PAPER

Modeling Complex Relationship Paths for Knowledge Graph Completion

Ping ZENG[†,††], ***Student Member***, **Qingping TAN**[†,††a)], **Xiankai MENG**[†,††], **Haoyu ZHANG**[†,††], ***and*** **Jianjun XU**[†,††], ***Nonmembers***

SUMMARY Determining the validity of knowledge triples and filling in the missing entities or relationships in the knowledge graph are the crucial tasks for large-scale knowledge graph completion. So far, the main solutions use machine learning methods to learn the low-dimensional distributed representations of entities and relationships to complete the knowledge graph. Among them, translation models obtain excellent performance. However, the proposed translation models do not adequately consider the indirect relationships among entities, affecting the precision of the representation. Based on the long short-term memory neural network and existing translation models, we propose a multiple-module hybrid neural network model called TransP. By modeling the entity paths and their relationship paths, TransP can effectively excavate the indirect relationships among the entities, and thus, improve the quality of knowledge graph completion tasks. Experimental results show that TransP outperforms state-of-the-art models in the entity prediction task, and achieves the comparable performance with previous models in the relationship prediction task.

key words: *knowledge graph completion, knowledge representation learning, knowledge graph*

1. Introduction

Knowledge graph techniques are widely used in several intelligent fields including intelligent search [1], question answering systems [2], expert systems [3], named entity recognition [4] and entity disambiguation [5]. These techniques represent the semantic network with a graph structure, expressing the knowledge entities as nodes and relationships as edges in the graph. The knowledge graph can be formally described as $G = (E, R, S)$, where $E = \{e_1, e_2, \ldots, e_n\}$ is the set of entities, $R = \{r_1, r_2, \ldots, r_m\}$ is the set of relationships, and $S \subseteq E \times R \times E$ represents the constraint of the entities and relationships, that is, the triples set T. Each triple in T can be expressed as $T_i = <h_i, r_i, t_i>$, where h is the head entity, t is the tail entity and r is the relationship between h and t.

There is often a large number of corrupted triples in the knowledge graph; problems in the triples include missing entities and incorrect relationships. It is difficult to ensure the correctness and completeness of the knowledge graph. The main purpose of the completion of the knowledge graph is to identify the missing entities and the incorrect relationships by entity prediction or relationship prediction. The main task of entity prediction is, for a given triple $T = <h, r, *>$ or $T = <*, r, t>$ ($*$ indicates the missing or incorrect entities), to find all entities which can make the triple correct. Relationship prediction is also known as link prediction, and the main task is to find the missing or incorrect relationships in the triple $T = <h, *, t>$ ($*$ indicates the missing or incorrect relationships) which make the triple correct.

Traditional knowledge representation has problems such as low computational efficiency, too-sparse data and the need to manually select features, which make completion tasks difficult in large-scale knowledge graphs. The deep learning technique based on representation learning provides a new perspective on this problem. This technique expresses the knowledge entities and relationships as low-dimensional real-valued vectors (vector **h** represents head entity h, vector **t** represents tail entity t and vector **r** represents relationship r), then calculates the semantic information by simple and fast mathematical methods to achieve the purpose of knowledge graph completion. Due to the high performance and the automatic learning of semantic features, this technique has developed rapidly. Researchers have proposed a lot of models such as Structured Embedding (SE) [6], Single Layer Model (SLM) [7] and Semantic Matching Energy (SME) [8], [9]. However, most of these models only consider the simple relationships among entities, with poor prediction performance.

Translation models such as TransE [10] have become a hotspot of research in recent years. Based on the translation invariance of the low-dimensional vector of the entities, translation models establish the loss function by measuring the Ln distance between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$, then train the final model. Based on the spatial projection and combinatorial operators, these models have provided a more useful perspective on the feature representation of complex relationships among entities, such as *1-to-N*, *N-to-1* and *N-to-M*. PTransE [11] and other models even consider the existence of information on paths among the entities in multiple-level relationships, modeling two to three indirect relationships. Compared with the previous models, these models have better performance in knowledge graph completion tasks, but still fail to represent the indirect relationships among long-distance entities.

Based on the existing translation models, we propose a method to construct the entity and relationship paths for knowledge graph, and propose a multiple-module deep

Manuscript received December 8, 2017.
Manuscript revised January 31, 2018.
Manuscript publicized February 20, 2018.

[†]The authors are with College of Computer, National University of Defense Technology, Changsha, China.
[††]The authors are with National Key Laboratory for Parallel and Distributed Processing, Changsha, China.
a) E-mail: eric_tan@nudt.edu.cn

DOI: 10.1587/transinf.2017EDP7398

learning model called TransP. TransP uses the long short-term memory (LSTM) to excavate the indirect relationships among the entities, and uses the method mentioned in ProjE [12], which takes reducing the collective ranking loss as the optimization goal to excavate the direct relationships, to enable effective excavation of rich information about the relationships among the entities. By using the step-wise training mechanism, TransP can effectively capture the rich relationship information among entities. Experimental results show that TransP outperforms state-of-the-art models in the entity prediction task, and achieves the comparable performance with the previous models in the relationship prediction task.

