KNET: A General Framework for Learning Word Embedding Using Morphological Knowledge

QING CUI, Tsinghua University
BIN GAO and JIANG BIAN, Microsoft Research
SIYU QIU, Nankai University
HANJUN DAI, Fudan University
TIE-YAN LIU, Microsoft Research

Neural network techniques are widely applied to obtain high-quality distributed representations of words (i.e., word embeddings) to address text mining, information retrieval, and natural language processing tasks. Most recent efforts have proposed several efficient methods to learn word embeddings from context such that they can encode both semantic and syntactic relationships between words. However, it is quite challenging to handle unseen or rare words with insufficient context. Inspired by the study on the word recognition process in cognitive psychology, in this article, we propose to take advantage of seemingly less obvious but essentially important morphological knowledge to address these challenges. In particular, we introduce a novel neural network architecture called KNET that leverages both words' contextual information and morphological knowledge to learn word embeddings. Meanwhile, this new learning architecture is also able to benefit from noisy knowledge and balance between contextual information and morphological knowledge. Experiments on an analogical reasoning task and a word similarity task both demonstrate that the proposed KNET framework can greatly enhance the effectiveness of word embeddings.

CCS Concepts: • **Information systems → Language models**; • **Computing methodologies → Lexical semantics**; • **Computing methodologies → Phonology/morphology**; • **Computing methodologies → Neural networks**

Additional Key Words and Phrases: Neural network, word embedding, morphological knowledge

ACM Reference Format:
Qing Cui, Bin Gao, Jiang Bian, Siyu Qiu, Hanjun Dai, and Tie-Yan Liu. 2015. KNET: A general framework for learning word embedding using morphological knowledge. ACM Trans. Inf. Syst. 34, 1, Article 4 (August 2015), 25 pages.
DOI: http://dx.doi.org/10.1145/2797137

1. INTRODUCTION

Neural network techniques have been widely applied to solve text mining, information retrieval (IR), and natural language processing (NLP) tasks, the basis of which yields obtaining high-quality distributed representations of words in a low-dimensional space (i.e., word embeddings). In recent years, efficient methods, such as the continuous bag-of-word (CBOW) model and the continuous Skip-gram (Skip-gram) model [Mikolov et al. 2013b], have been proposed to leverage the surrounding context of a word in

Authors' addresses: Q. Cui, Department of Mathematical Sciences, Tsinghua University, Beijing, 100084, P. R. China; email: cuiqing.thu@gmail.com; B. Gao, J. Bian, and T.-Y. Liu, Microsoft Research, 13F, Bldg 2, No. 5, Danling St Beijing, 100080, P. R. China; emails: bingao@microsoft.com, jiang.bian.prc@gmail.com, tyliu@microsoft.com; S. Qiu, Nankai University, Tianjin, 300071, P. R. China; email: ppqq2356@gmail.com; H. Dai, Fudan University, Shanghai, 200433, P. R. China; email: daihanjun@gmail.com.

DOI: http://dx.doi.org/10.1145/2797137

documents to transform words into vectors (i.e., word embeddings) in a continuous space, which captures both semantic and syntactic relationships between words. The underlying principle in these works lies in that words that are syntactically or semantically similar should have similar surrounding contexts.

While the aforementioned works have demonstrated their effectiveness in various tasks, they also suffer from a couple of limitations.

(1) It is difficult to obtain word embeddings for new words since they are not included in the previous vocabulary. Some previous studies [Mikolov 2012] used a default index to represent all unknown words, but such a solution will inevitably lose information for emerging words.
(2) The embeddings for rare words are unreliable due to the insufficient surrounding contexts. Since the aforementioned works adopt statistical methods, when a word has only a few occurrences in the training data, they will fail in extracting statistical clues to correctly map the word into the embedding space.

In sharp contrast, according to the studies on word recognition in cognitive psychology [Ehri et al. 1991; Ehri 2005], when a human looks at a word, no matter new or rare, he or she can figure out effective ways to understand it. For instance, one sometimes conducts phonological recoding through blending graphemes into phonemes and blending syllabic units into recognizable words; one may also analyze the root/affix of the new word so as to build its connections with his or her known words. Suppose the new word is *inconveniently*. Given its root and affixes (i.e., *in-convenient-ly*), it is natural to guess that it is the adverb form of *inconvenient* and the latter is probably the antonym of *convenient*. Henceforth, morphological word similarity can act as an effective bridge for understanding new or rare words based on known words in the vocabulary. Inspired by this word recognition process, we propose using morphological knowledge to enhance the deep learning framework for learning word embedding. In particular, beyond the contextual information already used in CBOW and Skip-gram, we take advantage of morphological similarity between words in the learning process so as to handle new or rare words.

Although the morphological knowledge contains invaluable information, it might be risky to blindly rely on it. The reason is that the prediction based on morphological word similarity is somehow only a kind of guess, and there exists many counterexamples inconsistent with it. For example, if only looking at the morphological similarity, one may link *convention* to *convenient* since they share a long substring. However, it is clear that these two words are neither syntactically nor semantically similar. In this case, if we stick to the morphological knowledge, the effectiveness of the learned word embeddings could be even worse. To tackle this issue, we once again leverage the findings regarding word recognition in cognitive psychology [Ehri et al. 1991; Ehri 2005]. It has been revealed that humans can take advantage of the contextual information (both the context at the reading time and the context in his or her memory) to correct the unreliable morphological word similarity. By comparing their respective contexts, one can distinguish between *convenient* and *convention* and weaken the morphological connection between these two words in his or her mind. Inspired by this, we also propose updating the morphological knowledge during our learning process. Specifically, we will not fully trust the morphological knowledge and will change it so as to maximize the consistency between contextual information and morphological word similarity.

To sum up the previous discussions, we actually develop a novel neural network architecture that can leverage morphological word similarity for word embedding. Our proposed framework consists of a contextual information branch and a morphological knowledge branch. On one hand, we adopt the state-of-the-art Skip-gram model [Mikolov et al. 2013b] as our contextual information branch for its efficiency and effectiveness. On the other hand, we explore edit distance, longest common substring similarity, morpheme similarity, and syllable similarity as morphological knowledge to build a relation matrix between words and put the relation matrix into the morphological knowledge branch. These two branches share the same word embedding space, and they are combined together using tradeoff coefficients in order to feed forward to the output layer to predict the target word. The back-propagation stage will modify the tradeoff coefficients, word embeddings, and weights in the relation matrix layer by layer. We call the proposed framework *KNET*, for it is a *K*nowledge-powered neural *NET*work. We have conducted experiments on a publicly available corpus, and the results demonstrate that our proposed KNET method can help produce improved word representations as compared with the state-of-the-art methods on an analogical reasoning task and a word similarity task.

The main contributions of the article include the following:

(1) We have proposed a general and robust neural network framework called KNET that can effectively leverage both contextual information and morphological knowledge to learn word embeddings.
(2) The KNET framework can learn high-quality word embeddings especially on rare words and new words with the help of morphological knowledge even when the knowledge is not very reliable.
(3) We also conduct some experimental studies to gain insight about how KNET can benefit from noisy knowledge and balance between contextual information and morphological knowledge.

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

2. RELATED WORK

Word embedding as continuous vectors has been studied for a long time [Hinton et al. 1986]. Many different types of models were proposed for learning continuous representations of words, such as the well-known Latent Semantic Analysis (LSA) [Hofmann 1999] and Latent Dirichlet Allocation (LDA) [Blei et al. 2003]. However, such probabilistic approaches usually yield limitations in terms of scalability. Recently, deep learning methods have been applied to obtain continuous word embeddings to solve a variety of text mining, information retrieval, and natural language processing tasks [Collobert and Weston 2008; Glorot et al. 2011; Mikolov et al. 2013a, 2013b; Socher et al. 2011; Turney 2013; Turney and Pantel 2010; Deng et al. 2013; Collobert et al. 2011; Mnih and Hinton 2008; Turian et al. 2010]. For example, Collobert and Weston [2008] and Collobert et al. [2011] proposed a unified neural network architecture that learns word representations based on large amounts of unlabeled training data to deal with several different natural language processing tasks.

Most recently, Mikolov et al. [2013a, 2013b] proposed the continuous bag-of-words model (CBOW) and the continuous skip-gram model (Skip-gram) for learning distributed representations of words also from a large amount of unlabeled text data; these models can map the semantically or syntactically similar words to close positions in the word embedding space, based on the intuition that the contexts of the similar words are similar. In particular, in the Skip-gram model, a sliding window is employed on the input text stream to generate the training samples. In each sliding window, the model tries to use the central word as input to predict the surrounding words. Specifically, the input word is represented in the 1-of-V format, where V is the size of the vocabulary of the training data and each word in the vocabulary is represented as a vector with only one nonzero element. In the feed-forward process, the input word is first mapped into the embedding space by the weight matrix M. After that, the 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 surrounding words after applying the *softmax* function on it. In the back-propagation process, the prediction errors are propagated back to the network to update the two weight matrices. When the training process converges, the weight matrix M is used as the learned word embeddings. Though the previous works like Skip-gram perform well on some NLP tasks, they still cannot produce high-quality word embeddings for rare words and unknown words since they do not leverage the rich extra knowledge when learning word embeddings.

