Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning

Wen Zhang
College of Computer Science,
Zhejiang University
wenzhang2015@zju.edu.cn

Bibek Paudel*
Stanford University
& University of Zürich
paudel@ifi.uzh.ch

Liang Wang
College of Computer Science,
Zhejiang University
21621254@zju.edu.cn

Jiaoyan Chen
Department of Computer Science,
University of Oxford
jiaoyan.chen@cs.ox.ac.uk

Hai Zhu
Alibaba Group
China
marvin.zh@alibaba-inc.com

Wei Zhang
Alibaba Group & AZFT Joint Lab for
Knowledge Engine
lantu.zw@alibaba-inc.com

Abraham Bernstein
Department of Informatics,
University of Zürich
Switzerland
bernstein@ifi.uzh.ch

Huajun Chen†
Zhejiang University & AZFT Joint
Lab for Knowledge Engine
China
huajunsir@zju.edu.cn

ABSTRACT

Reasoning is essential for the development of large knowledge graphs, especially for completion, which aims to infer new triples based on existing ones. Both rules and embeddings can be used for knowledge graph reasoning and they have their own advantages and difficulties. Rule-based reasoning is accurate and explainable but rule learning with searching over the graph always suffers from efficiency due to huge search space. Embedding-based reasoning is more scalable and efficient as the reasoning is conducted via computation between embeddings, but it has difficulty learning good representations for sparse entities because a good embedding relies heavily on data richness. Based on this observation, in this paper we explore how embedding and rule learning can be combined together and complement each other's difficulties with their advantages. We propose a novel framework IterE iteratively learning embeddings and rules, in which rules are learned from embeddings with proper pruning strategy and embeddings are learned from existing triples and new triples inferred by rules. Evaluations on embedding qualities of IterE show that rules help improve the quality of sparse entity embeddings and their link prediction results. We also evaluate the efficiency of rule learning and quality of rules from IterE compared with AMIE+, showing that IterE is capable of generating high quality rules more efficiently. Experiments show that iteratively learning embeddings and rules benefit each other during learning and prediction.

KEYWORDS

knowledge graph; reasoning; embedding; rule learning

ACM Reference Format:
Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Many Knowledge Graphs (KGs), such as Freebase [2] and YAGO [33], have been built in recent years and led to a broad range of applications, including question answering [4], relation extraction [36], and recommender system [49]. KGs store facts as triples in the form of *(subject entity, relation, object entity)*, abridged as (s, r, o). Some KGs also have an ontology with class and property expression axioms which place constraints on classes and types of relationships.

Knowledge graph reasoning (KGR) can infer new knowledge based on existing ones and check knowledge consistency. It is attracting research interest and is important for completing and cleaning up KGs. Two of the most common learning methods for KGR are embedding-based reasoning and rule-based reasoning [26].

One of the crucial tasks for embedding-based and rule-based reasoning is to learn embeddings and rules respectively. *Embedding learning methods* such as TransE [3], HolE [27] and ComplEx [35] learn latent representations of entities and relations in continuous vector spaces, called *embeddings*, so as to preserve the information and semantics in KGs. Embedding-based reasoning is more efficient when there are a large number of relations or triples to reason over. *Rule learning methods* such as AMIE[10] aim to learn deductive and interpretable inference rules. Rule-based reasoning is precise and can provide insights for inference results.

*Work done while at UZH.
†Corresponding author.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

With different advantages, both embedding and rule learning are important for KGR, while they still have their own difficulties and weaknesses.

Sparsity Problem for Embedding Learning. One of the main difficulties for embedding learning is the poor capability of encoding sparse entities. For example, Figure 1 shows correlation between entity frequency and entity link prediction results measured in mean reciprocal rank (MRR), where higher values means better results. In Figure 1, the blue line shows there are a large portion of entities having only a few triples, revealing the common existence of sparse entity. The yellow line shows that the prediction results of entities are highly related to their frequency, and the results of sparse entities are much worse than those of frequent ones.

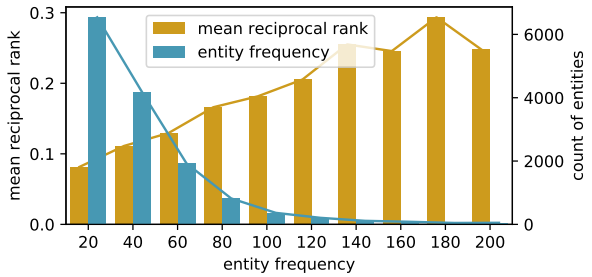

Figure 1: The numbers and mean reciprocal rank of different frequency entities based on ANALOGY results on FB15k-237.

Efficiency Problem for Rule Learning. The main difficulty in rule learning is the huge search space when determining rule structures and searching for support triples. For example, with a small KG containing 10 relations and 100 entities, the number of possible structures for a rule with 3 relations is 10^3 and the maximum number of supports for these rules is $100^{2*3} * 10^3 = 10^{15}$. Since the search space is exponential to the number of relations, it will be much larger for real KGs than this example.

