Incorporating Latent Meanings of Morphological Compositions to Enhance Word Embeddings

Yang Xu[†], **Jiawei Liu**[†], **Wei Yang**[‡*], and **Liusheng Huang**[‡]
School of Computer Science and Technology,
University of Science and Technology of China, Hefei, 230027, China
[†]{smallant, ustcljw}@mail.ustc.edu.cn
[‡]{qubit, lshuang}@ustc.edu.cn

Abstract

Traditional word embedding approaches learn semantic information at word level while ignoring the meaningful internal structures of words like morphemes. Furthermore, existing morphology-based models directly incorporate morphemes to train word embeddings, but still neglect the latent meanings of morphemes. In this paper, we explore to employ the latent meanings of morphological compositions of words to train and enhance word embeddings. Based on this purpose, we propose three Latent Meaning Models (LMMs), named LMM-A, LMM-S and LMM-M respectively, which adopt different strategies to incorporate the latent meanings of morphemes during the training process. Experiments on word similarity, syntactic analogy and text classification are conducted to validate the feasibility of our models. The results demonstrate that our models outperform the baselines on five word similarity datasets. On Wordsim-353 and RG-65 datasets, our models nearly achieve 5% and 7% gains over the classic CBOW model, respectively. For the syntactic analogy and text classification tasks, our models also surpass all the baselines including a morphology-based model.

1 Introduction

Word embedding, which is also termed distributed word representation, has been a hot topic in the area of Natural Language Processing (NLP). The derived word embeddings have been used in plenty of tasks such as text classification (Liu et al., 2015), information retrieval (Manning et al., 2008), sentiment analysis (Shin et al., 2016), machine translation (Cho et al., 2014) and so on. Recently, some classic word embedding methods have been proposed, like Continuous Bag-of-Word (CBOW), Skip-gram (Mikolov et al., 2013a), Global Vectors (GloVe) (Pennington et al., 2014). These methods can usually capture word-level semantic information but ignore the meaningful inner structures of words like English morphemes or Chinese characters.

The effectiveness of exploiting the internal compositions of words has been validated by some previous work (Luong et al., 2013; Botha and Blunsom, 2014; Chen et al., 2015; Cotterell et al., 2016). Some of them compute the word embeddings by directly adding the representations of morphemes/characters to context words or optimizing a joint objective over distributional statistics and morphological properties (Qiu et al., 2014; Botha and Blunsom, 2014; Chen et al., 2015; Luong et al., 2013; Lazaridou et al., 2013), while others introduce some probabilistic graphical models to build relationship between words and their internal compositions. *e.g.*, Bhatia et al. (2016) treat word embeddings as latent variables for a prior distribution, which reflects words' morphological properties, and feed the latent variables into a neural sequence model to obtain final word embeddings. Cotterell et al. (2016) construct a Gaussian graphical model that binds the morphological analysis to pre-trained word embeddings, which can help to smooth the noisy embeddings. Besides, these two methods also have the ability to predict embeddings for unseen words.

Different from all the above models (we regard them as *Explicit* models in Fig. 1) where internal compositions are directly used to encode morphological regularities into words and the

*This is the corresponding author.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1232–1242
Melbourne, Australia, July 15 - 20, 2018.

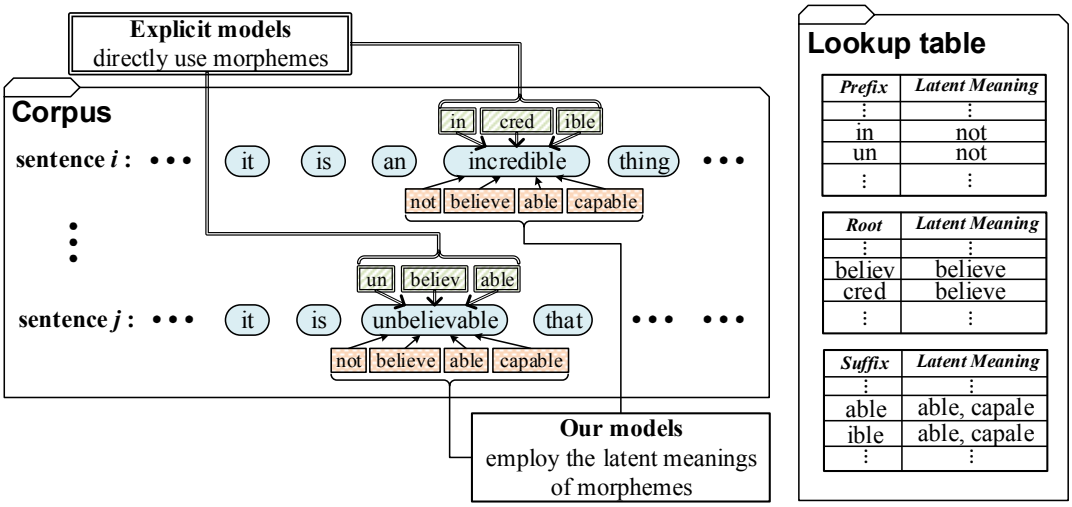

Figure 1: An illustration of explicit models and our models in an English corpus. Although *incredible* and *unbelievable* have different morphemes, their morphemes have the same latent meanings.

