical reasoning task and several word similarity tasks have demonstrated the effectiveness of our method in producing high-quality words embeddings compared with the state-of-the-art methods.

1 Introduction

Word representation is a key factor for many natural language processing (NLP) applications. In the conventional solutions to the NLP tasks, discrete word representations are often adopted, such as the 1-of-v representations, where v is the size of the entire vocabulary and each word in the vocabulary is represented as a long vector with only one non-zero element. However, using discrete word vectors cannot indicate any relationships between different words, even though they may yield high semantic or syntactic correlations. For example, while *careful* and *carefully* have quite similar semantics, their corresponding 1-of-v representations trigger different indexes to be the hot values, and it is not explicit that *careful* is much closer to *carefully* than other words using 1-of-v representations.

To deal with the problem, neural network models have been widely applied to obtain word representations. In particular, they usually take the 1-of-v representations as the word input vectors in the neural networks, and learn new distributed word representations in a low-dimensional continuous embedding space. The principle of these models is that words that are highly correlated in terms of either semantics or syntactics should be close to each other in the embedding space. Representative works in this field include feed-forward neural network language model (NNLM) (Bengio et al., 2003), recurrent neural network language model (RNNLM) (Mikolov et al., 2010), and the recently proposed continues bag-of-words (CBOW) model and continues skip-gram (Skip-gram) model (Mikolov et al., 2013a).

However, there are still challenges for using neural network models to achieve high-quality word embeddings. First, it is difficult to obtain word embeddings for emerging words as they are not included in the vocabulary of the training data. Some previous studies (Mikolov, 2012) used one or more default indexes to represent all the unknown words, but such solution will lose information for the new words.

Second, the embeddings for rare words are often of low quality due to the insufficient context information in the training data.

Fortunately, semantically or syntactically similar words often share some common morphemes such as roots, affixes, and syllables. For example, *probably* and *probability* share the same root, i.e., *probab*, as well as the same syllables, i.e., *pro* and *ba*. Therefore, morphological information can provide valuable knowledge to bridge the gap between rare or unknown words and well-known words in learning word representations. In this paper, we propose a novel neural network architecture that can leverage morphological knowledge to obtaining high-quality word embeddings. Specifically, we first segment the words in the training data into morphemes, and then employ the 1-of-v representations of both the words and their morphemes as the input to the neural network models. In addition, we propose to use morphological information as auxiliary supervision. Particularly, in the output layer of the neural network architecture, we predict both the words and their corresponding morphemes simultaneously. Moreover, we introduce extra coefficients into the network to balance the weights between word embeddings and morpheme embeddings. Therefore, in the back propagation stage, we will update the word embeddings, the morpheme embeddings, and the balancing coefficients simultaneously.

Our proposed neural network model yields two major advantages: on one hand, it can leverage three types of co-occurrence information, including co-occurrence between word and word (conventional), co-occurrence between word and morpheme (newly added), and co-occurrence between morpheme and morpheme (newly added); on the other hand, this new model allows to learn word embeddings and morpheme embeddings simultaneously, so that it is convenient to build the representations for unknown words from morpheme embeddings and enhance the representations for rare words. Experiments on large-scale public datasets demonstrate that our proposed approach can help produce improved word representations on an analogical reasoning task and several word similarity tasks compared with the state-of-the-art methods.

The rest of the paper is organized as follows. We briefly review the related work on word embedding using neural networks in Section 2. In Section 3, we describe the proposed methods to leverage morphological knowledge in word embedding using neural network models. The experimental results are reported in Section 4. The paper is concluded in Section 5.

2 Related Work

Neural Language Models (NLMs) (Bengio et al., 2003) have been applied in a number of NLP tasks (Collobert and Weston, 2008) (Glorot et al., 2011) (Mikolov et al., 2013a) (Mikolov et al., 2013b) (Socher et al., 2011) (Turney, 2013) (Turney and Pantel, 2010) (Weston et al.,) (Deng et al., 2013) (Collobert et al., 2011) (Mnih and Hinton, 2008) (Turian et al., 2010). In general, they learn distributed word representations in a continuous embedding space. For example, Mikolov et al. proposed the continuous bag-of-words model (CBOW) and the continuous skip-gram model (Skip-gram) (Mikolov et al., 2013a). Both of them assume that words co-occurring with the same context should be similar. Collobert et al. (Collobert et al., 2011) fed their neural networks with extra features such as the capital letter feature and the part-of-speech (POS) feature, but they still met the challenge of producing high-quality word embeddings for rare words.

