A Flexible Translation-Based Knowledge Graph Embedding Adapting Unobserved Entities

A-Yeong Kim, Seong-Bae Park and Sang-Jo Lee *

School of Computer Science and Engineering
Kyungpook National University
Daegu, 41566, Korea
{aykim, sbpark}@sejong.knu.ac.kr, sjlee@knu.ac.kr

Abstract

This paper proposes a flexible translation-based knowledge graph embedding that learns unobserved entities by moving positions of embedding vectors from existed embedding space. To reflect unobserved entities, previous methods tend to learn knowledge graphs all over again. This process causes high cost of calculation. Thus, this paper introduces an adjusting method which moves positions of learned embedding vectors according to unobserved entity. This idea is based on TransE model that is a one of translation-based methods. According to experiments, the proposed method shows the plausibility at link prediction task and triple classification task. These experimental results prove that reducing learning cost is a crucial issue for embedding knowledge graphs.

Keywords: Knowledge Graph Embedding, Adaptive Learning, Translation-Based Embedding, Knowledge Graph Completion, Link Prediction, Triple Classification

1. Introduction

To represent knowledge base as a graph is one of the powerful ways to utilize various large-scale knowledge graph such as Freebase [1], Wordnet [2] and Yago [3] which are available these days. Knowledge graph has been played a significant role in many AI related tasks, such as question answering, web search and so on. A typical knowledge graph easily covers a large amount of entities and their relations as the form of triple (head entity, relation, tail entity) (denoted as (h, r, t)). However, the traditional method for building knowledge graph had difficulty in aggregating large scale knowledge graphs. To resolve this problem, many embedding approaches have been proposed [4-7]. The advantage of knowledge graph embedding is that it can preserve certain properties of knowledge graph by embedding the knowledge graphs in a low dimensional continuous vector space. This method can simply measure the graph with algebraic operations in the vector space. Because of its advantages, knowledge graph embedding has received a lot attention.

Among the various knowledge embedding methods, the translation-based embedding model achieves outstanding performance in knowledge graph completion [8-11]. TransE [8], very simple and effective method, is a well-known approach to completion problem. The main idea of TransE is that relationships are represented as translations in the embedding space. For knowledge triple (h, r, t) composed of two entities (h and t) and relation (r), the embedding of the tail entity t should be close to the sum of the embedding of the head entity h and relation r. Since TransE embeds all relations in a single vector space, it can not cover any multiple relations such as 1-to-N,
N-to-1 and N-to-N relations. To resolve these issues, some proposed models assumed each relation has its own embedding space [9-11].

These translation-based embedding models show high performance in knowledge graph completion but there is still room for improvement. That is, model complexity in learning unobserved entities. If an unobserved entity is given, it can not apply to pre-trained embedding model directly. To learn unobserved entity for completing knowledge graph, existed models learn again the whole of knowledge graph including unobserved entities. Although translation-based embedding model has relatively lower model complexity than other embedding methods, they can encounter high model complexity in this situation that can not be ignored. As the expended knowledge graph, learning process will be more inefficient. Therefore, the solution of reducing process step for learning knowledge graph is needed.

In this paper, we propose an adaptive learning algorithm to reflect the situation that a new entity is observed. This algorithm projects a new entity onto the pre-trained embedding space and moves the position of existed embedding vector simultaneously. For instance, let assume that pre-trained embedding vectors and a new entity are given. If an embedding vector of new entity is determined, pre-trained embedding vectors tend to adjust according to a new entity. The pre-trained embedding vectors may be mixed harmoniously with a new entity at learning process. The plausibility of this idea is verified with two experiments by using standard benchmark datasets. We can believe that the partially learned embedding space with adaptive learning approach is similar to the embedding space of general translation-based knowledge graph embedding model.

2. Related Researches

With the sparsity of knowledge graph becoming one of the critical issues, a number of studies have been focused on solving sparsity issue. Many studies on completing knowledge graphs tried to predict new relations between entities on a knowledge graph from existing relations of the graph. It calls link prediction and there are three typical approaches for this task. One is based on graph features which are observable features composed of the paths between entity pairs [14, 15] and subgraphs [16]. Another approach is based on Markov random fields. Some studies by this approach inference new relations from probabilistic soft logic [17] and first-order logic [18]. The other is knowledge graph embedding which is promising approach recently [19].