The main contributions of this work are as follows. First, proposing a novel multiple-module hybrid neural network model, which jointly considers the direct and indirect relations among entities. Second, to the best of our knowledge, this study is the first to use LSTM to extract the long entity path information to enhance the learning ability in the knowledge graph completion task. Lastly, the proposed model achieves state-of-the-art performance in the entity prediction task.

2. Related Work

As the first translation model, TransE [10] assumed that the tail entity $\mathbf{t}$ is translated by the head entity $\mathbf{h}$ along the relationship $\mathbf{r}$. TransE measures the Ln distance between $\mathbf{h}+\mathbf{r}$ and $\mathbf{t}$, $f_{\mathbf{r}}(\mathbf{h},\mathbf{t}) = \|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_{L1/L2}$ as the score function, and the loss function defined as follows:

$$L = \sum_{<h,r,t>\in G} \sum_{<h',r,t'>\notin G} [\gamma + f_{\mathbf{r}}(\mathbf{h},\mathbf{t}) - f_{\mathbf{r}}(\mathbf{h}',\mathbf{t}')] \quad (1)$$

In the above formula, $<h,r,t>$ represents the existing triples in the knowledge graph, called golden triples, $<h',r,t'>$ represents the triples that do not exist in the knowledge graph, called corrupted triples, and γ is the minimum margin between a golden triple and a corrupted triple. TransE has the fewest parameters and the lowest complexity of all translation models, but it is only suitable for learning the representation of *1-to-1* relationships.

In order to learn complex relationships such as *1-to-N*, *N-to-1* or *M-to-N*, a series of variant models have been proposed. TransH [13] proposed to project the head entities and tail entities onto the hyperplane; TransR [14] proposed to project the entities into relationship space; CTransR [14] proposed to cluster the relationships, then divide them into multiple sub-relationships; KG2E [15] considered the uncertainty, imbalance and heterogeneity of entities and relationships, and proposed the use of Gaussian distribution modeling the relationships among entities, using the asymmetric energy function based on KL divergence and the symmetrical energy function based on expected probability; TransG [16] considered the multiple semantic properties of the relationships, using the Gaussian mixture model to describe the relationships among entities, and leveraging the Bayesian nonparametric infinite mixture embedding model to discover the multiple relationships semantics; TransA [17] proposed the use of Markov distance measurement loss function to learn different weights for each dimension; TransD [18] proposed projecting the entities into relationship space like TransR, but with fewer parameters; TransSparse [19] considered the heterogeneity and imbalance of the entities and relationships, using the sparse matrix instead of the dense matrix in TransR, and using different projection matrices for the head entities and tail entities; TransF [20] had flexibility, which ensured that the sum of the head entities and the corresponding relationships had the same direction as the tail entities, but with different sizes.

Most of the models consider the knowledge completion problem as a pairwise ranking problem, but ProjE [12] views the completion problem as a collective scores ranking problem of candidate entities of the head entity h and corresponding relationship r. ProjE establishes a three-layer neural network model that uses combinatorial operators to combine the input data into target vectors and improve performance by optimizing the collective rank loss of the candidate entities list (or relationships list). It is a self-contained model and does not depend on any preprocessing.

The above works have improved modeling of the direct relationships among entities, but do not take into account the indirect relationships among entities. In order to learn the representation of such relationships, PTransE [11], RTransE [21] and other models take into account the translation model and the indirect relationship path information among the entities. These models further improve the performance of knowledge graph completion tasks. However, the long paths can result in parameter explosion problem, and not all of the relationships are reliable, meaning these models usually only consider two or three level indirect relationships.

3. Methodology

To take into account the direct relationship and the indirect relationship among entities, TransP constructs the entity relationship paths and a multiple-module hybrid neural network model including LSTM unit, and uses the step-wise training mechanism to optimize the network parameters. In order to capture the rich and subtle relationships among entities in the knowledge graph, TransP reduces the overall collective loss as an optimization goal.

3.1 Term Definitions

This paper includes the following terms and definitions:

Fact Triple: For the knowledge graph G and any triples $T = <h,r,t>$, if the triple T exists in the knowledge graph G, that is, $T \in G$, then T is the Fact Triple in the knowledge graph G, referred to as Fact.

Indirect Connection Exists: Let $T_{set} = \{T_1 = <e_1, r_1, e_2>, T_2 = <e_2, r_2, e_3>, \ldots, T_{n-1} = <e_{n-1}, r_{n-1}, e_n>\}$ be the set of fact triples in knowledge graph G, such that for every two triples T_i, T_j ($j = i+1$, and where $j < n$) in T_{set},

