Knowledge-Enhanced Ensemble Learning for Word Embeddings

Lanting Fang
Southeast University
NKG, CN
lantingf@outlook.com

Yong Luo
Nanyang Technological University
SG
yluo180@gmail.com

Kaiyu Feng
Nanyang Technological University
SG
kaiyufeng@gmail.com

Kaiqi Zhao
Nanyang Technological University
SG
kzhao002@e.ntu.edu.sg

Aiqun Hu
Southeast University
NKG, CN
aqhu@seu.edu.cn

ABSTRACT

Representing words as embeddings in a continuous vector space has been proven to be successful in improving the performance in many natural language processing (NLP) tasks. Beyond the traditional methods that learn the embeddings from large text corpora, ensemble methods have been proposed to leverage the merits from pre-trained word embeddings as well as external semantic sources. In this paper, we propose a knowledge-enhanced ensemble method to combine both knowledge graphs and pre-trained word embedding models. Specifically, we interpret relations in knowledge graphs as linear translation from one word to another. We also propose a novel weighting scheme to further distinguish edges in the knowledge graph with same type of relation. Extensive experiments demonstrate that our proposed method is up to 20% times better than state-of-the-art in word analogy task and up to 16% times better than state-of-the-art in word similarity task.

KEYWORDS

Word embedding; Knowledge graph; Ensemble model

ACM Reference Format:
Lanting Fang, Yong Luo, Kaiyu Feng, Kaiqi Zhao, and Aiqun Hu. 2019. Knowledge-Enhanced Ensemble Learning for Word Embeddings. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313425

1 INTRODUCTION

Word embeddings are low dimensional distributed representations of words in a real-valued vector space. As they are both compact and informative, word embeddings have proven to be effective in improving the performance in many natural language processing (NLP) tasks, e.g., named entity recognition (NER) [32], POS tagging [26] and information retrieval [30]. Most of the existing methods for learning word embeddings are trained on large unlabeled text corpora and many pre-trained word embeddings are available online. Mikolov et al. [18] learn the continuous bag-of-words (CBOW) and Skip-Gram embeddings from Google News corpus. The training objective of CBOW is to combine the embeddings of context words to predict the target word, while Skip-Gram is to use the embedding of each target word to predict its context words. Pennington et al. [24] learn the GloVe embedding set from the common crawled web data. The training objective of GloVe is to let the dot product of the embeddings of two words approximately equal to the logarithm of the count of their co-occurrences.

As word embeddings learned by different methods often capture different statistics from different corpora and these corpora are usually not available publicly, a natural idea for gathering the merits from multiple pre-trained word embeddings is to learn an ensemble model taking the statistics from them. Muromagi et al. [22] propose to combine several word embedding models into a unified one. They prove that an ensemble model can not only cancels out random noise of individual models but also reinforces co-occurrence patterns in the input models. However, the above models are totally context-based and ignore the common sense relations between two words within the embeddings. As a result, these models are incapable of capturing semantic relatedness of words if they do not appear in similar context.

Several studies [3, 7, 28, 34, 36] have been proposed to incorporate external semantic sources. Knowledge graphs, such as WordNet [19] and ConceptNet [29], are considered as one of the most important semantic sources for improving word embeddings. A knowledge graph is a directed graph and can be represented by a set of (entity, *relation*, entity) tuples. Faruqui et al. [7] propose a "retrofitting" method to leverage the synonym, hypernym, hyponym and paraphrase relations in WordNet to learn the ensemble embeddings. Robert et al. [28] propose a more general method "ConceptNet-Numberbatch (CN)" to support other types of relations in ConceptNet. However, these methods underlook the importance of the relations and consider two words are related if they are connected by any relation in the knowledge graph. For instance, relations *EatenBy* and *Hit* in tuples (apple, *EatenBy*, man) and (apple, *Hit*, man) are viewed as equivalent.

In this paper, we propose a knowledge-enhanced ensemble learning model called *RA-Retrofit* for word embeddings. The proposed model is able to leverage all semantic information provided by knowledge graphs as well as the pre-trained word embedding models. Firstly, to use the semantic information provided by knowledge graph, inspired by the efforts on multi-relational data embedding

Yong Luo is also with the Peking University Shenzhen Graduate School, SZX, CN.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313425

[1], we learn embeddings for both nodes (each node represents a word) and edges (each edge represents the relation between two words) in the knowledge graph. The embedding of an edge is interpreted as linear translation from the source word to the target word. In the previous example, the embedding of "man" can be calculated by simply adding the embeddings of word "apple" and relation *EatenBy*. Different from [1], we propose a novel scheme to assign weights to the edges in the loss function. We compute the weight for each edge based on the relevance between the two words connected by the edge: a larger weight for two close related words. For example, we give higher weights for (Internet, *RelatedTo*, Computer) than for (Multipart, *RelatedTo*, Computer) because "Computer" is more related to "Internet" than "Multipart". Secondly, to preserve the information from both the input models and knowledge graphs in the ensemble word embeddings, we propose a joint learning method that minimizes (1) the training loss of the knowledge graph embeddings at the same time. Additionally, we show that *RA-Retrofit* is more general than several existing embedding methods through a theoretical analysis and (2) the difference between the pre-trained word embeddings and the linear projection of the ensemble word embeddings.

In summary, this paper has the following contributions.

- We propose a knowledge-enhanced method to incorporate common sense knowledge into pre-trained embeddings. We learn embeddings for both words and their relations in the knowledge graph and represent the relations as linear translation between the embedding vectors of words. Additionally, we propose a novel weighting scheme for different edges in knowledge graphs by considering the relatedness of the two words of the edge in terms of neighborhood similarity.
- We propose an effective joint learning algorithm that minimizes two losses simultaneously - (1) the loss of acquiring a word's representation vector via relation translation and (2) the loss of recovering the pre-trained word embeddings through linear projection of the ensemble embedding.
- We establish a thorough theoretical analysis to show that several embedding models are restricted version of the RA-Retrofit model proposed in this paper.
- We conduct extensive experiments to show the superiority of the proposed ensemble learning method. Our method significantly outperforms individual pre-training word embeddings (word2vec [18] and Glove [24]) and the state-of-the-art ensemble methods (CN [28] and Meta-Embedding [35]) in both word analogy and word similarity tasks.

2 RELATED WORKS

Our work is related to *word embedding*, *ensemble method*, and *knowledge graph embedding*. We relate research in these areas in turn.

2.1 Word Embedding

Learning word embeddings has been an active research problem for the past couple of years. A host of approaches has been proposed and they have been shown to be successful in improving and simplifying many natural language applications. Most of these word embeddings are trained on large unlabeled text corpora. Continuous bag-of-words (CBOW) and Skip-Gram [18] are two typical embedding methods that either use the context words to predict the target word or vice-versa. GloVe [24] is another unsupervised learning algorithm for learning word embeddings based on context words. It is trained with global word co-occurrence counts and learns a set of word embeddings such that the dot product of the embeddings of two words is approximately equal to the logarithm of their co-occurrence count. The above word embedding methods learn two vectors for each word – a representation vector for the word acting as a target word and a context vector for the word acting as a context word. Usually, only the representation vectors are used for computing similarity.