With different advantages and difficulties, we argue that embedding learning and rule learning can benefit and complement each other. On the one hand, deductive rules can infer additional triples for sparse entities and help embedding learning methods encode them better. On the other hand, embeddings encoded with rich semantics can turn rule learning from discrete graph search into vector space calculation, so that reduce the search space significantly. Thus we raise the research question: **whether it is possible to learn embeddings and rules at the same time and make their advantages complement to each other's difficulties.**

In this paper, we propose a novel framework **IterE** that iteratively learns embeddings and rules, which can combine many embedding methods and different kinds of rules. Especially, we consider linear map assumption (Section 2.1) for embedding learning because it is inherently friendly for rule learning as there are special rule conclusions for relation embeddings in rules (Table 2). We also consider a particular collection of object property axioms defined in OWL2 (Table 1) for rule learning considering that semantics included in web ontology language are important for the development of knowledge graph.

IterE mainly includes three parts: (1) embedding learning, (2) axiom induction, and (3) axiom injection. Embedding learning learns embeddings for entities and relations, with input including triples existing in KG and those inferred by rules. Axiom induction first generates a pool of possible axioms with an effective pruning strategy proposed in this paper and then assigns a score to each axiom in the pool based on calculation between relation embeddings according to rule conclusions from linear map assumption. Axiom injection utilizes axioms' deductive capability to infer new triples about sparse entities according to axiom groundings and injects these new triples into KG to improve sparse entity embeddings. These three parts are conducted iteratively during training.

We evaluate IterE from three perspectives, 1) whether axiom improves sparse embeddings' quality and their predictions, 2) whether embedding helps improve rule learning efficiency and quality, and 3) how iterative training improves both embedding and rule learning during training. The experiment results show that IterE achieves both better link prediction performance and high quality rule learning results. These support our goal of making IterE complement the strengths of embedding and rule learning.

Contributions of our work are as follows:

- We propose an iterative framework that combines embedding learning and rule learning to explore the mutual benefits between them. Experiments show that it leads to better link prediction results using rules and embeddings together.
- We present a new method for *embedding learning with rules* based on axiom injection through t-norm based fuzzy logics. Experiments show that IterE can significantly improve embedding quality for the sparse part of a knowledge graph.
- We further identify a portfolio of ontology axioms for *rule learning with embedding* based on linear map assumption. Experiments show that IterE learns more high quality rules more efficiently than conventional rule learning systems.

2 PRELIMINARIES

2.1 Knowledge graph embedding

A KG $\mathcal{K} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$ contains a set of entities $\mathcal{E}$, a set of relations $\mathcal{R}$ and a set of triples $\mathcal{T} = \{(s, r, o)|s, o \in \mathcal{E}; r \in \mathcal{R}\}$. In a triple (s, r, o), the symbols s, r, and o denote subject entity, relation, and object entity respectively. An example of such triple is (Tokyo, *locatedIn*, Japan).

Knowledge graph embedding (KGE) aims to embed all entities and relations in a continuous vector space, usually as vectors or matrices called *embeddings*. Embeddings can be used to estimate the likelihood of a triple to be true via a score function $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$. Concrete score functions are defined based on different vector space assumptions. We now describe two vector space assumptions commonly used in KGEs and their corresponding score functions.

(a) Translation-based assumption embeds entities and relations as vectors and assumes $\mathbf{v}_s + \mathbf{v}_r = \mathbf{v}_o$, in which $\mathbf{v}_s, \mathbf{v}_r$ and $\mathbf{v}_o$ are vector embeddings for s, r and o respectively. For a true triple, the relation-specific translation of subject embedding $(\mathbf{v_s} + \mathbf{v_r})$ is close to the object embedding $\mathbf{v_o}$ in embeddings' vector space.

(b) Linear map assumption embeds entities as vectors and relations as matrices. It assumes that the subject entity embedding $\mathbf{v}_s$ can be linearly mapped to object entity embedding $\mathbf{v}_o$ via relation embedding $\mathbf{M}_r$. In this case, for a true triple, the linear mapping of the subject embedding by the relation matrix $(\mathbf{v_s M_r})$ is close to the object embedding $\mathbf{v_o}$ in embeddings' vector space.

Table 1: Conditions for object property expression axioms in OWL2 and translated rule formulation. OP refers to "ObjectProperty". OPE, with or without subscript, denotes Object Property Expression and x, y, z are entity variables. $\triangle_I$ is a nonempty set called object domain. $\cdot^{OP}$ is an object property interpretation function. When translating axioms into rule forms according to condition, we replace OPE in axioms with binary relation r in the context of knowledge graph and the number of OPE in EquivalentOP and the OPChain in SubOP is set to 2.