composition embeddings like morpheme embeddings are generated as by-products, we explore a new way to employ the latent meanings of morphological compositions rather than the compositions themselves to train word embeddings. As shown in Fig. 1, according to the distributional semantics hypothesis (Sahlgren, 2008), *incredible* and *unbelievable* probably have similar word embeddings because they have similar context. As a matter of fact, *incredible* is a synonym of *unbelievable* and their embeddings are expected to be close enough. Since the morphemes of the two words are different, especially the roots *cred* and *believ*, the explicit models may not significantly shorten the distance between the words in the vector space. Fortunately, the latent meanings of the different morphemes are the same (*e.g.*, the latent meanings of roots *cred*, *believ* are "*believe*") as listed in the lookup table (derived from the resources provided by Michigan State University),[1] which evidently implies that *incredible* and *unbelievable* share the same meanings. In addition, by replacing morphemes with their latent meanings, we can directly and simply quantize the similarities between words and their sub-compositions with the same metrics used in most NLP tasks, *e.g.*, cosine similarity. Subsequently, the similarities are utilized to calculate the weights of latent meanings of morphemes for each word.

In this paper, we try different strategies to modify the input layer and update rules of a neural language model, *e.g.*, CBOW, Skip-gram, and propose three lightweight and efficient models, which are termed Latent Meaning Models (LMMs), to not only encode morphological properties into words but also enhance the semantic similarities among word embeddings. Usually, the vocabulary derived from the corpus contains vast majority or even all of the latent meanings. Rather than generating and training extra embeddings for latent meanings, we directly override the embeddings of the corresponding words in the vocabulary. Moreover, a word map is created to describe the relations between words and the latent meanings of their morphemes.

For comparison, our models together with the state-of-the-art baselines are tested on two basic NLP tasks, which are word similarity and syntactic analogy, and one downstream text classification task. The results show that LMMs outperform the baselines and get satisfactory improvement on these tasks. In all, the main contributions of this paper are summarized as follows.

- Rather than directly incorporating the morphological compositions (surface forms) of words, we decide to employ the latent meanings of the compositions (underlying forms) to train the word embeddings. To validate the feasibility of our purpose, three specific models, named LMMs, are proposed with different strategies to incorporate the latent meanings.

[1] https://msu.edu/~defores1/gre/roots/gre_rts_afx1.htm

- We utilize a medium-sized English corpus to train LMMs and the state-of-the-art baselines, and evaluate their performance on two basic NLP tasks, *i.e.*, word similarity and syntactic analogy, and one downstream text classification task. The results show that LMMs outperform the baselines on five word similarity datasets. On the golden standard Wordsim-353 and RG-65, LMMs approximately achieve 5% and 7% gains over CBOW, respectively. For the syntactic analogy and text classification tasks, LMMs also surpass all the baselines.

- We conduct experiments to analyze the impacts of parameter settings, and the results demonstrate that the performance of LMMs on the smallest corpus is similar to the performance of CBOW on the corpus that is five times as large, which convinces us that LMMs are of great advantages to enhance word embeddings compared with traditional methods.

2 Background and Related Work

Considering the high efficiency of CBOW proposed by Mikolov et al. (2013a), our LMMs are built upon CBOW. Here, we first review some backgrounds of CBOW, and then present some related work on recent word-level and morphology-based word embedding methods.

