Learning Term Embeddings for Lexical Taxonomies

Jingping Liu[1], Menghui Wang[1], Chao Wang[1], Jiaqing Liang[1], Lihan Chen[1], Haiyun Jiang[1*], Yanghua Xiao[1†], Yunwen Chen[2]
[1]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, China
[2]DataGrand Inc., Shanghai, China
{jpliu17, 19212010074, 17110240038, jianghy16, shawyh}@fudan.edu.cn
{l.j.q.light, lhc825}@gmail.com, chenyunwen@datagrand.com

Abstract

Lexical taxonomies, a special kind of knowledge graph, are essential for natural language understanding. This paper studies the problem of lexical taxonomy embedding. Most existing graph embedding methods are difficult to apply to lexical taxonomies since 1) they ignore implicit but important information, namely, sibling relations, which are not explicitly mentioned in lexical taxonomies and 2) there are lots of polysemous terms in lexical taxonomies. In this paper, we propose a novel method for lexical taxonomy embedding. This method optimizes an objective function that models both hyponym-hypernym relations and sibling relations. A term-level attention mechanism and a random walk based metric are then proposed to assist the modeling of these two kinds of relations, respectively. Finally, a novel training method based on curriculum learning is proposed. We conduct extensive experiments on two tasks to show that our approach outperforms other embedding methods and we use the learned term embeddings to enhance the performance of the state-of-the-art models that are based on BERT and RoBERTa on text classification.

Introduction

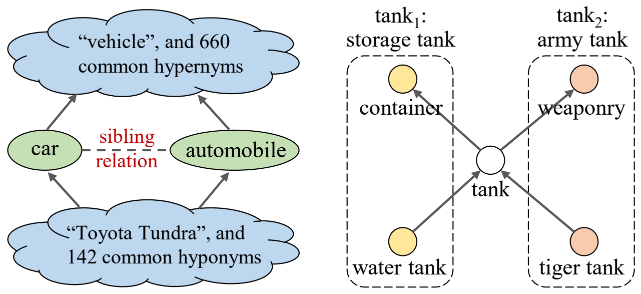
Knowledge Graphs (KGs), e.g., WordNet (Miller 1995) and Freebase (Bollacker et al. 2008), are an important resource for many applications, such as question answering (Cui et al. 2017) and recommendation (Wang et al. 2019). However, KGs adopting symbolic and logical representations are hard to support tasks with intensive numerical computation. Thus, many studies have been conducted to embed KGs into low-dimension spaces. These efforts are referred to as *knowledge graph embedding*.

Embedding techniques for KGs, in general, can be roughly divided into two categories. First, inspired by word2vec (Mikolov et al. 2013), a series of translation-based models have been proposed, such as TransE (Bordes et al. 2013) and RotatE (Sun et al. 2019). The basic idea of these models is to learn the embeddings $\vec{h}$, $\vec{r}$, and $\vec{t}$ satisfying $\vec{h} + \vec{r} \approx \vec{t}$ for a given triple (h, r, t), where r is the relation between the head entity h and the tail entity t. Second, semantic matching models like HolE (Nickel, Rosasco, and Poggio 2016) and ComplEx (Trouillon et al. 2016) are proposed to learn latent representations. They expect the embeddings for each triple to satisfy $\vec{h}^T \mathbf{M}_r \approx \vec{t}^T$, where $\mathbf{M}_r$ is a matrix associated with the relation r.

In this paper, we focus on the embedding of a special kind of KGs: *lexical taxonomy* (LT) built by data-driven approaches. LTs consist of *hyponym-hypernym* relations between terms. One term is a hypernym of another term if the meaning of the former covers the latter (Sang 2007). For example, `fruit` is a hypernym of `apple`. The opposite of hypernym is hyponym, so `apple` is a hyponym of `fruit`. In this paper, we use *hypo*(x, y) to represent a hyponym-hypernym relation, where x is a hyponym of y. The reason we study LTs is that hyponym-hypernym relations are the key to the proper understanding of natural language texts. For example, given a text "*Kobe Bryant died at age 41, along with his 13-year-old daughter Gianna and seven others ...*", Although recent pre-trained language models (e.g., BERT) have achieved remarkable improvements in a wide range of NLP tasks, it is difficult to accurately classify the text in a classification task. The main reason is that the models cannot capture that `Kobe Bryant` is an `NBA star`, which is useful for classifying the text into the class *sport*. Thus, to effectively integrate hyponym-hypernym relations into the numerical models, it is necessary to embed LTs into low-dimensional spaces.