Object Property Axiom	Condition	Rule Form
ReflexiveOP(OPE)	$\forall x : x \in \triangle_I$ implies (x,x) $\in$ (OPE)^{OP}	(x, r, x) [1]
SymmetricOP(OPE)	$\forall x, y : (x, y) \in (OPE)^{OP}$ implies (y, x) $\in (OPE)^{OP}$	$(y, r, x) \leftarrow (x, r, y)$
TransitiveOP(OPE)	$\forall x, y, z : (x, y) \in (OPE)^{OP}$ and (y, z) $\in (OPE)^{OP}$ imply (x, z) $\in (OPE)^{OP}$	$(x, r, z) \leftarrow (x, r, y), (y, r, z)$
EquivalentOP(OPE_1 ... OPE_n)	$(OPE_j)^{OP} = (OPE_k)^{OP}$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$	$(x, r_2, y) \leftarrow (x, r_1, y)$
SubOP(OPE_1 OPE_2)	$(OPE_1)^{OP} \subseteq (OPE_2)^{OP}$	$(x, r_2, y) \leftarrow (x, r_1, y)$
InverseOP(OPE_1 OPE_2)	$(OPE_1)^{OP} = \{(x, y) \mid (y, x) \in (OPE_2)^{OP}\}$	$(x, r_1, y) \leftarrow (y, r_2, x)$
SubOP(OPChain(OPE_1 ... OPE_n) OPE)	$\forall y_0, \ldots, y_n : (y_0, y_1) \in (OPE)^{OP}$ and ... and $(y_{n-1}, y_n) \in (OPE_n)^{OP}$ imply $(y_0, y_n) \in (OPE)^{OP}$	$(y_0, r, y_2) \leftarrow (y_0, r_1, y_1), (y_1, r_2, y_2)$

Thus the score function ϕ of two assumptions can be written as:

$$\phi_{translation} = sim(\mathbf{v}_s + \mathbf{v}_r, \mathbf{v}_o)$$
$$\phi_{linearmap} = sim(\mathbf{v}_s \mathbf{M}_r, \mathbf{v}_o) \quad (1)$$

where $sim(\mathbf{x}, \mathbf{y})$ calculates the similarity between vector $\mathbf{x}$ and $\mathbf{y}$.

From the assumption point of view, $\mathbf{v_s M_r} = \mathbf{v_o}$ (or $\mathbf{v_s} + \mathbf{v_r} = \mathbf{v_o}$) should exactly hold for true triples in the linear-map assumption (or translation-based assumption). From the modeling point of view, this is the optimization goal during learning, namely to be as close as possible to the equation in assumption. The learning process is done through either maximizing the objective or minimizing the error induced from assumptions given by their respective loss functions. Hence, the assumption equation usually does not exactly hold with learned embeddings, but their loss functions are designed to approach the assumption as much as possible.

In this paper, we adopt linear map assumption for embedding learning because many reasonable rule conclusions can be derived with relation embeddings based on this assumption (Table 2).

2.2 Rules Learning

Suppose $\mathcal{X}$ is a countable set of variables and C a countable set of constants. A rule is of the form $head \leftarrow body$, where $head$ is an atom over $\mathcal{R} \cup \mathcal{X} \cup C$ and $body$ is a conjunction of positive or negative atoms over $\mathcal{R} \cup \mathcal{X} \cup C$. An example of such rule can be:

$$(\mathrm{X}, hasMother, \mathrm{Y}) \leftarrow (\mathrm{X}, hasParent, \mathrm{Y}), (\mathrm{Y}, gender, \mathrm{Female}) \quad (2)$$

When replacing all variables in a rule with concrete entities in KG, we get a *grounding* of the rule. For example, one grounding of Rule (2) can be:

$$(\mathrm{Bob}, hasMother, \mathrm{Ann}) \leftarrow \quad (3)$$
$$(\mathrm{Bob}, hasParent, \mathrm{Ann}), (\mathrm{Ann}, gender, \mathrm{Female})$$

A grounding with all triples existing in knowledge graph is a *support* of this rule. For examples, if (Bob, *hasMother*, Ann) $\in \mathcal{K}$, (Bob, *hasParent*, Ann) $\in \mathcal{K}$ and (Ann, *gender*, Female) $\in \mathcal{K}$, then grounding (3) is a support for rule (2).

The results of rule learning is of the following form:

$$\alpha \; head \leftarrow body$$

in which $\alpha \in [0, 1]$ is a confidence score assigned to the rule by the learning method.

Incorporating logical rules into other learning system such as embedding learning is called *rule injection*. One way for rule injection is adding regularizer or other constraints to entity and relation representations by propositionalizing the rules. Another way is adding constraints to the constituent relations' representation in rules without direct effect on entity representations. As we want to get new information of sparse entities through rules, we chose propositionalization in this paper for rule injection.

2.3 OWL 2 Web Ontology Language Axioms

In this paper, instead of learning general Horn rules or closed-path rules as previous works[9][48], we are more interested in ontology axioms, the main components of knowledge graph ontologies, because they are important for enriching semantics in KGs.