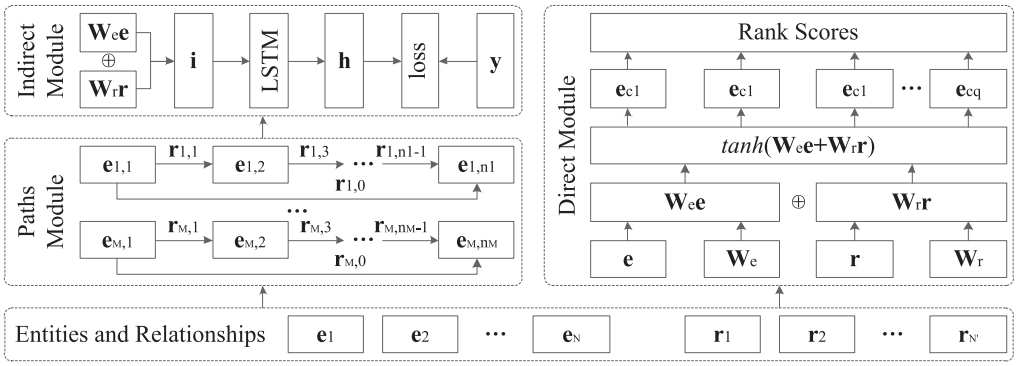

Fig. 1 The architecture of TransP. The details of the LSTM layer in the indirect module are hidden (see Sect. 3.5). $\mathbf{e}_*$ and $\mathbf{r}_*$ in the entities and relationships module denote the vectors of all entities and relationships in the knowledge graph, which will be fine-tuned during training. $\mathbf{e}$ and $\mathbf{r}$ in the indirect module denote the entities and relationships contains in the paths constructed by path construction module. $\mathbf{e}_{c*}$ in the direct module denote all the candidate entities corresponding to the entity $\mathbf{e}$ and relationship $\mathbf{r}$. Their scores will be calculated for $\mathbf{e}$ and $\mathbf{r}$ separately (see Sect. 3.6). $\mathbf{W}_e$ and $\mathbf{W}_r$ denote the weight matrices corresponding $\mathbf{e}$ and $\mathbf{r}$.

the tail entity of T_i is just the head entity of T_j. We say that Indirect Connection Exists between e_1 and e_n if there is another triplet $T_k = <e_1, r, e_n>$ in G.

Entity Relationship Path and Sub-Path: For entity pairs $<h, t>$, if there is a sequence of ordered fact triples inside the h and t arranged in a certain order, we call the set of entities $E = \{h = e_1, e_2, \ldots, t = e_n\}$ in the order as Entity Path, and the relationships set $R = \{r_1, r_2, \ldots, r_{n-1}\}$ is called the Relationship Path. The Entity Path and the Relationship Path are collectively referred to as the Entity Relationship Path. A subset of any length in the Entity Relationship Path is called its Sub-Path.

For example, consider the following set of fact triples: {<*Wall Street, located in, Manhattan*>, <*Manhattan, located in, New York City*>, <*New York City, located in, New York state*>, <*New York state, located in, USA*>, <*USA, located in, North America*>}. In this example, for triple <*Wall Street, located in, North America*>, there exists an entity path {*Wall Street, Manhattan, New York City, New York state, USA, North America*}. In this path, there are indirect connections between any two non-adjacent entities, and path {*New York City, New York state, USA*} is a sub-path.

3.2 Model Architecture

TransP is a hybrid neural network model with multiple modules; the overall architecture includes an entities and relationships embedding module, a path construction module, an indirect module and a direct module. The overall architecture is shown in Fig. 1.

The entities and relationships embedding module mainly maps all entities and relationships into their corresponding vector representation $\mathbf{E} \in \mathbb{R}^{d\times|E|}$ and $\mathbf{R} \in \mathbb{R}^{d\times|R|}$, where d is the vector dimension, $|E|$ and $|R|$ are the total number of entities and relationships in the knowledge graph. These vectors are the basis of the entire model and will be fine-tuned during training.

The path construction module is used for the entity relationship path construction (see Sect. 3.3). The indirect module is used to analyze the indirect relationship information contained in the path provided by the path construction module (see Sect. 3.5). The direct module is used to analyze the direct relationship information among the entity and the corresponding candidate entities (see Sect. 3.6). The candidate entity acquisition method in the direct module is described in Sect. 3.4. The inference process of model is described in detail in Sect. 3.7.

Entities and relationships vectorization, path construction, and candidate entity sampling are done before training. TransP uses a step-wise training mechanism, the training of the model is a two-step iterative process. First, the entities and relationships vectors in the path are updated according to the constructed indirect path information by indirect module. Then, according to the list of candidate entities, the direct module is used to further update the entities and relationships vector. The model performance will eventually stabilize and the trained entities and relationships vectors will be used for knowledge graph completion tasks.

3.3 Entity Relationship Path Construction

The basic method of entity relationship path construction is to find all the indirect relationships of entity pairs $<h, t>$ for all triples T in knowledge graph G, and generate the entity relationship path in sequence. The essence of finding an entity relationship path is to explore all paths among all nodes in a directed graph with a time complexity of $O(N^3)$, where N is the total number of nodes in the directed graph, that is, the number of entities in the corresponding knowledge graph.