Different from our proposed approach, the above word embedding models learn the representation only from text corpora. In our proposed approach, we incorporate knowledge graph into our ensemble model. We use the relations in knowledge graph to refine the word embeddings.

2.2 Ensemble Method

Motivated by the fact that different word embedding are trained on different corpora and these corpora are often not publicly available, a group of ensemble methods has been proposed to combine these pre-trained word embeddings. They can be categorized into two types.

The first type of ensemble methods [11, 15, 31, 32, 35] aims to combine multiple embeddings to generate an ensemble embedding with better quality. For instance, Tsuboi et al. [31] incorporate word2vec and GloVe embeddings into a POS tagging system and show that the ensemble model outperforms both individual models. Turian et al. [32] show that the ensemble model that combines CW embeddings [5] and HLBL embeddings [21] outperforms the individual models in natural language applications like named entity recognition and chunking. Luo et al. [15] train the CBOW model with three datasets and propose an ensemble model to combine the word embeddings learned from the three datasets. They show that the ensemble model achieves better performance than using a single word embedding. Hill et al. [11] propose an ensemble method based on machine translation. Yin et al. [35] propose an ensemble approach of combining different public embedding sets to learn meta-embeddings. They show that the meta-embeddings achieve better performance in word similarity and word analogy tasks compared to individual models. These approaches do not incorporate external semantic resources, and thus they are different from our proposed model.

The second type of ensemble methods incorporates external semantic resources to improve word embeddings [7, 28, 36]. One of the most famous semantic resources is knowledge graph, including, but not limited to PPDB [9], WordNet [19] and ConceptNet [28]. The second type of ensemble methods can be further divided into two categories. In the first category, the ensemble methods use semantic resources as a feature when learning the word embeddings [3, 34, 36]. Yu et al. [36] propose a new training objective for learning word embeddings that incorporates prior knowledge. The model extends the objective of word2vec [18] to take prior knowledge about synonyms from semantic resources into account. In the second category, the ensemble methods retrofit the semantic resources into existing word embeddings [7, 28]. Faruqui et al.

[7] introduce a technique known as "retrofitting", which combines embeddings learned from the existing corpus with a source of structured connections between words. Robert et al. [28] modify the retrofitting algorithm, making it not depending on the row order of its input matrix, and allowing it to propagate over the union of the vocabularies. These approaches differ from our proposed model in the aspect that they disregard the types of the relations in the knowledge graph. In contrast, the proposed model interprets relation as linear translation from a word to the other.

2.3 Knowledge graph embedding

The proposed model is also related to knowledge graph embedding. Various types of representation learning methods have been proposed to embed entities and relationships of knowledge graphs into low dimensional vector spaces. Given the representation vectors of entities and relations, these methods define a scoring function on each fact to measure its plausibility. For example, given two entity vectors, the model of Neural Tensor Network [27] represents each relation as a bilinear tensor operator followed by a linear matrix operator. The model of TransE [1], on the other hand, represents each relation as a single vector that linearly interacts with the entity vectors. Likewise, variations on entity representations also exist. RESCAL [23] models triples through bilinear operations over entity and relation representations.

Recently, more and more researchers have started to represent a graph as a document. These approaches sample sequences of nodes from the underlying graph and turn a graph into an ordered sequence of nodes. For example, DeepWalk [25] shows that the distribution of nodes appearing in short random walks is similar to the distribution of words in natural language and employes Skip-Gram to learn the representations of nodes. Node2vec [10] defines a flexible notion of a node's network neighborhood and design a biased random walk procedure. KGloVe [4] presents an approach in that exploits global patterns for creating vector space embeddings, which is inspired by the GloVe model.

Since the adjacency matrix of a knowledge graph is sparse [13], the entity embeddings learned from a knowledge graph cannot provide enough information. In the proposed method, we view the pre-trained word embeddings as distinct descriptions of words and combine it with the entity embeddings.

3 METHOD

In this section, we present an ensemble model that combines pre-trained word embedding models and knowledge graphs. Important notations are summarized in Table 1.

3.1 Problem Formulation

In this subsection we formulate the problem to be investigated. We start with some basic definitions.

Definition 3.1. (**Word Embedding Matrix**): Let V be a vocabulary of words. A word embedding matrix $W \in \mathbb{R}^{|V| \times d}$ contains d-dimensional continuous representation for each word in V. Each row w_v in W is the representation vector for word $v \in V$.

Definition 3.2. (**Knowledge Graph**): A knowledge graph $G(V, E)$ is a directed graph, where V is the set of nodes, or equivalently the vocabulary of words, and E is the set of edges. Each edge in E connects a source word $v \in V$ with target word $u \in V$ via a relation type r. Each edge in E is denoted as a tuple $e = (v, r, u)$, where $v, u \in V$, and $r \in R$ is a type of relation.

Table 1: Notations

Notation	Description
$w_v^i \in \mathbb{R}^{d_i}$	pre-trained word embedding for word v in model i
V	vocabulary
R	relation set
d_i	dimension of the i-th pre-trained word embeddings
$G(V, E)$	knowledge graph G with entity set V and edge set E
(v, r, u)	an edge in E specifying relation r between two words $v, u \in V$
$q_r \in \mathbb{R}^d$	d-dimensional embedding for relation r
$y_v \in \mathbb{R}^d$	the target d-dimensional ensemble word embedding for word v

Consider t pre-trained word embedding matrices $W^1, \ldots, W^t$ and a knowledge graph G on a global vocabulary V, we aim to train (1) an ensemble word embedding matrix Y, where each row $y_v \in \mathbb{R}^d$ is the d-dimensional representation vector for $v \in V$, and (2) the representation vector $q_r \in \mathbb{R}^d$ for each relation $r \in R$.

3.2 Overview

We propose an ensemble model to combine the linguistic information in a set of t pre-trained word embedding models and knowledge graphs. The learning objective of our ensemble model is twofold: (1) To leverage the knowledge graph, the representation vector y_v for word v should be close to the addition of the representation vector of its related word and the relation that connects the two words. (2) To leverage the t word embedding models, the representation vector y_v for word v in the target model should be close to its representation vector w_v^i in the pre-trained word embedding models via linear projection.

To achieve the learning objectives, we first introduce learning from knowledge graph (3.3) and learning from word embedding models (3.4) separately. Then we introduce the weighted global objective function for our ensemble model (3.5), and the corresponding weighting scheme by considering the relevance strength of each edge in the knowledge graph (3.6). To minimize the objective function, we also propose the positive and negative sampling methods (3.7). We elaborate on these issues in turn in the subsequent subsections.

3.3 Learning from Knowledge Graphs

The edges in the knowledge graph reveal a lot of syntactic and semantic relations between the words. A host of work has been proposed to leverage the information in the knowledge graph to improve the word embedding models. Faruqui et al. [7] introduce the "retrofitting" procedure, which refines dense matrices of embeddings to take into account external knowledge from a sparse semantic network. Robert et al. [28] improve the retrofitting algorithm by making it not depending on the row order of its input

matrix and allowing it to propagate over the union of the vocabularies. Both methods define the dissimilarity between two words in the Euclidean space, i.e., dissimilarity between two words is the Euclidean distance between the representation vectors of the two words in the edge. Specifically, given an edge (u, r, v) from the knowledge graph, the dissimilarity between u and v is defined as

$$\phi(u, v) = \| y_u - y_v \|, \tag{1}$$

where y_u and y_v are the learned representation vectors for u and v, respectively.