CBOW with Negative Sampling With a sliding window, CBOW utilizes the context words in the window to predict the target word. Given a sequence of tokens $T = \{t_1, t_2, \cdots, t_n\}$, where n is the size of a training corpus, the objective of CBOW is to maximize the following average log probability equation:

$$L = \frac{1}{n}\sum_{i=1}^{n} \log p\big(t_i|context(t_i)\big), \qquad (1)$$

where $context(t_i)$ represents the context words of t_i in the slide window, $p\big(t_i|context(t_i)\big)$ is derived by softmax. Due to huge size of English vocabulary, $p\big(t_i|context(t_i)\big)$ can not be calculated in a tolerable time. Therefore, negative sampling and hierarchical softmax are proposed to solve this problem. Owing to the efficiency of negative sampling, all our models are trained based on it. In terms of negative sampling, the log probability $\log p(t_O|t_I)$ is transformed as:

$$\log \delta\big(vec'(t_O)^T vec(t_I)\big) + \sum_{i=1}^{m} \log\big[1 - \delta\big(vec'(t_i)^T vec(t_I)\big)\big], \qquad (2)$$

where m denotes the number of negative samples, and $\delta(\cdot)$ is the sigmoid function. The first item of Eq. (2) is the probability of target word when its context is given. The second item indicates the probability that negative samples do not share the same context as the target word.

Word-level Word Embedding In general, word embedding models can mainly be divided into two branches. One is based on neural network like the classic CBOW model (Mikolov et al., 2013a), while the other is based on matrix factorization. Besides CBOW, Skip-gram (Mikolov et al., 2013a) is another widely used neural-network-based model, which predicts the context by using the target word (Mikolov et al., 2013a). As for matrix factorization, Dhillon et al. (2015) proposed a spectral word embedding method to measure the correlation between word information matrix and context information matrix. In order to combine the advantages of models based on neural network and matrix factorization, Pennington et al. (2014) proposed a famous word embedding model named GloVe, which is reported to outperform the CBOW and Skip-gram models on some tasks. These models are effective to capture word-level semantic information while neglecting inner structures of words. In contrast, the unheeded inner structures are utilized in both our LMMs and other morphology-based models.

Morphology-based Word Embedding Recently, some more fine-grained word embedding models are proposed by exploiting the morphological compositions of words, *e.g.*, root and affixes. These morphology-based models can be divided into two main categories.

The first category directly adds the representations of internal structures to word embeddings or optimizes a joint objective over distributional statistics and morphological properties (Luong et al., 2013; Qiu et al., 2014; Botha and Blunsom, 2014; Lazaridou et al., 2013; Chen et al., 2015; Kim et al., 2016; Cotterell and Schütze, 2015). Chen et al. (2015) proposed a character-enhanced Chinese word embedding model, which splits a Chinese word into several characters and add the characters into the input layer of their models.

Luong et al. (2013) utilized the morpheme segments produced by Morfessor (Creutz and Lagus, 2007) and constructed morpheme trees for words to learn morphologically-aware word embeddings by the recursive neural network. Kim et al. (2016) incorporated the convolutional character information into English words. Their model can learn character-level semantic information for embeddings, which is proved to be effective for some morpheme-rich languages. However, with a huge size architecture, it's very time-consuming. Cotterell et al. (2015) augmented the log linear model to make the words, which share similar morphemes, gather together in vector space.

The other category tries to use probabilistic graphical models to connect words with their morphological compositions, and further learns word embeddings (Bhatia et al., 2016; Cotterell et al., 2016). Bhatia et al. (2016) employed morphemes and made them as prior knowledge of the latent word embeddings, then fed the latent variables into a neural sequence model to obtain final word embeddings. Cotterell et al. (2016) proposed a morpheme-based post-processor for pre-trained word embeddings. They constructed a Gaussian graphical model which can extrapolate continuous representations for unknown words.

However, these morphology-based models directly exploit the internal compositions of words to encode morphological regularities into word embeddings, and some by-products are also produced like morpheme embeddings. In contrast, we employ the latent meanings of morphological compositions to provide deeper insights for training better word embeddings. Furthermore, since the latent meanings are included in the vocabulary, there is no extra embedding being generated.

3 Our Latent Meaning Models

We leverage different strategies to modify the input layer and update rules of CBOW when incorporating the latent meanings of morphemes. Three specific models, named Latent Meaning Model-Average (LMM-A), LMM-Similarity (LMM-S) and LMM-Max (LMM-M), are proposed. It should be stated that, for now, our models mainly concern the derivational morphemes, which can be interpreted to some meaningful words or phrases (*i.e.*, latent meanings), not the inflectional morphemes like tense, number, gender, *etc.*

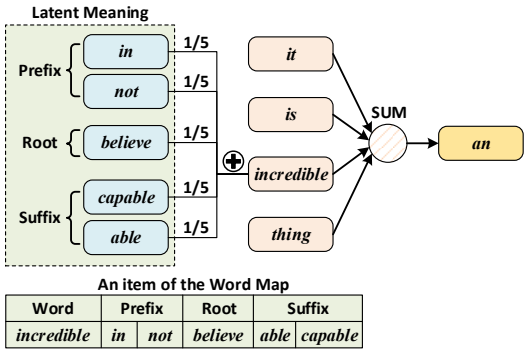

Figure 2: A paradigm of LMM-A. The sentence "*it is an incredible thing*" is selected as an example. When calculating the input vector of "*incredible*", we first find out the latent meanings of its morphemes in the word map, and add the vectors of all latent meanings to the vector of "*incredible*" with equal weights.