In order to reduce the time complexity, we propose a batch entity relationship path construction method. Assuming that the total number of triples in the knowledge graph G is N_t, the entities in each batch (which have M triples) made into a directed sub-graph, will make a total of N_t/M sub-graphs (M is the super-parameter). For any of the triples T_i

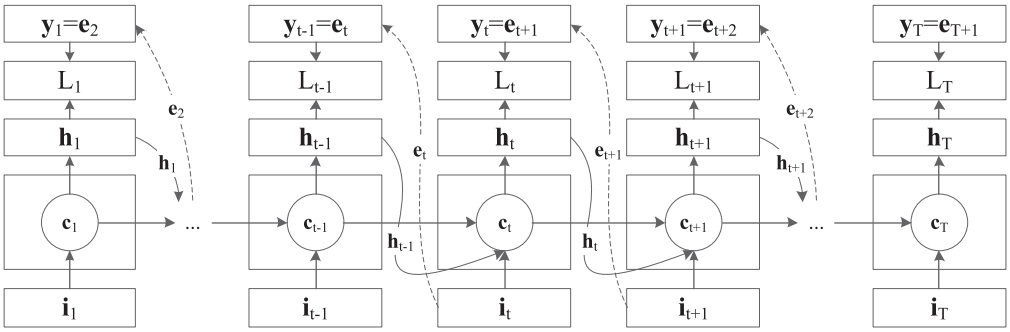

Fig. 2 The unrolled structure of LSTM by time steps. The solid arrows indicate the direction of the data flow, and the dashed arrows indicate the data correspondence. In the back-propagation phases, the residuals are passed in the opposite direction to the data stream. In the figure, circle $\mathbf{c}_*$ is the memory unit, L_* is the loss function, $\mathbf{i}_* = \mathbf{e}_* \oplus \mathbf{r}_*$ denotes the input vector and $\mathbf{h}_*$ denotes the output vector. The input gate, output gate and forgot gate of the LSTM block are hidden.

$= <h_i, r_i, t_i>$ in the sub-graph G_k, we first find all the paths of the h_i node to the t_i node, and append the entities in the path into the entity path set $PE_k = \{PE_{k1}, PE_{k2}, \ldots, PE_{kn}\}$ and append the relationships in the path into the relationship path set $PR_k = \{PR_{k1}, PR_{k2}, \ldots, PR_{kn-1}\}$.

During the training process, all sub-entity paths and sub-relationship paths are spliced into a long entity path and a long relationship path, where the sub-entity paths are separated by a special symbol and the sub-relationship paths are separated by two special symbols (since the number of relationships in the sub-relationship path is one less than the number of entities in the corresponding sub-entity path). After the splicing is complete, the path is segmented according to the batch size B and the time step T to obtain the final indirect module training data set.

3.4 Candidate Entity Sampling

A candidate entity is any entity that may make T become a fact. In a large-scale knowledge graph, if all the entities are treated as candidate entities for model training, it will bring great training costs. It is common practice to reduce the number of candidate entities by using candidate sampling to improve training efficiency [22]–[24]. As with ProjE, we use the negative sampling method used in word2vec to sample candidate entities [23].

In the knowledge graph G, the candidate entity that makes the triples become a fact is called a positive case; otherwise it is called a negative case. In the candidate entity choice process, we include all positive cases into the candidate entity set E^c. For the negative cases, we use a simplified 0-1 distribution $B(1, p_y)$ for sampling, where py is the probability of negative cases being accepted and $1 - p_y$ is the probability that negative cases are not accepted. The value range of p_y is [0%, 100%], where 0% means that no negative cases are sampled and 100% means that all negative cases are sampled.

3.5 Indirect Module

The indirect module consists of the entity relationship sub-layer, the combined sub-layer and the LSTM sub-layer. The LSTM sub-layer is a simple LSTM network [25], [29], which consists of input layer, hidden layer and output layer. The LSTM sub-layer is responsible for processing the tensor by time step, and its unrolled structure is shown in Fig. 2.

LSTM is a variant of recurrent neural network designed to cope with long-term dependency problems. A LSTM unit is composed of several gates to control the proportions of the input to give to the memory cell, and the proportion information to forget and to pass on to the time steps. We use the following implementation:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{g}_t &= \tanh(\mathbf{W}_g\mathbf{x}_t + \mathbf{U}_g\mathbf{h}_{t-1} + \mathbf{b}_g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}, \tag{2}$$

where σ is the element-wise sigmoid function, and $\odot$ is the element-wise product. $\mathbf{x}_t$ is the input vector at time t, $\mathbf{c}_t$ is the memory cell state at time t and $\mathbf{h}_t$ is the hidden state at time t. $\mathbf{W}_*$ and $\mathbf{U}_*$ denote the weight matrices of different gates, and $\mathbf{b}_*$ denote the bias vectors.