Knowledge graph embedding is the method that embeds entities and relations of knowledge graph into a continuous low dimensional vectors. Entities and relations vectors are optimized by a score function of embedding model. Bordes et. al., [19] proposed Semantic Matching Energy (SME) model which learns vector representations of entities and relations by using neural network. When a triple (h, r, t) is given, SME makes relation-dependent embeddings for pairs (h, r) and (r, t). The similarity between the embedding vectors is used for score function of a triple. The complexity of SME is relatively lower than other embedding methods since relations are represented as vectors. Jenatton et. al., [20] proposed Latent Factor Model (LFM) to capture multiple order interactions between two entities. A relation is encoded as bilinear operators on the entities and a weighted sum of latent factors to learning a large number of relations. Socher et. al., [4] suggested Neural Tensor Networks (NTN) model. In this model, the standard linear neural network layer is replaced by bilinear tensor layer. NTN model can process interactions of entity vectors via a tensor. Even though NTN model obtains the high accuracy for link prediction, it is difficult to process large-scale knowledge graphs because of its high complexity.
The model complexity of knowledge graph is one of the critical issues in utilizing knowledge graphs. To overcome this issue, translation-based embedding approach is proposed. As the result, it becomes a primary trend of knowledge graph recently. The basic idea of this approach is that every entities and relations are represented as vectors and relations are considered as translation in the embedding space. Thus, it finds vector representation of entities and relations so that the sum of head entity vector \( h \) and relation vector \( r \) becomes as similar as tail entity vector \( t \). TransE [8] is the first study based on translation-based graph embedding. However, it ignores the fact that entities can have multiple aspects and thus relations should be represented differently according to the aspects of entities.

Starting from TransH [9], some translation-based embedding methods [10, 11] try to cover the drawbacks of TransE. TransH [9] allows entities to play different roles according to relations. It projects an entity vector into relation-specific hyperplanes to have multiple representations of entity. TransR [10] also considers having multiple relations by mapping an entity vector into relation-specific spaces. Ji et al., proposed TransD that allows a relation to have multiple relation spaces according to its entities [11]. Every relation in TransD has multiple entity-specific spaces by constructing relation mapping matrices dynamically. As the result, TransD is able to handle multiple types of relations, but TransH and TransR cannot do it.

3. A Translation-Based Knowledge Graph Embedding Adapting Unobserved Entities

3.1. TransE

As mentioned in previous sections, translation-based embedding wants to find vector representations of knowledge graph when triples \((h, r, t)\). Figure 1 depicts the simple illustration of TransE. TransE models a relation \( r \) as a translation vector \( r \in \mathbb{R}^d \) and relation is modeled as an operation in the space. The score function of TransE is
where two entities $h, t \in E$, and $r \in R$ are the vectors of a head entity, tail entity, and their relation on a single embedding space. $E \subseteq \mathbb{R}^k$ is the set of entities and $R \subseteq \mathbb{R}^k$ is the set of relations. This score function is expected to be low value for a positive triple and high value for an incorrect triple.

To learn embeddings, TransE minimize a margin-based ranking criterion over the training set defined as

$$L = \sum_{(h,r,t) \in \mathcal{P}(h,r,t)} \sum_{(h',r,t') \in \mathcal{P}'(h,r,t)} [y + d(h + r, t) - d(h' + r, t')]_+$$

where $[x]_+$ denotes the positive triple of $x$, $y > 0$ is a margin hyperparameter, and

$$\mathcal{P}'(h,r,t) = \{(h', r, t') | h' \in E\} \cup \{(h, r, t') | t' \in E\}.$$

If the entity appears as the head or as the tail of a triple, its embedding vector is the same in this case. The optimization is performed by stochastic gradient descent in minibatch mode with additional constraints that the $L_2$-norm of the embeddings of the entities. This constraint prevents the training process to minimize $L$. 

**Figure 2. Overall process of adTransE**
Applying New Entities to Pre-trained Embedding Space

To apply new entities, we propose adjusting TransE (adTransE) which adjust pre-trained embedding vectors to new entities [14]. Figure 2 depicts the process of proposed model. The step of process is described as below. First, existing knowledge graph is embedded into low-dimensional embedding vector space same as the embedding process of TransE. Second, when a new entity appears, a new entity is embedded into existed embedding space. At that time, pre-trained embedding vectors slightly move their position according to a vector of new entity. These two steps would be repeated whenever new entities appear.