There are some knowledge-related word embedding works in the literature, but most of them were targeted at the problems of knowledge base completion and enhancement [Bordes et al. 2011; Socher et al. 2013; Weston et al. 2013] rather than producing high-quality word embeddings, which is different with our work. In contrast, some recent efforts have explored how to take advantage of knowledge to produce better word embedding. For example, Qiu et al. [2014] introduced a colearning framework to produce both the word representation and the morpheme representation such that each of them can be mutually reinforced. Yu and Dredze [2014] proposed a new learning objective that ingrates both a neural language model objective and a semantic prior knowledge objective, which can result in better word embedding for semantic tasks. Moreover, a recent work [Bian et al. 2014] took empirical studies on how to incorporate various types of knowledge in order to enhance word embedding. According to this work, morphological, syntactic, and semantic knowledge are all valuable to improve the quality of word embedding. In this article, as we aim at obtaining high-quality word embeddings for rare words and unknown words, we focus on leveraging morphological knowledge since it can generate critical correlation between rare/unknown words with popular ones.

Some previous works have attempted to include morphology in continuous models, especially in the speech recognition field, including 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 Boltzmann 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 142 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 rare/unknown words and to lower the out-of-vocabulary rate.

In the NLP and text mining domain, Luong et al. [2013] proposed a morphological Recursive Neural Network (morphoRNN) that combines recursive neural networks and neural language models to learn better word representations, in which they regarded each morpheme as a basic unit and leveraged neural language models to consider contextual information in learning morphologically aware word representations. We will compare our proposed model with morphoRNN in Section 4.4.

3. WORD EMBEDDING POWERED BY MORPHOLOGICAL KNOWLEDGE

We first introduce how people learn words and understand text by leveraging the morphological knowledge and then describe the knowledge-powered neural network architecture for learning effective word embedding based on both contextual information and morphological knowledge. Afterward, we mention four types of morphological knowledge that are often used by people as well as our framework.

3.1. Word Recognition Process

According to the study on word recognition in cognitive psychology [Ehri et al. 1991; Ehri 2005], when a human learns a new language, he or she usually starts from learning some basic words and gradually enlarges his or her vocabulary during the learning process. He or she also learns the language grammar and morphological knowledge so as to build cross-links between words in his or her knowledge base; for example, the adjective form of *care* is *careful* and its adverb form is *carefully*. When he or she encounters an unknown or unfamiliar word, he or she will try to explore several different channels to recognize it [Ehri et al. 1991]:

Recoding (or Decoding). One can either sound out and blend graphemes into phonemes or work with larger chunks of letters to blend syllables into recognizable words. For example, *psychology* can be pronounced as *psy-cho-lo-gy*, in which *psy* means know or study, *cho* means mind or soul, and *logy* means academic discipline. Thus, one may guess *psychology* is an academic discipline that studies something in the mind or soul.

Analogizing. One can use his or her known words to read the new word [Goswami 1986]. If the new word is morphologically similar to several known words, one will guess the meaning of the new words based on the meanings of these known words. For example, *admob* appears in a news article as a new word to a reader. The reader may quickly understand that it is related the advertisements on mobile devices, simply because *admob* is composed by *ad* and *mob*, which are substrings of *advertisement* and *mobile*, respectively.

Prediction. One can use context and letter clues to directly guess the meaning of the unknown word [Chapman 1998]. Sometimes, one may even retrieve the context of the word in his or her memory and make associations to the current context to guess the meaning of the word. For example, *inmate* is an unknown word to a reader, but according to the context *Inmates and police officers held a basketball game in the Fox River Prison last Tuesday evening*, one can easily guess that *inmate* means prisoner in the sentence.

In this process, the different channels may reinforce each other. On one hand, sometimes contexts could be insufficient; for example, there are simply not many contexts surrounding the unknown word, and there is no historical context in the memory either. In this case, it is extremely hard to directly predict the meaning of the word. In contrast, decoding and analogizing could do a good job since they can work in a context-free manner. On the other hand, sometimes decoding and analogizing can result in errors. For example, *convention* and *convenient* are morphologically very similar since they share a long substring *conven*; however, their meanings are quite different. In this case, blindly relying on morphological knowledge will bring in a lot of noise, but contextual information can help one to successfully distinguish these two words. By refining one's morphological knowledge with the help of the contextual information, he or she can avoid the misrecognition.

Please note that all of the previous process happens within just a second, which enables humans to be super powerful in recognizing unknown or unfamiliar words. This phenomenon strongly inspires us to leverage both morphological knowledge and contextual information to learn word embeddings. Accordingly, we propose a novel neural network architecture that consists of a morphological knowledge branch and a contextual information branch. Details will be given in the next subsection.

3.2. Neural Network Architecture for KNET

In this subsection, we describe our proposed new neural network architecture that leverages both contextual information and morphological knowledge to learn word embedding. We use the Skip-gram model [Mikolov et al. 2013b] as the basis of our proposed framework.[1] Skip-gram is a neural network model to learn word representations, the underlying principle of which is that similar words should have similar contexts.[2]

To be more specific, given a sequence of training words $w_1, w_2, \ldots, w_T$, the objective of the Skip-gram model is to maximize the following average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-N\leq j\leq N, j\neq 0}\log p(w_{t+j}|w_t), \tag{1}$$

where w_t and w_{t+j} are two words inside a sliding window, and N indicates that the size of the sliding window is $2N+1$. The conditional probability $p(w_{t+j}|w_t)$ is defined using the following *softmax* function:

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_w \exp({v'_w}^T v_{w_I})}, \tag{2}$$

where v_w and v'_w are the *input* and *output* representation vectors of w, and the sum in the denominator is over all words in the vocabulary. Here we use w_I to denote the input word (i.e., w_t) and use w_O to denote the output word (i.e., w_{t+j}).

It is difficult and impractical to directly optimize the previous objective because computing the derivative is proportional to the vocabulary size, which is often very large. Several approaches [Morin and Bengio 2005; Bengio et al. 2003; Bengio and Senecal 2008] have been employed to tackle this problem. The state-of-the-art method is noise-contrastive estimation (NCE) [Gutmann and Hyvärinen 2012], which aims at fitting unnormalized probabilistic models. NCE can approximate the log probability of the *softmax* function by performing logistic regression to discriminate between the observed data and some artificially generated noises. It was first adapted in the neural language model in Mnih and Teh [2012] and was then applied to the inverse vector log-bilinear model [Mnih and Kavukcuoglu 2013]. Another simpler method is negative sampling (NEG) [Mikolov et al. 2013b], which generates *k* noise samples for each input word to estimate the objective.

By using NEG, the *softmax* conditional probability $p(w_{t+j}|w_t)$ will be replaced by

$$J(\theta) = \log\sigma\big({v'_{w_O}}^T v_{w_I}\big) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\Big[\log\sigma\big(-{v'_{w_i}}^T v_{w_I}\big)\Big], \tag{3}$$

where θ is the model parameter including the word embeddings, σ denotes the logistic function, and $P_n(w)$ represents the noise distribution, which is set as the 3/4 power of the unigram distribution $U(w)$, that is, $P_n(w) = U(w)^{3/4}/Z$ (Z is a normalizer) [Mikolov et al. 2013b]. Then, we can estimate the gradient of $J(\theta)$ by computing

$$\frac{\partial J(\theta)}{\partial\theta} = \big(1-\sigma\big({v'_{w_O}}^T v_{w_I}\big)\big)\frac{\partial {v'_{w_O}}^T v_{w_I}}{\partial\theta} - \sum_{i=1}^{k}\left[\sigma\big({v'_{w_i}}^T v_{w_I}\big)\frac{\partial {v'_{w_i}}^T v_{w_I}}{\partial\theta}\right]. \tag{4}$$

By summing over *k* noise samples instead of a sum over the entire vocabulary, the training time yields a linear scale to the number of noise samples and becomes independent of the vocabulary size.

[1]Note that although we task the Skip-gram model as an example to illustrate our framework, a similar framework can be developed on the basis of any other word embedding models.
[2]In our model and experiments, we used the sliding window as the context of a word.

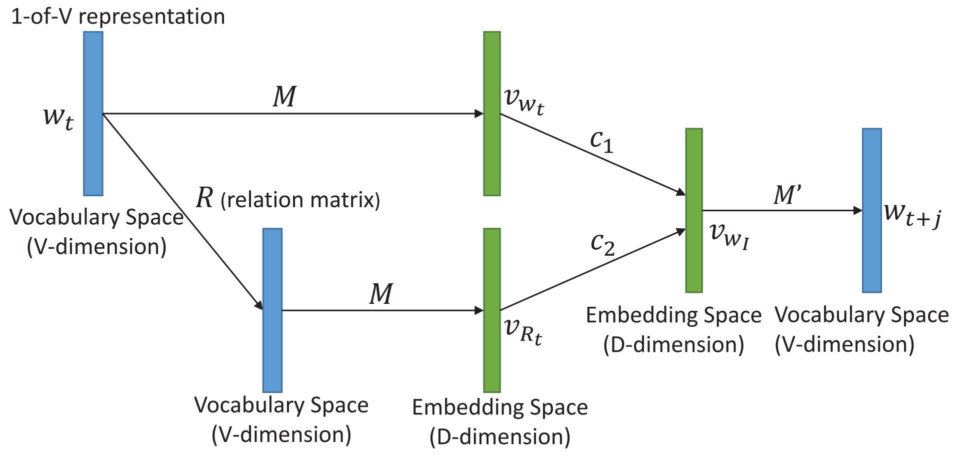
1-of-V representation
w_t
M
v_{w_t}
c_1
R (relation matrix)
Vocabulary Space (V-dimension)
c_2
M'
v_{w_I}
w_{t+j}
M
v_{R_t}
Embedding Space (D-dimension)
Vocabulary Space (V-dimension)
Vocabulary Space (V-dimension)
Embedding Space (D-dimension)

Fig. 1. The neural network architecture of the proposed KNET framework.