The input vector of the LSTM input layer is multiplied by the entity vector in the entity path and the matrix of the entity combination, plus the combination vector composed of the relationship vector in the corresponding relationship path multiplied by the relationship combination matrix. For the entity path matrix $\mathbf{E}$ and the corresponding relationship path matrix $\mathbf{R}$, the input tensor $\mathbf{IT}$ is defined as follows:

$$\mathbf{IT} = \begin{bmatrix} \mathbf{E}_1 \oplus \mathbf{R}_1 \\ \mathbf{E}_2 \oplus \mathbf{R}_2 \\ \cdots \\ \mathbf{E}_B \oplus \mathbf{R}_B \end{bmatrix}, \tag{3}$$

where

$$\mathbf{E}_i \oplus \mathbf{R}_i = \begin{bmatrix} \mathbf{W}_{ei1}\mathbf{e}_{i1} + \mathbf{W}_{ri1}r_{i1} \\ \mathbf{W}_{ei2}\mathbf{e}_{i2} + \mathbf{W}_{ri2}r_{i2} \\ \cdots \\ \mathbf{W}_{eiT}\mathbf{e}_{iT} + \mathbf{W}_{riT}r_{iT} \end{bmatrix}^{\mathrm{T}} \tag{4}$$

In the above equation, the $\mathbf{W}_{e*}$ and $\mathbf{W}_{r*}$ matrices are diagonal matrices. This means that in the pre-processing, we only consider the dimension weights of the entity vector and the relationship vector, regardless of the influence of dimensions. This approach, on the one hand, can simplify the calculation; on the other hand, the influence among the dimensions will be further considered in the hidden layer or directly related layer. In the entity and relationship prediction task, the indirect module input only considers $\mathbf{h} + \mathbf{r}$; even only considering $\mathbf{h}$ can obtain better results.

The output of the LSTM layer is a tensor **OT** which has the same form as the input tensor **IT**, and a tensor contains $B * T$ vectors with d dimensions (B is the batch size, T is the time step). For each time step $t(0 < t < T)$, the output is a matrix **OM** which contains B input vectors. Assuming that the symbol corresponding to the t-th time step in the l-th sequence is $y^l_{(t)}$, the total loss function is defined as:

$$L = \frac{1}{2}\sum_{l=1}^{B}\sum_{t=1}^{T}\left\|\mathbf{y}^l_{(t)} - \mathbf{h}^l_{(t)}\right\|^2 \tag{5}$$

In the above equation, we directly let $\mathbf{y}^l_{(t)}$=$\mathbf{e}^l_{(t+1)}$, that is, for the entity vector $\mathbf{e}_t$ and the relationship vector $\mathbf{r}_t$ in the entity relationship path, the target output vector is $\mathbf{e}_{t+1}$. Based on this, the BPTT algorithm [25], [26], which is commonly used in the recurrent neural network, is used to train the whole network parameters according to the time steps. After the indirect module training is completed, the trained entity vectors, relationship vectors and their corresponding combined operators are transferred to the direct module for further training.

3.6 Direct Module

The direct module is a three-layer simple feed-forward neural network, which contains three sub-layers of input layer, candidate entity combination layer and scores rank layer. The input layer data is transformed by the tanh function after the combination of the entity vectors and relationship vectors and the corresponding combination operator trained by the indirect module, defined as:

$$\mathbf{ID}_i = f(t(\mathbf{e}_i \oplus \mathbf{r}_i)) = tanh(\mathbf{e}_i \oplus \mathbf{r}_i) \tag{6}$$

Similarly to the ProjE model [12], we consider the knowledge graph completion problem as a ranking problem for candidate entities and use the list method [27] to handle the entity ranking task. According to this method, the candidate entity combination layer combines the input vector with all corresponding candidate entity vectors, and the loss function definition is also consistent with ProjE:

$$L(\mathbf{ID}_i, \mathbf{y}) = -\sum_{j}^{|\mathbf{y}|}\frac{log(h_d(\mathbf{ID}_i)_j)}{\sum_j \mathbf{1}(y_j = 1)} \tag{7}$$

In the above equation, $h_d(\mathbf{ID}_i)_j$ is defined as:

$$h_d(\mathbf{ID}_i)_j = softmax(\mathbf{e}_j tanh(\mathbf{e}_i \oplus \mathbf{r}_i) + \mathbf{b}) \tag{8}$$

In the above equation, $\mathbf{e}_j$ represents the vector corresponding to the j-th candidate entity, and $\mathbf{b}$ is the corresponding bias. This is a multi-class problem, and all positive cases in the candidate entity set (i.e., entities that are directly related to the input entity $\mathbf{e}$ and the input relationship $\mathbf{r}$) are scored as $1/|\mathbf{E}^c_+|$ ($|\mathbf{E}^c_+|$ representing the total number of positive cases in all candidate entities corresponding to $\mathbf{e}$ and $\mathbf{r}$), and the entities not directly related to the input entity $\mathbf{e}$ and the input relationship $\mathbf{r}$ are scored as 0 in the direct module.

The training method of the direct module is the same as the feedforward neural network, and the parameters of the direct path module are trained in the reverse direction. During the training process, the derivative is directly derived from the loss value, until the parameters in the entire direct module are trained.

3.7 Model Inference

All entity vectors, relationship vectors, and the weight parameters of the model will be fixed after the training process, and these data will be used in the inference process. The indirect module is mainly used to adjust the entity vectors and relationship vectors to capture potential relationship information among entities during training process. The inference process does not need to construct the entity relationship path, and does not require indirect relationship computing, so the inference process is simpler than the training process.