Besides using neural network, many different types of models were proposed for estimating continuous representations of words, such as the well-known Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). However, Mikolov et al. (Mikolov et al., 2013c) have shown that words learned by neural networks are signicantly better than LSA for preserving linear regularities while LDA becomes computationally expensive on large datasets.

There were a lot of previous attempts to include morphology in continuous models, especially in the speech recognition field. Represent works include Letter n-gram (Sperr et al., 2013) and feature-rich DNN-LMs (Mousa et al., 2013). The first work improves the letter-based word representation by replacing the 1-of-v word input of restricted Boltzman machine with a vector indicating all n-grams of order n and smaller that occur in the word. Additional information such as capitalization is added as well. In the model of feature-rich DNN-LMs, the authors expand the inputs of the network to be a mixture of selected full words and morphemes together with their features such as morphological tags. Both of these works intend to capture more morphological information so as to better generalize to unknown or rare words and to lower the out-of-vocabulary rate.

There are some other related works that consider morphological knowledge when learning the word embeddings, such as factored NLMs (Alexandrescu and Kirchhoff, 2006) and csmRNN (Luong et al., 2013), both of which are designed to handle rare words. In factored NLMs, each word is viewed as a vector of shape features (e.g., affixed, capitalization, hyphenation, and classes) and a word is predicted based on several previous vectors of factors. Although they made use of the co-occurrence of morphemes and words, the context information is lost after chopping the words and feeding the neural network with morphemes. In our model, we also utilize the co-occurrence information between morphemes, which has not been investigated before. In csmRNN, Luong *et al* proposed a hierarchical model considering the knowledge of both morphological constitutionality and context. The hierarchical structure looks more sophisticated, but the relatedness of words with morphological similarity are weaken by layers when combining morphemes into words. In addition, the noise accumulated in the hierarchical structure in building a word might be propagated to the context layer. In our model, the morphological and contextual knowledge are combined in parallel, and their contributions to the input vector are decided by a pair of learned tradeoff coefficients.

3 The Morpheme powered CBOW Models

In this section, we introduce the architecture of our proposed neural network model based on the CBOW model. In CBOW (see Figure 1), a sliding window is employed on the train text stream to obtain the training samples. In each sliding window, the model aims to predict the central word using the surrounding words as the input. Specifically, the input words are represented in the 1-of-v format. In the feed-forward process, these input words are first mapped into the embedding space by the same weight matrix M, and then the embedding vectors are summed up to a combined embedding vector. After that, the combined embedding vector is mapped back to the 1-of-v space by another weight matrix M', and the resulting vector is used to predict the central word after conducting softmax on it. In the back-propagation process, the prediction errors are propagated back to the network to update the two weight matrices. After the

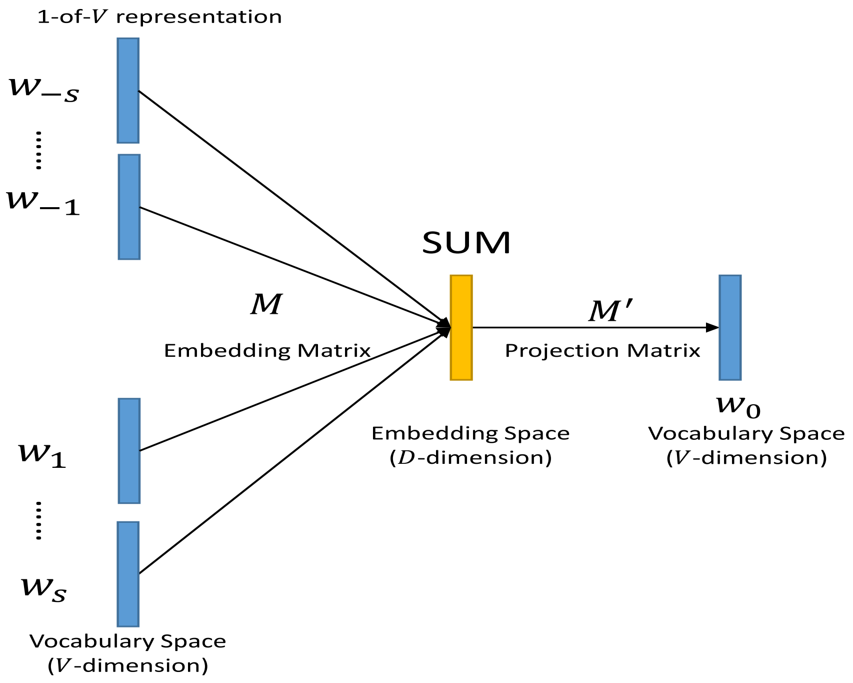

Figure 1: The CBOW model.