To incorporate morphological knowledge into the learning process, we propose a new neural network architecture. Beyond the basic Skip-gram model that predicts a target word based on its context, the proposed new method introduces a parallel branch that leverages morphological knowledge to assist predicting the target word, as shown in Figure 1. Intuitively, when a word w_t is the central word in the context window, we predict the surrounding words by leveraging not only the representation of word w_t as contextual information (referred to as *contextual information branch*) but also the representations of the words that are morphologically similar to w_t (referred to as *morphological knowledge branch*). Therefore, the objective of the proposed model is the same as Equation (1) (i.e., we want to maximize the average probability of word prediction) except that we replace the input word representation v_{w_I} in the softmax function in Equation (2) by a new formulation, which is combined from both the contextual information branch and the morphological knowledge branch. Now we introduce the detailed formulation of v_{w_I}.

According to Figure 1, to obtain the representation of a central word w_t from the morphological knowledge branch, it is necessary to find the set of words that are morphologically similar to w_t, which is denoted as R_t. Then, we can extract the embedding of each word in R_t from the embedding matrix M shared with the contextual information branch. After that, a corresponding knowledge representation of R_t can be computed by feeding forward the relationship layer, which is written as

$$v_{R_t} = \sum_{w \in R_t} s(w_t, w) v_w, \tag{5}$$

where $s(w_t, w)$ is initialized with the similarity score, the methods of computing which will be introduced in Section 3.3. Actually v_w is the ith row of matrix M, where i is the index of the word w in the vocabulary, and $s(w_1, w_2)$ is the element of relation matrix R at (i, j) that are the indices of words w_1 and w_2, respectively. To ensure the quality of morphological knowledge and control the number of parameters, we only leverage the top words with the highest morphological similarity scores as R_t. For example, in our experiments, an input word can only connect to at most five words in the relationship layer. This sparse structure will not change during training, and only the weights of these connections will be updated. Therefore, we will not suffer from a huge number of parameters even if R is learned. To sum up, the morphological similarity score calculated in Section 3.3 is only employed as the initial values of R as well as for determining the top five words in R_t.

Finally, an aggregated representation of the input word, denoted as v_{w_I}, can be calculated as the weighted sum of the representations from the contextual information branch and the morphological knowledge branch, that is,

$$v_{w_I} = c_1(w_t)v_{w_t} + c_2(w_t)v_{R_t}, \tag{6}$$

where $c_1(\cdot)$ and $c_2(\cdot)$ are the functions of w_t and yield much dependency on the word frequency. Intuitively, frequent words are associated with much more training samples than rare words, such that it is easy to collect rich contextual information for frequent words, while the contextual information for rare words might be insufficient. In contrast, the volume of morphological knowledge of a word usually has little correlation to the word frequency, and thus rare words can still rely more on the morphological knowledge even though the contextual information is not reliable. Therefore, the balancing function $c_1(\cdot)$ and $c_2(\cdot)$ should be related to word frequency. Specifically, we divide the words into a number of buckets according to their frequencies, and all the words in the same bucket will share the same values of $c_1(\cdot)$ and $c_2(\cdot)$.

A more explicitly intuitive way to interpret the previous model is as follows. For each word w_t, we use one row in the embedding matrix M to encode its contextual embedding. In addition, by using matrix R, we can identify a couple of morphologically similar words to w_t. Then we can also extract the contextual embeddings of these similar words from M and take the weighted average of these embedding vectors as the morphological embedding for the original word w_t. Then finally, the overall embedding of w_t is computed as the weighted combination of its contextual embedding and morphological embedding. Matrix M' is used to predict the surrounding word w_{t+j} based on the overall embedding of w_t. Similar to v_w, the output representation vector v'_w in Equation (2) is the ith row of matrix M', where i is the index of the word w in the vocabulary. In the back-propagation process, the parameters in M, R, M', and multiple pairs of c_1 and c_2 (corresponding to different frequency buckets) are updated. When the training process converges, we take the matrix M as the learned word embeddings. In our implementation, we take the NEG strategy to calculate the gradient in Equation (4), in which v_{w_I} is substituted by Equation (6), and learn the parameters with standard gradient descent techniques and error back-propagation procedure. The details of the back-propagation process and the learning process are described in Appendix A. We call the proposed framework as *KNET*, considering that it is a *K*nowledge-powered neural *NET*work.

3.3. Morphological Knowledge

As compared to Skip-gram, the uniqueness of our model lies in the introduction of the morphological knowledge branch. In this subsection, we will make discussions on how we realize this new branch. In particular, we propose four types of naturally defined knowledge. In the previous four types of knowledge, Edit Distance Similarity and Longest Common Substring Similarity are string similarity measures, while Morpheme Similarity and Syllable Similarity are based on morphemes and syllables accordingly. For ease of reference, we denote all these four types of knowledge as morphological knowledge. Note that this is not a complete study on morphological knowledge, but we can use these four specific types as examples to show the effectiveness of the proposed framework. Any other kinds of knowledge consisting of a pairwise relationship can be used under the KNET framework.

3.3.1. Edit Distance Similarity (Edit). Edit distance is a way of quantifying how dissimilar two strings (e.g., words) are by counting the minimum number of operations required to transform one string into the other. The operations might be *letter insertion*, *letter deletion*, or *letter substitution*. We calculate the edit distance similarity score for two words w_1 and w_2 as

$$s_{Edit}(w_1, w_2) = 1 - \frac{d(w_1, w_2)}{\max(l(w_1), l(w_2))},$$

where $d(w_1, w_2)$ represents the edit distance of the two words and $l(w_1), l(w_2)$ are the corresponding word lengths.

3.3.2. Longest Common Substring Similarity (LCS). Longest common substring similarity is defined as the ratio of the length of the longest shared substring of two words (denoted by $g(w_1, w_2)$) and the length of the longer word, that is,

$$s_{LCS}(w_1, w_2) = \frac{g(w_1, w_2)}{\max(l(w_1), l(w_2))}.$$

3.3.3. Morpheme Similarity (Morpheme). Morpheme similarity is calculated based on the shared roots (or stems) and affixes (prefix and suffix) of two words. Suppose each word of w_1 and w_2 can be split into a set of morphemes (denoted by $F(w_1)$ and $F(w_2)$); then the morpheme similarity of the two words is calculated as

$$s_{Morpheme} = \frac{|F(w_1) \bigcap F(w_2)|}{\max(|F(w_1)|, |F(w_2)|)},$$

where $|\cdot|$ outputs the size of the set.

3.3.4. Syllable Similarity (Syllable). Syllable similarity is calculated based on the shared syllables of two words. Suppose both w_1 and w_2 can be split into a set of syllables (denoted by $G(w_1)$ and $G(w_2)$); then the syllable similarity of the two words is calculated as

$$s_{Syllable} = \frac{|G(w_1) \bigcap G(w_2)|}{\max(|G(w_1)|, |G(w_2)|)}.$$

In addition to using these four types of morphological word similarity separately, one can also combine them together. In the next section, we will conduct experimental study on all these different choices.

4. EXPERIMENTAL EVALUATION

In this section, we report the experimental results regarding the effectiveness of our proposed KNET framework. Our experiments are mainly composed of three parts. In the first part, we compare KNET with several baselines based on Skip-gram to show the effectiveness and robustness of our framework. Then we compare KNET with morphoRNN on two word similarity tasks (one mainly contains frequent words while the other contains lots of rare and new words) to show that our framework can achieve high-quality word embedding on rare and new words. After that, we conduct some case studies to gain a deeper understanding about how KNET can benefit from noisy knowledge to obtain high-quality word embedding on rare words; we also give an empirical study to gain insight about the balancing function between the contextual information branch and the morphological knowledge branch.

4.1. Evaluation Tasks

We evaluated the performance of the learned word representations on the following two tasks.

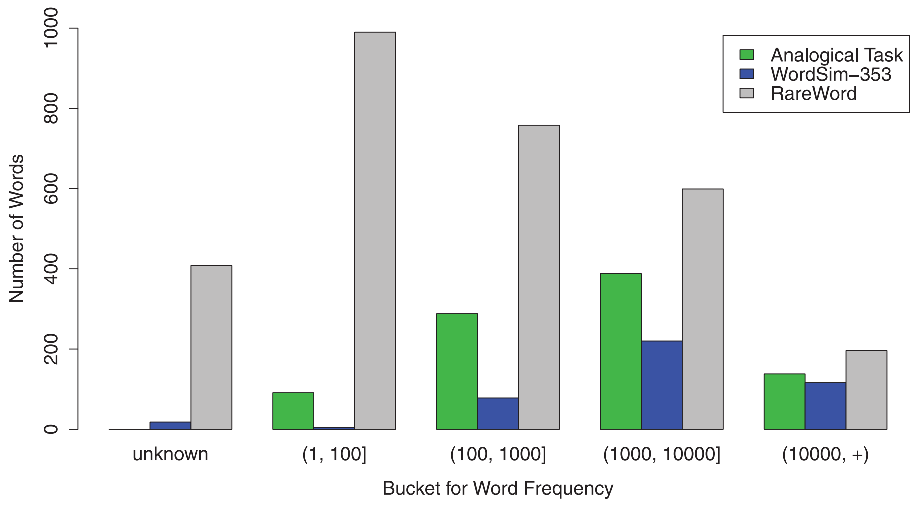

Fig. 2. Word frequency distributions for the word similarity test sets in the *enwiki9* corpus.