By using the Euclidean distance, it can bring the embeddings of two similar words closer to each other. However, this dissimilarity measure disregards the types of the relation in the edge. For instance, (obscurity, *RelatedTo*, darkness) and (sleeping, *HasPrerequisite*, darkness) are two edges in the knowledge graph. Though "obscurity" and "sleeping" are both connected to "darkness", they are connected by different types of edges. Apparently, it is not reasonable to encourage the embedding vectors of "obscurity" and "sleeping" to be close to each other.

How can we leverage the type of the relation while we are learning from knowledge graph? To answer the question, we first think about *discovering analogical relations*, which is one of the most popular benchmarks for word embeddings. This task, which is first proposed in [18], shows that the proportional analogies (A is to B as C is to D) can be solved by finding the vector closest to the hypothetical vector calculated as $w_C - w_A + w_B$, e.g. vec("king") - vec("man") + vec("woman") $\approxeq$ vec("queen"). In knowledge graph, proportional analogy (A is to B as C is to D) is relationally equal to edges (A, r, B) and (C, r, D) having the same type of relation. Hence, it is a natural idea to interpret an edge as linear translation operating on the embeddings of the two words. Specifically, given an edge (u, r, v), we represent the relation r with a vector based on its type. The embedding vector of v should be close to the addition of the embeddings of u and r. Therefore, we define the following function to evaluate the dissimilarity for an edge:

$$\phi(u, r, v) = \| y_u + q_r - y_v \|, \tag{2}$$

where y_u and y_v are the learned representation vectors for u and v, respectively, and q_r is the embeddings for relation r. Given two edges (A, r, B) and (C, r, D) from the knowledge graph, Equation 2 encourages the representation vector $y_v = w_D$ (w_B) to be close to the addition of $y_u = w_C$ and q_r (w_A and q_r). Hence, $w_B - w_A + w_C$ will be close to w_D.

In fact, the idea of interpreting edges as translation operating on vectors of nodes has been investigated by efforts on multi-relational data embedding [1], where a model called TransE is proposed to embed entities and relationships of multi-relational data into vectors. Since the TransE model is designed for multi-relational data embedding, we also combine it with other word embedding models to make it suitable for word embedding problem, denoted by TransE+. We will show in Section 5 that the ensemble model proposed in this paper outperforms both TransE and TransE+ in all tasks.

Let (u, r, v) be a positive edge, i.e., a real edge from E. Let (u', r, v') be a negative edge, i.e., (u', r, v') does not exists in E. In order to learn from knowledge graph, we minimize a weighted margin-based ranking criterion over the edges in the knowledge graph:

$$\mathcal{L}_1(u, u', v, v', r) = \mathcal{W}(u, r, v) \max\{m + \phi(u, r, v) - \phi(u', r, v'), 0\}, \tag{3}$$

where m is the margin, $\phi(\cdot)$ is the score function defined in Equation 2, and $\mathcal{W}(u, r, b)$ is the weight of an edge (to be discussed in section 3.6). The intuition is that the dissimilarity of positive edges should be smaller than the negative ones.

3.4 Learning from Word Embedding Models

We consider t word embedding models that are trained with different algorithms and corpora. They can be viewed as distinct descriptions of words. In this section, we first present how to pre-process these word embedding models. Then we present how to learn from these word embedding models.

3.4.1 Aligning Vocabularies. Different word embeddings are trained with different corpora. Moreover, these models apply different pre-processing steps to produce their own vocabularies. For instance, word2vec [18] joins multi-word phrases with underscores while GloVe [24] does not. These pre-processing steps produce different vocabularies for each model. We can combine these models only if their vocabularies are aligned. We next present a unified pre-processing procedure that combines all the pre-processing steps that the word embedding models apply.

The unified pre-processing includes five steps: tokenizing text to separate words from punctuation, joining multi-word phrases with underscores, removing a small list of stopwords from multi-word phrases, folding the text to lowercase, and replacing multiple digits with the character '#'.

This pre-processing procedure is a many-to-one procedure. For instance, "When", "WHEN", and "when" are all in the vocabulary of word2vec. Each of the three terms corresponds a pre-trained representation vector. After the unified pre-processing procedure, they are all transformed to "when" in the new vocabulary. In this case, the term "when" in the new vocabulary is associated with three pre-trained representation vectors. As a result, we need to merge the associated pre-trained vectors for the transformed terms. In this paper, we follow the strategy proposed in a previous work [28] to merge the associated representation vectors by computing a weighted average based on terms' frequencies. Specifically, for a pre-trained word embedding model W, let $v_1, \ldots, v_k$ be the set of terms that will be transformed to a term v after the unified pre-processing step. The merged representation vector for term v is defined as

$$w'_v = \sum_{j=1}^{k} freq(v_j) w_{v_j} \tag{4}$$

where w'_v is the merged representation vector for v, and w_{v_j} is the representation vector of v_i in the pre-trained embedding matrix, and $freq(v_j)$ is the frequency of v_j

In the remaining of the paper, we assume the vocabularies of all pre-trained word embedding models have been aligned. With a slight abuse of notation, we still use w^i_v to denote the merged representation vector of term v in model W^i.

3.4.2 Learning from Pre-trained Word Embedding Models. Given a word $v \in V$ and t word embedding models $W^1, \ldots, W^t$, we use a

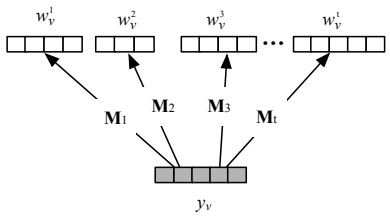


Figure 1: Network for Combining Pre-trained Embeddings

simple neural network to learn the ensemble embeddings, as shown in Figure 1. Let Y be the ensemble embedding model. We define a projection from space Y to space W^i, $i \in [1, t]$ as follows:

$$\hat{w}_v^i = \mathbf{M}_i y_v, \tag{5}$$

where $y_v \in \mathbb{R}^d$ is the embedding of word $v \in V$, $\hat{w}_v^i \in \mathbb{R}^{d_i}$ is the predicted embedding vector of v in W^i and d_i is the dimension of the representation vectors in model W^i, $\mathbf{M}_i \in \mathbb{R}^{d_i \times d}$ is the transformation matrix.

The predicted embedding vector $\hat{w}_v^i$ of word v should be close to the embedding vector w_v^i in model W^i. Thus, we define the objective function for a word v as:

$$\begin{aligned} \mathcal{L}_2(v) &= \sum_{i=1}^{t} \parallel \hat{w}_v^i - w_v^i \parallel \\ &= \sum_{i=1}^{t} \parallel \mathbf{M}_i y_v - w_v^i \parallel, \end{aligned} \tag{6}$$

The intuition behind Equation 6 is that we treat each individual embedding vector as a projection of the ensemble embedding vector. An individual embedding is a description of the word based on the corpus and the model that were used to create it. The ensemble embedding model tries to recover a more comprehensive description of the word when it is trained to predict the individual descriptions.