OWL2 Web Ontology Language, informally OWL2[2], is an ontology language for Semantic Web with formally defined meaning and is a W3C recommendation. It defines multiple types of axioms, from which we select some of them as a guidance of rule structures. The selection is based on following principles: (i) the axioms are related with binary relations, the main components of rules in KG and (ii) they can infer new triples because rules are used to help add new information about sparse entities in this paper. Thus, we focus on object property expression axioms in OWL2 which are composed of binary relations in the context of KG. Finally, 7 types of object property expression axioms out of 14 are selected. The unselected axioms are mainly applied to help check the consistency in knowledge graph.

In OWL2, each axiom has its own condition revealing its semantics. The axiom is *satisfied* if its condition hold. We introduce the selected 7 types of object property expression axioms and their conditions in Table 1. We also translate the conditions of axioms into rule-form including a *head* and a *body* as introduced in the previous subsection. The translated rule forms are used to guide the structures of rules to be learned in this paper.

3 METHOD

Given a knowledge graph $\mathcal{K} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, our goal is to learn embeddings and rules at the same time and make their advantages complement each other's difficulties. As discussed in Section 1, embedding learning methods suffer from the problem of data sparsity and rule learning methods have a very large search space.

In this paper, we propose a general framework **IterE** which learns embeddings and rules in an iterative manner and can be applied to many KGEs that are based on linear map assumption. It includes three main parts:(i) embedding learning, (ii) axiom induction and (iii) axiom injection. Figure 2 shows the key idea of IterE with these three iterative parts.

- **Embedding learning** learns entity embeddings **E** and relation embeddings **R** with a loss function $L_{embedding}$ to be minimized, calculated with input triples (s, r, o), each with a label related

[1] the translated rule form of ReflexiveOP(OPE) only contain a *head*.
[2] https://www.w3.org/TR/owl2-primer/

Table 2: Seven types of object property expression axioms selected from OWL2 ontology language. `OP` **is the short for** `ObjectProperty`. $\mathcal{K}$ **denotes a KG and** x, y, z **are entity variables. v and M denote entity and relation embeddings respectively. I is identity matrix.**

Object Property Axioms	Rule Form	According to Linear Map Assumption	Rule Conclusion
`ReflexiveOP`(r)	(x, r, x)	$\mathbf{v}_x\mathbf{M}_r = \mathbf{v}_x$	$\mathbf{M}_r = \mathbf{I}$
`SymmetricOP`(r)	$(y, r, x) \leftarrow (x, r, y)$	$\mathbf{v}_y\mathbf{M}_r = \mathbf{v}_x; \mathbf{v}_x\mathbf{M}_r = \mathbf{v}_y$	$\mathbf{M}_r\mathbf{M}_r = \mathbf{I}$
`TransitiveOP`(r)	$(x, r, z) \leftarrow (x, r, y), (y, r, z)$	$\mathbf{v}_x\mathbf{M}_r = \mathbf{v}_z; \mathbf{v}_x\mathbf{M}_r = \mathbf{v}_y,\ \mathbf{v}_y\mathbf{M}_r = \mathbf{v}_z,$	$\mathbf{M}_r\mathbf{M}_r = \mathbf{M}_r$
`EquivalentOP`(r_1, r_2)	$(x, r_2, y) \leftarrow (x, r_1, y)$	$\mathbf{v}_x\mathbf{M}_{r_2} = \mathbf{v}_y,\ \mathbf{v}_x\mathbf{M}_{r_1} = \mathbf{v}_y$	$\mathbf{M}_{r_1} = \mathbf{M}_{r_2}$
`subOP`(r_1, r_2)	$(x, r_2, y) \leftarrow (x, r_1, y)$	$\mathbf{v}_x\mathbf{M}_{r_2} = \mathbf{v}_y,\ \mathbf{v}_x\mathbf{M}_{r_1} = \mathbf{v}_y$	$\mathbf{M}_{r_1} = \mathbf{M}_{r_2}$
`inverseOP`(r_1, r_2)	$(x, r_1, y) \leftarrow (y, r_2, x)$	$\mathbf{v}_x\mathbf{M}_{r_1} = \mathbf{v}_y,\ \mathbf{v}_y\mathbf{M}_{r_2} = \mathbf{v}_x$	$\mathbf{M}_{r_1}\mathbf{M}_{r_2} = \mathbf{I}$
`subOP(OPChain`(r_1, r_2)`)`, r)	$(y_0, r, y_2) \leftarrow (y_0, r_1, y_1), (y_1, r_2, y_2)$	$\mathbf{v}_{y_0}\mathbf{M}_r = \mathbf{v}_{y_2},\ \mathbf{v}_{y_0}\mathbf{M}_{r_1} = \mathbf{v}_{y_1},\ \mathbf{v}_{y_1}\mathbf{M}_{r_2} = \mathbf{v}_{y_2}$	$\mathbf{M}_{r_1}\mathbf{M}_{r_2} = \mathbf{M}_r$