4.1.1. Analogical Reasoning Task. The analogical reasoning task was introduced by Mikolov et al. [2013a]. The task consists of 19,544 questions of the form "a is to b is as c is to _," denoted as $a : b \rightarrow c : ?$. Suppose $\overrightarrow{w}$ is the learned word representation vector of word w normalized to unit norm. Following Mikolov et al. [2013a], we answer this question by finding the word d^* whose representation vector is the closest to vector $\overrightarrow{b} - \overrightarrow{a} + \overrightarrow{c}$ according to cosine similarity excluding b and c, that is,

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

The question is regarded as answered correctly only when d^* is exactly the answer word in the evaluation set. There are two categories in the task, with 8,869 semantic analogies (e.g., *England* : *London* $\rightarrow$ *China* : *Beijing*) and 10,675 syntactic analogies (e.g., *amazing* : *amazingly* $\rightarrow$ *unfortunate* : *unfortunately*).

4.1.2. Word Similarity Task. WordSim-353 [Finkelstein et al. 2001] is a standard dataset for evaluating vector space models on word similarity. It contains 353 pairs of nouns without context. Each pair is associated with 13 to 16 human judgments on similarity and relatedness on a scale from 0 to 10. 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 by word embeddings and the human judgments.

In addition to WordSim-353, we also used the RareWord dataset [Luong et al. 2013] to test the performance of the proposed model, which contains 2,034 pairs of rare words. According to the frequency distribution in the training data *enwik9* (see Figure 2), half of the words in the RareWord are tail words (word frequency < 100), while WordSim-353 is mainly composed of frequent words. Furthermore, the RareWord dataset contains more than 400 unknown words (which have not appeared in the training data and thus do not have word embeddings available by themselves). We make use of these unknown words to test the capability of our KNET model in dealing with new words. We use v_{R_t} in Equation (5) as the embedding for an unknown word. Specifically, we computed its similarity to all the known words using a certain type of morphological knowledge, and then we selected the top five closest known words and calculated the linear combination of their embedding vectors as the representation for the unknown word (the normalized similarity scores were used as the combination weights).

4.2. Experimental Setup

4.2.1. The Construction of Relation Matrix R. In our experiments, we employed four types of morphological knowledge. *Edit* and *LCS* can be computed directly from the definitions. For *Morpheme*, we used a public tool called Morfessor [Creutz and Lagus 2007], which can split a word into morphological segments with prefix, stem, and suffix tags. For *Syllable*, we implemented the hyphenation tool proposed by Liang [1983], which has been used in many editing software programs including LATEX to break words by syllables.

For each of them, given a word w, we calculated its similarities to all the other words and selected the top five words with the highest similarity to build the relation edges in the weight matrix R, leaving the other edges with zero.[3] We tested the R matrix built based on each single type of knowledge, and we also tested the R matrix built based on several types of knowledge through combination. Specifically, given the four ranked lists of words from the morphological knowledge, we combined them into a union set and selected the top five words that got more votes by the four knowledge types, denoted as *Combination*. We tried to directly use the top five similar words as the representation of the word and use the cosine similarity to evaluate the representations. We find that the representations are so sparse that almost all pairs of words have zero similarity scores and it is worthless for word representations. Therefore, we do not report the results of this trivial method here.

4.2.2. The Balancing Parameters. As discussed in Section 3.2, the balancing parameters in KNET are related to word frequency. Since they are not the main focus of this article, for simplicity, we used a greedy algorithm to divide the words into a certain number of buckets. Specifically, if we intend to split words into b buckets, we first rank the words in the vocabulary by their frequencies in descending order and put the words into the first bucket sequentially until the sum of the word's frequency in the first bucket reaches the $1/b$ of the total word frequency. After that, we feed the rest of the buckets consecutively in a similar way, and eventually the sum of the frequency in each of the b buckets is approximately equal to $1/b$ of the total frequency. We let all words in one bucket share the same balance coefficients. In our experiments, we initialize both c_1 and c_2 with 0.5 in each bucket. We set the number of buckets to 1,000 in the analogical reasoning task and WordSim-353 word similarity task since they are mainly composed of frequent words, while we set the number of buckets to 100,000 in the RareWord word similarity task since it contains many rare or unknown words. More discussions about the balancing between contextual information and morphological knowledge as well as the relationship with the number of buckets can be found in Section 4.6. Note that we choose the different number of buckets with the highest performance on separate tasks, respectively.

4.3. Comparison with Baselines Related to Skip-gram

4.3.1. Datasets. The training set used in this part of experiments is the *enwik9* corpus,[4] which is built from the first billion characters from Wikipedia. This corpus contains a total of 123.4 million words. We used Matt Mahoney's text preprocessing script[5] to process the corpus. After preprocessing, all digits were replaced with English words

[3]In our experiments, the performance varies little when the number of similar words varies from three to 50.
[4]http://mattmahoney.net/dc/enwik9.zip.
[5]http://mattmahoney.net/dc/textdata.html.

(e.g., *3* was replaced with *three*), and the metadata and hyperlinks were removed. Furthermore, all words that occurred less than five times in the training data were discarded from the vocabulary, resulting in a vocabulary of 220,000 words. The out-of-vocabulary words were ignored in training.

4.3.2. Compared Methods and Experimental Settings. In our experiments, we compare the following methods:

Skip-gram: this is a popularly used baseline model introduced by Mikolov et al. [2013b].

Skip-gram + Edit/LCS/Morpheme/Syllable/Combination Input Feature: this is a group of baselines using the morphological features as additional inputs during training of the Skip-gram model. Specifically, the input is no longer a 1-of-V representation. Instead, we will append the morphological feature, which is the corresponding row of the relation matrix R, to the 1-of-V vector. Thus, the input is a vector of length $2V$ and the projection matrix has the size of $2V \times D$, where D is the dimension of word embeddings. We denote this group of baselines as Skip-gram + Input Feature.

Skip-gram + Fixed Edit/LCS/Morpheme/Syllable/Combination Relation Matrix: this is the same with our proposed model except that we do not update the relation matrix while learning the word embedding. We design this baseline to verify that blindly sticking to the morphological knowledge may even hurt in some cases, which is coherent with human cognitive psychology. We denote this group of baselines as Skip-gram + Fixed Relation Matrix.

Skip-gram + Edit/LCS/Morpheme/Syllable/Combination Relation Matrix: this is the proposed KNET model, in which we employed the same types of morphological knowledge and updated all the parameters in the training process. Note that our model can be degraded to Skip-gram + Input Feature by fixing c_1, c_2, R and not sharing M in the training process. If we only fix R in KNET, we get Skip-gram + Fixed Relation Matrix.

In all of these methods, we set the dimension of word embeddings to 100 and the context windows size to 5. We employed the negative sampling technique to train these models and the number of negative samples was set to 3.

With these settings, the training time of the proposed model was only about 1.5 times of the original Skip-gram model, showing that the KNET framework is very efficient. Actually, its training can finish in about 15 minutes on a single machine with four cores.

4.3.3. Results. Table I shows the performance of the methods on the two tasks, respectively. RareWord shows the results obtained by representing all unknown words as a default zero vector. RareWord* shows the results obtained by predicting the embedding of unknown words with the relation matrix and the embedding of known words using the method described in Section 4.1.2. In the vertical direction, we can find that the performance of model groups follows the order of **Skip + Input Feature $\prec$ Skip-gram $\prec$ Skip-gram + Fixed Relation Matrix $\prec$ Skip-gram + Relation Matrix (KNET)**, where $\prec$ means worse than.