In the entity prediction task, given the head entity vector $\mathbf{h}_e$ and relationship vector $\mathbf{r}_e$, all the entities except the head entity will be scored in turn by the following formula:

$$score(e_i) = \mathbf{e}_i tanh(\mathbf{h}_e \oplus \mathbf{r}_e) + \mathbf{b} \tag{9}$$

In the above equation, $\mathbf{e}_i$ represents the vector corresponding to the i-th entity. Ranking all the entities according to the score in descending order can determine the most likely tail entity corresponding to the head entity h_e and relationship r_e. The head entity inference process is similar, only need to replace $\mathbf{h}_e$ in the above formula with tail entity vector $\mathbf{t}_e$.

In the relationship prediction task, given the head entity vector $\mathbf{h}_e$ and tail entity vector $\mathbf{t}_e$, all the relationship will be scored in turn by the following formula:

$$score(r_j) = \mathbf{t}_e tanh(\mathbf{h}_e \oplus \mathbf{r}_j) + \mathbf{b} \tag{10}$$

In the above equation, $\mathbf{r}_j$ represents the vector corresponding to the j-th relationship. Ranking all the entities according to the score in descending order can determine the most likely relationship corresponding to the head entity h_e and tail entity t_e.

4. Experiments

We used Python to implement the TransP model in the TensorFlow [28] framework and evaluated TransP's entity prediction and relationship prediction capabilities based on

the commonly used data set FB15K [10]. FB15K contains 1,345 relationships and 14,951 entities, corresponding to 483,142 training triples, 50,000 validated triples and 59,071 test triples.

To evaluate the impact of the indirect module, we perform ablation tests in the entity prediction task and the relation prediction task respectively, by removing the indirect module in the whole TransP model.

4.1 Parameter Setting

The indirect module is optimized using the GradientDescent optimizer. The super-parameters that need to be set in the whole module include the number of sub-graph triples M, time step t, hidden layer dimension h, hidden layer number l, mini-batch size bi and the maximum iteration period ei.

The direct module is optimized by the Adam optimizer, and the L1 regularization is also used to prevent overfitting. The super-parameter that needs to be set by the whole module includes the negative sampling probability ps, the maximum number of training iterations ed, the regularization weight a, the mini-batch size bd, and the parameters $\beta 1, \beta 2$ and ε that the optimizer needs to set.

In addition, we introduced dropout layers in both the indirect module and the direct module. The whole model needs to set the public hyperparameters including the entity and relationship dimension k, the learning rate r and the dropout probability d.

Inspired by ProjE [12], in our experiments, the parameters were initially set to $M = 5,000$, $p_y = 0.25$, $t = 5$, $h = 200$, $l = 1$, $bi = 20$, $ei = 4$, $ps = 0.5$, $ed = 100$, $a = 1e{-}5$, $bd = 200$, $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\varepsilon = 1e{-}8$, $k = 200$, $r = 0.01$ and $d = 0.5$.

The initial value of the learning rate of the model can also be set larger, and then gradually decreased with the increase in the number of training sessions. All entity vectors, relationship vectors and parameters that need to be initialized in the model are initialized using TransE's recommended uniform distribution $U[-6/\sqrt{k}, 6/\sqrt{k}]$.

4.2 Entity Prediction

For the entity prediction task, we used the same evaluation criteria mentioned in TransE, DKRL, TransH, TransR, PTransE, RTransE, TransA and ProjE. That is, for each test triple $T = <h, r, t>$, h and t are replaced by each entity e_i in the data set respectively; the score is then calculated for $Tr = <e_i, r, t>$ and $Tr = <h, r, e_i>$, and the entities sorted in descending order according to the scores.

Based on the above entity ranking, we consider the collective average ranking (Mean Rank) and the top 10 hit rate (HITS10) metrics. The Mean Rank refers to the average number of correct entities in the ranking. HITS10 refers to the proportion of the correct entities which appear in the top 10.

In the original measurement results, there may be some misclassified triples T, T does not exist in the test set, but exists in the training set and the validation set. To reflect the predictive performance objectively, we process the original results and remove the triples from the list. In the experimental results, we mark the untreated original measure as Raw, and the result after processing is called Filter. All triples that are judged to be wrong in the filter do not exist in the knowledge graph.

Table 1 The results of the entity prediction on the FB15K dataset. ADD-2 means use the ADD method and consider a two-length path; RNN-2 means use the RNN method and consider a two-length path; ADD-3 means use the ADD method and consider a three-length path. As the same data set and evaluation metrics are used, the data are directly drawn from the related work.

Models	Mean Rank		HITS10	
	Raw	Filter	Raw	Filter
TransE	243	125	34.9	47.1
DKRL CNN	200	113	44.3	57.6
TransH	212	87	45.7	64.4
TransR	198	77	48.2	68.7
TransE+Rev	205	63	47.9	70.2
PTransE ADD-2	200	54	51.8	83.4
PTransE RNN-2	242	92	50.6	82.2
PTransE ADD-3	207	58	51.4	84.6
RTransE	-	50	-	76.2
TransA	155	74	56.1	80.4
TransF	220	89	40.5	61.2
ProjE_pointwise	174	104	56.5	86.6
ProjE_listwise	146	76	54.6	71.2
ProjE_wlistwise	124	34	54.7	88.4
TransP-Ind	117	27	61.3	86.4
TransP	**107**	**17**	**62.4**	**89.3**

We compare a number of models and the results are shown in Table 1. From the experimental results it can be seen that TransP outperforms state-of-the-art models. For MeanRank (Raw) and MeanRank (Filter), TransP improved by 17 (about 13.7%) and 17 (about 50.0%), respectively. And for HITS10 (Raw) and HITS10 (Filter), TransP improved by 7.7 (about 14.1%) and 0.9 (about 1.0%), respectively. The results indicate that TransP has better prediction capabilities in entity prediction tasks.

