Knowledge Graph Embedding for Hyper-Relational Data

Chunhong Zhang*, Miao Zhou, Xiao Han, Zheng Hu, and Yang Ji

Abstract: Knowledge graph representation has been a long standing goal of artificial intelligence. In this paper, we consider a method for knowledge graph embedding of hyper-relational data, which are commonly found in knowledge graphs. Previous models such as Trans (E, H, R) and CTransR are either insufficient for embedding hyper-relational data or focus on projecting an entity into multiple embeddings, which might not be effective for generalization nor accurately reflect real knowledge. To overcome these issues, we propose the novel model TransHR, which transforms the hyper-relations in a pair of entities into an individual vector, serving as a translation between them. We experimentally evaluate our model on two typical tasks—link prediction and triple classification. The results demonstrate that TransHR significantly outperforms Trans (E, H, R) and CTransR, especially for hyper-relational data.

Key words: distributed representation; transfer matrix; knowledge graph embedding

1 Introduction

Recently, knowledge graph representation has become a hot topic in artificial intelligence, as it has played an increasingly important role in facilitating various applications, such as question answering, semantic searches, and information inference. Knowledge graphs, such as Wordnet[1], Freebase[2], and Yago[3], usually contain huge amounts of structured data that is generated daily in many application domains. Hence, knowledge graph representation must be flexible, compact, and have the ability to generalize. However, traditional knowledge graph representations are symbolic and logical[4,5] and cannot efficiently measure the semantic relatedness of entities. To address this problem, knowledge graph representation uses the idea of distributed representation, whereby objects are represented as dense, real-valued, and low-dimensional vectors. Thus, knowledge graph embedding has been proposed for embedding entities and relations into a low-dimensional continuous vector space and to intuitively regard the triple as a computable equation. Of the various embedding models, there is a series of translation-based models including TransE[6], TransH[7], TransR[8], and others.

An elementary fact of the knowledge graph is represented in the form of a triple with two entities and a relation, i.e., (head, relation, tail) denoted by (h;r;t). For example, (Obama, born here, USA) corresponds to the knowledge that Obama was born in the USA. The basic idea behind all translation-based models is that the relation is regarded as a translation from head to tail when it is encoded into a metric space, that is, \( h \approx r \approx t \) holds for the triple \((h,r,t)\). This assumption results in relation completion by finding an \( r^* \) such that it corresponds to one of the nearest neighbors of \( r \), that is, \( h + r^* \approx t \) for a given entity pair \((h,t)\).

Difficulties arise when there are multiple relations between a pair of entities, which is common in knowledge graphs, since only one legal \( r \) is allowed by the equation in the metric space. Multi-metric space based mechanisms are then proposed, in which
their main difference is how to translate the vector representations from one space to the others. For example, TransR\(^8\) embeds entities and relations into distinct spaces, and project entities from the entity space into the relation space by a translation matrix \(M_r\) such that \(hM_r + r \approx tM_r\).

However, these separate projections disconnect diverse aspects of the same entities that may have been highly semantically related. For an instance extracted from FB40K\(^9\) as shown in Table 1, there are a total of 14 relations between the person entity Jude Law and the person entity Sienna Miller, which indicates the changing relationship of the celebrity couple at different periods of time. For example, the relation “/people/person/spouse_s” indicates Jude Law’s spouse is Sienna Miller. Obviously in order to reflect real knowledge, it is more reasonable to model multi-relations as an individual vector rather than dividing one person into 14 different parts.

Motivated by the above observation, we propose a novel translation-based schema, TransHR, to specifically address issues related to embedding multi-relations between entity pairs. First, to clearly distinguish our approach, we must re-define the relation patterns. In previous works, relations in a knowledge graph were classified into four types in terms of the number of entities: 1-to-1, 1-to-many, many-to-1, and many-to-many\(^{6–8}\). Although intuitively reasonable, this straightforward relation classification is somewhat ambiguous and not sufficient for our approach.

We start from the view point of the number of relations and classify them into two categories: sole-relation and hyper-relation. For a given pair of entities, if there is more than one relation simultaneously between them, then each of these relations is called a hyper-relation. Conversely, if a relation has never co-occurred with another relation in some entity pairs, it is called a sole-relation. Figure 1 shows a simple abstraction of two hyper-relations \(r_1, r_2\) for the entity pair \((h, t)\). An instance based on this figure scenario can be illustrated by two triplets of \((Obama, born here, USA)\) and \((Obama, live here, USA)\), where the relations born here and live here are both hyper-relations since they connect the same entity pair (Obama, USA). In contrast, the relation capital in the triplet (China, capital, Beijing) is a sole-relation when based on the assumption that the capital never appears with other relations in any triplets. We note that for a given relation type \(r\), it can typically appear in multiple triplets. As long as an instance of \(r\) appears together with other relations for some entity pairs, the \(r\) is treated as a hyper-relation even if it might exist alone in other entity pairs. That is, the notion hyper-relation mainly concerns the overall relation semantic of \(r\) rather than its specific instances.