By **Skip-gram $\prec$ Skip-gram + Fixed Relation Matrix** and **Skip-gram $\prec$ Skip-gram + Relation Matrix (KNET)**, we can observe the following:

(1) Adding morphological knowledge, either single type or combined knowledge, to the Skip-gram model can consistently increase all types of accuracies in the analogical reasoning task and word similarity task. This shows that morphological knowledge can effectively improve the quality of the learned word embeddings.
(2) Looking inside the Skip-gram + Relation Matrix (KNET) group, we can find that *Morpheme* performs the best among the five types of knowledge in terms of semantic

Table I. Comparison Between KNET and Baselines Related to Skip-gram on the Analogical Reasoning Task and the Word Similarity Task

	Analogical Reasoning Task			Word Similarity Task		
Model	Semantic Accuracy	Syntactic Accuracy	Total Accuracy	WordSim-353	RareWord	RareWord*
Skip-gram	21.85%	34.64%	28.84%	0.6283	0.1685	-
+ Edit Input Feature	13.67%	27.85%	21.41%	0.5788	0.1625	0.3087
+ LCS Input Feature	13.65%	28.30%	21.65%	0.6055	0.1679	0.3180
+ Morpheme Input Feature	13.55%	23.66%	19.07%	0.5954	0.1595	0.3068
+ Syllable Input Feature	11.94%	25.30%	19.24%	0.5657	0.1554	0.2944
+ Combination Input Feature	13.67%	28.52%	21.78%	0.5759	0.1659	0.3228
+ Fixed Edit Relation Matrix	21.42%	40.62%	31.91%	0.6384	0.1962	0.3595
+ Fixed LCS Relation Matrix	23.48%	41.24%	33.18%	0.6452	0.1982	0.3609
+ Fixed Morpheme Relation Matrix	23.94%	41.04%	33.28%	0.6451	0.1800	0.3235
+ Fixed Syllable Relation Matrix	22.48%	40.61%	32.38%	0.6482	0.1814	0.3301
+ Fixed Combination Relation Matrix	21.17%	43.60%	33.42%	0.6423	0.2085	0.3686
+ Edit Relation Matrix	23.59%	43.49%	34.46%	0.6532	0.2103	0.3797
+ LCS Relation Matrix	23.70%	44.50%	35.06%	0.6545	0.2043	0.3700
+ Morpheme Relation Matrix	**24.86%**	43.68%	35.14%	**0.6612**	0.1909	0.3347
+ Syllable Relation Matrix	24.53%	41.88%	34.01%	0.6607	0.1916	0.3371
+ Combination Relation Matrix	23.58%	**46.90%**	**36.32%**	0.6495	**0.2191**	**0.3932**

We report the semantic/syntactic/total accuracy in the analogical reasoning task and Spearman's ρ correlation in the word similarity task. The word embeddings are trained on the *enwiki9* data with dimension 100.

accuracy in the analogical reasoning task and WordSim-353 in the word similarity task. Since these two tasks focus on the semantic relationship, we hypothesize the reason is that morphemes (like roots and affixes) are basic units in word composition, and it implies accurate semantic correlation if two words share the same root. On the other hand, *Combination* performs the best in terms of syntactic accuracy in the analogical reasoning task and RareWord in the word similarity task. Besides, *Edit* and *LCS* are always better than *Morpheme* and *Syllable* in these two tasks. Considering that these tasks contain many rare words and building a connection to relatively frequent words is the key to learning reasonable word embedding for rare words, the recall of the truly similar words in these tasks is more critical than the precision. *Edit* and *LCS* naturally have high recall of the truly similar words because they directly calculate the similarity score in the letter level. Even though *Edit* and *LCS* have high recall, every single type of morphological knowledge has its own limitations, and thus combining them together will further increase the recall of truly similar words, which leads to better performance on these two tasks.

(3) Focusing on the performance of Skip-gram + Relation Matrix (KNET) on the word similarity task, we have the following observations. The average gain on RareWord (10.08%) is much higher than that on WordSim-353 (4.38%), which illustrates that leveraging morphological knowledge will especially benefit rare words. Since there is no sufficient contextual information for the rare words in the training data, building connections between words using the morphological knowledge will provide additional evidence for us to generate effective embeddings for these rare words. While the rare words can benefit from the morphological knowledge, we can keep the noise brought by it away from the frequent words. The secret is that in our KNET framework, frequent words rely more on the contextual information while rare words rely more on the morphological knowledge by balancing between these two branches. More discussion about this can be seen in Section 4.6. By using the embeddings of known words and relation matrix to predict those of the unknown

words, we can achieve significant improvement on the RareWord, with almost a 100% increment compared with Skip-gram. This indicates that our proposed KNET framework can effectively deal with new emerging words, which yields a potential impact for natural language processing applications in the real world.

By **Skip-gram + Input Features $\prec$ Skip-gram $\prec$ Skip-gram + Fixed Relation Matrix**, we can observe that simply adding morphological knowledge as additional input features does not work as expected and conversely hurts the language model. Recall that our KNET framework can be degraded to Skip-gram + Input Features by fixing c_1, c_2, R and not sharing M in the training process. We can also obtain Skip-gram + Fixed Relation Matrix by fixing R from KNET. Thus, the difference between Skip-gram + Input Features and Skip-gram + Fixed Relation Matrix is fixing c_1, c_2 and not sharing M, which leads to the great gap of performance; that is, one is worse than Skip-gram and the other is better than it. It indicates that c_1, c_2 and sharing M bring the core effectiveness of our proposed KNET framework. Actually, M is the channel that the contextual information branch and the morphological knowledge branch used to communicate with each other, while c_1 and c_2 are the key factors of balancing between these two branches. With both of these aspects, our KNET framework can effectively leverage the morphological knowledge while keeping consistent with the context. In this perspective, we can easily understand that the language model suffers from the artificially appended identification when we simply add the noisy morphological knowledge as additional input features.

By **Skip-gram + Fixed Relation Matrix $\prec$ Skip-gram + Relation Matrix (KNET)**, we verified the hypothesis that blindly sticking to the morphological knowledge may even hurt in some cases and we can leverage the context to avoid the misrecognition brought by the morphological knowledge that is coherent with the human cognitive psychology introduced in Section 3.1. In this manner, the contextual information branch and the morphological knowledge branch can reinforce each other. The morphological knowledge helps when the context is insufficient, while the context can correct and refine the noisy morphological knowledge.

To sum up, through the comparison of experiment results among these models, we can claim that our KNET framework is a general, effective, and robust framework that can leverage both contextual information and morphological knowledge while making them harmonize with each other. Specifically, with sharing M, these two branches can communicate; with updating c_1, c_2, these two branches can balance; and with updating R, M, these two branches can reinforce each other as a united framework. By analyzing the results of KNET on two word similarity tasks, we find that our framework can learn the effective word embedding especially on rare words, which will be further verified in the next subsection.

4.4. Comparison with the MorphoRNN Model

4.4.1. Datasets and Experimental Settings. To make the comparison fair, we used the same corpus as Luong et al. [2013], which is the April 2010 snapshot of the Wikipedia corpus denoted as *wiki2010*. After the preprocessing similar to *enwiki9* and ignoring the words that occurred less than 10 times, there are 487 million tokens with a vocabulary of 466,000 words. The experimental settings are almost the same as in the previous experiments except that we set the dimension of word embeddings to 50 to be consistent with Luong et al. [2013]. Because they did not publish the codes and only published the trained word embedding on *wiki2010*, considering there are many words in the analogical reasoning task but not in the vocabulary of *wiki2010*, we cannot fairly compare morphoRNN with KNET on the analogical reasoning task. Therefore, we will focus on the word similarity task, which is also the main part in their work, and we

Table II. Comparison Between KNET and MorphoRNN on the Word Similarity Task Measured by the Spearman's ρ Correlation

Model	WordSim-353	RareWord	UnknownWords
HSMN + csmRNN	**0.6458**	0.2231	0.1694
C&W + csmRNN	0.5701	0.3436	0.0946
Skip-gram	0.6010	0.2855	-
+ Edit Relation Matrix	0.5953	0.3714	0.3087
+ LCS Relation Matrix	0.6076	**0.3780**	0.3559
+ Morpheme Relation Matrix	0.5983	0.3647	0.4250
+ Syllable Relation Matrix	0.6021	0.3715	0.3528
+ Combination Relation Matrix	0.6094	0.3752	**0.4467**

The word embeddings are trained on the *wiki2010* data with dimension 50. Here we refer the numbers reported in their paper directly.

will refer to the numbers reported in their paper. Besides the results on WordSim-353 and RareWord, we also extracted the unknown words that appear in RareWord but not in the vocabulary of wiki2010 and built a new test dataset consisting of the word pairs in RareWord that contain at least one of the previous unknown words. We denote this test dataset as UnknownWord. There are 60 unknown words and 64 word pairs in UnknownWord. For the UnknownWord dataset, we used the published word embeddings of Luong et al. [2013] in the evaluation.[6]

4.4.2. Compared Methods and Results. The morphoRNN was proposed by Luong et al. [2013], which has been introduced in Section 2. In their work, they proposed two kinds of morphoRNNs: cimRNN, which is context insensitive, and csmRNN, which is context sensitive. Since the context-sensitive models are consistently better than the context-insensitive models as expected, we only compare KNET with their context-sensitive models. In their experiments, they make use of two publicly available embeddings provided by Collobert et al. [2011] and Huang et al. [2012] to initialize their models. Following their notation, we denote these two morphoRNN models as **C&W + csmRNN** and **HSMN + csmRNN,** which are the best models in their work. The results of these two models and our KNET models are shown in Table II.

From the results, we can observe that the best model on WordSim-353 is HSMN + csmRNN. As explained in Luong et al. [2013], the reason is that HSMN performs well on frequent words and HSMN + csmRNN uses the word embedding produced by HSMN in the initialization. However, although HSMN + csmRNN can do a great job on frequent words, its performance on RareWord is bad. C&W + csmRNN performs better than HSMN + csmRNN on RareWord, but they are both beaten by the proposed KNET models powered by different types of morphological knowledge, showing the effectiveness of KNET. The advantage of our models is even greater on UnknownWords in which every word pair contains at least one unknown word. Compared with Skip-gram, our proposed model can greatly improve the performance on RareWord, while the performance on WordSim-353 is flat. This is reasonable because the morphological knowledge can help improve the quality of word embeddings for rare words, while it can barely help words with already plenty context information especially in the low-dimension embedding space. Note that the performance of the models in Table II is a little worse than those in Table I, because the dimension of word embeddings is 50 on *wiki2010,* while it is 100 on *enwiki9*.