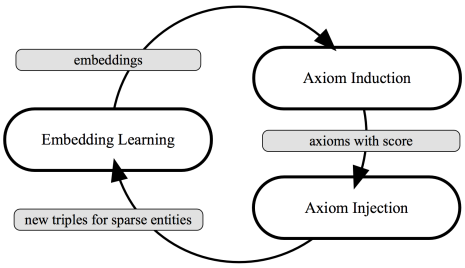

Figure 2: Overview of our IterE.

with its truth value. The inputs are of two types: triples existing in $\mathcal{K}$ and triples that do not exist in $\mathcal{K}$ but are inferred by axioms.

- **Axiom Induction** inducts a set of axioms $\mathcal{A}$ based on relation embeddings $\mathbf{R}$ from the embedding learning step, and assigns each axiom with a score s_{axiom}.
- **Axiom Injection** injects new triples about sparse entities in $\mathcal{K}$ to help improve their poor embeddings caused by insufficient training. The new triples are inferred from groundings of quality axioms with high scores in $\mathcal{A}$ from axiom induction. After axiom injection, with $\mathcal{K}$ updated, the process goes back to embedding learning again.

IterE is proposed based on the observation that embeddings learned with linear map assumption can fully support the axiom selected in this paper, while other assumptions such as translation-based assumption can't as pointed out in [50]. Inherently, for each type of axioms, a meaningful conclusion can be drawn with relation embeddings according to linear map assumption. For example, considering axiom `inverse`($hasParent$, $hasChild$), if ($Mike$, $hasParent$, $John$) exists in knowledge graph, according to the condition and rule form of inverse axiom in Table 1, another triple ($John$, $hasChild$, $Mike$) can be inferred. Suppose embeddings of $Mike$, $John$, $hasParent$ and $hasChild$ are $\mathbf{v}_{Mike}$, $\mathbf{v}_{John}$, $\mathbf{M}_{hasChild}$ and $\mathbf{M}_{hasParent}$ respectively. According to the linear assumption for individual triples, we can get following two equations: $\mathbf{v}_{Mike}\mathbf{M}_{hasParent} = \mathbf{v}_{John}$ and $\mathbf{v}_{John}\mathbf{M}_{hasChild} = \mathbf{v}_{Mike}$. With these two equations, another equation can be deduced: $\mathbf{M}_{hasParent}\mathbf{M}_{hasChild} = \mathbf{I}$. Note that this conclusion equation is only related with two corresponding relation embeddings and is unrelated with concrete entities, thus it can be regarded as a general conclusion for axiom `inverse`($hasParent$, $hasChild$). For other types of axioms, a general conclusion can be drawn in the same way. We list details of the conclusion for each type of axioms in Table 2. These conclusions about relation embeddings help guide axiom induction in this paper.

3.1 Embedding Learning

The input $\mathcal{I}$ of embedding learning is a set of triples with labels. Unlike previous KGEs, which only learn embeddings but not rules, positive input triple in IterE contains two parts: triples $(s, r, o) \in \mathcal{T}$ existing in original knowledge graph and triples $(s, r, o) \in \mathcal{T}_{axiom}$, where $\mathcal{T}_{axiom}$ is a set of triples inferred by axioms learned from embeddings. Negative input triples $(s', r', o') \in \mathcal{T}_{negative}$ are generated by randomly replacing s or o with $e \in \mathcal{E}$ or replacing r with $r' \in \mathcal{R}$ for $(s, r, o) \in \mathcal{T}$. Thus the input set $\mathcal{I}$ in IterE is as follows:

$$\mathcal{I} = \{((s, r, o), l_{sro}) | ((s, r, o)) \in \mathcal{T} \wedge \mathcal{T}_{axiom} \wedge \mathcal{T}_{negative}\} \quad (4)$$

where l_{sro} is the label for triple (s, r, o) to evaluate its truth value. $l_{sro} = 1$ for $(s, r, o) \in \mathcal{T}$ and $l_{sro} = 0$ for $(s, r, o) \in \mathcal{T}_{negative}$. For triples $(s, r, o) \in \mathcal{T}_{axiom}$ we assign $l_{sro} = \pi(s, r, o)$, where $\pi(s, r, o)$ is triple truth value predicted by axiom injection (Section 3.3).