This process can expand knowledge graph fast and easily then other methods. To reduce the time complexity the proposed method randomly samples 10 percent of new data. For instance, let assume that there are 10 million triples from training set and 1 million triples from new entities. In case of TransE, this model totally calculates 21 million times for training triples. This calculation time comes from 10 million times for triples from training set and 11 million times from training set and new entities. The training calculation time of TransE depends on the size of training set. On the other hand, the calculation time of proposed model is increased about only 1 percent on the same condition as TransE.

Algorithm 1. Learning adTransE

```plaintext
input Training set \( S = \{(h,r,t)\} \), entities and relation sets \( E \) and \( R \), margin \( \gamma \), embeddings dimension \( k \).

1: initialize \( r, e \leftarrow \text{uniform}\left(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right) \) for each \( r \in R \), each entity \( e \in E \)
2: \( r \leftarrow r/\|r\| \) for each \( r \in R \)
3: loop
4: \( e \leftarrow e/\|e\| \) for each \( e \in E \)
5: \( S_{\text{batch}} \leftarrow \text{sample}(S, b, o) \) // sample a minibatch of size \( b \) with ratio \( o \)
6: \( T_{\text{batch}} \leftarrow \emptyset \) // initialize the set of pairs of triplets
7: for \((h, r, t) \in S_{\text{batch}}\) do
8: \( (h', r, t') \leftarrow \text{sample}(S_{(h, r, t)}) \) // sample a corrupted triple
9: \( T_{\text{batch}} \leftarrow T_{\text{batch}} \cup \{(h, r, t), (h', r, t')\} \)
10: end for
11: \[ \sum_{((h, r, t), (h', r, t')) \in T_{\text{batch}}} \nabla[\gamma + d(h + r, t) - d(h' + r, t')]_+ \]
12: end loop
```

3.2. Applying New Entities to Pre-trained Embedding Space

To apply new entities, we propose adjusting TransE (adTransE) which adjust pre-trained embedding vectors to new entities [14]. Figure 2 depicts the process of proposed model. The step of process is described as below. First, existing knowledge graph is embedded into low-dimensional embedding vector space same as the embedding process of TransE. Second, when a new entity appears, a new entity is embedded into existed embedding space. At that time, pre-trained embedding vectors slightly move their position according to a vector of new entity. These two steps would be repeated whenever new entities appear.

This process can expand knowledge graph fast and easily then other methods. To reduce the time complexity the proposed method randomly samples 10 percent of new data. For instance, let assume that there are 10 million triples from training set and 1 million triples from new entities. In case of TransE, this model totally calculates 21 million times for training triples. This calculation time comes from 10 million times for triples from training set and 11 million times from training set and new entities. The training calculation time of TransE depends on the size of training set. On the other hand, the calculation time of proposed model is increased about only 1 percent on the same condition as TransE.
The detailed optimization procedure is described in Algorithm 1. All embeddings for entities and relations are initialized following the procedure proposed in [8, 13]. First, the embeddings vectors of the entities are normalized at iteration of the algorithm. Then a small set of triples is sampled from existed entities (the training sets) and new entities with ratio. The ratio of existed entities and new entities is randomly determined. A set of triples will serve as the training triples of the minibatch. For each triple, we sample a corrupted triple from minibatch. The parameters are updated by taking a gradient step with constant learning rate. This algorithm is stopped based on its performance on a validation set.

4. Experiments

In this paper, the plausibility of proposed model is shown through two kinds of tasks, link prediction [8] and triple classification [4]. Link prediction task is that completing a triple when an entity is missing. For instance, model predicts head entity when tail entity and relation is given or predicts tail entity when head entity and relation is given. In this task, the rank a set of candidate entities is deducted by a system. Triple classification task is that confirming whether a given triple is correct or not. It is used for evaluating NTN model.

We will compare with some related work [3,4,7,8,17,19,20], including the TransE, and noadTransE (without adjusting process) model. The noadTransE is a baseline model of the proposed model. The difference between noadTransE model and the proposed model is whether trained entity vector is fixed in vector space. The baseline model also can adopt new entities without moving the positions of existed embedding vector when it trains the data. On the other hand, the proposed model changes the existed entity vectors when new entities come. By doing this, the proposed model can adjust new entities and expand knowledge graph.