Besides the promising performance on rare words, KNET has several other advantages over morphoRNN.

[6]http://stanford.edu/ lmthang/morphoNLM/.

(1) KNET is much more efficient, since it does not need initialization by other word embeddings. In contrast, C&W + csmRNN is initialized with the C&W embeddings, which were trained for about 2 months. Furthermore, KNET is much more efficient than morphoRNN models in both the language model and the recursive structure, so that it can be trained in less than 20 minutes on a single machine with four cores.
(2) KNET is more robust, since it can benefit from the noisy knowledge by updating the relation matrix and balancing between contextual information and morphological knowledge. In contrast, morphoRNN models used a hierarchical structure to co-consider the morphological knowledge and the contextual information, and thus the noise accumulated in the morphological layer (the RNN structure) might be propagated to the context layer (the language model).
(3) KNET is more flexible, since it can leverage not only the morpheme knowledge but also other morphological knowledge types such as *Edit* and *LCS*, which is not applicable for morphoRNN. Actually, KNET can leverage any kind of pairwise relationship that can cover most of the relations in knowledge bases such as *WordNet* and *Freebase*. We leave this for future work.

The best result of Qiu et al. [2014] on the RareWord dataset is 0.3288, which does not beat C&W + csmRNN. In our understanding, the main reason is that directly leveraging morphemes would bring more noise than word-level morphological knowledge, but their model could not handle the noise that morphemes bring in since they did not update the relation matrix. In addition, the flexibility of our framework is also an advantage against their model. Their model can be regarded as a special case of our framework if we replace the top similar words by morphemes.

4.5. Case Study

To further understand how KNET benefits from the noisy morphological knowledge, we sampled some rare words and compared the closest words to them in different word embedding spaces and morphological knowledge to check the effect of the learning process. Specifically, for a given word, we extracted its representation vector in the 100-dimension embedding space, which we obtained in Section 4.3, and calculated its cosine similarity with the representation vectors of all the other words. Then we showed the five most similar words generated by the methods under investigation in Table III. According to Table I, the combination of four types of knowledge achieved the best performance on most tasks; therefore, we only show the results for the baseline method (Skip-gram) and the combination method (Skip-gram + Combination Relation Matrix). We also show the most similar words directly given by the combination of the four types of knowledge without going through the learning process (denoted as Combined Knowledge), which can give us an overview of how the original morphological knowledge looks. Note that the baseline actually does a good job on frequent words and the results of our model on those words are similar to the baseline, so we only sampled some rare words to demonstrate the power of the KNET model.

From Table III, we have the following observations:

(1) We can see that the Skip-gram method often fails in finding reasonable semantically or syntactically related words for rare words. For example, *uninformative* only appears 18 times in the training corpus, and thus its nearest neighbors are almost random. According to the morphological knowledge (see the column of Combined Knowledge), this word may have a relation with *informative* and *formative*. By leveraging these relatively frequent words to enhance the embedding for *uninformative*, our model eventually generated a very effective embedding for this rare word, and its similar words in the learned embedding space became much more reasonable.

Table III. Top Five Similar Words in the Embedding Spaces Produced by KNET Using the Combination of Morphological Knowledge

Example Word	Skip-gram	Combined Knowledge	Skip-gram + Combination Relation Matrix
uninformative	monotherapy	informative	problematic
	lcg	inchoative	fallacious
	electrodeposition	inoperative	inaccurate
	astrophotography	interrogative	uninteresting
	ultrafilters	formative	precisely
stepdaughter	grandaughter	daughters	grandaughter
	swynford	daughter	daughter
	caesaris	grandaughter	daughters
	theling	steptoe	wife
	stepson	slaughter	stepfather
uncompetitive	overvalued	competitively	competitive
	monopsony	competitive	noncompetitive
	skyrocketing	noncompetitive	profitable
	dampened	competitiveness	competetive
	undervalued	competetive	lucrative
tasteful	hackneyed	wasteful	tastes
	freshest	distasteful	piquant
	haircuts	tasted	pretentious
	nutritive	distaste	taste
	teapots	tastes	elegance
weirdest	swordfight	weird	weird
	merseybeat	weirdos	fun
	sty	widest	nostalghia
	oversoul	wildest	weirdos
	washroom	nordeste	skinflint

(2) We can also see that the morphological knowledge could be noisy in some cases. For example, it suggests *inchoative* and *interrogative* to *uninformative*, because these words share a substring *ative* with *uninformative*. However, they are neither syntactically similar nor semantically similar. The power of our proposed framework lies in that it can distinguish useful knowledge and noise by seeking help from the contextual information and refine the tradeoff coefficients and the relationship matrix to ensure the generation of a more reliable embedding. We can see that the most similar words to *uninformative* in the final embedding space, such as *problematic* and *inaccurate,* are more semantically correlated to *uninformative* than *inchoative* and *interrogative*.

To sum up, the examples in Table III indicate that for rare words, (1) it is unreliable to learn their embeddings only from contexts, (2) morphological knowledge can significantly improve the learned word representations if we can effectively deal with the noise it brings in, and (3) contextual information can help in distinguishing useful knowledge and noise. In this manner, our proposed KNET framework can achieve the best performance, while the contextual information and morphological knowledge reinforce each other.

4.6. Analysis of the Balancing Function

In this subsection, we give some empirical results on the influence of the balancing function between the contextual information branch and the morphological knowledge branch in the KNET framework. We expect that the greater the absolute value of

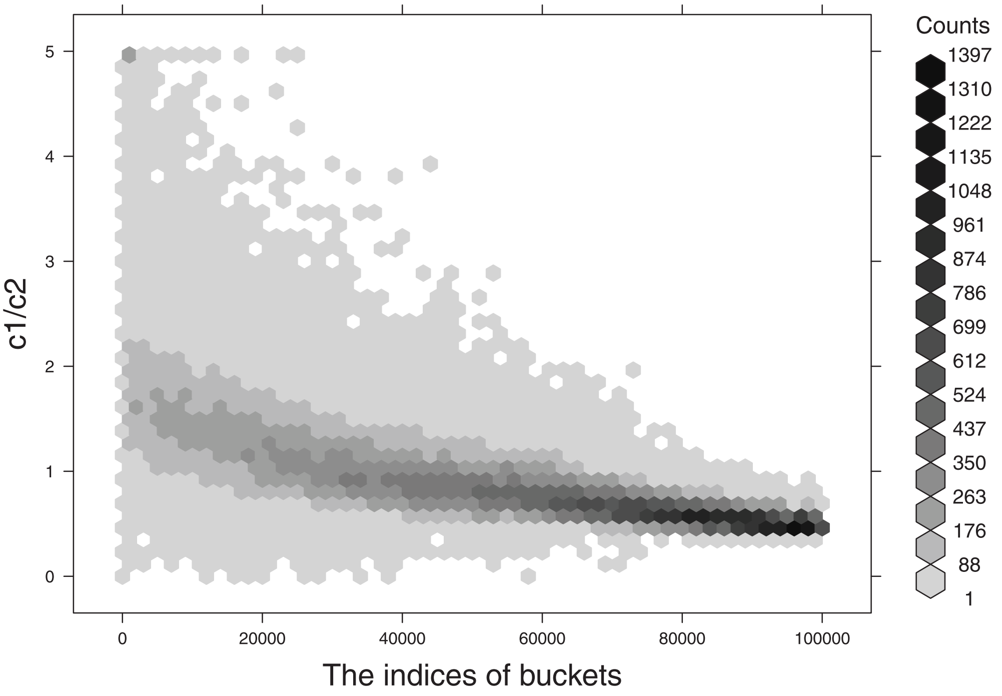

Fig. 3. The ratio of c_1/c_2 in different buckets.

the ratio of the tradeoff coefficients (i.e., c_1/c_2) is, the more the model relies on the contextual information branch. By analyzing the variation of this ratio under different settings, we can draw the following two conclusions:

(1) For a specific model, frequent words rely more on contextual information, while rare words rely more on morphological knowledge.
(2) By comparing the overall weighted ratios of different models under different settings, we can observe that (a) models relying more on contextual information perform better than those relying more on morphological knowledge on tasks mainly composed of frequent words, and (b) models relying more on morphological knowledge perform better than those relying more on contextual information on tasks containing many rare words.

We give more detailed discussions about the two conclusions next.

4.6.1. Rare Words Rely More on Morphological Knowledge. We use the results of Skip-gram + Combination Relation Matrix to illustrate the first conclusion. Note that we observed a similar phenomenon for other models. As we want to carefully analyze the behavior of words with different frequencies, we set the number of buckets b to 100,000 so that each bucket may only contain two words on average. The hexagon binning plot of ratios c_1/c_2 in different buckets is shown in Figure 3, in which the indices of the buckets are in the descending order according to the frequency; that is, the first bucket contains the most frequent words and the last bucket contains the rarest words. We took the absolute value for each ratio and fixed the ratios greater than 5 to be 5 so as to make the figure more readable. The grayscale of the hexagon represents the number of points falling in that hexagon; that is, the hexagon is darker when more points fall in it.