In our proposed model, we address the challenge of producing high-quality word embeddings for rare words and unknown words by leveraging the three types of co-occurrence information between words and morphemes.

On the input side, we segment the words into morphemes and put both the words and the morphemes as input. That is, the vocabulary for the 1-of-v representation contains both words and morphemes. As shown in Figure 2, the surrounding words in the sliding window are $w_{-s}, \cdots, w_{-1}, w_1, \cdots, w_s$ and their corresponding morphemes are $m_{-s,1}, m_{-s,2}, \cdots, m_{-s,t_{-s}}; \cdots; m_{-1,1}, m_{-1,2}, \cdots, m_{-1,t_{-1}}; m_{1,1}, m_{1,2}, \cdots, m_{1,t_1}; \cdots; m_{s,1}, m_{s,2}, \cdots, m_{s,t_s}$, where $2s$ is the number of the surrounding words and t_i is the number of morphemes for w_i $(i = -s, \cdots, -1, 1, \cdots, s)$. Note that t_i depends on the formation of w_i so that it may vary from word to word. If a word is also a morpheme, there will be two embedding vectors which are tagged differently. We use v_{w_i} and $v_{m_{i,j}}$ to represent the 1-of-v vectors of word w_i and morpheme $m_{i,j}$ respectively. On the input side, both the words and their morphemes are mapped into the embedding space by the same weight matrix M, and then the weighted sum v_I of the combination of word embeddings and the combination of morpheme embeddings is calculate as below,

$$v_I = \phi_w \cdot \sum_{\substack{i=-s \\ i\neq 0}}^{s} v_{w_i} + \phi_m \cdot \sum_{\substack{i=-s \\ i\neq 0}}^{s} \sum_{j=1}^{t_i} v_{m_{i,j}},$$

where ϕ_w and ϕ_m are the tradeoff coefficients between the combination of word embeddings and the combination of morpheme embeddings.

On the output side, we map the combined embedding vector v_I back to the 1-of-v space by another weight matrix M' to do the prediction. We have four settings of the structure. In the first setting, we only predict the central word w_0, and we name the model under this setting as *MorphemeCBOW*. In the second setting, we predict both the central word w_0 and its morphemes $m_{0,1}, m_{0,2}, \cdots, m_{0,t_0}$, and we name this setting as *MorphemeCBOW+*. In the above two settings, the tradeoff weights ϕ_w and ϕ_m are fixed. If we update the two weights in the learning process of *MorphemeCBOW*, we will get the third setting and we name it as *MorphemeCBOW**, while updating the two weights in *MorphemeCBOW+* yields the forth setting named *MorphemeCBOW++* .

Take *MorphemeCBOW+* as example, the objective is to maximize the following conditional co-occurrence probability,

$$\log(P(w_0 \mid \{w_i\}, \{m_{i,j}\})) + \log(\sum_{j=1}^{t_0} P(m_{0,j} \mid \{w_i\}, \{m_{i,j}\})), \tag{1}$$

where $\{w_i\}, \{m_{i,j}\}$ represent the bag of words and bag of morphemes separately. The conditional probability in the above formula is defined using the softmax function,

$$P(w_0 \mid \{w_i\}, \{m_{i,j}\}) = \frac{\exp(v'^T_{w_0} \cdot v_I)}{\sum_{v' \in V_O} \exp(v'^T \cdot v_I)} \quad , \quad P(m_{0,j} \mid \{w_i\}, \{m_{i,j}\}) = \frac{\exp(v'^T_{m_{0,j}} \cdot v_I)}{\sum_{v' \in V_O} \exp(v'^T \cdot v_I)}, \tag{2}$$

where V_O is the set of the output representations for the whole vocabulary; v' is used to differentiate with input representations; and $v'_{w_0}, v'_{m_{0,j}}$ represent the output embedding vectors of w_0 and $m_{0,j}$ respectively.