LMM-A assumes that all latent meanings of morphemes of a word have equal contributions to the word. LMM-A is applicable to the condition where words are correctly segmented into morphemes and each morpheme is interpreted to appropriate latent meanings. However, refining the latent meanings for morphemes is time-consuming and needs vast human annotations. To address this concern, LMM-S is proposed. Motivated by the attention scheme, LMM-S holds the assumption that all latent meanings have different contributions, and assigns the outliers small weights to let them have little impact on the representation of the target word. Furthermore, in LMM-M, we only keep the latent meanings which have the greatest contributions to the corresponding word. In what follows, we are going to introduce each of our LMMs in detail. At the end of this section, we will introduce the update rules of the models.

3.1 LMM-A

Given a sequence of tokens $T = \{t_1, t_2, \cdots, t_n\}$, LMM-A assumes that morphemes' latent meanings of token t_i ($i \in [1, n]$) have equal contributions to t_i, as shown in Fig. 2. The item for t_i in the word map is $t_i \mapsto M_i$. M_i is a set of latent meanings of t_i's morphemes, and it consists of three sub-parts P_i, R_i and S_i corresponding to the latent meanings of prefixes, roots and suffixes of t_i, respectively. Hence, at the input layer, the

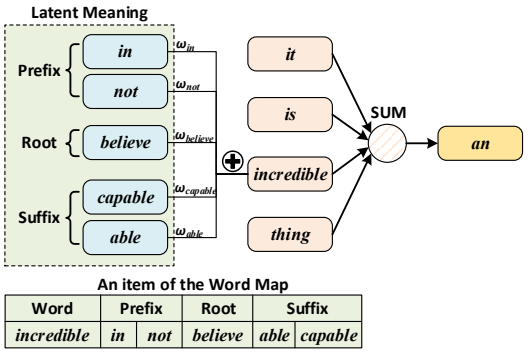

Figure 3: A paradigm of LMM-S. In this model, all latent meanings of morphemes of "*incredible*" are added together with different weights.

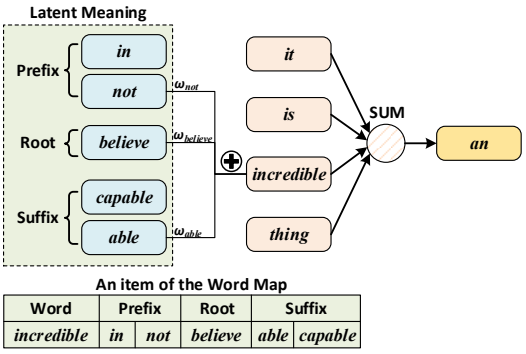

Figure 4: A paradigm of LMM-M. The latent meanings with maximum similarities towards "*incredible*" are selected.

modified embedding of t_i can be expressed as

$$\widehat{v}_{t_i} = \frac{1}{2}\big(v_{t_i} + \frac{1}{N_i}\sum_{w \in M_i} v_w\big), \tag{3}$$

where v_{t_i} is the original word embedding of t_i, N_i denotes the length of M_i and v_w indicates the embedding of latent meaning w. Meanwhile, we assume the original word embedding and the average embeddings of v_w ($w \in M_i$) have equal weights, *i.e.*, 0.5. Eventually, $\widehat{v}_{t_i}$ rather than v_{t_i} is utilized for training in CBOW.

3.2 LMM-S

This model is proposed based on the attention scheme. We observe that many morphemes have more than one latent meaning. For instance, prefix *in-* means "*in*" and "*not*", and suffix *-ible* means "*able*" and "*capable*".[2] As Fig. 3 shows, for the item *incredible* $\mapsto$ $\{[in, not], [believe], [able, capable]\}$ in the word map, the latent meanings have different biases towards "*incredible*". Therefore, we assign different weights to latent meanings. We measure the weights of latent meanings by calculating the normalized similarities between token t_i and the corresponding latent meanings. For LMM-S, the modified embedding of t_i can be rewritten as

$$\widehat{v}_{t_i} = \frac{1}{2}\big[v_{t_i} + \sum_{w \in M_i} \omega_{<t_i,w>} \cdot v_w\big], \tag{4}$$

where v_{t_i} is the original vector of t_i, and $\omega_{<t_i,w>}$ denotes the weight between t_i and the latent meaning w ($w \in M_i$). We use $cos(v_a, v_b)$ to denote the cosine similarity between v_a and v_b, then $\omega_{<t_i,w>}$ is expressed as follows:

$$\omega_{<t_i,w>} = \frac{cos(v_{t_i}, v_w)}{\sum_{x \in M_i} cos(v_{t_i}, v_x)}. \tag{5}$$