3.5 Global Objective Function

Now we proceed to give the global objective function. The global objective function is the linear combination of the two aforementioned objective functions, which is defined as follows:

$$\begin{aligned} \mathcal{L}(u, u', v, v', r) =& \mathcal{L}_1(u, u', v, v', r) + \mu \sum_{s \in \{u, u', v, v'\}} \mathcal{L}_2(s) \\ &+ \eta_1 \parallel \Theta_1 \parallel + \eta_2 \parallel \Theta_2 \parallel, \end{aligned} \tag{7}$$

where μ is a parameter to balance the two aforementioned loss functions, $\Theta_1 = \{ y_u, y_{u'}, y_v, y_{v'}, q_r \}$ is the set of embedding vectors of words u, u', v, v' and relation r, $\Theta_2 = \{\mathbf{M}_1, \ldots, \mathbf{M}_t\}$ is the set of transformation matrices that are defined to project the embedding vector in the ensemble model to the space of each individual embedding model.

In the global objective function, the first term $\mathcal{L}_1(u, u', v, v', r)$ makes the ensemble model to learn the information from knowledge graphs. The second term $\mathcal{L}_2(s)$ allows the ensemble model to capture the information in each individual embedding model. The last two terms are L2 regularization to avoid over-fitting, and $\eta_1 > 0$ and $\eta_2 > 0$ are the regularization coefficients.

Algorithm 1 Learning Ensemble Model

Input: Knowledge graph $G = (V, E)$, pre-trained embedding models $W^1, \ldots, W^t$, learning rate γ, ensemble embedding dimension d, component weight μ, η_1, η_2
Output: Ensemble embedding for each words $i \in V$ and relation $r \in R$; transformation parameters $\mathbf{M}_1, \ldots, \mathbf{M}_t$
Initialize ensemble embedding for each $i \in V$, $r \in R$ and parameters $\mathbf{M}_1, \ldots, \mathbf{M}_t$ with uniform distribution $(-\frac{1}{d}, \frac{1}{d})$
repeat
  for $(u, r, v) \in E$ **do**
    $P_N \leftarrow \emptyset$
    for k steps **do**
      $(u', r, v') \leftarrow$ Sample a negative edge
      $P_N \leftarrow (u', r, v')$
    end for
    Extract embeddings of all words in positive and negative edges from $W^1, \ldots, W^t$
    Compute loss function $\mathcal{L}$
    Update gradient $\Theta \leftarrow \Theta + \gamma \frac{\partial}{\partial \Theta} \mathcal{L}$
    Normalize word embeddings
  end for
until *convergence*

The optimization is carried out by stochastic gradient descent. In each step, we update all parameters $\Theta = \Theta_1 \cup \Theta_2$ by calculating the gradient:

$$\Theta \leftarrow \Theta + \gamma \frac{\partial}{\partial \Theta} \mathcal{L}, \tag{8}$$

where γ is the learning rate and $\frac{\partial}{\partial \Theta}$ indicates the gradient of parameters Θ. Finally, we normalize the embedding vectors for word v in the ensemble model Y:

$$\hat{y}_v = \frac{y_v}{max(\parallel y_v \parallel, 1)}. \tag{9}$$

The detailed optimization procedure of ensemble approach is summarized in Algorithm 1. All embedding vectors in the ensemble model are first randomly initialized. At each main iteration of the algorithm, the embedding vectors of the words are first normalized. For each positive edge in the knowledge graph, we first generate k negative edges. We utilize a list P_N to store all negative edges. Then for each individual word embedding model W^i, we extract the pre-trained embedding vectors for the words in the positive and negative edges. After that, we compute the loss function. The parameters are then updated by taking a gradient step with constant learning rate. The algorithm is stopped based on its performance on a validation set.

We also consider the out-of-vocabulary (OOV) words in the vocabulary union i.e., words that not covered by vocabulary intersection. For the words not appearing in the pre-trained embedding model, we initialize the OOV embeddings randomly and use the same mapping formula as for RA-retrofit to connect a ensemble embedding with the pre-trained embeddings. Both ensemble embedding and initialized OOV embeddings are updated during training. For the words not appearing in the graph model, we use equation 6 to learn the ensemble embeddings.

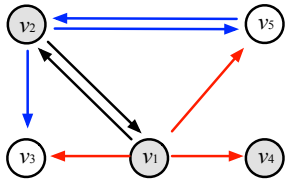


Figure 2: An example of knowledge graph

3.6 Weighting Scheme for an Edge

The relevances between the words of the two edges may be different even if they have same type of relation. For example, although edges (Internet, *RelatedTo*, Computer) and (Multipart, *RelatedTo*, Computer) have the same relation "RelatedTo", the relevance between the words "Internet" and "Computer" should be higher than that between words "Multipart" and "Computer". Intuitively, given an edge (u, r, v) from the knowledge graph, the relevance between words u and v is not only dependent on the type of relation r, but also dependent on the two words themselves. We have shown that we use a representation vector q_r to capture the relation r between two words. We next present how to capture the relevance between the two words in an edge.

Some prior work proposes to use manually defined constraints to distinguish the relevance between the words. For example, Ding et al. [6] propose to impose non-negativity constraints on vector representations, in which only positive samples will be used to learn the representations. Their approach has many limitations. First, it requires expertise to define the most appropriate constraint. Second, it is difficult to compare the relevance between words, especially when the number of words is large. Third, the relevance defined in their approach is highly subjective.

To solve this problem, we propose a method to compute the weight for each edge. Firstly, we define the neighbors of a word v in a knowledge graph as the words that can be connected with v via any relation r. Intuitively, if two words share more common neighbors, they are more likely to be highly relevant. Figure 2 shows an example of the knowledge graph. It can be seen in Figure 2 that v_1 shares more common neighbors with v_2 than with v_4. Specifically, the common neighbors of v_1 and v_2 are v_3 and v_5, while v_1 and v_4 have no common neighbors. Hence, v_1 and v_2 is likely to be more relevant.

Given a word v, let v^N be the set of neighbors of v. We define the weight of an edge (u, r, v) as the Jaccard similarity between the neighbors of u and v:

$$\mathcal{W}(u, r, v) = \epsilon_1 + \frac{|u^N \cap v^N|}{|u^N \cup v^N| + \epsilon_2}, \tag{10}$$

where $\epsilon_2 = 0.001$ is used to prevent division by zero, ϵ_1 is equal to the average value of $|u^N \cap v^N|/(|u^N \cup v^N| + \epsilon_2)$ in all edges such that the weight of an edge is always larger than 0.

3.7 Positive and Negative Sampling

The most frequent words can easily occur hundreds of millions of times. Such words usually provide less information value than the rare words. To counter the imbalance between the rare and frequent words, Mikolov et al. [18] use a simple subsampling approach: Each word v in the training set is discarded with probability computed by the equation:

$$P(v) = 1 - \sqrt{\frac{s}{f(v)}}, \tag{11}$$

where $f(v)$ is the frequency of word v and s is a chosen threshold.