However, the above KG embedding approaches are not suitable for lexical taxonomy embedding (LTE) due to their ignorance of the inherent hierarchical structure of LTs. For example, considering *hypo*(h, m), *hypo*(m, t), and *hypo*(h, t) in translation-based models, if $\vec{h} + \vec{r} \approx \vec{m}$ and $\vec{m} + \vec{r} \approx \vec{t}$ hold where r is the hyponym-hypernym relation, it is impossible for $\vec{h} + \vec{r} \approx \vec{t}$ since $\vec{r} \neq 0$ (Chen et al. 2018). Semantic matching models are confronted with the similar problem. To take into account the hierarchies, some studies such as the dynamic distance-margin model (DDMM) (Yu et al. 2015), TransC (Lv et al. 2018), and On2Vec (Chen et al. 2018), employ hierarchical neighborhood information in translation-based models to enforce the close representation between nodes. Other efforts (Nickel and Kiela 2017) use hyperbolic space to model hierarchical structure with relatively fewer dimensions. Although these methods take into account the

[*]Now he works for Tencent AI Lab in Shenzhen, China.
[†]Corresponding author



(b) Polysemous term

Figure 1: Examples of sibling relations and polysemous terms in Probase. The term at the beginning of the arrow is a hyponym, while the term at the end is a hypernym.

hierarchical structure of taxonomy, they ignore implicit but important information, namely, *sibling relation*, which is not explicitly mentioned in LTs. Terms within a sibling relation[1] mean that they share many common direct/indirect neighbors in an LT, which indicates that they are often semantically similar. In other words, two terms with a sibling relation should be close to each other in the embedding space. As an example in Figure 1(a), the vectors of `car` and `automobile` in the embedding space should be similar since these two terms share 143 hyponyms and 661 hypernyms in Probase (Wu et al. 2011), which is one of the largest LTs. Thus, in this paper, we incorporate sibling relations into the embedding learning model to learn a more reasonable representation for each term.

Besides, existing taxonomy embedding methods are difficult to be used for LTs[2] since, in LTs, there are lots of **polysemous terms** that have a negative impact on the embedding learning of other terms. Existing methods often learn term embeddings through the semantic correlation between terms x and y for a given *hypo*(x, y). However, when a term is polysemous, its neighbors will introduce bias in terms of semantics. Take using hyponym to predict hypernym as an example. Given *hypo*(`tank`, `weaponry`), it is difficult to learn an accurate representation for the term `weaponry` from `tank` since the embedding of `tank` reflects the semantics of `water tank` through *hypo*(`water tank`, `tank`), where `tank` has two different meanings: $\texttt{tank}_1$ =`storage tank` and $\texttt{tank}_2$ =`army tank`, as shown in Figure 1(b).

To address the above problems, in this paper, we propose a novel deep model for LTE. First, to fully use information including both explicit and implicit relations, i.e., hyponym-hypernym and sibling relations, we design a joint model that reflects these two relations in an LT. Then, to reduce the negative impact of a polysemous term on another term in a hyponym-hypernym relation, a term-level attention mechanism is proposed. For a target term, the core idea of this mechanism is to explore the importance of its hyponyms/hypernyms to the target. Besides, since sibling relations are not explicitly mentioned in LTs, a metric based on the random walk is proposed to obtain siblings in advance to provide training data for the deep model. Finally, inspired by the learning process of humans, which generally starts with learning easier samples and then gradually considers complex ones, we propose a novel method to determine the training order of samples based on curriculum learning (Bengio et al. 2009).

Contributions. The contributions of this paper are summarized as follows:

- We present the first effort to work on embedding learning for a special kind of KG: LT. One of the significant characteristic of LTs is that the terms in LTs are not disambiguated.
- We design a joint model to characterize hyponym-hypernym and sibling relations and propose a novel training method based on curriculum learning.
- Experimental results on two tasks show that our model outperforms other embedding methods and the learned vectors of terms can be used to improve the performance of BERT and RoBERTa on text classification.

Problem Definition

In this section, we first formalize the problem of LTE. Then, we give two properties that we expect the term embeddings to satisfy.

Given a lexical taxonomy $LT = \{\mathcal{T}, R\}$, where $\mathcal{T}$ is the set of terms and R is the hyponym-hypernym relations between terms, the goal of LTE is to represent each term $t \in \mathcal{T}$ into a low-dimensional space $\mathbb{R}^d$ where $d \ll |\mathcal{T}|$. Specifically, we assign two embeddings[3] to each term t: the *hyponym embedding* $\vec{v}_t$ and the *hypernym embedding* $\vec{u}_t$. The rationale is that the hyponym-hypernym relation is asymmetric and we need to distinguish the roles of two terms in a hyponym-hypernym relation. We use $\vec{v}_t$ to represent t when t is treated as a hyponym and use $\vec{u}_t$ to represent t when t is treated as a hypernym.

We expect to learn term embeddings that encode hyponym-hypernym and sibling relations in an LT. More specifically, we hope that the learned embeddings reflect two properties:

- Hyponym-hypernym relation. If *hypo*(x, y) holds, then $\vec{v}_x$ and $\vec{u}_y$ are similar under a bias b, which is denoted as $(\vec{v}_x \simeq \vec{u}_y)_{|b}$. For example, $(\vec{v}_{apple} \simeq \vec{u}_{fruit})_{|b}$. For the details of b, we will elaborate on it in the next section.
- Sibling relation. If two terms t_1 and t_2 are siblings (denoted as *sib*(t_1, t_2)), then $\vec{u}_{t_1}$ and $\vec{u}_{t_2}$ are similar, which is denoted as $\vec{u}_{t_1} \simeq \vec{u}_{t_2}$. For example, $\vec{u}_{\texttt{dog}} \simeq \vec{u}_{\texttt{cat}}$. Similarly, $\vec{v}_{t_1} \simeq \vec{v}_{t_2}$.