Usually, the computation cost for Formula (2) is expensive since it is proportional to the vocabulary size. In our model, we use negative sampling discussed in (Mikolov et al., 2013b) to speed up the computation. Particularly, we random select k negative samples $u_1, u_2, \cdots, u_k$ for each prediction target (word or morpheme). By using this technique, Formula (1) can be equally written as,

$$G(v_I) \equiv \log \sigma(v'^T_{w_0} \cdot v_I) + \sum_{j=1}^{t_0} \log \sigma(v'^T_{m_{0,j}} \cdot v_I) + \sum_{\substack{i=1 \\ u_i \neq w_0 \\ u_i \neq \forall m_{0,j}}}^{k} E_{u_i \sim P_n(u)}[\log \sigma(-v'^T_{u_i} \cdot v_I)],$$

where σ denotes the logistic function, and $P_n(u)$ is the vocabulary distribution used to select the negative samples. $P_n(u)$ is set as the 3/4rd power of the unigram distribution $U(u)$[1]. The negative samples should not be the same as any of the prediction targets w_0 and $m_{0,j}$ $(j = 1, \cdots, t_0)$. By using negative sampling, the training time spent on summing up the whole vocabulary in Formula (2) is greatly reduced so that it becomes linear with the number of the negative samples. Thus, we can calculate the gradient of $G(v_I)$ as below,

$$\begin{aligned}\frac{\partial G(v_I)}{\partial v_I} =& (1-\sigma(v'^T_{w_0}\cdot v_I))\cdot\frac{\partial(v'^T_{w_0}\cdot v_I)}{\partial v_I} + \sum_{j=1}^{t_0}(1-\sigma(v'^T_{m_{0,j}}\cdot v_I))\cdot\frac{\partial(v'^T_{m_{0,j}}\cdot v_I)}{\partial v_I}\\ &-\sum_{\substack{i=1\\u_i\neq w_0\\u_i\neq\forall m_{0,j}}}^{k}[\sigma(v'^T_{u_i}\cdot v_I)\cdot\frac{\partial(v'^T_{u_i}\cdot v_I)}{\partial v_I}].\end{aligned}$$

In the back-propagation process, the weights in the matrices M and M' are updated. When the training process converges, we take the matrix M as the learned word embeddings and morpheme embeddings.

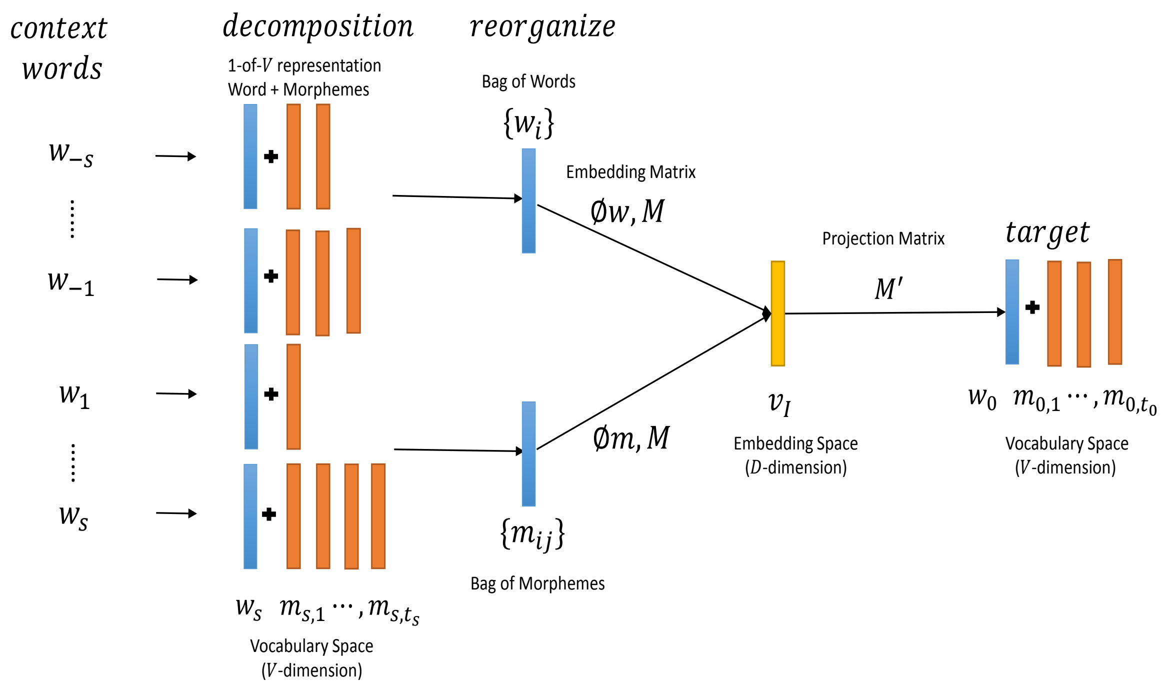

Figure 2: The proposed neural network model.