3.3 LMM-M

To further eliminate the impacts of some uncorrelated latent meanings to a word, in LMM-M, we only select the latent meanings that have maximum similarities to the token t_i from P_i, R_i and S_i. As is shown in Fig. 4, the latent meaning "*not*" of prefix *in* is finally selected since the similarity between "*not*" and "*incredible*" is larger than that between "*in*" and "*incredible*". For token t_i, LMM-M is mathematically defined as

$$\widehat{v}_{t_i} = \frac{1}{2}\big[v_{t_i} + \sum_{w \in M^i_{max}} \omega_{<t_i,w>} \cdot v_w\big], \tag{6}$$

where $M^i_{max} = \{P^i_{max}, R^i_{max}, S^i_{max}\}$ is the set of latent meanings with maximum similarities towards token t_i, and P^i_{max}, R^i_{max}, S^i_{max} are obtained by the following equations:

$$\begin{aligned} P^i_{max} &= arg\max_w cos(v_{t_i}, v_w), w \in P_i, \\ R^i_{max} &= arg\max_w cos(v_{t_i}, v_w), w \in R_i, \\ S^i_{max} &= arg\max_w cos(v_{t_i}, v_w), w \in S_i. \end{aligned} \tag{7}$$

The normalized weight $\omega_{<t_i,w>}$ ($w \in M^i_{max}$) can similarly be derived like Eq. (5).

[2] All the latent meanings of roots and affixes are referred to the resources we mentioned before.

3.4 Update Rules for LMMs

After modifying the input layer of CBOW, Eq. (1) can be rewritten as

$$\widehat{L} = \frac{1}{n}\sum_{i=1}^{n} \log p\big(v_{t_i} | \sum_{t_j \in context(t_i)} \widehat{v}_{t_j}\big), \quad (8)$$

where $\widehat{v}_{t_j}$ is the modified vector of v_{t_j} ($t_j \in context(t_i)$). Since the word map describes top-level relations between words and the latent meanings, these relations don't change during the training period. All parameters introduced by our models can be directly derived using the word map and word vectors, thus no extra parameter needs to be trained. When the gradient is propagated back to the input layer, we update not just the word vector v_{t_j} ($t_j \in context(t_i)$) but the vectors of the latent meanings in the vocabulary with the same weights as they are added to the vector v_{t_j}.

4 Experimental Setup

Before conducting experiments, some experimental settings are firstly introduced in this section.

4.1 Corpus and Word Map

We utilize a medium-sized English corpus to train all word embedding models. The corpus stems from the website of the 2013 ACL Workshop on Machine Translation[3] and is used in (Kim et al., 2016). We choose the news corpus of 2009 whose size is about 1.7GB. It contains approximately 500 million tokens and 600,000 words in the vocabulary. To get better quality of the word embeddings, we filter all digits and some punctuation marks out of the corpus.

For many languages, there exist large morphological lexicons or morphological tools that can analyze any word form (Cotterell and Schütze, 2015). To create the word map, we need to obtain the morphemes of each word and interpret them with the lookup table mentioned above to get the latent meanings. Usually, the lookup table can also be derived from the morphological lexicons for different languages, although it costs some time and manpower, we can create the lookup table once for all since it represents the common knowledge with respect to a certain language. Specifically, we first perform an unsupervised morpheme segmentation using Morefessor (Creutz and Lagus, 2007) for the vocabularies. Then we execute matching between the segmentation results and the morphological compositions in the lookup table, and the character sequence with largest overlap ratio will be viewed as a final morpheme and further be replaced by its latent meanings. Although the lookup table employed in this paper contains latent meanings for only 90 prefixes, 382 roots and 67 suffixes, we focus on validating the feasibility of enhancing word embeddings with the latent meanings of morphemes, and expending the lookup table is left as future work.