Hyper-relations are commonly found in multi-relation graph data such as the aforementioned knowledge bases (Freebase, Wordnet, Yago)\(^6,10\), in which nodes correspond to entities and edges indicate the relations between these entities. This concept naturally coincides with the complicated interaction between entities and will enhance the representation learning of any polysemous, redundant, or heterogeneous semantics inherently embedded in the triplets of multi-relation graphs. Table 2 presents statistics of head-tail entity pairs in terms of the number

| Head | Relation | Tail |
|------|----------|------|
| /people/person/spouse_s | /celebrities/celebrity/sexual_relationships | Sienna Miller |
| /base/popstra/dated/participant | /base/popstra/public_insult/victim | Sienna Miller |
| Jude Law | /base/popstra/dated/participant | (https://en.wikipedia.org/wiki/Jude_Law) |
| Sienna Miller | /base/popstra/dated/participant | (https://en.wikipedia.org/wiki/Sienna_Miller) |

Table 1 A case of hyper-relation in FB40K.
Table 2 Number of head-tail entity pairs in terms of the number of relations between them. For instance, the number 139 739 means there are 139 739 head-tail entity pairs linked by only one relation.

| Data set | Number of relations |
|----------|---------------------|
| FB38K    | 139 739             |
| FB15K[9] | 330 948             |
| WN18[10] | 140 884             |
| FB13     | 280 335             |
| WN11[11] | 107 854             |

of relations for five data sets, where FB38K is a subgraph that we extracted from FB40K[9] (discussed later in detail) to evaluate our algorithm. We can see that FB38K and FB15K have the most head-tail pairs with a large number of relations. Hence, for these two data sets, we also count the number of sole- and hyper-relations in Table 3, which shows that FB38K contains only 39 sole-relations but 568 hyper-relations. Similarly, FB15K also has more hyper-relations than sole-relations. This indicates that hyper-relations are much more common than sole-relations, so the embedding accuracy of the hyper-relations is clearly important in the knowledge graph embedding.

Following the definition of hyper-relation, the algorithm TransHR tries to learn the appropriate vector representations for both the entities and relations, and then projects the relations from the relation space into the entity space by the translation matrix \( M_r \). The advantages of TransHR are twofold. First, it is computationally efficient in comparison to TransR and CTransR, because the expensive operation of matrix projection is applied to individual relations rather than entities, and whose number is usually several orders of magnitude smaller than the number of entities. Second, TransHR can capture entity-independent properties and thus might come closer to real knowledge.

The contributions of this paper are as follows: (1) We propose a new relation category based on the topologies of knowledge graphs to embed triplets more effectively, and (2) we propose a new model, TransHR, which outperforms previous models including Trans (E, H, R) and CTransR, with respect to link prediction and triple classification, as demonstrated experimentally.

Table 3 Number of relations.

| Data set   | Number of sole-relations | Number of hyper-relations |
|------------|--------------------------|----------------------------|
| FB38K      | 39                       | 568                        |
| FB15K[9]   | 390                      | 955                        |

This paper is organized as follows. We discuss a number of related models in Section 2 and describe our model TransHR in Section 3. We describe in detail an experimental study in Freebase, comparing TransHR with many other methods in Section 4, and suggest an application in Section 5. In Section 6, we draw our conclusions.

2 Related Work

Previous models of knowledge graph embedding can be divided into four main categories: translation-based embedding, neural network embedding, graph-based embedding, and other models. Before proceeding, we define the mathematical notations. We denote a triple by \((h, r, t)\) and their embedding vectors by the same letters in boldface, \(\mathbf{h}, \mathbf{r}, \mathbf{t}\). We represent a relation-specific matrix as \(M_r\) and express the score function of a triple as \(f_r(h, t)\).

2.1 Translation-based embedding

Translation-based models, which treat a relationship as a translation from a head entity to a tail entity, have a close connection with our model and we consider some of them as baselines in our experiments. As mentioned above, TransE[6] generates \(\mathbf{h} + \mathbf{r} \approx \mathbf{t}\) when \((h, r, t)\) holds, which indicates that \(t\) should be the nearest neighbor of \(h + r\). The score function is

\[
    f_r(h, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2^2
\]

However, this model has issues when modeling hyper-relational data. For example, as we see in Fig. 1, TransE tends to embed vectors \(r_1\) and \(r_2\) into the same vector, which is obviously inconsistent with the facts. The reason for this can be the insufficient representation of single space.

To improve this relationship classification problem related to the mapping properties of relations, several models have been proposed. TransH[7], an improvement of TransE, introduces the mechanism of projecting to the relation-specific hyperplane and enables entities that have different representations in different relations. Similar to TransE, TransH uses the following score function:

\[
    f_r(h, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2^2
\]

Although TransH embeds triplets by translating them onto the hyperplane, it does not break the restraint of single space. TransR[8] was developed to make...
an appropriate adjustment by modeling entities and relations in distinct spaces, i.e., entity space and multiple relation spaces, and then performs translation in the corresponding relation space. The score function is obtained by the following:

$$f_r(h,t) = \|hM_r + r - tM_r\|_2^2$$

(3)

where $M_r \in \mathbb{R}^{k \times d}$, $h, t \in \mathbb{R}^k$, and $r \in \mathbb{R}^d$. The motivation behind TransR boils down to the multiple aspects of each entity, in which different relations focus differently. However, typically, head-tail entity pairs manifest diverse patterns under a specific relation. So, CTransR was developed as an extension of TransR, which clusters diverse head-tail entity pairs into groups and learns distinct relation vectors for each group. To avoid overfitting, CTransR initializes entity and relation embeddings with the results of TransR. Then, as shown in Eq. (3), TransR and CTransR require two matrix-vector multiplications, which involve lengthy calculations and cannot be applied on large-scale knowledge graphs.