In this paper, we extend the probability equation for edges in the knowledge graph. Given an edge (u, r, v), we define a combine frequency $f(u, v)$ over the corresponding frequency $f(u)$ and $f(v)$. We consider several choices for the combining operator which are summarized in Table 2. We define the subsampling formula for each edge as following:

$$P((u, r, v)) = 1 - \sqrt{\frac{s}{g(u, v)}}, \tag{12}$$

where $g(u, v)$ is the combined frequency of edge (u, r, v) (all combined frequencies are normalized to the interval $[0, 1]$) and s is a chosen threshold (we define $s = 1 \times 10^{-5}$ in this paper).

We next present the negative sampling technique to construct negative edges. Negative sampling assumes that two random words from the vocabulary V should be unrelated with high probability. Given an edge (u, r, v), we generate a set $E'_{(u,r,v)}$ of k negative edges by replacing either u or v by a random word v' from vocabulary V excluding u and v.

$$E'_{(u,r,v)} = \{(u, r, v')|v' \in V \setminus \{u, v\}\} \cup \{(u', r, v)|u' \in V \setminus \{u, v\}\} \tag{13}$$

Table 2: Combining operator $g(u, v)$ for u and v

Id	Operator	Definition
1	Average	$g(u, v) = \frac{f(u)+f(v)}{2}$
2	Max value	$g(u, v) = max(f(u), f(v))$
3	Geometric mean	$g(u, v) = \sqrt{f(u) \times f(v)}$

4 RELATIONSHIP TO OTHER MODELS

In the following we provide a unified view of several embedding models, by showing that they are restricted versions under our framework.

4.1 ConceptNet-Numberbath (CN)

Given an edge (u, r, v), the CN model [28] encourages the embedding vector of u to be close to both the pre-trained embedding vector of u and the embedding vector of its neighbor v. The objective of CN is to minimize:

$$\mathcal{L}_{cn}(u) = \alpha \parallel y_u - w_u \parallel + \beta_{u,v} \parallel y_u - y_v \parallel, \tag{14}$$

where w_u is pre-trained embedding vector of u, y_u and y_v are the embedding vectors of u and v in the ensemble model, respectively.

Proposition 1. CN can be fully recovered by RA-Retrofit embedding when $m = 0$, $\mathcal{W}(u, r, v) = 2\beta_{u,v}$, $t = 1$, $\alpha = \mu$, $\mathbf{M}$ is identity matrix, the number of negative samples $N_s = 0$, and remove the relation embedding q_r and regularization terms.

Proof. For each edge (u, r, v) from the knowledge graph, we remove the regularization terms and set $m = 0, N_s = 0, \mathcal{W}(u, r, v) =$

$2\beta_{u,v}$ and $t = 1$. We can get the objective function of RA-Retrofit as

$$\begin{aligned}\mathcal{L}(u,v,r) =& \mathcal{L}_1(u,v,r) + \mu\mathcal{L}_2(u) + \mu\mathcal{L}_2(v)\\ =& \mathcal{W}(u,r,v)\max\{\phi(u,r,v),0\} + \mu\mathcal{L}_2(u) + \mu\mathcal{L}_2(v)\\ =& 2\beta_{u,v} \parallel y_u + q_r - y_v \parallel + \mu \parallel \mathbf{M}y_u - w_u \parallel\\ &+ \mu||\mathbf{M}y_u - w_v||.\end{aligned} \tag{15}$$

By removing the relation embedding q_r, the objective function $\mathcal{L}$ can be rewrite as

$$\mathcal{L}(u,v) = 2\beta_{u,v} \parallel y_u - y_v \parallel + \mu \parallel y_u - w_u \parallel + \mu \parallel y_v - w_v \parallel . \tag{16}$$

The objective of an edge (u,r,v) to be minimized in CN can be viewed as the addition of the objective functions for u and v:

$$\begin{aligned}\mathcal{L}_{cn}(u,v) =& \mathcal{L}_{cn}(u) + \mathcal{L}_{cn}(v)\\ =& \alpha \parallel y_u - w_u \parallel + \beta_{u,v} \parallel y_u - y_v \parallel + \alpha \parallel y_v - w_v \parallel\\ &+ \beta_{u,v} \parallel y_v - y_u \parallel\\ =& \alpha \parallel y_u - w_u \parallel + 2\beta_{u,v} \parallel y_u - y_v \parallel + \alpha \parallel y_v - w_v \parallel .\end{aligned} \tag{17}$$

When $\alpha = \mu$, this function can be fully recovered by the proposed model. □

4.2 Meta-Embedding (MetaE)

MetaE [35] leverages multiple word embedding models and defines the objective function as:

$$\sum_{k=1}^{t} f_m(\parallel \mathbf{M}_k y_v - w_v^k \parallel + \gamma \parallel \mathbf{M}_k \parallel), \tag{18}$$

where y_v is the learned embedding of word $v \in V$, w_v^k is the embedding of word v in pre-trained embedding model W^k, $\mathbf{M}_k$ is transformation matrix.

Proposition 2. MetaE can be fully recovered by RA-Retrofit embeddings when $\mathcal{W}(u,r,v) = 0$, $f_m = \mu$, $\eta_1 = 0$ and $\eta_2 = f_m \times \gamma$.

Proof. When $\mathcal{W}(u,r,v) = 0$, the objective function of RA-Retrofit can be reduced to

$$\mathcal{L}(u,v) = \mu\mathcal{L}_2(u) + \eta_1 \parallel \Theta_1 \parallel + \eta_2 \parallel \Theta_2 \parallel . \tag{19}$$

For each word v in the edge, the objective of RA-Retrofit to be minimized can be reduced to

$$\begin{aligned}\mathcal{L}(v) =& \mu\mathcal{L}_2(v) + \eta_1 \parallel \Theta_1 \parallel + \eta_2 \parallel \Theta_2 \parallel\\ =& \mu \sum_{k=1}^{t} \parallel \mathbf{M}_k y_v - w_v^k \parallel + \eta_1 \parallel y_v \parallel + \eta_2 \sum_{k=1}^{t} \parallel \mathbf{M}_k \parallel .\end{aligned} \tag{20}$$

When $f_m = \mu$, $\eta_1 = 0$, $\eta_2 = f_m \times \gamma$, the proposed model can recover the meta-embed model. □

4.3 Translating Embeddings (TransE)

TransE [1] is a knowledge graph embedding model. The objective of the TransE is to minimize:

$$\begin{aligned}&\max\{m + \phi(u,r,v) - \phi'(u',r,v'), 0\}\\ &s.t. \quad \phi(u,r,v) = \parallel y_u + q_r - y_v \parallel,\end{aligned} \tag{21}$$

where where y_u and y_v are the learned representation vectors for u and v, respectively, q_r is the embeddings for relation r, $\phi(u,r,v)$ and $\phi'(u',r,v')$ are the scores for positive and negative edges, respectively.

Proposition 3. TransE can be fully recovered by RA-Retrofit when $\mu = \eta_1 = \eta_2 = 0$ and $\mathcal{W}(u,r,v) = 1$.