4.2 Baselines

For comparison, we choose three word-level state-of-the-art word embedding models including CBOW, Skip-gram (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014), and we also implement an Explicitly Morpheme-related Model (EMM), which is a variant version of the previous work (Qiu et al., 2014). The architecture of EMM is based on our LMM-A, where latent meanings are replaced back to morphemes and the embeddings of morphemes are also learned when training word embeddings. This enables our evaluation to focus on the critical difference between our models and the explicit model (Bhatia et al., 2016). We utilize the source code of word2vec[4] to train CBOW and Skip-gram. GloVe is trained based on the code[5] provided by Pennington et al. (2014). We modify the source of word2vec and train our models and EMM.

4.3 Parameter Settings

Parameter settings have a great effect on the performance of word embeddings (Levy et al., 2015). For fairness, all models are trained based on equal parameter settings. In order to accelerate the training process, CBOW, Skip-gram and EMM together with our models are trained by using the negative sampling technique. It is suggested that the number of negative samples in the range 5-20 is useful for small corpus (Mikolov et al., 2013b). If large corpus is used, the number of negative samples can be as small as 2-5. According to the size of corpus we used, the number of negative samples is empirically set to be 20 in this paper.

[3] http://www.statmt.org/wmt13/translation-task.html

[4] https://github.com/dav/word2vec

[5] http://nlp.stanford.edu/projects/glove

Name	Pairs	Name	Pairs
RG-65	65	RW	2034
SCWS	2003	Men-3k	3000
Wordsim-353	353	WS-353-REL	252

Table 1: Details of datasets. The column "Pairs" shows the number of word pairs in each dataset.

The dimension of word embedding is set as 200 like that in (Dhillon et al., 2015). We set the context window size as 5 which is equal to the setting in (Mikolov et al., 2013b).

4.4 Evaluation Benchmarks

4.4.1 Word Similarity

This experiment is conducted to evaluate the ability of word embeddings to capture semantic information from corpus. For English word similarity, we employ two gold standard datasets including Wordsim-353 (Finkelstein et al., 2001) and RG-65 (Rubenstein and Goodenough, 1965) as well as some other widely-used datasets including Rare-Word (Luong et al., 2013), SCWS (Huang et al., 2012), Men-3k (Bruni et al., 2014) and WS-353-Related (Agirre et al., 2009). More details of these datasets are shown in Table 1. Each dataset consists of three columns. The first two columns stand for word pairs and the last column is human score. We utilize the cosine similarity, which is used in many previous works (Mikolov et al., 2013b; Pennington et al., 2014), as the metric to measure the distance between two words. The Spearman's rank correlation coefficient (ρ) is employed to evaluate the similarity between our results and human scores. Higher ρ means better performance.

4.4.2 Syntactic Analogy

Based on the learned word embeddings, the core task of syntactic analogy is to answer the analogy question "a is to b as c is to _ ". We utilize the Microsoft Research Syntactic Analogies dataset, which is created by Mikolov (Mikolov et al., 2013c) with size of 8000. To answer the syntactic analogy question "a is to b as c is to d" where d is unknown, we assume that the word representations of a, b, c, d are v_a, v_b, v_c, v_d, respectively. To get d, we first calculate $\widehat{v}_d = v_b - v_a + v_c$. Then, we find out the word d' whose cosine similarity to $\widehat{v}_d$ is the largest. Finally, we set d as d'.

4.4.3 Text Classification

To further evaluate the learned word embeddings, we also conduct 4 text classification tasks using the 20 Newsgroups dataset.[6] The dataset totally contains around 19000 documents of 20 different newsgroups, and each corresponding to a different topic, such as guns, motorcycles, electronics and so on. For each task, we randomly select the documents of 10 topics and split them into training/validation/test subsets at the ratio of 6:2:2, which are emplyed to train, validate and test an L2-regularized 10-categorization logistic regression (LR) classifier. As mentioned in (Tsvetkov et al., 2015), here we also regard the average word embedding of words (excluding stop words and out-of-vocabulary words) in each document as the feature vector (the input of the classifier) of that document. The LR classifier is implemented with the scikit-learn toolkit (Pedregosa et al., 2011), which is an open-source Python module integrating many state-of-the-art machine learning algorithms.