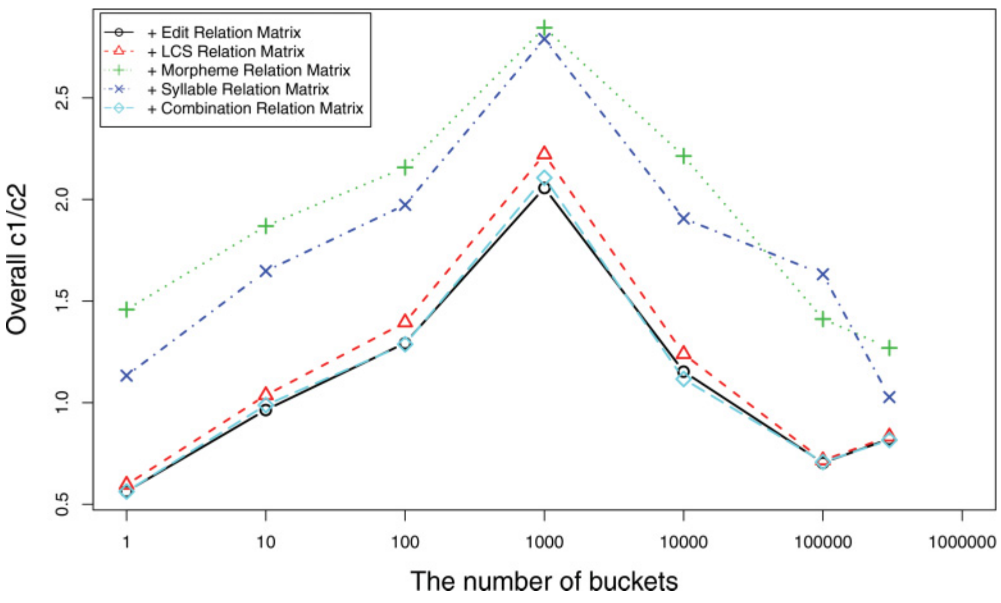

Fig. 4. The overall ratios of c_1/c_2 in different models while the number of buckets varies.

We can see in Figure 3 that the ratio c_1/c_2 is approximately decreasing as the indices of the buckets increase, which indicates that frequent words have relatively higher ratios of c_1/c_2 than rare words. In other words, frequent words rely more on contextual information and rare words rely more on morphological knowledge. By learning the tradeoff coefficients, our model learns how to leverage morphological knowledge according to word frequency automatically and effectively.

4.6.2. Models Relying More on Morphological Knowledge Perform Better on Rare Words. We compare all the proposed knowledge-powered word embedding models under different settings to illustrate the second conclusion. Specifically, we evaluated Skip-gram + Edit/LCS/Morpheme/Syllable/Combination Relation Matrix that were trained with different numbers of buckets on WordSim-353 and RareWord. As WordSim-353 mainly contains frequent words and RareWord contains many rare words, we can analyze the results on these two datasets to estimate the performance on frequent words and rare words. The overall ratio of each model is computed as the weighted sum of c_1 over all buckets divided by the weighted sum of c_2 over all buckets as follows:

$$c_1/c_2 = \frac{\sum_{i=1}^{b} c_{1i} n_i}{\sum_{i=1}^{b} c_{2i} n_i},$$

where n_i is the number of words in the ith bucket.

The overall ratio of c_1/c_2 of different models while the number of buckets varies is shown in Figure 4.[7] We can see that the overall ratio is the highest when the number of buckets lies in the middle, while it drops down as the number of buckets moves to the two extremes. Besides, we can observe that the ratios of Skip-gram + Morpheme/Syllable Relation Matrix are consistently higher than those of Skip-gram + Edit/LCS/Combination Relation Matrix. Similar to Section 4.6.1, the model with the

[7]We only run experiments when the number of buckets is 1, 10, 100, 1,000, 10,000, 100,000, or 300,000 (the maximum value) and connect the points in the figure.

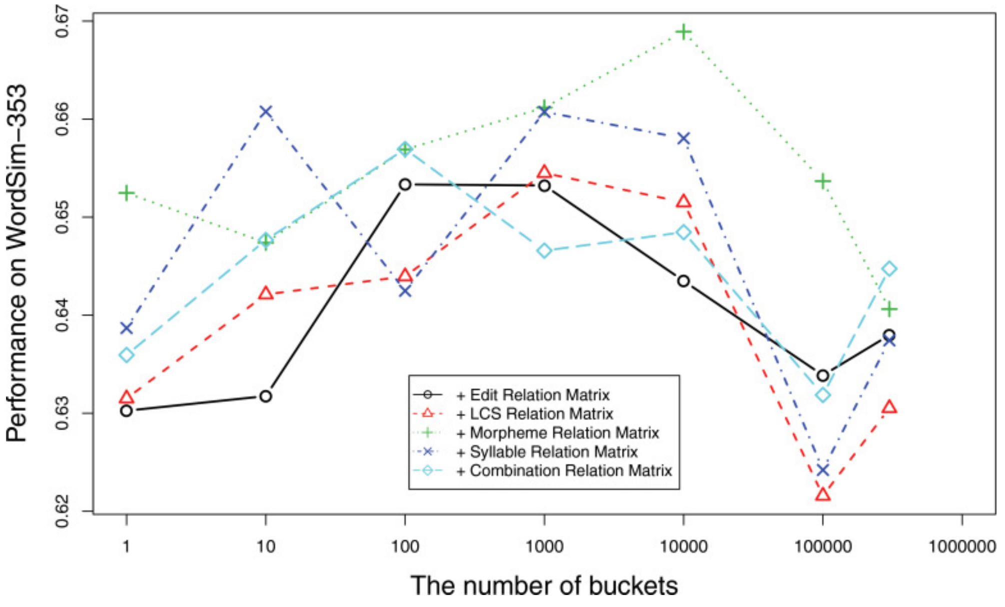

Fig. 5. The performance of different models on WordSim-353 while the number of buckets varies.

lower ratio indicates that the model relies more on morphological knowledge. Therefore, we can conclude that the models in the extremes with one or 300,000 buckets rely more on morphological knowledge, and Skip-gram + Edit/LCS/Combination Relation Matrix rely more on morphological knowledge. In Figure 6, we want to show that these models perform better on datasets with many rare words, that is, RareWord, and thus, the points of Skip-gram + Edit/LCS/Combination Relation Matrix should be on the top and the curves should be convex downward. In Figure 5, we want to show the opposite conclusion, that models in the middle with 1,000 buckets rely more on contextual information and perform better on datasets with more frequent words, that is, WordSim-353, and thus, the curves should be roughly convex upward. Here follows the detailed discussion.

The performance of these models on WordSim-353 is shown in Figure 5. The trend is not stable, probably due to the uncertainty brought by the small size of WordSim-353. However, we can roughly draw the conclusion that (1) most models in the middle perform better than those in the extremes, and (2) Skip-gram + Morpheme/Syllable Relation Matrix perform better than other models in most cases. The trend of the model performance on WordSim-353 is consistent with the overall ratio in Figure 4, which implies that models relying more on contextual information perform better on frequent words.

The performance of these models on RareWord is shown in Figure 6. The trend is very clear and we can easily observe that it is strictly opposite to the overall ratio of c_1/c_2 in Figure 4. The models in the two extremes achieve the best performance and Skip-gram + Edit/LCS/Combination Relation Matrix perform better than others. Considering these models have a lower overall ratio of c_1/c_2 and thus rely more on morphological knowledge, we can conclude that models relying more on morphological knowledge perform better on rare words.

Note that the points at which the number of buckets is the maximal (which means there is only one word in each bucket) look a little strange. The reason is that the c_1 and c_2 of rare words have very little opportunity to be updated when the buckets are

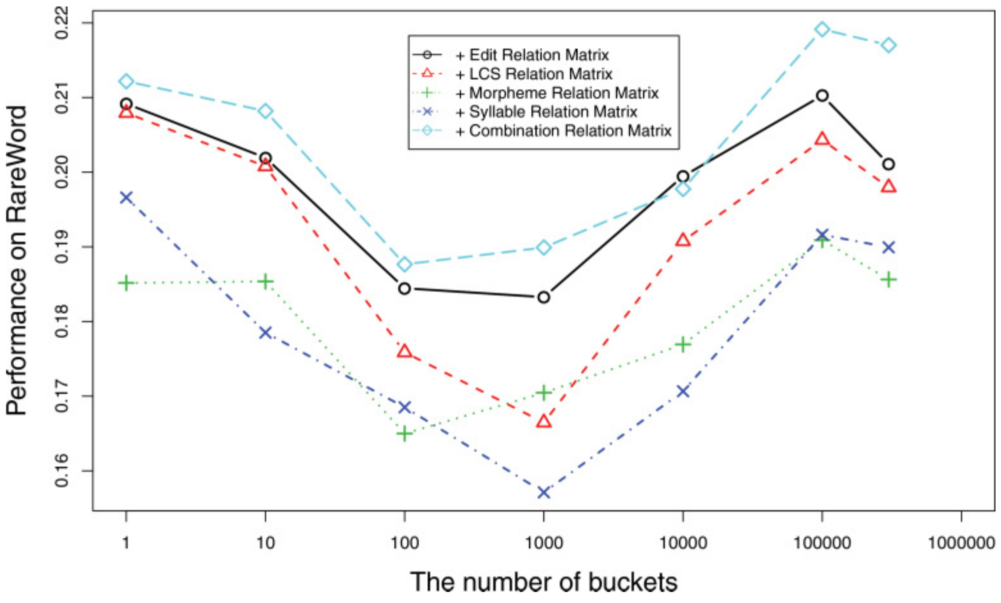

Fig. 6. The performance of different models on RareWord while the number of buckets varies.

very sparse. Further considering that the initializations of c_1 and c_2 are both 0.5, which results in the middle values of the ratio c_1/c_2, we can understand why the corresponding model performance tends to be in the middle.