Proof. It is easy to verify that RA-Retrofit can recover TransE if we set $\mathcal{W}(u,r,v) = 1$ and $\mu = \eta_1 = \eta_2 = 0$. □

5 EXPERIMENTAL STUDY

In this section, we first describe the datasets that are used in the experimental study. Then we present the settings of our experiments. Finally, we investigate the performance of the evaluated methods. All the experiments are done on a computer having Intel(R) Core(TM) i7-6700 CPU @3.40 GHz processor, 6 GB RAM, and running Ubuntu 16.04. Our source codes will be released at https://github.com/fanglanting/RA-retrofit.

5.1 Datasets

We use GloVe[1] and word2vec[2] as the pre-trained embedding vectors for learning. We use ConceptNet (CNet)[3], PPDB[4] and WordNet (WNet)[5] as the knowledge graphs for retrofitting.

5.1.1 Pre-trained embeddings. **Glove** consists of 300 dimensional embedding vectors for 1.9 million words. These vectors were trained on 6 billion words from Wikipedia and English Gigaword. **Word2vec** are trained on 100 billion words of Google news dataset and are of length 300. It consists of embedding vectors for 3 million words.

5.1.2 Knowledge graphs. **CNet** originated as a machine-parsed version of the early crowd-sourcing project called Open Mind Common Sense, and has expanded to include several other data sources. It is a graph whose edges express common sense relationships between two short phrases or words, known as concepts. The edges are labeled from a defined set of relations, such as IsA, HasA, or UsedFor, expressing what relationship holds between the concepts. It contains a total of 2.8 millions of edges, in which 842, 194 edges remain after filtering out the edges whose words do not appear in the vocabularies of both word2vec and GloVe.

PPDB is another resource that is useful for learning about word similarity. It lists pairs of words that are translated to the same word in parallel corpora. Most of its edges are labeled with "Equivalent" and a small part is labeled with other types such as "OtherRelated" and "ForwardEntailment". It contains a total of 7.4 million edges for English language, in which 6 million edges remain in our experiments.

WNet is a lexical database for English. It groups English words into sets of synonyms called synsets, and records a number of relations among these synsets. It explicitly relates concepts with semantically aligned relations such as "hypernyms" and "hyponyms". For example, the word dog is a synonym of canine, a hypernym of puppy and a hyponym of animal. It contains a total of 843, 856 edges, in which 688, 147 edges remain in our experiments.

[1] https://nlp.stanford.edu/projects/glove/
[2] https://code.google.com/archive/p/word2vec/
[3] https://github.com/commonsense/conceptnet5/wiki
[4] http://paraphrase.org/#/download
[5] https://wordnet.princeton.edu/

5.2 Evaluated Methods

We evaluate three types of methods in our experimental study: individual embedding models, ensemble models, and RA-Retrofit model.

Individual embedding models. In the first type of methods, we evaluate the performance of three well-known embedding models: **GloVe** [24], word2vec [18], denoted by **W2V**, and **TransE** [1]. We use the pre-trained vectors for GloVe and word2vec that are published online. We combine CNet, PPDB and WNet as the knowledge graph to train the TransE embeddings.

Ensemble models. In the second type of methods, we consider two state-of-the-art ensemble models **CN** [28] and **MetaE** [35]. We use the published pre-trained vectors for CN directly. We train MetaE to combine word2vec and GloVe with the suggested hyper-parameters [35]. In addition, we adapt the TransE model, denoted by **TransE+**, to make it suitable for embedding words into vector space. Specifically, **TransE+** uses TransE, word2vec and GloVe as the pre-trained embeddings models, and use Eq. (6) to learn the ensemble embedding.

RA-Retrofit model. In the third type of methods, we evaluate the proposed RA-Retrofit model by varying the components in the model. Recall that the RA-Retrofit model combines knowledge graph and pre-trained word embeddings to learn the ensemble model. We vary the knowledge graphs and the pre-trained word embedding models that RA-Retrofit uses and get different variations of RA-Retrofit model. All these variations of RA-Retrofit models are trained by back-propagation with mini-batches. In the experiments, we set the hyper-parameters margin ridge m as 1, parameters μ as 0.01, η_1 and η_2 as 5×10^{-5}, learning rate γ as 0.02, the number of negative samples as 5 and iteration epoch as 100. When retrofitting a single knowledge graph, we set the batch size as 128. When retrofitting multiple knowledge graphs, we set the batch size as 256. In the subsampling process, we use geometric mean as the combine operator.

5.3 Evaluation Tasks

We conduct experiments on three tasks, word visualization, word similarity and word analogy to evaluate the performance of the proposed ensemble model.

The word anology task consists of questions like, "a is to b as u is to ?". In our experimental study, we use the state-of-the-art Google analogy test set [6]. Note that not all questions in the test set are covered by vocabularies of the embedding models. We filter out the uncovered questions and use 17,809 remaining questions in the test set. These questions are divided into two subsets: a semantic subset of size 8,733 and a syntactic subset of size 9,076. Given three words a, b, and u, we find the optimal word v for the analogy "a is to b as u is to v" by using the 3CosMul function [14] defined as follows:

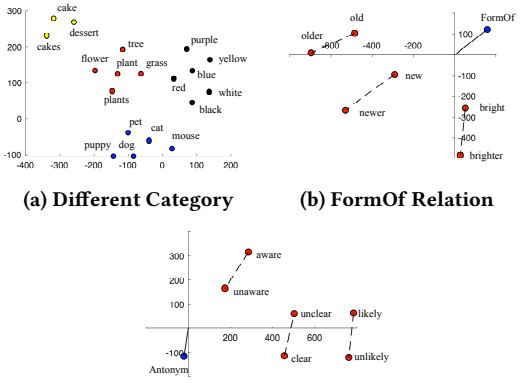
$$v = \arg\max_{y_v} \frac{(\cos(y_v, y_b) + 1) \cdot (\cos(y_v, y_u) + 1)}{2(\cos(y_v, y_a) + 1) + 0.004} \tag{22}$$

where y_a, y_b, y_u and y_v are the embeddings of word a, b, u and v.

In the word similarity task, we use the following word-similarity gold standards: WordSim-353 (WS353) [8], The Stanford Rare Words (RW) [16], MC-30 (MC) [20], MEN-3000 (MEN) [2] and Stanford Contextual Word Similarity (SCWS) [12]. Each test set contains pairs of words together with their human-assigned similarity scores. We only consider the pairs of words that are also in the vocabularies of the elvauated models. The test set WS353, RW, MC, MEN, and SCWS contains 353, 1707, 30, 3000, and 1971 valid pairs, respectively. For each test set, we calculate cosine similarity between the vectors of two words, and report Spearman's rank correlation coefficient [33] between the rankings produced by our model against the human rankings:

$$\rho = 1 - \frac{6 \sum_{i=1}^{N} \Delta_i^2}{N(N^2 - 1)}, \tag{23}$$

where Δ_i is the difference between the two ranks of each observation and N is the number of observations.



(a) Different Category **(b) FormOf Relation**

(c) Antonym Relation

Figure 3: Visualization of word embeddings by t-SNE

5.4 Word Visualization

An intuitive way to understand the performance of a word embedding model is to visualize the embedding vectors. This task allows us to make qualitative assessment of the learned word embeddings.