In this paper, we use Probase[4] (Wu et al. 2011) as an ex-

[1]Since Probase is a graph, the sibling relation we define is different from that in a tree where sibling nodes share the same parent node.

[2]Distinct from general taxonomies, the terms in LTs are not disambiguated since LTs are often built by data-driven methods.

[3]Due to the incompleteness of LTs, that is, the leaf/root nodes may have hyponyms/hypernyms, we still assign two vectors to the leaf and root nodes, respectively.

[4]Our method is also applicable to other LTs.

ample to learn term embeddings. Probase contains lots of hyponym-hypernym relations. Each hyponym-hypernym relation is associated with a frequency observed on corpora. Based on the frequency, we obtain the probability $p_c(x|y)$ of a hyponym x given a hypernym y and the probability $p_c(y|x)$ of y given x:

$$p_c(x|y) = \frac{n(x,y)}{\sum_{x_i} n(x_i,y)},\ p_c(y|x) = \frac{n(x,y)}{\sum_{y_i} n(x,y_i)}, \quad (1)$$

where $n(x,y)$ is the co-occurrence frequency of x and y on corpora. We define $p_c(x|y)$ and $p_c(y|x)$ as the *typicality* of x and y, respectively. Intuitively, the typicality tells us how popular x (y) is as far as y (x) is concerned. For example, given a term dog, people are more likely to think of it as an animal than a mammal, which is embodied by $p_c(\texttt{animal}|\texttt{dog}) > p_c(\texttt{mammal}|\texttt{dog})$.

LTE: Lexical Taxonomy Embedding

In this section, a novel method for LTE is first proposed to model both hyponym-hypernym and sibling relations. Then, a term-level attention mechanism and a random walk based metric are proposed to assist in modeling these two kinds of relations, respectively. Since hyponym-hypernym relations are asymmetric, we model hyponym-hypernym relations from two directions: from hyponym to hypernym and vice versa. For convenience, we discuss our solution for the hypernym with respect to hyponym.

LTE with Hyponym-hypernym Relation

To model the hyponym-hypernym relation from the hyponym x to the hypernym y, we expect $\vec{v}_x$ and $\vec{u}_y$ to be similar under a bias b, i.e., $(\vec{v}_x \simeq \vec{u}_y)_{|b}$, where b is defined as the bias of the target term y, i.e., b_y, which is used to measure the importance of y. Formally, we define a potential function $\psi(y|x) = \exp(\vec{u}_y^T \vec{v}_x + b_y)$ to judge whether y is a hypernym of x. If $\psi(y|x)$ is large enough, we consider y to be a hypernym of x. Furthermore, the function $\psi(y|x)$ in the embedding space is normalized as follows:

$$p(y|x) = \frac{\exp(\vec{u}_y^T \vec{v}_x + b_y)}{\sum_{i=1}^{|\mathcal{T}|} \exp(\vec{u}_i^T \vec{v}_x + b_i)}, \quad (2)$$

where $|\mathcal{T}|$ is the number of terms in an LT and b_i is the bias of the i-th term. Eq. (2) defines a conditional probability of the hypernym y generated by the hyponym x. Thus, our goal is to maximize the likelihood of the above objective function.

Unfortunately, computing Eq. (2) is expensive since it requires summing overall terms in $\mathcal{T}$, which is in general very large. To address this problem, we employ the method of negative sampling (Mazur et al. 2019), which samples multiple wrong hypernyms of x for each relation *hypo*(x, y). Thus, the above objective function is approximated as:

$$p(y|x) = \frac{\exp(\vec{u}_y^T \vec{v}_x + b_y)}{\exp(\vec{u}_y^T \vec{v}_x + b_y) + \sum\limits_{y'} \exp(\vec{u}_{y'}^T \vec{v}_x + b_{y'})}, \quad (3)$$

where $y' \in S_n(x)$ ($y' \neq y$) and $S_n(x)$ is a set of wrong hypernyms of x, which is randomly sampled from all terms, and n is the number of wrong hypernyms. For each hyponym x, this objective function enforces the correct hypernym y to be larger in terms of $\psi(\cdot|\cdot)$ than the wrong hypernym y'. To train the model, we have the loss $loss_h = -\log p(y|x)$.

Term-level Attention Mechanism

For a relation *hypo*(x, y), if the hyponym x is polysemous, it will have a large negative impact on the learning of embedding for the hypernym y. Thus, we expect to reduce the negative impact of x on y so that the semantic information that $\vec{u}_y$ captures from $\vec{v}_x$ can be reduced. To this end, we employ a term-level attention mechanism (Bahdanau, Cho, and Bengio 2014).