By analyzing the balancing function, we can also draw an empirical conclusion about how to tune the number of buckets. Setting the number of buckets to the minimum (one bucket) or near the maximum (the size of vocabulary) will help learning better word representations of rare words, while setting the number of buckets to some middle quantity will benefit the frequent words. One can choose the proper range of this parameter according to his or her specific task and conduct a grid search in this range.

5. CONCLUSIONS AND FUTURE WORK

We proposed a novel neural network framework called KNET to leverage morphological word similarity to learn high-quality word embeddings. The framework contains a contextual information branch to leverage word co-occurrence information and a morphological knowledge branch to leverage the morphological relationship between words. We tested the framework on several tasks, and the results show that it can produce enhanced word representations compared with the state-of-the-art models.

The KNET framework is robust because it can refine the noisy knowledge and balance between contextual information and morphological knowledge. In real applications, the robustness of our framework allows us to leverage more general knowledge.

The KNET framework is also very flexible. One can easily change the word embedding model and use another type of knowledge or use another balance function, because the KNET framework provides the two-branch structure, the relation layer with update procedure, and the balancing principle accordingly.

The KNET framework is particularity beneficial for rare words in the application, and thus it is suitable for other morphologically complex languages such as Finnish or

Turkish, especially when the amount of text data is limited but the vocabularies are huge. We will leave it as our future work.

APPENDIX

A. THE BACK-PROPAGATION ALGORITHM AND THE LEARNING PROCESS OF KNET

We will introduce the standard back-propagation algorithm in Section A.1 and show how to apply the algorithm to our model in Section A.2.

A.1. The Back-Propagation Algorithm

This algorithm can be found in every textbook related to neural networks. We just give a simple introduction.[8] As shown in Figure 1, the KNET framework has two branches, the contextual information branch and the morphological knowledge branch. For simplicity, we will use the morphological knowledge branch as an example to explain the back-propagation process in Section A.1.

We let n_l denote the number of layers in the neural network; thus, we have $n_l = 5$ in the morphological knowledge branch. We label layer l as L_l, so layer L_1 is the input layer, and layer L_{n_l} is the output layer. Our morphological branch has parameters $(W, b) = (W^{(1)}, W^{(2)}, W^{(3)}, W^{(4)})$, where we write $W_{ij}^{(l)}$ to denote the parameter (or weight) associated with the connection between unit j in layer l and unit i in layer $l+1$. (Note the order of the indices.) In our morphological branch, $W^{(1)} = R^T$, $W^{(2)} = M^T$, $W^{(3)} = \text{diag}(c_2)$, $W^{(4)} = M'^T$, where M^T is the transpose of M and $\text{diag}(c_2)$ is the diagonal matrix with the diagonal elements all equal to c_2. We use $a_i^{(l)}$ to denote the activation (meaning output value) of unit i in layer l. We also let $z_i^{(l)}$ denote the total weighted sum of inputs to unit i in layer l, so that $a_i^{(l)} = f(z_i^{(l)})$. In our models, we do not use the activation function (except the output layer) and bias term, so the forward propagation is simply $a^{(l+1)} = W^{(l)} a^{(l)}$.

We have introduced the forward propagation in Section 3.2, such as Equations (5) and (6). Our output layer outputs the conditional probability $p(w_{t+j}|w_t)$. By using NEG, we actually approximate $p(w_{t+j}|w_t)$ with $\sigma({v'_{w_{t+j}}}^T v_{w_I})$. We define the cost function as Equation (3), denoted as $J(W)$ here. In the experiments, we initialized the weights in R with the similarity scores, $c_1 = c_2 = 0.5$, and all the other weights with small random values.

Next, we will describe how back-propagation can be used to compute $\frac{\partial}{\partial W_{ij}^{(l)}} J(W; x, y)$ and the partial derivatives of the cost function $J(W, b; x, y)$ defined with respect to a single example (x, y).

The intuition behind the back-propagation algorithm is as follows. Given a training example (x, y), we will first run a forward pass to compute all the activations throughout the network, including the output value of the conditional probability. Then, for each node i in layer l, we would like to compute an error term $\delta_i^{(l)}$ that measures how much that node was responsible for any errors in our output. In detail, here is the back-propagation algorithm:

(1) Perform a feed-forward pass, computing the activations for layers L_2, L_3, and so on up to the output layer L_{n_l}.

[8]http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial.

(2) For each output unit i in layer n_l (the output layer), set

$$\begin{aligned}\delta_i^{(n_l)} &= \frac{\partial}{\partial z_i^{(n_l)}}\left[y_i \log\sigma\left({v'_{w_i}}^T v_{w_I}\right) + (1-y_i)\log\sigma\left(-{v'_{w_i}}^T v_{w_I}\right)\right]\\ &= y_i\left(1-\sigma\left({v'_{w_i}}^T v_{w_I}\right)\right) - (1-y_i)\sigma\left({v'_{w_i}}^T v_{w_I}\right),\end{aligned}$$

where $y_i = 1$ when it is the positive sample; that is, w_i is the real word in the context, and $y_i = 0$ when it is the negative sample.

(3) For $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$, for each node i in layer l, set

$$\delta_i^{(l)} = \sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)}.$$

(4) Compute the desired partial derivatives, which is given as

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W; x, y) = a_j^{(l)}\delta_i^{(l+1)}.$$

With the gradient, we can apply gradient descent to update the parameters W as follows:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}} J(W).$$

Here α is the learning rate. We set it to 0.025 in all our experiments.

A.2. The Learning Process of the KNET Framework

With the help of the standard back-propagation algorithm, our learning process is quite natural except for these three aspects:

(1) The KNET framework has two branches. In the architecture, they share the input layer, the v_{w_I} layer, and the output layer. Thus, in the back-propagation process, we only need to update M' once. For the other layers, we regard the architecture as two separate branches and update the parameters separately.
(2) For the shared parameters matrix M in two branches, we simply store just one copy and update it twice, that is, one round in the contextual information branch and the other round in the morphological knowledge branch.
(3) For the shared parameters c_1, c_2 in the diagonal matrix, we actually store just one copy and sum the gradient of all edges to update it. You can also regard the procedure as updating it several times. Since we have different c_1, c_2 for different words, we will use the proper parameter for each input word.

Besides these three aspects, the learning process is the same with the standard back-propagation algorithm introduced in Section A.1.

REFERENCES

Y. Bengio and J.-S. Senecal, and others. 2003. Quick Training of Probabilistic Neural Nets by Importance Sampling.

Y. Bengio and J.-S. Senecal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Trans. Neur. Netw.* 19, 4, 713–722.

J. Bian, B. Gao, and T.-Y. Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proc. of ECML/PKDD*.

D. M. Blei, A. Y. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research* 3, 993–1022.

A. Bordes, J. Weston, R. Collobert, Y. Bengio, and others. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

J. W. Chapman. 1998. Language prediction skill, phonological recoding ability, and beginning reading. *Reading and Spelling: Development and Disorders*, 33.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*. ACM, New York, NY, 160–167.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR* 12, 2493–2537.

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

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

L. C. Ehri. 2005. Learning to read words: Theory, findings, and issues. *Scientific Studies of Reading* 9, 2, 167–188.

L. C. Ehri, R. Barr, M. L. Kamil, P. Mosenthal, and P. D. Pearson. 1991. Development of the ability to read words. *Handbook of Reading Research* 2, 383–417.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web.* ACM, 406–414.

X. Glorot, A. Bordes, and Y. Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 513–520.

U. Goswami. 1986. Children's use of analogy in learning to read: A developmental study. *Journal of Experimental Child Psychology.* 42, 1, 73–83.

M. U. Gutmann and A. Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.* 13, 307–361.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Distributed representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 3:1137–1155.

T. Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 289–296.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1.* Association for Computational Linguistics, 873–882.

F. M. Liang. 1983. *Word Hy-phen-a-tion by Com-put-er (Hyphenation, Computer).* Stanford University, Stanford, CA, USA.

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

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

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*. 3111–3119.

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

A. Mnih and K. Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*. 2265–2273.

A Mnih and Y. W. Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*. Omnipress, New York, NY, 1751–1758.

F. Morin and Y. Bengio. 2005. Hierarchical probabilistic neural network language model. In *AISTATS*. 246–252.

A. El-Desoky Mousa, H.-K. J. Kuo, L. Mangu, and H. Soltau. 2013. Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8435–8439.

S. Qiu, Q. Cui, J. Bian, B. Gao, and T.-Y. Liu. 2014. Co-learning of word representations and morpheme representations. In *Proc. of COLING*.

R. Socher, D. Chen, C. D. Manning, and A. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.

R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 129–136.

H. Sperr, J. Niehues, and A. 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*. 30–39.

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

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

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

J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*.

M. Yu and M. Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Association for Computational Linguistics (ACL)*. 545–550.

Received May 2014; revised June 2015; accepted June 2015