We project word emebeddings into a 2-dimensional space using the t-SNE toolbox [17], as shown in Figure 3 (a). We can observe from Figure 3 (a) that words sharing similar semantics are closer to each other in the 2-dimensional space than the other words. For example, the words at the right side of Figure 3 (a) are all about color, while the words at the bottom are all related to animal.

To evaluate the analogy feature of the RA-Retrofit embeddings, we randomly select three edges with "FormOf" relation and three edges with "Antonym" relation from the knowledge graphs. Relation "FormOf" represents *A is an inflected form of B and B is the root word of A*. Relation "Antonym" represents *A and B are opposites in some relevant way*. We project the embedding vectors of words and relations in the edges into the 2-dimensional space. The embeddings of the selected edges are reported in Figure 3 (b) and Figure 3 (c). These results show that the embeddings can capture the analogy relationships between the words, i.e. A is to B as C is to D. For instance, vec("aware")-vec("unaware") is similar to vec("likely")-vec("unlikely"). Moreover, we observe that the embedding vector of a relation is similar to the subtraction between the embedding vectors of the two words. For instance, vec("aware")-vec("unaware") is similar to vec("'Antonym").

[6] http://download.tensorflow.org/data/questions-words.txt

Table 3: Results on word analogy tasks. If a result is the best result in a block and better than all models in "Ind", then it is bolded. The best result for each benchmark is marked with †.

	Model	Semantic	Syntactic	Full
Ind	W2V	75.51	72.73	74.09
	GloVe	81.17	70.81	75.89
	TransE	31.65	6.17	18.66
Ens	CN	71.54	**76.43**	74.04
	MetaE	**82.10**	70.82	76.35
	TransE+	80.08	74.78	**77.38**
RA-Retrofit	GloVe			
	+PPDB	78.90	**73.89**	76.34
	+WNet	81.59	66.44	73.87
	+CNet	**84.40**	70.02	**77.07**
	+All	82.01	66.91	74.32
	W2V			
	+PPDB	72.56	**75.85**	74.24
	+WNet	78.63	67.76	73.09
	+CNet	**83.17**	69.46	**76.18**
	+All	81.56	68.41	74.86
	W2V + GloVe			
	+PPDB	77.32	75.66	76.47
	+WNet	81.46	68.88	75.05
	+CNet	**85.61**†	73.83	79.61
	+All	85.51	**78.46**†	**81.92**†

Table 4: Results on word similarity tasks. If a result is the best result in a block and better than all models in "Ind", then it is bolded. The best result for each benchmark is marked with †.

	Model	WS353	RW	MC	MEN	SCWS
Ind	W2V	70.35	53.41	80.25	77.56	66.28
	GloVe	76.25	52.00	83.99	81.95	66.14
	TransE	64.35	36.01	76.02	69.26	55.99
Ens	CN	**81.74**	**64.11**†	**88.51**	**86.40**	**73.13**†
	MetaE	78.39	53.67	83.54	83.18	66.37
	TransE+	76.54	54.86	87.93	83.06	66.38
RA-Retrofit	GloVe					
	+PPDB	77.89	54.01	85.61	84.93	67.07
	+WNet	79.88	56.78	86.28	84.19	66.98
	+CNet	81.87	57.00	84.28	85.10	68.14
	+All	**83.60**†	**62.38**	**87.93**	**86.21**	**71.03**
	W2V					
	+PPDB	70.64	53.74	81.78	81.19	66.87
	+WNet	72.23	58.43	**87.75**	80.01	65.92
	+CNet	75.33	59.09	84.79	82.70	67.00
	+All	**77.33**	**63.88**	85.39	**83.29**	**70.53**
	W2V + GloVe					
	+PPDB	76.40	53.85	84.05	83.49	68.31
	+WNet	80.48	61.47	89.11	84.22	69.71
	+CNet	**83.18**	62.24	89.86	**86.95** †	70.93
	+All	81.74	**62.64**	**90.20**†	86.13	**71.20**

5.5 Word Analogy

We next conduct experiments to evaluate the capability of the proposed model for semantic deduction with the word analogy task [18].

Table 3 reports the performance of each model in word analogy tasks. Block "Ind" lists the performance of individual embedding models W2V, GloVe and TransE. Block "Ens" lists the performance of state-of-the-art ensemble models CN, MetaE and TransE+. Block "RA-Retrofit" lists all the variations of the proposed RA-Retrofit models. Specifically, we use GloVe, W2V, and both GloVe and W2V as the pre-trained word embeddings for the RA-Retrofit model, respectively. We vary the knowledge graphs by using PPDB, WNet, CNet, and all three graphs (All).

From block "Ind" and "Ens", we observe that the analogy test accuracy of CN is comparable with W2V and GloVe in the full test set. This is because CN does not utilize the relation information, which is very important to analogy tasks. On the contrary, MetaE outperforms W2V and GloVe w.r.t. analogy test accuracy in the full test set. The results show that combing multiple embeddings can generate an ensemble embedding that contains more analogy information. Moreover, we observe that TransE has the worst accuracy compared with other methods. This is because the semantic information contained in knowledge graphs is not enough for word embeddings. When combined with pre-trained embeddings, TransE+ achieves a better accuracy than the other individual models or state-of-the-art ensemble models.

When the RA-Retrofit model is trained with GloVe or with W2V, we observe that combining knowledge graphs can improve the accuracy compared with each individual embedding model. Among the knowledge graphs, CNet achieves the best increase in the accuracy. This is because CNet contains more edges than WNet, and more relation categories than PPDB (CNet has 37 categories of relations, while PPDB has 6 categories).

When the RA-Retrofit model is trained with both GloVe and W2V, we observe that using all knowledge graphs achieves the best performance. This full model is 20% times better than CN in semantic test set, and 11% times better than W2V, 8% times better than GloVe, 11% times better than MetaE and 6% times better than TransE+ in the full test set.

Based on the observations, we conclude that the word representations generated by the RA-Retrofit model achieves the best performance in the word analogy task.

5.6 Word Similarity

Table 4 reports the empirical results on the word similarity task. We observe that the full RA-Retrofit model (trained with both pre-trained word embeddings and all knowledge graphs) outperforms all individual word embedding models. For instance, the full model is 16% times better than W2V and 7% times better than GloVe in WS353. Additionally, the full RA-Retrofit model performs better than ensemble models MetaE and TransE+. Although CN model generates competitive results compared with the proposed model, its analogy results are much worse than the proposed model. These results show that the proposed model can improve the performance of the word embedding not only at the analogy tasks but also at the word similarity tasks.

According to the results, we conclude that the proposed RA-Retrofit model is capable to generate high quality word embeddings compared with state-of-the-art methods.

Table 5: The effects of various modifications to the RA-Retrofit. Bold indicates the best result.