4 Experimental Evaluation

In this section we test the effectiveness of our model in generating high-quality word embeddings. We first introduce the experimental settings, and then we report the results on one analogical reasoning task and several word similarity tasks.

4.1 Datasets

We used two datasets for training: *enwiki9*[2] and *wiki2010*[3].

[1] http://www.cs.bgu.ac.il/~yoavg/publications/negative-sampling.pdf
[2] http://mattmahoney.ent/dc/enwik9.zip
[3] http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html

- The *enwiki9* dataset contains about 123.4 million words. We used Matt Mahoney's text pre-processing script[4] to process the corpus. Thus, we removed all non-Roman characters and mapped all digits to English words. In addition, words occurred less than 5 times in the training corpus were discarded. We used the learned word embeddings from *enwiki9* to test an analogical reasoning task described in (Mikolov et al., 2013a).
- The *wiki2010* dataset contains about 990 million words. The learned embeddings from this dataset were used on word similarity tasks as it was convenient to compare with the csmRNN model (Luong et al., 2013). We did the same data pre-processing as csmRNN did. That is, we removed all non-Roman characters and mapped all digits to zero.

4.2 Settings

In the analogical reasoning task, we used the CBOW model as the baseline. In both CBOW and our proposed model, we set the context window size to be 5, and generated three dimension sizes (100, 200, and 300) of word embeddings. We used negative sampling (Mikolov et al., 2013b) in the output layer and the number of negative samples is chosen as 3.

In the word similarity tasks, we used the csmRNN model as the baseline. The context window size of our model was set to be 5. To make a fair comparison with the csmRNN model, we conducted the same settings in our experiments as csmRNN. First, as csmRNN used the *Morfessor* (Creutz and Lagus, 2007) method to segment words into morphemes, we also used *Morfessor* as one of our word segmentation methods to avoid the influence caused by the segmentation methods. Second, as csmRNN used two existing embeddings C&W[5] (Collobert et al., 2011) and HSMN[6] (Huang et al., 2012) to initialize the training process, we also used the two embeddings as the initial weights of M in our experiments. Third, we set the dimension of the embedding space to 50 as csmRNN did.

In our model, we employed three methods to segment a word into morphemes. The first method is called *Morfessor*, which is a public tool implemented based on the minimum descriptions length algorithm (Creutz and Lagus, 2007). The second method is called *Root*, which segments a word into roots and affixes according to a predefined list in Longman Dictionaries. The third method is called *Syllable*, which is implemented based on the hyphenation tool proposed by Liang (Liang, 1983). Besides, the architecture of the proposed model can be specified into four types: *MorphemeCBOW*, *MorphemeCBOW**, *MorphemeCBOW+*, and *MorphemeCBOW++*. For the model *MorphemeCBOW* and *MorphemeCBOW+* with fixed tradeoff coefficients, we set the weights ϕ_w and ϕ_m to be 0.8 and 0.2 respectively; while for the other two models with updated tradeoff weights, the weights ϕ_w and ϕ_m are initialized as 1. These weight settings are chosen empirically.

4.3 Evaluation Tasks

4.3.1 Analogical reasoning task

The analogical reasoning task was introduced by Mikolov et al (Mikolov et al., 2013a). All the questions are in the form "a is to b is as c is to ?", denoted as $a : b \rightarrow c : ?$. The task consists of 19,544 questions involving semantic analogies (e.g., England: London → China: Beijing) and syntactic analogies (e.g., amazing: amazingly → unfortunate: unfortunately). Suppose that the corresponding vectors are $\overrightarrow{a}$, $\overrightarrow{b}$, and $\overrightarrow{c}$, we will answer the question by finding the word with the representation having the maximum cosine similarity to vector $\overrightarrow{b} - \overrightarrow{a} + \overrightarrow{c}$, i.e,

$$\max_{x \in V, x \neq b, x \neq c} (\overrightarrow{b} - \overrightarrow{a} + \overrightarrow{c})^T \overrightarrow{x}$$

where V is the vocabulary. Only when the computed word is exactly the answer word in evaluation set can the question be regarded as answered correctly.