Since entities linked by a relation contain various types and attributes, a more fine-grained model TransD[12] was proposed as an improvement upon TransR and CTransR, which defines two vectors for each entity and relation, and uses them to generate a unique mapping matrix for every entity-relation pair. There are some other translation-based models inspired by other points, such as PTransE[9], which considers relation paths as translations between entities for representation learning and TranSparse[13], which explores sparse projection matrices to deal with the fact that entities and relations are heterogeneous and unbalanced. K2GE[14], which uses density-based embedding, was proposed for explicitly modeling the certainty of entities and relations and to learn knowledge graph embedding in the spaces between multi-dimensional Gaussian distributions. Each entity or relation is represented by a Gaussian distribution, in which the mean denotes its position and the covariance properly represents its certainty. TransG[15] leverages a Bayesian non-parametric infinite mixture model to handle multiple relation semantics by generating multiple translation components for a relation.

### 2.2 Neural network embedding

The Structured Embedding (SE) model of Ref. [16] introduces two independent projections to entities in a relation and measures the $L_1$ distance between these two projections to score relationships. The main problem with this model is that the parameters of the two entity vectors do not interact, as they are independently mapped to a common space. As pointed out by the authors in Ref. [17], this model performs weakly in capturing correlations between entities and relations as it uses two separate matrices. The Single-Layer Model (SLM) tries to alleviate the problems of the SE by connecting the entity vectors implicitly through nonlinear transformations of a standard, single-layer neural network, using the score function $f_r(h, t) = u_r^T g(M_r h + M_r t)$, where $M_{r,1}$ and $M_{r,2}$ are relation-specific weight matrices. While this represents an improvement over SE, the non-linear function provides only a weak interaction between the two entity vectors. The latent factor model[18] considers second-order correlations between entity embeddings so that each component of an entity interacts with each component of the other entity. The Semantic Matching Energy (SME) model[10, 19] represents each relation using a single vector in the same embedding space as the entities. Interaction between entities and relation types are performed via linear matrix products. Neural Tensor Networks (NTNs)[17] replace the standard linear neural network layer with a bilinear tensor layer that directly relates two entity vectors across multiple dimensions. However, neural network embedding methods do not explicitly address the relation classification problem.

### 2.3 Graph-based embedding

Many research approaches to knowledge representation in Knowledge Bases (KBs) have taken graph-based methods into consideration, making some kind of inference regarding graphs, e.g., knowledge base completion. Markov logic networks[20] fall into this category, as does ProPPR[21] and many other logic-based systems. The Path Ranking Algorithm (PRA)[4, 22] also fits into this category. PRA work has ranged from incorporating a parsed corpus as additional evidence for random walk inference[23], to introducing better representations of the text corpus[24, 25], to using PRA in a broader context as part of Googles Knowledge Vault[22]. Other interesting work that combines embedding methods with graph-based methods is that by the authors in Ref. [26], which uses a Recurrent Neural Network (RNN) to create embedded representations of PRA-style paths. The authors in Ref. [27] improved this model by accounting for both relations and entities and training a single high-
capacity RNN to compose Horn-clause chains across all predicted relation types.

2.4 Other models

Before the emergence of embedding with semantic features, a number of models were used for knowledge representation, such as clustering and tensor factorization methods. The infinite relational model\(^{[28]}\), the first of the clustering approaches, is a nonparametric Bayesian model that uses latent variables to discover meaningful partitions among entities and relation types. To improve the predictive power of this model, the authors in Ref. \([29]\) proposed multiple relation clustering, which considers several relations simultaneously while clustering entities. Then, the authors in Ref. \([30]\) proposed the Bayesian tensor clustered factorization model, which requires multiple embeddings per entity. Although they both provide data interpretation, the clustering approaches must rely on inference and cannot be scaled to very large databases. A typical model based on tensor factorization is RESCAL\(^{[31]}\), which interprets entities as low-dimensional vectors and represents relation types by low-rank matrices. Despite the fact that both clustering and tensor factorization methods provide a distributed representation of entities, they do not contain any semantic information and relations are obtained via factorizing or clustering the original data.

3 Translating for Hyper-Relational Data

As noted above, the challenges associated with hyper-relation embedding have been implicitly addressed by existing approaches by decomposing entities into multiple semantic parts, each of which corresponds to an individual relation, rather than directly manipulating the hyper-relation as a whole\(^{[6-8]}\). Our model differs from previous works mainly in its straightforward approach to hyper-relation by learning its own projection matrix to preserve its integrated semantic embedding. To illustrate the embedding problem, we categorize relations into two classes: sole-relation and hyper-relation.

3.1 Relation category

Previous models have usually adopted two kinds of relation category. The first categorizes the relations into four classes: 1-to-1, 1-to-many, many-to-1, and many-to-many, according to the cardinalities of their head and tail arguments. For instance, if the average number of tails per head is greater than 1.5 and the average number of heads per tail is less than 1.5, the relation will be treated as 1-to-many. However, this result might not always be consistent with fact and might be somewhat complex. The other approach\(^{[13]}\) categorizes relations into two classes: complex and simple relations, whereby the complexity of a relation is proportional to the number of triplets (or entities) linked by it. Although this approach reduces the number of relation classes, it can be hard to distinguish due to its fuzzy definition.

In this paper, we categorize relations into sole- and hyper-relations. For all entity pairs linked by a relation \(r\), if there exists no other relation between them, i.e., the relation \(r\) exists alone between those entity pairs, we call it a sole-relation. Otherwise, when other relations between a pair of entities appear simultaneously, it is called hyper-relation. We denote all triples \((h, r, t)\) in the knowledge graph as \(T\), the whole relation set as \(R\), the sole-relation set as \(R_s\), and the hyper-relation set as \(R_h\). \(R_r\) denotes the whole relation set \(R\) which has eliminated the relation \(r\). Then, we can define sole- and hyper-relations as follows:

\[
R_s = \{r | (h, r, t) \in T \land \forall r' \in R_r, (h, r', t) \notin T\},
\]

\[
R_h = \{r | (h, r, t) \in T \land \exists r' \in R_r, (h, r', t) \in T\}.
\]

By the above definitions, we find our relation category to be definitive and always consistent with fact.