With input $\mathcal{I}$, the loss function of embedding learning is calculated by mean of cross entropy loss among all n input triples and the training goal is to minimize the following loss function:

$$\min L_{embedding} = \frac{1}{n} \sum_{((s,r,o),l_{sro}) \in \mathcal{I}} [-l_{sro} \log(\phi(s, r, o)) - (1 - l_{sro}) \log(1 - \phi(s, r, o))] \quad (5)$$

As we adopt linear map assumption, the score function $\phi(s, r, o)$ for each triple (s, r, o) is defined as:

$$\phi(s, r, o) = sim(\mathbf{v}_s\mathbf{M}_r, \mathbf{v}_o) = \sigma(\mathbf{v}_s^\top \mathbf{M}_r \mathbf{v}_o) \quad (6)$$

in which $\mathbf{v}_s \in \mathbb{R}^{1\times d}$, $\mathbf{v}_o \in \mathbb{R}^{1\times d}$ are vector embeddings for subject and object entity. $\mathbf{M}_r \in \mathbb{R}^{d\times d}$ is matrix embedding for relation and σ denotes the sigmoid function. d is the embedding dimension. The similarity between two vectors is evaluated via dot product.

Our approach can be combined with many KGEs based on linear map assumption, such as DistMult[47] and ANALOGY[23]. In this paper, we adopt ANALOGY as it achieves state-of-the-art results on link prediction. ANALOGY is proposed to deal with an important kind of reasoning, analogy reasoning, in embedding learning. It imposes analogical structures among embeddings and requires linear maps associated with relations to form a commuting family of normal matrices. Specifically, the relation matrix embeddings are constrained as block-diagonal matrices with each diagonal block is either a real scalar, or a 2-dimensional real matrix in the form of $\begin{bmatrix} a & -b \\ b & a \end{bmatrix}$, where both a and b are real scalars.

After embedding learning, a collection of entity embeddings $\mathbf{E}$ and relation embeddings $\mathbf{R}$ are learned and $\mathbf{R}$ will be used in axiom induction.

3.2 Axiom Induction

Given relation embedding $\mathbf{R}$, axiom induction aims to induce a set of axioms $\mathcal{A}$ and assign a confidence score to each axiom in $\mathcal{A}$. To achieve this, IterE firstly generates a pool of possible axioms $\mathcal{P}$ with an effective pruning strategy. Then it predicts a score of each axiom $a \in \mathcal{P}$ based on calculation with $\mathbf{R}$.

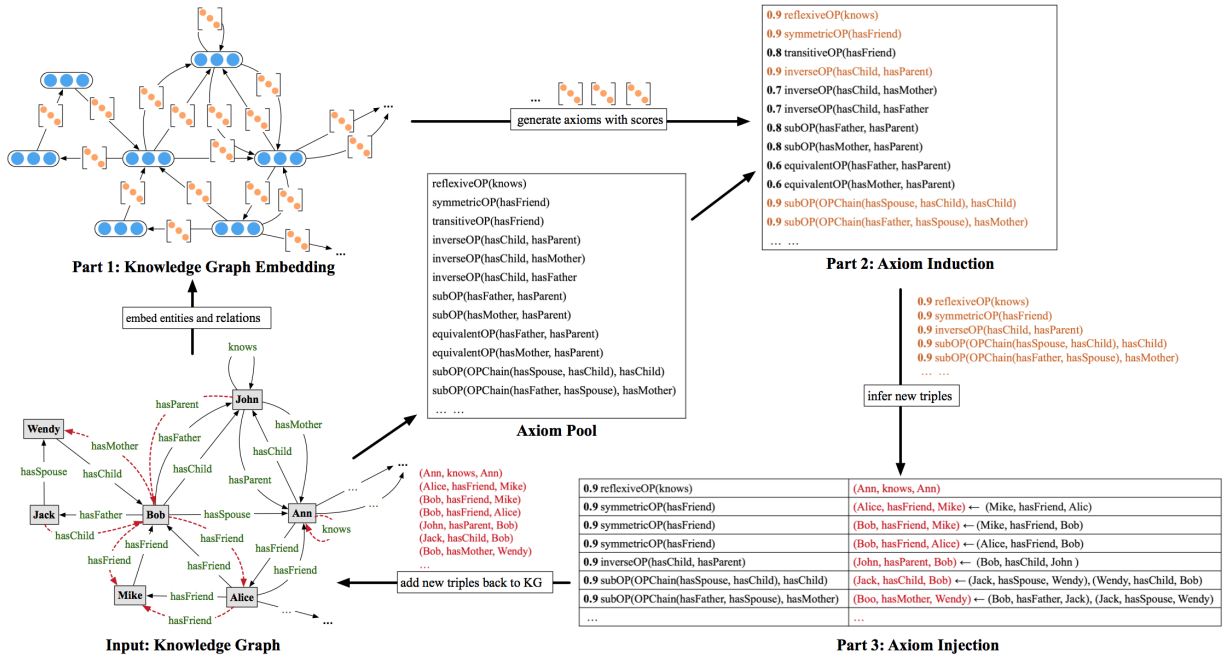

Figure 3: Detailed example for the process of IterE.

3.2.1 Axiom pool generation. Before calculating axiom scores with relation embedding $\mathbf{R}$, relation variables r, r_1, r_2, and r_3 in Table 2 should be replaced with concrete relations. Axiom pool generation generates a pool of possible axioms $\mathcal{P}$ by searching for possible axioms with the number of *support* greater than 1.