[4]`http://mattmahoney.net/dc/textdata.html`

[5]`http://ronan.collobert.com/senna/`

[6]`http://ai.stanford.edu/~ehhuang/`

4.3.2 Word similarity task

The word similarity task was tested on five evaluation sets: WS353 (Finkelstein et al., 2002), SCWS* (Huang et al., 2012), MC (Miller and Charles, 1991), RG (Rubenstein and Goodenough, 1965) and RW (Luong et al., 2013), which contain 353, 1,762, 30, 65 and 2,034 pairs of words respectively. Table 1 shows some statistics about the datasets. Furthermore, the words in WS353, MC, RG are mostly frequent words, while SCWS* and RW have much more rare words and unknown words (i.e., unseen words in the training corpus) than the first three sets. The word distributions of these datasets are shown in Figure 3, from which we can see that RW contains the largest number of rare and unknown words. For the unknown words, we segmented them into morphemes, and calculated their word embeddings by summing up their corresponding morpheme embeddings. Each word pair in these datasets is associated with several human judgments on similarity and relatedness on a scale from 0 to 10 or 0 to 4. For example, (*cup*, *drink*) received an average score of 7.25, while (*cup*, *substance*) received an average score of 1.92. To evaluate the quality of the learned word embeddings, we computed Spearman's ρ correlation between the similarity scores calculated on the learned word embeddings and the human judgments.

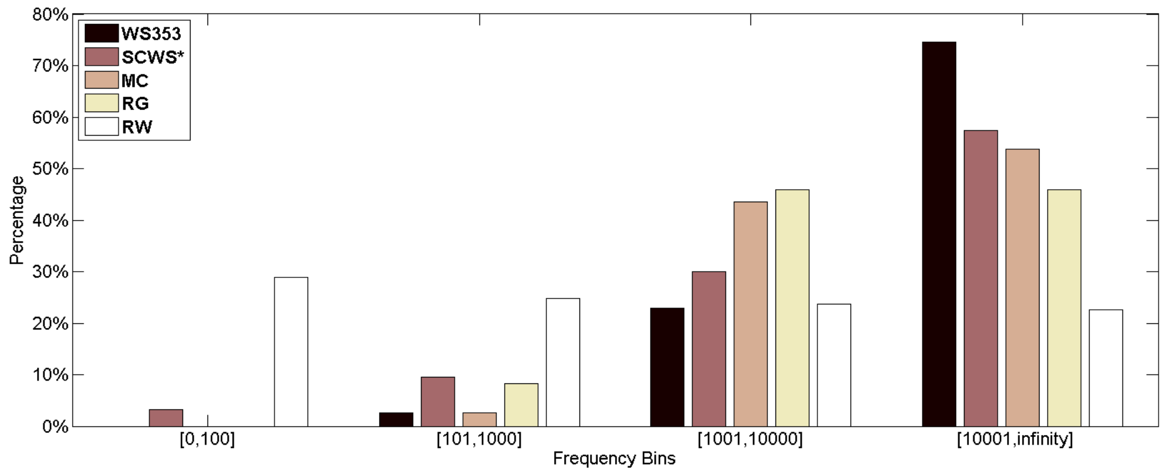

Figure 3: **Word distribution by frequency**. Distinct words in each test dataset are grouped according to frequencies. The figure shows the percentage of words in each bin.

Table 1: Statistics on the word similarity evaluation sets.

Dataset	Number of pairs	Number of words	Percentage of multi-segments words by *Morfessor*
WS353	353	437	28.15%
SCWS*	1726	1703	34.00%
RW	2034	2951	69.06%

4.4 Experimental Results

4.4.1 Results on analogical reasoning task

The experimental results on the analogical reasoning task are shown in Table 2, including semantic accuracy, syntactic accuracy, and total accuracy of all competition settings. Semantic/syntactic accuracy refers to the number of correct answers over the total number of all semantic/syntactic questions. From the results, we have the following observations:

- In MorphemeCBOW, we used the surrounding words and their morphemes to predict the central word. The total accuracies are all improved compared with baseline using the three word segmentation methods across three different dimensions of the embedding space. Generally, the improvements on semantic accuracies are less than those on syntactic accuracies. The reason is that the morphological information favors more for the syntactic tasks than the semantic tasks. Furthermore, the *Root* method achieved the best among the three segmentation methods, showing that the roots and affixes from the dictionary can help produce a high-quality morpheme segmentation tool.