3.2 Energy-based framework

The algorithm TransHR is learned under the supervision of the general energy-based framework\(^{[14]}\). Specifically, each triple in the knowledge graph is first mapped by a score function to a real value, called the score of the raw data, and this score is used to estimate the agreement between the predicted triple and the ground truth by a loss function. In general, the design of the score function usually depends on a specific target, such as knowledge graph completion or link prediction. For example, correct triples should have a lower score than incorrect ones. The selection of the loss function \(L\) is usually determined by the definition of the score and a set of parameters, which must be learned by the algorithm. This loss is generally obtained by comparing the scores of positive and negative samples, also referred to as ranking loss or hinge loss. Training of the algorithm TransHR induces a particular set of appropriate parameters to finally obtain a low loss, which indicates a good agreement.
between the model output and the ground truth. Since the loss function quantifies the quality of the set of parameters, the parameters are then generally optimized by the minimization of the loss function gradient. This optimization process iteratively continues until the loss of the model converges.

3.3 TransHR

When modeling hyper-relational data, existing models are either unable to distinguish hyper-relations between pairs of entities or have concentrated on breaking the entities into several parts, which does not identify properties of entity-independence and might work against generalization. Intuitively, it is more reasonable to model hyper-relations in a pair of entities as an individual vector rather than to disconnect the diverse aspects of the entities. Hence, we propose TransHR, which transforms vectors of the hyper-relations in a pair of entities from the relation space into one vector that performs as a translation in the entity space. Since we do not destroy any independent entity or relation, our model may more closely reflect real knowledge.

In TransHR, for each triple \((h, r, t)\), we denote entity embeddings by \(h, t \in \mathbb{R}^k\) and relation embedding by \(r \in \mathbb{R}^d\). Note that the dimension of the entity embeddings is not necessarily equal to the dimension of the relation embeddings, i.e., \(k \neq d\).

Assume that there are \(n\) relations between a pair of entities and the \(i\)-th relation is denoted by \(r_i (i = 1, 2, ..., n)\). To project relations vectors from the relation space to the entity space, we denote a transfer matrix \(M_r \in \mathbb{R}^{d \times k}\) for each relation \(r_i\). With these transfer matrices, we can obtain the projected vector by the following:

\[
r_{(h, t)} = r_i M_r, \quad i = 1, 2, ..., n \tag{4}
\]

where \(r_{(h, t)} \in \mathbb{R}^k\). Therefore, each relation-specific matrix \(M_r\) transforms its corresponding \(d\)-dimensional vector \(r_i\) into an individual \(k\)-dimensional vector \(r_{(h, t)}\), thereby acting as a translation between the head entity vector \(h\) and the tail entity vector \(t\). Then, we can correspondingly define the score function as follows:

\[
f_r(h, t) = \|h + r_{(h, t)} - t\|_2^2 \tag{5}
\]

where function \(f_r\) is a dissimilarity measure, which we take as either \(L_1\) or \(L_2\)-norm. Hence, \(f_r(h, t)\) indicates the dissimilarity of \(h + r_{(h, t)}\) and \(t\).

To illustrate how to solve the problem of modeling hyper-relational data, we take the two triples in Fig. 1 as an example. According to Eqs. (4) and (5), the score functions of these triples are as follows:

\[
f_{r_1}(h, t) = \|h + r_1 M_r - t\|_2^2 \tag{6}
\]

\[
f_{r_2}(h, t) = \|h + r_2 M_r - t\|_2^2 \tag{7}
\]

Since both \(f_{r_1}(h, t)\) and \(f_{r_2}(h, t)\) represent the dissimilarity of \(h + r_{(h, t)}\) and \(t\), their values should be equal. As such, we can conclude the following:

\[
r_1 M_r = r_2 M_r \Rightarrow r_{(h, t)} \tag{8}
\]

In Eq. (8), the hyper-relation vectors \(r_1\) and \(r_2\) are each finally transformed into a vector \(r_{(h, t)}\) (actually representing the vector \(t - h\)), respectively, by their transfer matrices \(M_r\). Thus, TransHR models hyper-relational data by expanding the parameter space so that the relations have more opportunity to be embedded into the correct position in the vector space.

Actually, TransE is a special case of our model, in which \(M_r\) can be treated as an identity matrix. This means that the entity space and the relation space are rolled into one space, which is insufficient for embedding both entities and relations. TransH also suffers from the problem of modeling in single space. Though TransR and CTransR embed in different semantic spaces, they ignore the properties of entity-independence and are at least twice as slow as TransHR. This is true because: (1) TransR and CTransR contain two matrix-vector multiplications while TransHR only has one, and (2) the two matrix-vectors of TransR and CTransR are applied to many more entities than the number of relations to which TransHR is applied.

3.4 Training

To train the parameters of the score function \(f_s(h, t)\), we follow the method in Ref. [6], which minimizes the margin-based score function as its training objective:

\[
L = \sum_{(h, r, t) \in S} \sum_{(h', r', t') \in S'} \max(0, f_s(h, t) + \gamma - f_s(h', t')) \tag{9}
\]

where \(\max(0, x)\) maximizes the margin between 0 and \(x\) and \(\gamma > 0\) is a margin hyperparameter (which generally takes the value 1) separating positive and negative triples, which is commonly used in margin-based models such as SVM\(^{[32]}\). As the embeddings of entities and relations are normalized, the margin \(\gamma\) can actually regularize the above objective and keep the weights from collapsing or deviating. \(S\) denotes the set of positive triples and \(S'\) denotes the set of negative triples, which are generated artificially. \(f_s(h, t)\) is the score of the positive triple, and \(f_s(h', t')\) denotes the score of corresponding negative
triple. The loss function (9) prefers lower positive triple scores than negative triple scores. This learning process is performed by Stochastic Gradient Descent (SGD). If \( f_r(h, t) > f_r(h', t) - \gamma \), SGD is performed to minimize the objective function.