5 Experimental Results

5.1 The Results on Word Similarity

Word similarity is conducted to test the semantic information which is encoded in word embeddings, and the results are listed in Table 2 (first 6 rows). We observe that our models surpass the comparative baselines on five datasets. Compared with the base model CBOW, it is remarkable that our models approximately achieve improvements of more than 5% and 7%, respectively, in the performance on the golden standard Wordsim-353 and RG-65. On WS-353-REL, the difference between CBOW and LMM-S even reaches 8%. The advantage demonstrates the effectiveness of our methods. Based on our strategy, more semantic information will be captured in corpus when adding more latent meanings in the context window. By incorporating mophemes, EMM also performs better than other baselines but fails to get the performance as well as ours. Actually, EMM mainly tunes the distributions of words in vector space to let the morpheme-similar words gather closer, which means it just encodes more morphological properties into word embeddings but lacks the ability to capture more semantic information. Specially, because of the medium-

[6] http://qwone.com/~jason/20Newsgroups

	CBOW	Skip-gram	GloVe	EMM	LMM-A	LMM-S	LMM-M
Wordsim-353	58.77	61.94	49.40	60.01	62.05	**63.13**	61.54
RW	40.58	36.42	33.40	40.83	**43.12**	42.14	40.51
RG-65	56.50	62.81	59.92	60.85	62.51	62.49	**63.07**
SCWS	**63.13**	60.20	47.98	60.28	61.86	61.71	63.02
Men-3k	68.07	66.30	60.56	66.76	66.26	**68.36**	64.65
WS-353-REL	49.72	57.05	47.46	54.48	56.14	**58.47**	55.19
Syntactic Analogy	13.46	13.14	13.94	17.34	**20.38**	17.59	18.30
Text Classification	78.26	79.40	77.01	80.00	80.67	80.59	**81.28**

Table 2: Performance comparison (%) of our LMMs and the baselines on two basic NLP tasks (word similarity & syntactic analogy) and one downstream task (text classification). The bold digits indicate the best performances.

size corpus and the experimental settings, GloVe doesn't perform as well as that described in (Pennington et al., 2014).

5.2 The Results on Syntactic Analogy

In (Mikolov et al., 2013c), the dataset is divided into adjectives, nouns and verbs. For brevity, we only report performance on the whole dataset. As the middle row of Table 2 shows, all of our models outperform the comparative baselines to a great extent. Compared with CBOW, the advantage of LMM-A even reaches to 7%. Besides, we observe that the suffix of "*b*" usually is the same as the suffix of "*d*" when answering question "*a* is to *b* as *c* is to *d*". Based on our strategy, morpheme-similar words will not only gather closer but have a trend to group near the latent meanings of their morphemes, which makes our embeddings have the advantage to deal with the syntactic analogy problem. EMM also performs well on this task but is still weaker than our models. Actually, syntactic analogy is also a semantics-related task because "*c*" and "*d*" are with similar meanings. Since our models are better to capture semantic information, they lead to higher performance than the explicitly morphology-based models.

5.3 The Results on Text Classification

For each one of the 4 text classification tasks, we report the classification accuracy over the test set. The average classification accuracy across the 4 tasks is utilized as the evaluation metric for different models. The results are displayed in the bottom row of Table 2. Since we simply use the average embedding of words as the feature vector for 10-categorization classification, the overall classification accuracies of all models are merely aroud 80%. However, the classification accuracies of our LMMs still surpass all the baselines, especailly CBOW and GloVe. Moreover, it can be found that incorporating morphological knowledge (morphemes or latent meanings of morphemes) into word embeddings can contribute to enhancing the performance of word embeddings in the downstream NLP tasks.

5.4 The Impacts of Parameter Settings

Parameter settings can affect the performance of word embeddings. For example, the corpus with larger corpus size (the ratio of tokens used for training) contains more semantic information, which can improve the performance on word similarity. We analyze the impacts of corpus size and window size on the performance of word embeddings. In the analysis of corpus size, we hold the same parameter settings as before. The sizes of tokens used for training are separately 1/5, 2/5, 3/5, 4/5 and 5/5 of the entire corpus mentioned above. We utilize the result of word similarity on Wordsim-353 as the evaluation criterion. From Fig. 5, we observe several phenomena. Firstly, the performance of our LMMs is better than CBOW at each corpus size. Secondly, the performance of CBOW is sensitive to the corpus size. In contrast, LMMs' performance is more stable than CBOW. As we analyzed in word similarity experiment, LMMs can increase the semantic information of word embeddings. It is worth noting that the performance of LMMs on the smallest corpus is even better than CBOW's performance on the largest corpus. In the analysis of window size, we observe that the performance of all word embeddings trained by different models has a trend to ascend with the increasing of window size as illustrated in Fig. 6. Our LMMs outperform CBOW under all the pre-set conditions. Besides, the worst performance of LMMs is nearly equal to the best performance of CBOW.