4.1. Datasets

For evaluating the proposed model, two typical knowledge graphs (WordNet and Freebase) are employed. WordNet is a lexical database that provides semantic knowledge of words. In WordNet, words are grouped by sets of synonym (synsets) that are connected to other synsets by semantic relations such as hypernym, hyponym, meronym and holonym. Freebase is a knowledge base that represents general facts from online databases such as Wikipedia, NNDB and so on. For example, the triple (Larry Page, founded, Google Inc.) builds a relation of founded between the name entity Larry Page and the organization entity Google Inc.

In this paper, two data sets are used from WordNet and Freebase respectively. Bordes et al. [8] used WN18 that contains 18 relation types and 151,442 triples from WordNet and FB15k that composed of 1,345 relation types and 592,213 from Freebase. Socher et al. [4] used WN11 that contains 11 relation types and 125,734
triples from WordNet and FB13 that holds 13 relation types and 345,873 triples from Freebase. The statistics of these data sets are shown as Table 1.

This paper assumed that new entities are presented after building knowledge graph. Thus the setting for new entities is needed on training process. In training process, we randomly split data into 2 groups. The data of group1 is composed of entities which are not showed until training and the data of group2 is the rest. The ratio of new entities is about 20 percent. That is, 20 percent of data is composed with new entities for minibatch.

Table 2. Experimental Results on Link Prediction

<table>
<thead>
<tr>
<th>Datasets</th>
<th>WN18</th>
<th>FB15K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean Rank</td>
<td>Hits@10</td>
</tr>
<tr>
<td></td>
<td>Raw</td>
<td>Filter</td>
</tr>
<tr>
<td>Unstructured [19]</td>
<td>315</td>
<td>304</td>
</tr>
<tr>
<td>RESCAL [3]</td>
<td>1,180</td>
<td>1,163</td>
</tr>
<tr>
<td>SE [7]</td>
<td>1,011</td>
<td>985</td>
</tr>
<tr>
<td>SME (linear) [19]</td>
<td>545</td>
<td>533</td>
</tr>
<tr>
<td>SME (bilinear) [19]</td>
<td>526</td>
<td>509</td>
</tr>
<tr>
<td>LFM [20]</td>
<td>469</td>
<td>456</td>
</tr>
<tr>
<td>TransE [8]</td>
<td>263</td>
<td>251</td>
</tr>
<tr>
<td>oadTransE</td>
<td>576</td>
<td>569</td>
</tr>
<tr>
<td>Proposed Model (adTransE)</td>
<td>549</td>
<td>542</td>
</tr>
</tbody>
</table>

4.2. Link Prediction

Following the previous work of experimental protocols [4, 8, 10, 11], two measures are used for evaluation metric. One is the average rank of all correct entities (Mean Rank) and another is proportion of correct entities in top-10 ranked entities (Hits@10). All of the testing triples are evaluated by these two measures, this setting is “raw” setting. If a corrupted triple exists in knowledge graph, it should be regard as a correct triple. Thus, the corrupted triples which have appeared in knowledge graph may be filtered out. This setting is “Filter” setting. In both settings, a lower Mean Rank or higher Hits@10 mean that the system is a good link predictor. Two datasets WN18 and FB15K are used for evaluating proposed model.

In training proposed model, we used learning rate $\alpha$ for SGD among {0.001, 0.005, 0.01, 0.05}, the margin $\gamma$ among {0.25, 0.5, 1}, the embedding dimension $k$ among {50, 75, 100, 150}, and batch size $B$ among {50, 100, 150}. The optimal parameters are determined by the validation set. We used “bern” which is the strategy of constructing negative labels. The way of “bern” is reducing false negative way by replacing head or tail with different probabilities. Under the “bern” setting, the optimal parameters are $\alpha=0.01, \gamma=1, k=100, B=100$ on both data sets.

The results for link prediction are shown in Table 2. The proposed model fell far behind the models of related work in Mean Rank metrics of both WN18 and FB15k. However, performance of the proposed model gets better in Hits@10 matrix of two datasets. Unlike the case of WN18, the performance of proposed model has come close
to the TransE. Even though proposed model didn’t outperform on every metric from two datasets, we can believe that our proposed model helps improve performance with low model complexity. Especially our proposed model cuts an appearance in Hits@10 from FB15k data. These results mean that our proposed model can facilitate to split knowledge base in several parts with low errors. As shown in Table 2, adTransE outperforms noadTransE. The method with moving existed entities works better than the method without moving existed entities.