### 3.5 Implementation on Mapreduce

Due to the performance of matrix multiplication (in which a relation vector multiplies its corresponding transfer matrix) and the large number of entities (usually up to hundreds of thousands), the computational complexity of TransHR is relatively expensive. Thus, during the process of testing data to determine the strength of the learned model, we use an implementation available on the distributed computing platform Hadoop, which can speed up the calculation approximately one hundred times that of the original central version of the algorithm.

Here, we exploit Hadoop streaming, which allows the use of any executable or script files as a mapper and reducer. The streaming tool creates a Mapreduce job, sends it to each task tracker, and then monitors the execution of the entire job. Figure 2 shows the framework of the distributed implementation for the algorithm test data evaluation, wherein test data with containing 100 triples is partitioned into, for instance, ten divisions, for the sake of illustration convenience.

The input of the distributed framework is the test data arranged with each triple in a line. First, we split the entire test data file into several subfiles. The lines of each subfile (each with 10 triples in Fig. 2) can be randomly chosen. Fortunately, each node (task tracker) in Mapreduce is independent, so there is no cost with respect to overhead traffic. To take full advantage of the multi-tasktrackers’ distributed parallel computing, subfiles must be generated with the same lines. Empirically, each line consumes several minutes to calculate results for our test data. Note that the time is not clearly fixed and strongly depends on the configuration and the actual resources allocation of the hadoop cluster. Next, the mapper and reducer will read the data from the standard input, deal with the data line by line, and send the result to the standard output. During mapping, the algorithm for the test data is executed to compute the result of each triple and during reduction, we do nothing but output the results of the mapper. Finally, we obtain the resulting rank value of each triple from the output of the framework. Obviously, whereas the reducer in Fig. 2 is designed very simply, functions such as estimating the average rank could be easily deployed on the reducer as needed to perform more statistical tasks on the test data using our proposed framework.

There must be a one-to-one correspondence between each input triple and its output rank. However, the shuffling procedure between mapping and reducing can change the order of the output data with respect to the input data, which means that the input triple cannot then find its corresponding output rank. To solve this problem, we set an identifier for each line before splitting the test file. For example, we record the line number before each line (as shown in Fig. 2). This identifier is taken during each stage of Mapreduce and is displayed in the final rank to identify the result as that of a certain triple (as shown in the final rank column in Fig. 2, where the output after reduction is disordered and is then sorted in the final rank column).

### 4 Experiments and Analysis

We conducted our experiments on data extracted from Freebase. Our task was to predict missing entities \( h \) or \( t \) for a correct triple (link prediction) and determine whether the triple is correct or not (triple classification).

---

**Fig. 2** Framework of the Mapreduce implementation for the TransHR model testing data.
Below, we describe the data sets and compare the performance of our model with respect to the above tasks with those of current state-of-the-art methods.

4.1 Data sets

As shown in Table 2, we extracted two kinds of data sets, one from Wordnet[1] and the other from Freebase[2]. Wordnet is a linguistic knowledge graph whose entities (termed synsets) correspond to word senses, and in which relations are defined between synsets by pointing to their lexical relations, e.g., (itinerant_NN, _hyponym, _swagman_NN). Freebase is a world knowledge graph that encodes general world facts. For example, the triple (Obama, born here, USA) in Freebase builds a relation born here as a translation from the name entity Obama to the location entity USA.

Table 2 lists five knowledge graph data sets, not all of which are appropriate for our experiment. The released FB40K[9] contains many relations that appear only once or twice. This low relation frequency will impact the performance evaluation, and especially the hyper-relation embedding. To solve this problem, we extracted triplets from FB40K with relations occurring more than ten times to construct the FB38K data, in which the ratio of the hyper-relations is much higher than that in FB40K. Meanwhile, the population of the remaining entities was slightly reduced from $40 \times 10^3$ to $38 \times 10^3$, which does not significantly affect the scalability of the model. The relative high density of the hyper-relations contained in the dataset suggests the possibility for a more convincing performance comparison with our model. For the same reason, we also selected FB15K for the experiment since it contains many hyper-relations, as shown in Table 1. Table 4 shows statistics relating to FB38K and FB15K.

4.2 Implementation

We utilized the codes in C++, which we downloaded from https://github.com/mrlyk423/relation_extraction. We obtained TransHR by modifying the codes, which we compare with Trans (E, H, R) and CTransR below. We conducted the experiments on a Ubuntu server with an Intel Xeon (R) CPU E5-26200 (2.00 GHz) and 12 GB RAM.

During the training phase, we used SGD for optimization. Following the procedure in Ref. [6], we initialized entity and relation embeddings with the same random procedure proposed in Ref. [33]. To avoid overfitting, we initialized the transfer matrix $M_f$ as an identity matrix. In each main iteration of the algorithm, we first randomly sampled a matrix $M_f$ as an identity matrix. In each main iteration of the algorithm, we first randomly sampled a triple from the training set to serve as the positive triple, and sampled a single corrupted triple as the negative triple. Then, we normalized the embedding vectors of the entities and relations. We updated the parameters by taking a gradient step with a constant learning rate and stopped the algorithm based on its performance on a validation set.