We removed the indirect module from the whole TransP model, to evaluate the impact of the indirect module (denoted TransP-Ind). The results shows that the performance degraded after removing the indirect module. For MeanRank (Raw) and MeanRank (Filter), TransP-Ind decreased by 10 (about 9.3%) and 10 (about 58.8%), respectively. And for HITS10 (Raw) and HITS10 (Filter), TransP-Ind decreased by 1.1 (about 1.8%) and 2.9 (about 3.2%), respectively.

We speculate that the main reason for this disparity is that by modeling the indirect relationship of entities, some entities with implicit relationships can be predicted. For example, there is no direct relationship between *New York City* and *USA* for entity paths {*New York City, New York State, USA*}, but there is an indirect relationship that can be predicted through indirect path modeling.

4.3 Relationship Prediction

For the relationship prediction task, we consider two metrics

Table 2 The results of the relationship prediction on the FB15K dataset. As the same data set and evaluation metrics are used, the data are directly drawn from the related work.

Models	Mean Rank		HITS1	
	Raw	Filter	Raw	Filter
TransE	2.8	2.5	65.1	84.3
TransE+Rev	2.6	2.3	67.1	86.7
DKRL CNN	2.9	2.5	69.8	89.0
PTransE ADD-2	1.7	1.2	69.5	93.6
PTransE RNN-2	1.9	1.4	68.3	93.2
PTransE ADD-3	1.8	1.4	68.5	94.0
ProjE_pointwise	1.6	1.3	75.6	95.6
ProjE_listwise	**1.5**	1.2	**75.8**	**95.7**
ProjE_wlistwise	**1.5**	1.2	75.5	95.6
TransP-Ind	**1.5**	1.2	75.5	94.8
TransP	**1.5**	**1.1**	**75.8**	95.1

of Mean Rank and HITS1 (top 1 hit rate). For the original measurement results (Raw), we used the same methods to process the results in the entity prediction task and all the triples which in the knowledge graph will be removed from the results list, recorded as Filter.

As shown in Table 2, TransP has the best effect on the Mean Rank (Raw and Filter) and HITS1 (Raw) which similar to the ProjE_listwise model. For HITS1 (Filter), TransP is slightly lower than ProjE by 0.6 (about 0.6%), but remarkably higher than the other models. This shows that TransP has comparable performance with the previous models in the relationship prediction task.

After removing the indirect module, we found that the Mean Rank (Raw) metrics of TransP-Ind is exactly the same as the TransP and ProjE. The Mean Rank (Filter) and HITS1 (Raw and Filter) metrics of the TransP-Ind decreased slightly, of which the Mean Rank (Filter) decreased by 0.1 (about 9.1%), the HITS1 (Raw) decreased by 0.3 (about 0.3%) and the HITS1 (Filter) decreased by 0.3 (about 0.3%). This shows that the indirect module also has a promotion effect in the relationship prediction task. But under the high prediction performance, the improvement effect of the model is not obvious.

We speculate that in the relationship prediction task based on FB15K, the performance of the neural network model is strongly affected due to the data noise or other factors such as gradient vanishing, and eliminating data noise may further improve the performance. In addition, the results may fluctuate due to the random initialization of parameters.

5. Conclusions

Based on ProjE and other translation models, this paper proposes a new knowledge representation learning model, called TransP. TransP is a multiple-module hybrid deep neural network model using step-wise training mechanism learning knowledge representations. TransP considers the rich entity relationship path information in the knowledge graph. By constructing the entity relationship path, the LSTM is used to model the relationship among these entities, and the distant relationship among the entities is excavated. Experiments show that this indirect relationship considered by TransP has a significant effect on learning knowledge's vector representations, which is of great significance in knowledge graph completion tasks.

In addition to the methods mentioned in this paper, TransP also has a strong scalability, can be combined with other knowledge learning models, and can even be adjusted using these combinations to adapt to different tasks. After combining with other models, the indirect entity relationship is extracted by the method proposed by TransP, and the model parameter space is optimized to further optimize the learning ability of the knowledge models.

In the future, we will consider two aspects: Firstly, we will consider the use of an attention mechanism, concerned with the higher-frequency entities and relationships in the knowledge graph, especially for the entities and relationships that contribute greatly to the relationships in the process of the entity relationship path construction, to further enhance the model's learning ability. Secondly, we will consider the excavation of the description information corresponding to the entities or relationships contained in the knowledge graph, to further enhance the model capacity.

Acknowledgments

This study is supported by the Scientific Research Fund of Hunan Provincial Education Department (No. 15A007) and the National Natural Science Foundation of China (No. 61202116).