- In MorphemeCBOW*, we predicted the central word, and updated the tradeoff coefficients in the learning process. We can see that the results are comparable or slightly better than MorphemeCBOW using the three word segmentation methods across three different dimensions of the embedding space, showing that updating the tradeoff coefficients may further boost the model performance under some specific settings.
- In MorphemeCBOW+, we predicted both the central word and its morphemes. MorphemeCBOW+ can provide slightly better results compared with MorphemeCBOW and MorphemeCBOW*, indicating that putting morphemes (especially roots) in the output layer can do extra help in generating high-quality word embeddings.
- In MorphemeCBOW++, we predicted the central word and its morphemes, and updated the tradeoff coefficients in the learning process. The performance under all of the three word segmentation methods got further improved compared with MorphemeCBOW+. It tells that the contributions from words and morphemes are different to the analogical reasoning task. According to our observations, the weight for words is usually higher than that for morphemes.
- By comparing MorphemeCBOW with MorphemeCBOW* as well as MorphemeCBOW+ with MorphemeCBOW++, we can observe that updating the weights of tradeoff coefficients seem to essentially boost syntactic accuracy by trading off a bit of semantic accuracy. As introduced in Section 4.2, in the fixed weight model the ratio of weight of morphemes to the weight of word is 0.25; while our experiment records show that the averaged ratio are 0.43 if the two weights are updated, meaning that the weight of the combination of morphemes increases and the contribution of the original word to the final combined embedding decreased. As a result, the syntactic accuracy which largely reflected in the morphological structure of a word increased, but the semantic accuracy hurts a little.

4.4.2 Results on word similarity task

Experimental results on the word similarity tasks are shown in Table 3[7],where the labels of C&W + csmRNN and HSMN + csmRNN mean that using C&W and HSMN to initialize csmRNN model as what had been introduced in the paper of Luong et al. In our experiments, the architecture of MorphemeCBOW* performs the best, so we only show the results related to MorphemeCBOW* in the table. We have the following observations from the results:

- On WS353, MC, RG, and SCWS*, MorphemeCBOW* performs consistently better than the csmRNN model, showing that our model can achieve better representations for common words.

[7]csmRNN embeddings are available on `http://www-nlp.stanford.edu/~lmthang/morphoNLM/`, Performances are tested based on the two embeddings.

Table 2: Performance of leveraging morphological information on the analogical reasoning task.

(a) Baseline			(b) MorphemeCBOW			(c) MorphemeCBOW*			(d) MorphemeCBOW+			(e) MorphemeCBOW++		
Dimension	*(%)*	CBOW	*Morfessor*	*Syllable*	*Root*	*Morfessor*	*Syllable*	*Root*	*Morfessor*	*Syllable*	*Root*	*Morfessor*	*Syllable*	*Root*
100	Total	26.49	31.99	31.28	**32.49**	33.07	31.16	**34.04**	**33.26**	31.12	32.77	**38.86**	34.42	35.78
	Semantic	17.51	19.44	18.76	21.77	15.20	15.68	17.87	22.82	20.80	22.79	21.12	22.58	22.43
	Syntactic	33.96	42.42	41.68	41.40	47.92	44.02	47.48	41.93	39.70	41.07	53.59	44.26	46.87
200	Total	30.50	34.04	34.71	**36.29**	34.69	33.13	**36.50**	38.28	39.32	**39.53**	40.32	41.79	**43.29**
	Semantic	19.71	19.10	19.13	22.45	11.53	15.91	18.92	25.94	27.99	28.29	24.20	24.05	25.04
	Syntactic	39.46	46.45	47.65	47.79	53.92	47.44	51.10	48.52	48.74	48.86	53.72	56.53	58.45
300	Total	29.04	31.27	32.45	**36.12**	31.21	32.16	**35.63**	38.01	39.56	**39.70**	37.65	41.64	**41.96**
	Semantic	17.58	15.45	15.63	20.79	8.85	12.54	15.75	25.11	26.94	27.80	13.97	26.64	25.82
	Syntactic	38.56	44.41	46.44	48.86	49.79	48.47	52.14	48.72	50.05	49.58	57.32	54.10	55.36

Table 3: Performance of leveraging morphological information on the word similarity task.