We selected the learning rate $\alpha$ from $\{0.001, 0.005, 0.01\}$, the margin $\gamma$ from $\{0.1, 0.5, 1, 2\}$, and the dimensions of entity vectors, $k$, and relation vectors, $d$, from $\{20, 50, 80, 100\}$. We found the best configuration to be: $\alpha = 0.001, \gamma = 1, m, n = 100$, taking $L_1$ as a dissimilarity. For both datasets, we performed training for 500 rounds.

4.3 Link prediction

The goal of link prediction is to complete the missing entities $h$ or $t$ to generate a positive triple $(h, r, t)$, i.e., complete $t$ given $(h, r)$ or complete $h$ given $(r, t)$. Rather than giving one best result, we rank a set of candidate entities from the knowledge graph.

Evaluation Setup. Following the same setup as that in Ref. [8], for each test triple $(h, r, t)$, $h$ or $t$ is deleted and substituted by each entity in the knowledge graph in proper sequence. Then, the models calculate the scores of those corrupted triples and ranks them in ascending order, thereby recording the rank of the correct entity. Similar to Ref. [6], we report the mean rank of the predicted ranks and the hits@10, i.e., the ratio of ranks in the top ten. This evaluation setting is labeled “Raw”, which may be unfair to the models because the knowledge graph may contain corrupted triples and rank them before finding the original correct entity should not be regarded as wrong. Hence, before ranking, we must eliminate corrupted triples appearing in the knowledge graph. We named the new evaluation setting “Filter”. For both settings, a lower mean rank and a higher hits@10 value is the mark of a better model.

Results Analysis. Table 5 shows the overall evaluation results for link prediction, from which we
can conclude that TransHR consistently outperformed its counterparts in both FB15K and FB38K. TransR and CTransR are in second place, whereas TransE and TransH generated the worst results, which may prove that a single space is insufficient for modeling hyper-relational data. Upon investigation of the experiment results of TransD[12] in FB15K, we found them to be approximately comparable with those of our algorithm. In addition, Tables 6 and 7 show detailed results from different points of view. Table 6 shows the sole- and hyper-relation results in FB38K, and Table 7 shows the individual hits@10 results of the head and tail predictions. To summarize, TransHR performs either the best or very close to it, and can be considered to be the best performing model on average.

However, TransHR is primarily designed for modeling hyper-relational data and we hypothesis that the evident improvements are due to its advantage in modeling hyper-relational data. To confirm this point, we examined the results of the most frequently occurring ten hyper-relations, as shown in Table 8. We can see that TransHR realizes significant improvements most of the time. For instance, the relation “/location/location/contains” appears 20,597 times in the training set and TransHR achieved an accuracy of 97.8% (an increase of three percentage points) in predicting the head and 78.2% (an increase of 30 percentage points) in predicting the tail. Although the results of CTransR are close to ours and sometimes even better than ours, TransHR is at least twice as fast as CTransR, since CTransR has two matrix-vector

### Table 5 Evaluation results for link prediction.

| Data set | Model | Mean rank Raw | Hits@10 (%) Raw Filter |
|----------|-------|---------------|------------------------|
| FB15K    | TransE[6] | 243 125 | 34.9 47.1 |
|          | TransH[7]  | 211 84  | 42.5 58.5 |
|          | TransR[8]  | 226 78  | 43.8 65.5 |
|          | CTransR[8] | 233 82  | 44 66.3 |
|          | TransHR    | 209 67  | **47.8 70.0** |
| FB38K    | TransE[6] | 564 197  | 55.3 76.7 |
|          | TransH[7]  | 560 190 | 54.6 66.1 |
|          | TransR[8]  | 567 189 | 58.6 73.7 |
|          | CTransR[8] | 568 186 | 59.5 76.7 |
|          | TransHR    | 559 183 | **59.6 76.5** |

### Table 6 Results for FB38K by relation category.

| Relation category | Model | Mean rank Raw | Hits@10 (%) Raw Filter |
|-------------------|-------|---------------|------------------------|
| Sole-relation     | TransE[6] | 514 159 | **51.2 59.3** |
|                   | TransH[7]  | 544 172 | 50.6 56.5 |
|                   | TransR[8]  | 503 112  | 49.8 63.2 |
|                   | CTransR[8] | 517 103 | 50.0 66.0 |
|                   | TransHR    | 475 92  | **51.2 65.8** |
| Hyper-relation     | TransE[6] | 565 198  | 55.4 67.5 |
|                   | TransH[7]  | 565 196 | 54.7 66.2 |
|                   | TransR[8]  | 568 191 | 58.8 74.0 |
|                   | CTransR[8] | 569 188 | 60.0 77.1 |
|                   | TransHR    | 561 185 | **59.8 76.8** |

### Table 7 Hits@10 (%) for FB38K.

| Task (Hits@10) | Model | Sole-relation | Hyper-relation |
|----------------|-------|---------------|----------------|
| Predicting head | TransE[6] | 57.1 63.9 | 53.3 65.6 |
|                | TransH[7]  | 56.2 62.3 | 52.5 64.1 |
|                | TransR[8]  | 55.5 68.4 | 56.8 72.6 |
|                | CTransR[8] | 54.7 68.4 | **58.5 76.1** |
|                | TransHR    | 56.3 **70.4** | 57.0 74.7 |
| Predicting tail | TransE[6] | 45.3 54.7 | 57.6 69.5 |
|                | TransH[7]  | 44.9 50.6 | 57.0 68.3 |
|                | TransR[8]  | 44.1 57.9 | 60.9 75.5 |
|                | CTransR[8] | 44.9 57.1 | 62.0 78.5 |
|                | TransHR    | **46.2 61.1** | 62.7 78.9 |