One intuitive way of searching for possible axioms is generating axioms by replacing all relation variable in each type of axioms with each relation and then check the number of support for them, but this suffers from a huge search space. Another way is to generate axioms via random walk on knowledge graph while this method can't ensure coverage of axioms. Therefore axiom pool generation is not an easy task because it has to achieve a good balance between search space and coverage of highly possible axioms.

In this paper, we propose a pruning strategy combining traversing and random selection. There are two steps for generating possible axioms for each relation $r \in \mathcal{R}$ in IterE.

- *step 1*: generate axioms or partial axioms: $\texttt{ReflexiveOP}(r)$, $\texttt{SymmetricOP}(r)$, $\texttt{TransitiveOP}(r)$, $\texttt{EquivalentOP}(r', r)$, $\texttt{subOP}(r', r)$, $\texttt{inverseOP}(r', r)$ and $\texttt{subOP}(\texttt{OPChain}(r', r''), r)$, , in which $\{r', r''\}$ are relation variables to be replaced with relations in KG and r participants in the head of their rule forms.
- *step 2*: complete partial axioms via randomly selecting k triples $(e', r, e'') \in \mathcal{T}$ related with r. Replacing r' or r'' in partial axioms with relations that directly link to e' or e''.
- *step 3*: search for support of each axiom, and add those axioms with number of support larger than 1 into axiom pool $\mathcal{P}$.

The key point of whole process is choosing k, which is not trivial because a large k will lead to dramatic increase of search space while a small k will decrease coverage of axioms immediately. Thus a good choice of k should achieve followings: 1) the probability for $\mathcal{P}$ covering all highly possible axioms higher than t, named *including probability*. 2) k is as small as possible. We defined the highly possible axioms as axioms with existing probability larger than p in this paper and we name p as *minimum axiom probability*.

We estimate the probability $p(r, a_x)$ that r replaces the relation variable in *head* of axiom a_x as:

$$p(r, a_x) = \frac{n}{N} \tag{7}$$

where N is the number of triples $(e', r, e'') \in \mathcal{T}$ and n is the number of support for axiom a_x in $\mathcal{K}$. Thus choosing k triples to satisfy two requirements mentioned above can be formulated as follows:

$$\min k \quad s.t.\ 1 - \frac{C^k_{(1-p)N}}{C^k_N} > t \tag{8}$$

From above inequality in (8), following inequality can be reached:

$$k > N - N(1-t)^{\frac{1}{pN}} \tag{9}$$

Thus with fixed p, t, the best choice of k should be the upper bound v_u of equation $f(N) = N - N(1-t)^{\frac{1}{pN}}$. Fortunately, with $N \in [0, 10^{15}]$, this equation is monotone increasing and has small upper bound v_u. For example, when $p = 0.5, t = 0.95$, v_u is 6.

As the input of axiom pool generation is a fixed $\mathcal{K}$ during learning, axiom pool $\mathcal{P}$ only need to be generated once.

3.2.2 Axiom Predicting. Given current relation embedding $\mathbf{R}$ and axiom pool $\mathcal{P}$, axiom predicting predicts a score s_a for each axiom $a \in \mathcal{P}$ based on the rule conclusion for each type of axioms (column 4 in Table 2), in the form that $\mathbf{M}_1^a = \mathbf{M}_2^a$, where $\mathbf{M}_1^a$ and $\mathbf{M}_2^a$ are matrices either from a single matrix or a product of two matrices. As rule conclusions are derived from ideal linear map assumption, $\mathbf{M}_1^a$ and $\mathbf{M}_1^a$ usually are not equal but similar during training. Thus, instead of matching $\mathbf{M}_1^a$ and $\mathbf{M}_2^a$, we estimate the truth of axiom a by similarity between $\mathbf{M}_1^a$ and $\mathbf{M}_2^a$ which is supposed to be related with Frobenius norm of their difference matrix:

$$s_a(F) = \|\mathbf{M}_1^a - \mathbf{M}_2^a\|_F \tag{10}$$

$s_a(F)$ is then normalized as follows because the value of $s_a(F)$ for different type of axioms vary dramatically:

$$s_a = \frac{s_{max}(t) - s_a(F)}{s_{max}(t) - s_{min}(t)} \tag{11}$$

in which t is the type of axiom that a belongs to. $s_{max}(t)$ and $s_{min}(t)$ is the maximum and minimum Frobenius norm score among all type t axioms in $\mathcal{P}$. $s_a \in [0, 1]$ is the final score for axiom a and the higher s_a is the more confident that axiom a is.

3.3 Axiom Injection

Given knowledge graph $\mathcal{K}$ and possible axiom set $\mathcal{A}$, axiom injection utilizes axiom' deductive capability to infer a set of new triples $\mathcal{T}_{axiom}$ for sparse entities and predict their labels. $\mathcal{T}_{axiom}$ will be injected into embedding learning to reduce the sparsity of entities.