References

[1] A. Uyar and F.M. Aliyu, "Evaluating search features of Google Knowledge Graph and Bing Satori," Online Information Review, vol.39, no.2, pp.197–213, 2015.
[2] W.-T. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base," In: Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, pp.1321–1331, 2015.
[3] J. Hyeon, K.-J. Oh, Y.J. Kim, H. Chung, B.H. Kang, and H.-J. Choi, "Constructing an initial knowledge base for medical domain expert system using induct RDR," In: IEEE International Conference on Big Data and Smart Computing IEEE, pp.408–410, 2016.
[4] D. Ye, Z. Xing, C.Y. Foo, Z.Q. Ang, J. Li, and N. Kapre, "Software-Specific Named Entity Recognition in Software Engineering Social Content," In: IEEE Interna-tional Conference on Software Analysis, Evolution, and Reengineering, pp.90–101, 2001.
[5] S. Cucerzan, Large-scale named entity disambiguation based on Wikipedia data, EMNLP-CoNLL, 7, pp.708–716, 2007.
[6] A. Bordes, et al., Learning Structured Embeddings of Knowledge Bases, In: 2011 AAAI Conference on Artificial Intelligence, San Francisco, DBLP, 2011.
[7] R. Socher, et al., Reasoning with neural tensor networks for knowledge base completion, International Conference on Intelligent Control & Information Processing, pp.464–469, 2013.
[8] A. Bordes, X. Glorot, and J. Weston, Joint Learning of Words and Meaning Representations for Open-Text Semantic Parsing, International Conference on Artificial Intelligence & Statistics, 2012.
[9] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," Machine Learning, vol.94, no.2, pp.233–259, 2014.

[10] A. Bordes, et al., Translating Embeddings for Modeling Multi-relational Data, Advances in Neural Information Processing Systems, 2013, pp.2787–2795, 2013.
[11] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, "Modeling Relation Paths for Representation Learning of Knowledge Bases," Computer Science, pp.705–714, 2015.
[12] B. Shi and T. Weninger, ProjE: Embedding Projection for Knowledge Graph Completion, AAAI Association for the Advancement of Artificial Intelligence, 2017.
[13] Z. Wang, et al., Knowledge Graph Embedding by Translating on Hyperplanes, AAAI Association for the Advancement of Artificial Intelligence, 2014.
[14] Y. Lin, et al., Learning entity and relation embeddings for knowledge graph completion, 29th AAAI Conference on Artificial Intelligence, pp.2181–2187, 2015.
[15] S. He, K. Liu, G. Ji, and J. Zhao, "Learning to Represent Knowledge Graphs with Gaussian Embedding," In: ACM International Conference on Information and Knowledge Management, pp.623–632, 2015.
[16] H. Xiao, M. Huang, and X. Zhu, "TransG: A Generative Mixture Model for Knowledge Graph Embedding," Computer Science, pp.2316–2325, 2015.
[17] Y. Jia, et al., Locally Adaptive Translation for Knowledge Graph Embedding, Computer Science, 2015.
[18] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge Graph Embedding via Dy-namic Mapping Matrix," In: Meeting of the Association for Computational Linguistics and the International Joint Con-ference on Natural Language Processing, pp.687–696, 2015.
[19] J. Guoliang, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," Proceedings of AAAI 2016, Phoenix, USA, 12-17 Feb. 2016.
[20] J. Feng, et al., Knowledge Graph Embedding by Flexible Translation, Computer Science, 2015.
[21] A. García-Durán, A. Bordes, and N. Usunier, "Composing Relationships with Translations," In EMNLP, pp.286–290, 2015.
[22] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, On Using Very Large Target Vocabulary for Neural Machine Translation, ACL, pp.1–10, 2015.
[23] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," NIPS, 2013.
[24] G. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," AISTATS, pp.297–304, 2010.
[25] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol.9, no.8, pp.1735–1780, 1997.
[26] A. Graves, "Long Short-Term Memory," Supervised Se-quence Labelling with Recurrent Neural Networks, Springer, Berlin Heidelberg, vol.385, pp.37–45, 2012.
[27] H.H. Pareek and P.K. Ravikumar, "A representation theory for ranking functions," NIPS, pp.361–369, 2014.
[28] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., "Tensorflow: Large-scale machine learning on heteroge-neous distributed systems," arXiv preprint arXiv:1603.04467, 2016.
[29] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," arXiv preprint arXiv:1409.2329, 2014.

Ping Zeng is a Ph.D. candidate in College of Computer at National University of Defense Technology, China. His research focuses on intelligent software, knowledge graph and cloud robotics.

Qingping Tan is a Professor and Ph.D. supervisor in College of Computer at National University of Defense Technology, China. His research interests include intelligent software, distributed software engineering, and high dependable software.

Xiankai Meng is a Ph.D. candidate in College of Computer at National University of Defense Technology, China. His research focuses on intelligent software and high dependable software.

Haoyu Zhang is a Ph.D. candidate in College of Computer at National University of Defense Technology, China. His research focuses on intelligent software and natural language processing.

Jianjun Xu is a lecturer in College of Computer at National University of Defense Technology, China. His research interests include intelligent software and high dependable software.