### Table 8 Hits@10 for Trans (E, H, R), CTransR, and TransHR of the top ten frequently occurring hyper-relations.

| Relation                               | Frequency | Head Hits@10 (TransE/TransH/TransR/CTransR/TransHR) (%) |
|----------------------------------------|-----------|--------------------------------------------------------|
| /people/person/nationality             | 21,496    | 2.5 / 3.4 / 3.4 / 8.5 / 3.9 | 28.3 / 27.3 / 30.2 / 27.0 / 45.9 |
| /location/location/contains           | 20,597    | 95.0 / 94.4 / 96.0 / 95.9 / 97.8 | 52.5 / 53.2 / 71.8 / 74.5 / 78.2 |
| /location/location/containsby         | 20,578    | 46.4 / 47.3 / 67.3 / 71.0 / 72.5 | 95.5 / 96.2 / 91.0 / 95.0 / 98.3 |
| /people/person/place_lived             | 14,146    | 47.5 / 42.2 / 53.7 / 53.9 / 61.1 | 86.6 / 86.4 / 92.0 / 94.5 / 93.8 |
| /people/place_lived/location           | 14,119    | 87.4 / 87.4 / 93.5 / 96.3 / 94.6 | 50.3 / 46.9 / 55.7 / 60.5 / 66.2 |
| /location/location/people_born_here   | 13,715    | 91.4 / 90.0 / 96.3 / 97.9 / 97.1 | 57.5 / 56.0 / 61.9 / 67.0 / 73.2 |
| /people/person/place_of_birth          | 13,607    | 57.5 / 58.2 / 62.0 / 68.9 / 69.6 | 91.9 / 91.3 / 96.6 / 97.8 / 96.9 |
| /education/education/institution       | 11,739    | 98.8 / 98.5 / 98.8 / 99.2 / 100 | 77.3 / 73.5 / 85.8 / 94.2 / 97.3 |
| /education/educational_institution/students_graduates | 11,609 | 99.2 / 97.5 / 99.0 / 99.5 / 100 | 78.4 / 76.9 / 88.2 / 95.0 / 96.0 |
| /education/education/student           | 11,556    | 76.0 / 76.0 / 89.0 / 94.2 / **95.1** | 99.7 / 98.3 / **99.7 / 99.7 / 99.7** |
multiplications whereas TransHR has only one and the matrices of CTransR apply to entities whose number is much larger than relations in the matrix to which TransHR is applied.

4.4 Triple classification

In this task, we identify whether a given triple \((h, r, t)\) is correct or not, based on a binary classification of each triple, as in Ref. [11]. This metric determines accuracy based on how many triples are identified correctly.

**Evaluation Setup.** We used FB38K in this task and followed the same settings as in NTN[11]. As knowledge graphs have only positive triples and classification evaluation requires negative labels, we corrupted each positive triple in the selected testing set to create a corresponding negative triple to generate a total set of double testing triples with equal numbers of positive and negative examples. When generating the negative triples, we followed the approach in Ref. [11], in which only the entities appearing in the same position of the replaced entity in the data set are qualified for the possible answer set. For each relation, we produced a homologous entity set. Since we only replaced each tail entity of one positive triple to generate its negative counterpart, the homologous entity set contains all the tail entities appearing after this relation. This forces the model to focus on harder cases and makes the evaluation more difficult since it does not include obvious non-relations.

We equipped each relation with a relation-specific threshold \(\delta_r\). The decision rule for the classification is simple: for a triple \((h, r, t)\), if the dissimilarity score (by the score function \(f_r\)) is below \(\delta_r\), then predict positive, otherwise predict negative. We determine the relation-specific threshold \(\delta_r\) by the classification accuracy of the validation set.

We compared our model with the same models used in the link prediction, which were also optimized with SGD. Note that the best settings of the entity vector dimension \(k\) and relation vector dimension \(d\) here are 50.

**Results.** Table 9 shows the evaluation results of the triples classification and we can see that TransHR outperformed all the other methods.

| Model     | Classification (%) |
|-----------|--------------------|
| TransE[6] | 80.6               |
| TransH[7] | 78.6               |
| TransR[8] | 80.2               |
| CTransR[9] | 70.4               |
| TransHR | **81.8**           |

5 Application

We have shown that TransHR can achieve high performance for knowledge graph embedding. In this section, we present an application of TransHR: Question Answering (QA), which explores large-scale knowledge graphs to semantically understand users’ queries and then rank reasonable answers.

QA is defined as a task of retrieving the correct entity or set of entities from a knowledge graph, given a query expressed as a question in natural language. Traditional methods answer questions by learning to map questions to logical forms or database queries[34–37]. Even if such systems are effective, they require large amounts of human-labeled data to define lexicons and grammars, which might be insufficiently generic for new large-scale knowledge graphs with other grammars. In this section, we follow previous models[38, 39] to convert questions and answers to vectorial feature representations without requiring any grammar or lexicon.

Figure 3 illustrates how TransHR works in QA. First, TransHR must be trained by large-scale triples and then generates an entity-to-vector (each entity in training triples and its corresponding vector) and a relation-to-vector (each relation in training triples and its corresponding vector). For the question “where was Obama born?” , we detect the entity Obama and the relation born in the question[40] and find their corresponding vector representations from the...
training result of the TransHR, for which the entity-to-vector and relation-to-vector are already obtained, respectively. Here we assume that there is one and only one entity and one relation in the training triples appears in each question, and that all the answer entities should be included in the training triples. The entity vector and the relation vector are taken as the inputs of TransHR and the problem of answering a question becomes a problem of link prediction. TransHR must then predict the missing \( t \) for the given \((h, r)\). To do this, TransHR regards each entity \( t_i \) in the training triples as the missing \( t \) and calculates the score of each \((h + rM_r - t_i)\) sequentially, then ranks the scores in ascending order to identify the top ten closest candidate answers (USA, Canada, UK, ...), which are the final results generated by TransHR for the question.