Given a relation *hypo*(x, y) and the hyponym set $\mathcal{X} = \{x_1, ..., x_m\}$ of y where $x \in \mathcal{X}$, the term-level attention aims to compute the alignment score between y and its hyponym x_i with function $f(x_i, y) = \vec{u}_y^T \vec{v}_{x_i} + b_y$, which reflects the attention of x_i on y. A softmax function is then used to convert the alignment scores into a probability distribution $p(x_i|\mathcal{X}, y)$. A small $p(x_i|\mathcal{X}, y)$ means that the i-th hyponym is likely to be polysemous or noise for the hypernym y. This process can be formulated as

$$p(x_i|\mathcal{X}, y) = \text{softmax}(h), h = [f(x_i, y)]_{i=1}^m. \quad (4)$$

Thus, by combining Eq. (4) and $loss_h$, we obtain the final loss function for the relation *hypo*(x, y) as follows:

$$loss_{ha} = p(x|\mathcal{X}, y) \cdot loss_h. \quad (5)$$

LTE with Sibling Relation

To model *sib*(t_1, t_2), we expect their embeddings to be similar, i.e., $\vec{u}_{t_1} \simeq \vec{u}_{t_2}$ and $\vec{v}_{t_1} \simeq \vec{v}_{t_2}$. Dot-products, i.e., $\vec{u}_{t_1}^T \cdot \vec{u}_{t_2}$ and $\vec{v}_{t_1}^T \cdot \vec{v}_{t_2}$, are the popular ways of measuring similarity between hyponym/hypernym embeddings. Thus, we define the joint probability between terms t_1 and t_2 to model the relation *sib*(t_1, t_2) as follows:

$$p(t_1, t_2) = \frac{1}{1 + \exp(-d(t_1, t_2))}, \quad (6)$$

and

$$d(t_1, t_2) = \alpha \vec{u}_{t_1}^T \cdot \vec{u}_{t_2} + (1-\alpha) \vec{v}_{t_1}^T \cdot \vec{v}_{t_2}, \quad (7)$$

where α is a hyperparameter used to balance the similarity of hyponym and hypernym embeddings. For each *sib*(t_1,t_2), we randomly sample multiple negative sibling relations to design the loss as follows:

$$loss_s = -\log p(t_1, t_2) - \sum_{(t_1', t_2')} \log(1 - p(t_1', t_2')), \quad (8)$$

where $(t_1', t_2') \in S_n(t_1, t_2) = \{(t_1', t_2)\} \cup \{(t_1, t_2')\}$, $S_n(t_1, t_2)$ is the set of negative sibling relations, $t_1' \neq t_1$ and $t_2' \neq t_2$ are randomly sampled from all terms, and n is the number of negative sibling relations.

Acquisition of Sibling Relation

Unfortunately, there are no positive sibling relations to train Eq. (8) since these relations are not explicitly mentioned in an LT. Thus, we need a method to tell us which term pair in an LT are siblings.

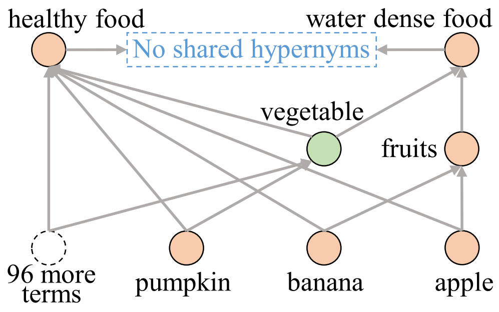


Figure 2: The two terms `healthy food` and `water dense food` have no common hypernyms and only one common hyponym, but they share many indirect neighbors in Probase.

Actually, the model with Eq. (5) can learn the similarity between hypernyms $\vec{u}_{t_1}$ and $\vec{u}_{t_2}$ by sharing hyponyms. However, this similarity is not sufficient. There are two reasons: 1) Two terms sharing many hypernyms may also be similar. 2) Similarity measurement should not only focus on the term's direct neighbors (its hyponyms and hypernyms), but also indirect neighbors (two or more steps away) due to the data sparsity of LTs. As shown in Figure 2, although `healthy food` and `water dense food` have only one common hyponym and no shared hypernyms, they are similar to each other since they share many indirect neighbors. To solve this problem, a random walk based metric is employed to learn more sufficient similarity between terms (Lovász et al. 1993). The core idea is that the two terms t_1 and t_2 are similar to each other if they have a similar probability to other words in an LT with a random walk process.

More specifically, let $|\mathcal{T}|$ be the number of terms in an LT, we construct a vector $\vec{s}_t \in \mathbb{R}^{2|\mathcal{T}|}$ for each term $t \in \mathcal{T}$ by concatenating two random walk vectors $\vec{s}_{te} \in \mathbb{R}^{|\mathcal{T}|}$ and $\vec{s}_{to} \in \mathbb{R}^{|\mathcal{T}|}$. $\vec{s}_{te}$ ($\vec{s}_{to}$) is the probability distribution over the $\mathcal{T}$ terms along the edge from hyponyms (hypernyms) to hypernyms (hyponyms). Each element in $\vec{s}_{te}$ and $\vec{s}_{to}$ is the probability of reaching each element from term t along with two directions by a random walk process. After the i-th step of random walk, $\vec{s}_{te}$ and $\vec{s}_{to}$ can be expressed as

$$\begin{aligned} \vec{s}_{te}^{(i)} &= \vec{s}_{te}^{(0)} + M\vec{s}_{te}^{(i-1)}, \\ \vec{s}_{to}^{(i)} &= \vec{s}_{to}^{(0)} + M\vec{s}_{to}^{(i-1)}, \end{aligned} \tag{9}$$

where $M \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}$ is the column-normalized adjacency matrix. $\vec{s}_{te}^{(0)}$ and $\vec{s}_{to}^{(0)}$ are the initial one-hot vector (only the dimension corresponding to t is set to 1 and the rest elements are set to 0). In general, the results of $\vec{s}_{te}^{(\infty)}$ and $\vec{s}_{to}^{(\infty)}$ are a stationary distribution (Lovász et al. 1993). However, the cost of each iteration in a random walk procedure is expensive since LTs always have millions of terms and hyponym-hypernym relations. Thus, we speed up the computation by limiting the number of iterations. Specifically, with acceptable experimental precision, we set the number of iterations to 2.