Table 3. Experimental Results on Triple Classification

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>WN11</th>
<th>FB13</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>53.0</td>
<td>75.2</td>
</tr>
<tr>
<td>SME (bilinear)</td>
<td>70.0</td>
<td>63.7</td>
</tr>
<tr>
<td>SLM [4]</td>
<td>69.9</td>
<td>85.3</td>
</tr>
<tr>
<td>LFM</td>
<td>73.8</td>
<td>84.3</td>
</tr>
<tr>
<td>NTN [4]</td>
<td>70.4</td>
<td>87.1</td>
</tr>
<tr>
<td>TransE</td>
<td>75.9</td>
<td>81.5</td>
</tr>
<tr>
<td>noadTransE</td>
<td>61.8</td>
<td>77.8</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>63.1</td>
<td>78.6</td>
</tr>
</tbody>
</table>

4.3. Triple Classification

Triple classification is task that judges whether a given triple \((h, r, t)\) is correct or not. This task is a binary classification, which had been explored previous works \([4, 9, 10]\). In this paper, two datasets WN11 and FB13 which contain golden and negative triples to evaluate our proposed model. A threshold \(\delta_r\) for each relation \(r\) is set in this task and the optimized threshold is obtained by validation set. If a classification score of given triple is larger than threshold, it will be classified as positive, otherwise negative.

For training proposed model, we used learning rate \(\alpha\) for SGD among \{0.005, 0.01, 0.05\}, the margin \(\gamma\) among \{0.25, 0.5, 1\}, the embedding dimension \(k\) among \{75, 100, 150\}, and batch size \(B\) among \{50, 100, 150\}. The optimal parameters are determined by the validation set. Under this setting, the optimal parameters are \(\alpha=0.01\), \(\gamma=1\), \(k=100\), \(B=100\) on two data sets.

Our proposed model is compared with previous works which is shown in Table 3. Table 3 shows the accuracies of triple classification on the two datasets. Unlike the previous task, proposed model didn’t affect this task for two tasks. These results imply that our proposed model is not suitable for this task. However, proposed model outperforms method without moving existed vector. That is, moving the position of existed vector represents much to training process for new entities. Especially, the proposed model shows its plausibility by closing the performance of TransE in FB13 dataset.

5. Conclusion

This paper has proposed a flexible translation-based knowledge graph embedding that adjusting new entities. Translation-based knowledge graph embedding is very simple model to embed huge knowledge graphs. Knowledge graphs are quite variable since knowledge can be generated all the time. Because of the scale of knowledge
graphs, model complexity is very important characteristics of expanding knowledge graphs with entities which are not showed in existed knowledge graphs. In order to expanding existed knowledge graph, entity vectors of existed knowledge graph move their positions according to new entity vectors. By doing this, the existed embedding vector adjusts to new entities during training process.

The plausibility of proposed model was shown through two tasks of link prediction and triple classification. The adjusting embeddings showed the improved performance in link prediction task over baseline and some previous models. Especially, proposed model showed the best performance in FB15K dataset. These results imply that the proposed projection is plausible to reflect new entities efficiently. There is still a room for improvement of the proposed model. Like other translation-based knowledge graph embedding models, proposed model can consider mapping matrices according to relation vector.

Acknowledgments

This work was supported by ICT R&D program of MSIP/IITP [B0126-16-1002, Development of smart broadcast service platform based on semantic cluster to build an open-media ecosystem]

References


Authors

A-Yeong Kim, she received M.S. degree from Kyungpook National University, Korea, in 2014. Her current research interests include the Natural Language Processing, Machine Learning, Semantic Web, and etc. She is currently a Ph.D. student in Kyungpook National University, Korea.

Seong-Bae Park, he received Ph.D. degree from Seoul National University, Korea, in 2002. His current research interests include the Machine Learning, Natural Language Processing, Text Mining, Information Extraction, and Bioinformatics. He is currently a professor at School of Computer Science and Engineering, Kyungpook National University, Korea.

Sang-Jo Lee, he received Ph.D. degree from Seoul National University, Korea. His current research interests include the Natural Language Processing, Information Extraction, Information Retrieval, Machine Learning, Ontology, and Semantic Web. He is currently a professor at School of Computer Science and Engineering, Kyungpook National University, Korea.