6 Conclusion

In this paper, we proposed a novel knowledge graph embedding model TransHR for modeling hyper-relational data. TransHR transforms the vectors of hyper-relations between a pair of entities from the relation space into an individual vector that serves as a translation in the entity space. Experiments on the tasks of link prediction and triple classification show that TransHR achieves promising improvements compared to the results of Trans (E, H, R) and CTransR. In addition, we found the relation category we introduced in this paper to be effective.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (Nos. 61302077, 61520106007, 61421061, and 61602048).

References

[1] G. A. Miller, WordNet: A lexical database for English, *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, Vancouver, Canada, 2008, pp. 1247–1250.
[3] M. S. Fabian, K. Gjergji, and W. Gerhard, Yago: A core of semantic knowledge unifying wordnet and wikipedia, presented at the 16th International World Wide Web Conference, WWW, Banff, Canada, 2007.
[4] N. Lao, T. Mitchell, and W. W. Cohen, Random walk inference and learning in a large scale knowledge base, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Edinburgh, UK, 2011, pp. 529–539.
[5] W. Y. Wang, K. Mazaitis, N. Lao, and W. W. Cohen, Efficient inference and learning in a large knowledge base, *Machine Learning*, vol. 100, no. 1, pp. 101–126, 2015.
[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, Translating embeddings for modeling multi-relational data, presented at Advances in Neural Information Processing Systems, Lake Tahoe, CA, USA, 2013.
[7] Z. Wang, J. Zhang, J. Feng, and Z. Chen, Knowledge graph embedding by translating on hyperplanes, presented at the Association for the Advance of Artificial Intelligence, Quebec City, Canada, 2014.
[8] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, presented at the Association for the Advance of Artificial Intelligence, Austin, TX, USA, 2015.
[9] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, Modeling relation paths for representation learning of knowledge bases, arXiv preprint arXiv:1506.00379, 2015.
[10] 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.
[11] R. Socher, D. Chen, C. D. Manning, and A. Ng, Reasoning with neural tensor networks for knowledge base completion, presented at the Advances in Neural Information Processing Systems, Lake Tahoe, CA, USA, 2013.
[12] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, Knowledge graph embedding via dynamic mapping matrix, presented at the Association for Computational Linguistics, Beijing, China, 2015.
[13] G. Ji, K. Liu, S. He, and J. Zhao, Knowledge graph completion with adaptive sparse transfer matrix, presented at the Association for the Advance of Artificial Intelligence, Phoenix, AZ, USA, 2016.
[14] S. He, K. Liu, G. Ji, and J. Zhao, Learning to represent knowledge graphs with gaussian embedding, in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, Melbourne, Australia, 2015, pp. 623–632.
[15] H. Xiao, M. Huang, Y. Hao, and X. Zhu, TransG: A generative mixture model for knowledge graph embedding, arXiv preprint arXiv:1509.05488, 2015.
[16] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, Learning structured embeddings of knowledge bases, presented at the Association for the Advance of Artificial Intelligence, San Francisco, CA, USA, 2011.
[17] D. Chen, R. Socher, C. D. Manning, and A. Y. Ng, Learning new facts from knowledge bases with neural tensor networks and semantic word vectors, arXiv preprint arXiv:1301.3618, 2013.
[18] R. Jenatton, N. L. Roux, A. Bordes, and G. Obozinski, A latent factor model for highly multi-relational data, presented at Advances in Neural Information Processing...
Xiao Han is a PhD candidate of School of Information and Communication Engineering at the Beijing University of Posts and Telecommunications (BUPT). She received the MEng degree in information and communication engineering from BUPT in 2014. Her research interest includes deep learning and distributed representation of knowledge graph.

Miao Zhou is a master student of School of Information and Communication Engineering at the Beijing University of Posts and Telecommunications (BUPT). She received the BEng degree from BUPT in 2014. Her research interests include deep learning and distributed representation of knowledge graph.
Chunhong Zhang is a lecturer with School of Information and Communication Engineering at the Beijing University of Posts and Telecommunications (BUPT). She received the PhD degree in computer science from BUPT in 2013, MEng degree in information technology from BUPT in 1996, and BEng degree from BUPT in 1993. She was a visiting scholar at Illinois Institute of Technology in 2015. Her research interest includes deep learning, ubiquitous computing, and data mining. Her research has been supported by Ministry of Science and Technology of China and the National Natural Science Foundation of China.

Zheng Hu is working at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT), China, and with the Institute of Sensing Technology and Business, BUPT, Wuxi, China. He received the PhD degree in circuit and system from BUPT in 2008 and BEng degree from Nanjing University of Posts and Telecommunications in 2002. He is engaging in the research on the ubiquitous networking and service computing. His current interests lie in the user behavior modeling and analysis in mobile internet and social networks.

Yang Ji is now a professor with Beijing University of Posts and Telecommunications (BUPT). He received the PhD degree in electronic engineering from BUPT in 2002. His research interest includes ubiquitous computing, web information system, and network technology. He has conducted a lot of significant projects with domestic industry and academic partners with the support from Ministry of Science and Technology, Ministry of Industry and Information Technology of China, and he also had a lot of cooperation with international partners with the support from EU commission in the past years.