To obtain a similar score for a pair of terms, we need to compare the stationary distribution $\vec{s}_{t_1}$ of term t_1 with the stationary distribution $\vec{s}_{t_2}$ of term t_2 both generated by a random walk procedure. In this paper, we adopt the cosine function to measure the similarity (or divergence) from a pair of distributions:

$$sim(t_1, t_2) = \frac{\vec{s}_{t_1} \cdot \vec{s}_{t_2}}{\|\vec{s}_{t_1}\|\|\vec{s}_{t_2}\|}. \tag{10}$$

Finally, to obtain positive samples needed for modeling sibling relations, we first calculate the cosine similarity of all pairs of terms in an LT. Then, we choose the pairs of terms with the similarity greater than the threshold ε as the positive samples.

Training Method

To model hyponym-hypernym relations from hyponym to hypernym and sibling relations, we adopt an alternating training strategy to combine the loss functions $loss_{ha}$ and $loss_s$. That is, we perform $loss_{ha}$ training and $loss_s$ training in turn. Similarly, we employ the same training method as above to model hyponym-hypernym relations from hypernym to hyponym and sibling relations with another set of parameters.

According to the learning principle of human beings in the cognitive process, we should start with simple and typical samples when learning a new model and then gradually consider more complex samples. To this end, we employ curriculum learning algorithm (Bengio et al. 2009) to determine the training order of sibling relations and hyponym-hypernym relations in advance. For the curriculum of sibling relations, we rank sibling relations in descending order of their similarity calculated by Eq. (10). In practice, the similarity score between two terms is used to measure their likelihood being siblings. Thus, through the similarity ranking, we hope that the model first learns the representation of siblings with high probabilities, which is then used to guide the model to learn the ones with relatively low probabilities. For the curriculum of hyponym-hypernym relations, we sort the samples in the descending order according to the following measurement

$$C(x, y) = p_c(x|y) \cdot p_c(y|x). \tag{11}$$

Eq. (11) actually measures the typicality of *hypo*(x,y) through the typicality of terms x and y defined in Eq. (1). A large $C(x, y)$ means that the relation *hypo*(x,y) tends to be an easy-to-learn sample.

Experiments

In this section, we conduct extensive experiments to evaluate our proposed models on two tasks: link prediction and hyponym-hypernym relation classification. Furthermore, we use the term embeddings learned by our model to enhance the performance of the models BERT and RoBERTa on text classification.

Datasets

Most previous knowledge graph embedding work used FB15K and WN18 (Bordes et al. 2013) for evaluation.

Datasets	# Train	# Valid or Test	# Hyponym	# Hypernym
			Max / Avg	Max / Avg
Pro5K	153,145	19,143	2,596 / 56	437 / 41
Pro300K	1,090,811	128,221	18,650 / 8	1,464 / 8

Table 1. The statistics of Pro5K and Pro300K. Among all hypernyms (hyponyms), "Max" and "Avg" are the maximum and average number of its hyponyms (hypernyms), respectively.

However, these two datasets are not suitable for our problem since the former is mainly composed of *non-hyponym-hypernym* relations (e.g., `/film/film/music`) and the latter is manually constructed and all terms in WordNet are disambiguated. Thus, we use Probase for evaluation, which is one of the largest LTs. To ensure the accuracy of the inputs, we filter the low-frequency relations with $n(x, y) \leq 5$. Then, similar to TransE (Bordes et al. 2013), we create two datasets constructed by selecting relations and the frequency of terms in these relations needs to be ranked in Top-5K and Top-300K in Probase. These two datasets are denoted as Pro5K and Pro300K. The statistics of these two datasets are shown in Table 1.

Experiment Setup

Baselines. We compare our proposed methods with three kinds of embedding models as follows.

- **Translation-based models.** The typical translation-based models mainly include TransE (Bordes et al. 2013), TransH (Wang et al. 2014), TransR (Lin et al. 2015), and RotatE (Sun et al. 2019).
- **Semantic matching models.** The representative semantic matching models include DistMult (Yang et al. 2014), HolE (Nickel, Rosasco, and Poggio 2016), and ComplEx (Trouillon et al. 2016).
- **Encoding hierarchical structure models.** The typical models used for taxonomy embedding learning mainly include DDMM (Yu et al. 2015) and On2Vec (Chen et al. 2018).