Model	WS353 (%)	SCWS* (%)	MC(%)	RG(%)	RW(%)
C&W	49.73	48.45	57.33	48.22	21.93
C&W + csmRNN	58.27	49.09	60.22	58.92	**31.77**
C&W + MorphemeCBOW*	**63.81**	**53.30**	**74.33**	**61.22**	31.14
HSMN	62.58	32.09	66.18	64.51	1.97
HSMN + csmRNN	64.58	44.08	71.88	65.15	22.31
HSMN + MorphemeCBOW*	**65.19**	**53.40**	**81.62**	**67.41**	**32.13**
MorphemeCBOW*	63.45	53.40	77.40	63.78	32.88

- On RW, MorphemeCBOW* performs better than the csmRNN model when using the HSMN embeddings as the initialization. When using the C&W embeddings as the initialization, the performance of MorphemeCBOW* is also comparable with that of csmRNN. In particular, if we do not use any pre-trained embeddings to initialize our mode, it performed the best (32.88%), and it even beats the best performance of csmRNN with initializations (31.77%)[8]. The initialization is very important to a neural network. Suitable initialization will help increase the embedding quality which works like training with multi-epochs. However, as there are two matrix M and M' in our network structure, the initialization of both of them are more sensible. Furthermore, considering that the recursive structure of csmRNN will bring higher computation complexity, we can conclude that our model has excellent ability in learning the embeddings of rare words from pure scratch.
- The improvement on RW is more significant than those on the other four datasets. Considering that RW contains more rare and unknown words (See Figure 3), we verified our idea that leveraging morphological information will especially benefit the embedding of low-frequency words. More specifically, without sufficient context information for the rare words in the training data, building connections between words using morphemes will provide additional evidence for the model to generate effective embeddings for these rare words; and, by combining the high-quality morpheme embeddings to obtain the representations of the unknown words, the model does a good job in dealing with the new emerging words.

5 Conclusions and Future Work

We proposed a novel neural network model to learn word representations from text. The model can leverage several types of morphological information to produce high-quality word embeddings, especially for rare words and unknown words. Empirical experiments on an analogical reasoning task and several word similarity tasks have shown that the proposed model can generate better word representations compared with several state-of-the-art approaches.

For the future work, we plan to separate words and morphemes into several buckets according to their frequencies. Different buckets will be associated with different coefficients, so that we can tune the coefficients to approach even better word embeddings. We also plan to run our model on more training corpus to obtain the embedding vectors for rare words, especially those new words invented out recently. These emerging new words usually do not exist in standard training corpus such as Wikipedia, but exists in some noisy data such as news articles and web pages. How well our model performs on these new training corpus is an interesting question to explore.

References

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 1–4, New York City, USA, June. Association for Computational Linguistics.

[8]34.36% in the paper of Luong et al; 32.06% in their project website, see note7

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12.

M. Creutz and K. Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *TSLP*.

L. Deng, X. He, and J. Gao. 2013. Deep stacking networks for information retrieval. In *ICASSP*, pages 3153–3157.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2002. Placing search in context: The concept revisited. In *ACM Transactions on Information Systems*.

X. Glorot, A. Bordes, and Y. Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

F. M. Liang. 1983. Word hy-phen-a-tion by com-put-er. Technical report.

M.-T. Luong, R. Socher, and C. D. Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013a. Efficient estimation of word representations in vector space. ICLR '13.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

T. Mikolov, W.-T. Yih, and G. Zweig. 2013c. Linguistic regularities in continuous space word representations. In *In NAACL-HLT*, pages 746–751.

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

G.A. Miller and W.G. Charles. 1991. Contextual correlates of semantic similarity. 6(1):1–28.

A. Mnih and G. E. Hinton. 2008. A scalable hierarchical distributed language model. In *NIPS*, pages 1081–1088.

Amr El-Desoky Mousa, Hong-Kwang Jeff Kuo, Lidia Mangu, and Hagen Soltau. 2013. Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic. In *ICASSP*, pages 8435–8439.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October.

R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*.

Henning Sperr, Jan Niehues, and Alex Waibel. 2013. Letter n-gram-based input encoding for continuous space language models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 30–39, Sofia, Bulgaria, August. Association for Computational Linguistics.

J. P. Turian, L.-A. Ratinov, and Y. Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

P. D. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

P. D. Turney. 2013. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *TACL*, pages 353–366.

J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*.