Model	Semantic	Syntactic	Total
RA-Retrofit-SVD	84.44	77.10	80.70
RA-Retrofit-unweighted	85.17	76.83	80.92
RA-Retrofit-Avg	85.15	77.58	81.29
RA-Retrofit-Max	84.47	78.24	81.30
RA-Retrofit-LogLoss	78.82	70.42	74.54
RA-Retrofit	**85.51**	**78.46**	**81.92**

5.7 Miscellaneous

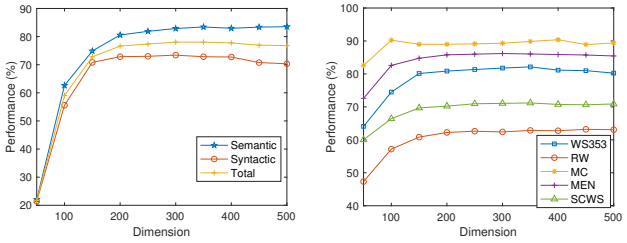
5.7.1 Ablation study for the RA-Retrofit model. We first conduct a set of experiments to justify the design of RA-Retrofit model by varying the components.

Effectiveness of transformation matrix. When learning from pre-trained word embeddings, we use a transformation matrix to project the ensemble embedding of a word such that the projection should be close to the word's pre-trained embedding vector. In this set of experiments, we modify the RA-Retrofit by replacing the transformation matrix with singular value decomposition (SVD). Specifically, we first concatenate the GloVe (300-dimensional) and word2vec (300-dimensional) vectors into 600-dimensional vectors. We then discount redundancy between its features by transforming these 600-dimensional vectors to 300-dimensional vectors with SVD. We encourage the ensemble embedding vector to be close to the decomposed vector. We refer to this modification as *RA-Retrofit-SVD*. The first row in Table 5 reports the accuracy of *RA-Retrofit-SVD*. We observe that using transformation matrix achieves better accuracy than using SVD.

Effectiveness of weight. As discussed in Section 3.6, we propose a scheme to assign weights to each edge. In this set of experiments, we modify the RA-Retrofit by using unweighted global loss to show the effectiveness of our weighting scheme. The modification is referred to as *RA-Retrofit-unweighted* and its accuracy is reported in the second row of Table 5. The results show that the weighting scheme improves the accuracy of the proposed model.

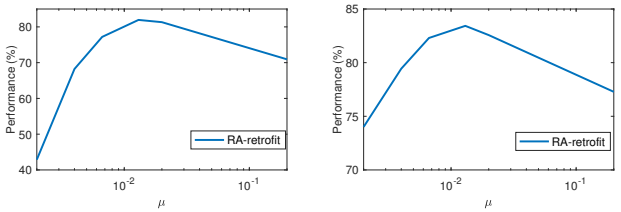
Effectiveness of combining operator. Recall that we discussed subsampling technique in Section 3.7, where we propose several combining operator to combine frequency of an edge. In this set of experiments, we replace the geometric mean in RA-Retrofit with *average* and *max*, respectively. The modifications are referred to as *RA-Retrofit-Avg* and *RA-Retrofit-Max*. The third and fourth rows in Table 5 report the accuracies of the two modifications. We observe that the RA-Retrofit model achieves the best performance when geometric mean is used as the combining operator.

Effectiveness of $\mathcal{L}_1$. As discussed in Section 3.3, we use margin-based ranking criterion to learn the information in the knowledge graphs. Another popular criterion that is often used to learn the knowledge graph embeddings is logistic loss. We modify the RA-Retrofit by changing the margin-based ranking criterion to logistic loss. The modification is referred to as *RA-Retrofit-LogLoss* and its accuracy is reported in the fifth row in Table 5. We observe that the performance drops rapidly. This is because the margin-based ranking criterion tend to assign low scores to positive edges and high scores to negative edges, which makes the learned representations more reasonable.



(a) Word Analogy **(b) Word Similarity**

Figure 4: Effect of Dimensions



(a) Word Analogy (Full) **(b) Word Similarity (WS353)**

Figure 5: Effect of parameter μ

5.7.2 Effects of dimensions and hyper-parameter μ. Figure 4 shows the effects of dimensionality $d \in [50, 500]$ for RA-Retrofit. There are no big differences in the word analogy and similarity tasks when the dimension d is high enough, roughly in the interval $[300, 500]$. In summary, as long as d is chosen to be large enough, the performance of the proposed model is robust.

Figure 5 demonstrates how hype-parameter $\mu \in [0.002, 0.2]$ affects RA-Retrofit in word similarity and word analogy tasks. We observe that the its performance degrades when the value of μ is either too small or too large. The best performance is obtained when μ is set to a value between 0.007 and 0.04. When μ is too small, the result is similar to only using the knowledge graph. When μ is too large, the result is similar to only using the pre-trained word embeddings. This experiment justifies that both knowledge graph and pre-trained word embeddings actually help improving word similarity computation and word analogy.

6 CONCLUSION

In this paper, we proposed a knowledge-enhanced ensemble learning method named RA-Retrofit for word embeddings. RA-Retrofit integrates common sense knowledge into an ensemble model of pre-trained distributional word embeddings to learn high quality word embeddings that is suitable for both word similarity and analogy. Our contributions include a method for translating embedding, equipped by a novel edge weighting scheme for knowledge graph, a joint learning method for optimizing both objectives from knowledge graph and pre-trained embeddings, and a thorough theoretical analysis among RA-Retrofit and other ensemble methods. Extensive experiments demonstrate that our proposed method is up to 20% times better than CN in word analogy task and up to 16% times better than W2V in word similarity task.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

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

[3] Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1602–1612.

[4] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. 2017. Global rdf vector space embeddings. In *International Semantic Web Conference*. 190–207.

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

[6] Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving Knowledge Graph Embedding Using Simple Constraints. *arXiv preprint arXiv:1805.02408* (2018).

[7] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1606–1615.

[8] 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*. 406–414.

[9] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 758–764.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[11] Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448* (2014).

[12] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew 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*. 873–882.

[13] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge Graph Completion with Adaptive Sparse Transfer Matrix.. In *AAAI*. 985–991.

[14] Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*. 171–180.

[15] Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. 2014. Pre-Trained Multi-View Word Embedding Using Two-Side Neural Network.. In *AAAI*. 1982–1988.

[16] Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the 17th Conference on Computational Natural Language Learning*. 104–113.

[17] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008), 2579–2605.

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

[19] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[20] George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6, 1 (1991), 1–28.

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

[22] Avo Muromägi, Kairit Sirts, and Sven Laur. 2017. Linear Ensembles of Word Embedding Models. *arXiv preprint arXiv:1704.01419* (2017).

[23] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *ICML*, Vol. 11. 809–816.

[24] 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*. 1532–1543.

[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[26] Richard Socher, John Bauer, Christopher D Manning, et al. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 455–465.

[27] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*. 926–934.

[28] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 4444–4451.

[29] Robyn Speer and Catherine Havasi. 2013. ConceptNet 5: A large semantic network for relational knowledge. In *The People's Web Meets NLP*. Springer, 161–176.

[30] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. 41–47.

[31] Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 938–950.

[32] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. 384–394.

[33] Arnold D Well and Jerome L Myers. 2003. *Research design & statistical analysis*. Psychology Press.

[34] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1219–1228.

[35] Wenpeng Yin and Hinrich Schütze. 2016. Learning Word Meta-Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 1351–1360.

[36] Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Vol. 2. 545–550.