Parameter Setting. For the experiment with our model, we set $\alpha = 0.9$ and $\varepsilon = 0.5$, respectively. For the joint model, We select the learning rate $\lambda = 0.001$ for Adam. The number of negative samples n in both Eq. (3) and (8) is set to 5. The dimension of term vector d in this paper and other compared methods is set to 128 and the vector of each term t is normalized by setting $||\vec{u}_t||_2 = 1$ and $||\vec{v}_t||_2 = 1$. During alternating training, we perform $loss_{ha}$ with 4 epochs and $loss_s$ with 1 epoch in turn and our models run on Windows 10 with Intel(R) Core(TM) i7-4790K CPU, GeForce GTX 980 and 32GB of RAM. The computation time of alternating training for one iteration is about $\frac{2}{3}$ and 30 minutes for Pro5K and Pro300K, respectively.

Link Prediction

Link prediction is to predict the missing hyponyms or hypernyms for hyponym-hypernym relations. For each relation *hypo*(x,y), we first remove x or y and replace it with all terms in an LT in turn. Then, we rank these terms in descending order according to the value of the potential function and record the ranking of x or y. We call this setting as "Raw". In fact, the relations whose hyponyms or hypernyms replaced by other terms may also exist in LT and these relations should be considered as correct predictions. Thus, we remove these relations and record the ranking of x or y. This setting is called "Filter".

Similar to TransE (Bordes et al. 2013), two evaluation metrics are reported: the mean rank of all correct terms (denoted as MR) and the proportion of correct terms in the Top-10 (denoted as Hits@10). A good embedding model should achieve lower MR and higher Hits@10.

The experimental results are shown in Table 2. From the table, we conclude that: 1) Our methods outperform most competitors under almost all metrics, which verifies the effectiveness of our method. 2) In general, the performance of the models on hyponym predicting is worse than that on hypernym predicting. The reason may be that for a given term, in general, the number of its hyponyms is more than the number of its hypernyms, as shown in Table 1. For example, given a hypernym y, if it has only one hyponym x, the embedding of y can accurately capture the semantics of x. If it has two hyponyms x_1 and x_2, the embedding of y will capture the semantics of both x_1 and x_2 so that y cannot accurately reflect the semantics of a single x_1 or x_2. Thus, when predicting x for a given y, the effectiveness of models will decrease as the number of hyponyms increases.

Hyponym-hypernym Relation Classification

Hyponym-hypernym relation classification is to judge whether a given *hypo*(x,y) is correct or not, which is a binary classification task. As to our datasets, the validation and test sets only have positive relations, which requires us to provide wrong relations. To solve this problem, we construct negative samples by randomly selecting terms from all terms in LT to replace hyponyms or hypernyms.

For relation classification, we set a threshold δ. For each *hypo*(x,y), if its potential function is larger than δ, *hypo*(x,y) will be classified as positive, otherwise negative. δ is optimized by maximizing the accuracy on the valid set.

The results are reported in Table 3. From the table, we conclude that: 1) Compared to other embedding methods, our model achieves the best performance on both datasets. The highest accuracy is close to 90% on Pro5K. 2) Although the data is expanded from 5K to 300K, the accuracy drops only about 1%, which indicates that our method is also applicable to large-scale LTs.

Text Classification

To further verify the effectiveness of our method, we use the learned term embeddings from Pro5K as additional features to enhance the performance of existing text classification models.

In this paper, we employ a standard text classification dataset AG's News (Zhang, Zhao, and LeCun 2015) for the evaluation, which contains 496,835 news articles about 4 classes. Based on this dataset, a Deep Neural Networks (DNN) with 3 layers (the hidden sizes are set to 768,

Task	Hypernym Predicting						Hyponym Predicting					
Datasets	Pro5K				Pro300K		Pro5K				Pro300K	
Metric	MR		Hits@10 (%)		MR	Hits@10	MR		Hits@10		MR	Hits@10
Eval. setting	*Raw*	*Filt.*	*Raw*	*Filt.*	*Filt.*	*Filt.*	*Raw*	*Filt.*	*Raw*	*Filt.*	*Filt.*	*Filt.*
TransE	286	247	8.7	16.5	9,959	10.7	847	605	3.7	10.5	15,275	10.1
TransH	285	247	8.6	16.9	11,049	9.4	844	602	3.4	10.6	16,225	7.0
TransR	290	252	8.8	16.3	11,576	9.4	852	610	3.7	10.8	16,888	6.9
RotatE	207	159	5.5	**38.6**	21,631	20.5	638	345	0.8	22.2	24,269	9.2
HolE	393	335	3.4	19.5	21,907	10.0	943	558	2.3	13.5	24,737	6.3
DistMult	335	276	2.7	18.2	11,558	15.7	858	454	1.7	11.1	14,459	10.3
ComplEx	301	237	10.6	22.7	11,588	15.8	781	365	1.1	19.8	14,480	10.4
DDMM	696	692	**18.8**	18.9	30,376	2.8	986	981	4.4	4.5	38,499	0.7
On2Vec	857	848	0.7	0.8	19,701	0.3	1,676	1,655	0.8	0.9	38,083	2.2
Ours (H)	212	152	3.6	34.6	10,435	21.2	754	448	3.6	18.9	13,243	8.8
Ours (HA)	188	145	4.4	36.1	10,069	21.8	688	383	4.5	20.0	12,729	10.1
Ours (HAS)	176	135	14.9	38.0	9,359	22.5	594	302	5.6	22.8	11,909	11.4
Ours (HASC)	**171**	**130**	15.2	38.2	**9,297**	**23.4**	**571**	**300**	**6.0**	**25.8**	**11,862**	**12.6**

Table 2. The results on link prediction. “Ours (H)”, “Ours (HA)”, and “Ours (HAS)” mean the models with the loss $loss_h$, $loss_{ha}$, and the combination of $loss_{ha}$ and $loss_s$, respectively. “Ours (HASC)” means the joint model trained by ranked samples with Eq. (10) and (11).