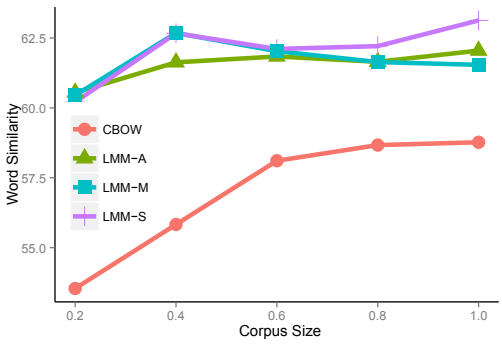

Figure 5: Parameter analysis of corpus size. X-axis denotes the ratio of tokens used for training, and Y-axis denotes the Spearman rank (%) of word similarity.

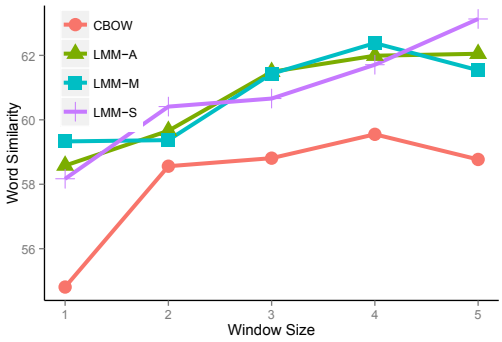

Figure 6: Parameter analysis of window size. X-axis and Y-axis denote the window size and Spearman rank (%) of word similarity, respectively.

5.5 Word Embedding Visualization

To visualize the embeddings of our models, we randomly select several words from the results of LMM-A. The dimensions of the selected word embeddings are reduced from 200 to 2 using Principal Component Analysis (PCA), and the 2-D word embeddings are illustrated in Fig. 7. The words with different colors reflect that they have different morphemes. It is apparent that words with similar morphemes have a trend to group together and stay near the latent meanings of their morphemes. In addition, we can also find some syntactic regularities in Fig. 7, for example, "*physics*" is to "*physicist*" as "*science*" is to "*scientist*", and "*physicist*" and "*scientist*" stay near the latent meaning, *i.e.*, "*human*", of the suffix *-ist*.

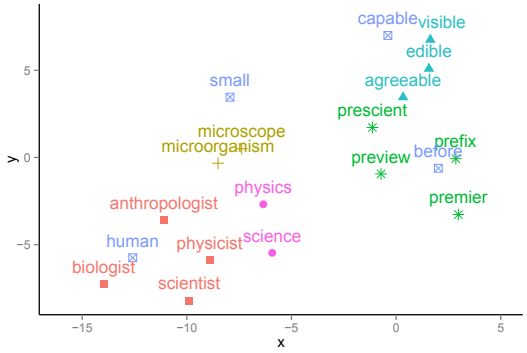

Figure 7: The visualization of word embeddings. Based on PCA, we randomly select several words from word embedding of LMM-A and illustrate them in this figure, "⊠" indicates the latent meanings of morphemes.

6 Conclusion

In this paper, we explored a new direction to employ the latent meanings of morphological compositions rather than the internal compositions themselves to train word embeddings. Three specific models named LMM-A, LMM-S and LMM-M were proposed by modifying the input layer and update rules of CBOW. The source code of LMMs is avaliable at https://github.com/Y-Xu/lmm.

To test the performance of our models, we chose three word-level word embedding models and implemented an Explicitly Morpheme-related Model (EMM) as comparative baselines, and tested them on two basic NLP tasks of word similarity and syntactic analogy, and one downstream text classification task. The experimental results demonstrate that our models outperform the baselines on five word similarity datasets. On the syntactic analogy as well as the text classification tasks, our models also surpass all the baselines including the EMM. In the future, we intend to evaluate our models for some morpheme-rich languages like Russian, German and so on.

Acknowledgments

The authors are grateful to the reviewers for constructive feedback. This work was supported by the National Natural Science Foundation of China (No.61572456), the Anhui Province Guidance Funds for Quantum Communication and Quantum Computers and the Natural Science Foundation of Jiangsu Province of China (No.BK20151241).

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.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 490–500. Association for Computational Linguistics.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificail Intelligence Research (JAIR)*, 49(1–47).

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *International Conference on Artificial Intelligence*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1287–1292.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1651–1660. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Paramveer S Dhillon, Dean P Foster, and Lyle H Ungar. 2015. Eigenwords: spectral word embeddings. *Journal of Machine Learning Research*, 16:3035–3078.

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.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Meeting of the Association for Computational Linguistics: Long Papers*, pages 873–882.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *The Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1517–1526. Association for Computational Linguistics.

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.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

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 *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 141–150.

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

Magnus Sahlgren. 2008. The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53.

Bonggun Shin, Timothy Lee, and Jinho D Choi. 2016. Lexicon integrated cnn models with attention for sentiment analysis. *arXiv preprint arXiv:1610.06272*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054.