Datasets	Pro5K	Pro300K
TransE	76.1	84.3
TransH	76.2	84.0
TransR	75.9	83.5
RotatE	86.5	81.5
HolE	78.8	80.9
DistMult	81.4	87.9
ComplEx	83.4	87.6
DDMM	83.1	73.2
On2Vec	79.8	76.3
Ours (H)	85.3	84.4
Ours (HA)	86.7	85.6
Ours (HAS)	89.1	88.4
Ours (HASC)	**89.9**	**89.2**

Table 3. The accuracy (%) of different models on relation classification.

1024, and 4) is trained as a classifier, where the input is the distributed representation of an article. To obtain textual representations, we adopt two strategies: 1) encoded by BERT[5]$_{BASE}$ (Devlin et al. 2018) and 2) encoded by RoBERTa[6]$_{BASE}$ (Liu et al. 2019). For these two situations, we use the dataset AG’s News to fine-tune the model and consider the encoding of the token `CLS` as textual representations. The dimensions of text vectors produced by these two strategies are set to 768.

To evaluate the effectiveness of term embeddings learned by our method, we employ a feature fusion-based method to incorporate the learned term vectors into the article vectors generated by the above two methods, respectively. Specifically, for each article, we first find the terms that exist in Pro5K through string matching. Then, a new vector is obtained by the average of the corresponding elements of learned embeddings for all terms. Finally, we concatenate the new vector and the article vector as the input of the classifier and thus the hidden sizes of DNN are set to 896 (768 + 128), 1024 and 4.

Input	BERT	RoBERTa
Basic	92.0	93.4
+ TransE	92.7	93.5
+ RotatE	93.0	94.1
+ DistMult	92.2	94.1
+ Ours (HASC)	93.4	94.1

Table 4. The accuracy (%) of different models on text classification. “Basic” means the input of the classifier is the article vector obtained by methods (BERT and RoBERTa). “+ TransE”, “+ RotatE”, “+ DistMult”, and “+ Ours (HASC)” mean the input of the classifier is the concatenation of the article vector and the new vector learned by TransE, RotatE, DistMult and Ours (HASC), respectively.

In this task, we compare several typical methods and the results are reported in Table 4. From the results, we conclude that: 1) The term embeddings learned by the methods including TransE, RotatE, DisMult and Ours (HASC) can be used to enhance the performance of BERT and RoBERTa on the text classification task. This shows that adding the term embeddings learned from LT on the basis of BERT and RoBERTa can improve the performance of the text classification task. 2) Compared with TransE, RotatE, and Dis-

[5]https://github.com/google-research/bert

[6]https://github.com/pytorch/fairseq/blob/master/examples/roberta/README.md

hyponym	hypernym	Ours (H)	Ours (HA)
scientist	person	8.35	7.25
scientist	profession	6.68	4.99
photograph	**figure**	1.92	1.85
plato	**figure**	10.89	4.63

Table 5. The results of $\psi(\cdot|\cdot)$ with models "Ours (H)" and "Ours (HA)". Terms in bold are polysemous.

Term	Siblings
powerpoint	excel, graphic, word, browser, ...
petrol	petroleum, natural gas, crude oil, ...
lavender	ginger, sunflower, turmeric, basil, ...
designer	engineer, teacher, lawyer, analyst, ...
white	purple, yellow, almond, green, ...

Table 6. Examples of terms' siblings.

Mult, adding the vector learned by our method can significantly improve the accuracy of the model, which proves the effectiveness of our method. 3) The methods RotatE, DisMult, and ours have the same improvement on the basis of RoBERTa. A possible reason is that RoBERTa is trained on a very large corpus and it is difficult to further improve by adding 5K terms.

Case Study

Next, we present some examples to analyze the effectiveness of the term-level attention mechanism and the random walk based metric.

Table 5 gives the potential function values of several relations whose hyponym or hypernym is polysemous with two models including "Ours (H)" and "Ours (HA)". From the table, we can see that the potential function value with the model "Ours (HA)" is lower than that with "Ours (H)", which verifies the effectiveness of our designed term-level attention mechanism. Because a decrease in the value of the potential function indicates that the semantic correlation of these relations is weakened. In other words, the semantic information that the model learns from the polysemous terms decreases.

Table 6 gives some examples of finding siblings using the metric based on the random walk for a given term. From the results, we can see that terms and their siblings are semantically similar, which verifies the effectiveness of the random walk based metric.

Related Work

KG Embedding. Recently, KG embedding has attracted a great deal of research interest. Embedding methods for KG can be divided into two groups: translation-based models and semantic matching models. The core idea of the former is to measure the plausibility of facts with a distance-based score function. The typical methods along this line include TransE (Bordes et al. 2013), TransR (Lin et al. 2015), and RotatE (Sun et al. 2019). The latter try to measure the likelihood of facts through a similarity-based scoring function and its representative work includes DistMult (Yang et al. 2014), HolE (Nickel, Rosasco, and Poggio 2016), and ComplEx (Trouillon et al. 2016). However, the above methods are hard to apply to LTs since they mainly focus on multi-relational graphs that are very different from LTs in structures. That is, LTs have only one relation, i.e., hyponym-hypernym relation, and each term in LTs usually has thousands of hyponyms or hypernyms.

Taxonomy Embedding. The goal of taxonomy embedding is to learn term representations in hyponym-hypernym relations. Embedding methods for taxonomies can be roughly classified into two categories. The first is to model the hierarchical structure of taxonomy in the embedding space with text corpora (Nguyen et al. 2017). A dynamic weighting neural network (Luu et al. 2016) is proposed to learn term embeddings based on the hyponym-hypernym relation and contextual information. (Wang, He, and Zhou 2019) proposes a taxonomy enhanced adversarial learning model to project a term to its hypernyms and non-hypernyms with text corpora. The second is to learn term embeddings only from taxonomies (Nickel and Kiela 2017), which is also the focus of this paper. DDMM (Yu et al. 2015) uses a neural network to encode hyponym-hypernym relations into term embeddings. On2Vec (Chen et al. 2018) is a hierarchy model that performs learning of hyponym-hypernym relations. TransC (Lv et al. 2018) encodes each concept as a sphere and each instance as a vector to learn the hierarchical structure of taxonomy. However, the above methods of learning term embeddings from taxonomies are not suitable for LTs since they do not take into account polysemous terms and ignore the implicit but important information, i.e., sibling relations.

Conclusion and Future Work

In this paper, we focus on the embedding learning of LTs. We first design a joint model to reflect hyponym-hypernym and sibling relations. Then, to assist in modeling these two relations, a term-level attention mechanism and a random walk based metric are proposed. Besides, we design a novel training method based on curriculum learning. Finally, extensive experiments are conducted to verify the effectiveness of our model.

In the future, we try to integrate the information (e.g., *entity-attribute-value*) in KG into our model to learn more effective representation for each term, thereby further improving the performance of downstream tasks. For example, the entity `Kobe Bryant` has an attribute-value pair `jersey number-24`, which is also an effective feature for the text classification task.

Acknowledgments

This paper was supported by National Key Research and Development Project (No. 2020AAA0109302) and Shanghai Science and Technology Innovation Action Plan (No.19511120400).

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48. ACM.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250. AcM.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Chen, M.; Tian, Y.; Chen, X.; Xue, Z.; and Zaniolo, C. 2018. On2vec: Embedding-based relation prediction for ontology population. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, 315–323. SIAM.

Cui, W.; Xiao, Y.; Wang, H.; Song, Y.; Hwang, S.-w.; and Wang, W. 2017. KBQA: learning question answering over QA corpora and knowledge bases. *Proceedings of the VLDB Endowment* 10(5): 565–576.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2181–2187.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

Lovász, L.; et al. 1993. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty* 2(1): 1–46.

Luu, A. T.; Tay, Y.; Hui, S. C.; and Ng, S. K. 2016. Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 403–413.

Lv, X.; Hou, L.; Li, J.; and Liu, Z. 2018. Differentiating concepts and instances for knowledge graph embedding. *arXiv preprint arXiv:1811.04588* .

Mazur, D.; Egiazarian, V.; Morozov, S.; and Babenko, A. 2019. Beyond Vector Spaces: Compact Data Representation as Differentiable Weighted Graphs. In *Advances in Neural Information Processing Systems*, 6903–6913.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11): 39–41.

Nguyen, K. A.; Köper, M.; Walde, S. S. i.; and Vu, N. T. 2017. Hierarchical embeddings for hypernymy detection and directionality. *arXiv preprint arXiv:1707.07273* .

Nickel, M.; and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, 6338–6347.

Nickel, M.; Rosasco, L.; and Poggio, T. 2016. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*, volume 30, 1955–1961.

Sang, E. T. K. 2007. Extracting hypernym pairs from the web. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, 165–168.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* .

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, 2071–2080.

Wang, C.; He, X.; and Zhou, A. 2019. Improving Hypernymy Prediction via Taxonomy Enhanced Adversarial Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7128–7135.

Wang, X.; Wang, D.; Xu, C.; He, X.; Cao, Y.; and Chua, T.-S. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5329–5336.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, volume 28, 1112–1119.

Wu, W.; Li, H.; Wang, H.; and Zhu, K. 2011. Towards a probabilistic taxonomy of many concepts. *Microsoft Res. Redmond, WA, USA, Tech. Rep. MSR-TR-2011-25* .

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* .

Yu, Z.; Wang, H.; Lin, X.; and Wang, M. 2015. Learning term embeddings for hypernymy identification. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 1390–1397.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.