3.3.1 Sparse entities. We evaluate the sparsity of entities as follows:

$$sparsity(e) = 1 - \frac{freq(e) - freq_{min}}{freq_{max} - freq_{min}} \tag{12}$$

where $freq(e)$ is the frequency of entity e participating in a triple, as subject or object entity. $freq_{min}$ and $freq_{max}$ are the minimum and maximum frequency among all entities. $sparsity(e) \in [0, 1]$. $sparsity(e) = 1$ means e is the most sparse entity and $sparsity(e) = 0$ means the most frequent one. With $sparsity(e) > \theta_{sparsity}$, we regard entity e as a sparse entity, where $\theta_{sparsity}$ is a sparse threshold. We use $\mathcal{E}_{sparse}$ to denote the set of sparse entity in $\mathcal{K}$.

During axiom injection, triples related with sparse entities are injected into the input of embedding learning. In other words, in new inferred triples (s^a, r^a, o^a), either $s^a \in \mathcal{E}_{sparse}$ or $o^a \in \mathcal{E}_{sparse}$ or $\{s^a, o^a\} \in \mathcal{E}_{sparse}$. Thus after inferring all possible new triples, we filter those unrelated to sparse entities.

3.3.2 Predicting new triples and their labels. We utilize groundings to infer new triples, and the grounding for axioms considered in this paper can be generalized as the following form:

$$(s^a, r^a, o^a) \leftarrow (s_1, r_1, o_1), (s_2, r_2, o_2), ..., (s_n, r_n, o_n) \tag{13}$$

where the right side triples $(s_k, r_k, o_k) \in \mathcal{T}$ with $k \in [1, n]$ are generated from the body of axiom rule form and $(s^a, r^a, o^a) \notin \mathcal{T}$ is new inferred triples to be added into knowledge graph.

To predict label for (s^a, r^a, o^a), we first translate the grounding form in (13) in propositional logical expression:

$$(s_1, r_1, o_1) \wedge (s_2, r_2, o_2) \wedge ... \wedge (s_n, r_n, o_n) \Rightarrow (s^a, r^a, o^a) \tag{14}$$

then we model groundings through t-norm based fuzzy logics[17]. It regards the truth value of a propositional logical expression as a composition of constituent triples' truth value, through specific logical connectives (e.g. $\wedge$ and $\Rightarrow$). For example, the truth value of a propositional logical expression $(s_1, r_1, o_1) \Rightarrow (s_2, r_2, o_2)$ is determined by the true value of two component triples (s_1, r_1, o_1) and (s_2, r_2, o_2), via a composition defined by logical implication $\Rightarrow$. We follow the definition of composition associated with logical conjunction ($\wedge$), disconjunction ($\vee$) and negation ($\neg$) as [14]:

$$\pi(a \wedge b) = \pi(a) \cdot \pi(b) \tag{15}$$

$$\pi(a \vee b) = \pi(a) + \pi(b) - \pi(a) \cdot \pi(b) \tag{16}$$

$$\pi(\neg a) = 1 - \pi(a) \tag{17}$$

$$\pi(a \Rightarrow b) = \pi(\neg a \vee b) \tag{18}$$

in which a, b are logical expressions and $\pi(x)$ is the truth value of x. Given these compositions, the truth value of any propositional logical expression can be calculated recursively through Equation (15)-(18). For example, applying to propositional logical expression of grounding $g: (s_1, r_1, o_1) \wedge (s_2, r_2, o_2) \Rightarrow (s^a, r^a, o^a)$:

$$\pi(g) = 1 - \pi(s_1, r_1, o_1)\pi(s_2, r_2, o_2) + \pi(s_1, r_1, o_1)\pi(s_2, r_2, o_2)\pi(s^a, r^a, o^a)$$

To predict the truth value $\pi(s^a, r^a, o^a)$ for triples (s^a, r^a, o^a), inferred by axiom a according to grounding g_a, based on t-norm fuzzy logics, $\pi(s^a, r^a, o^a)$ can be calculated according to $\pi(g)$ and $\pi(s_1, r_1, o_1), \pi(s_2, r_2, o_2), ..., \pi(s_n, r_n, o_n) \in g_a$. In former example,

$$\pi(s^a, r^a, o^a) = \frac{\pi(g) - 1 + \pi(s_1, r_1, o_1)\pi(s_2, r_2, o_2)}{\pi(s_1, r_1, o_1)\pi(s_2, r_2, o_2)}$$

For the truth value of axiom a's grounding g, we evaluate it as the score of a generated by axiom induction, namely $\pi(g_a) = s_a$. For the truth value of triples existing in knowledge graph, we evaluate them with their training labels, thus $\pi(s, r, o) = 1$ for $(s, r, o) \in \mathcal{T}$, because existing triples are absolutely true. With these two types of truth value assignment, we can easily get the following result for triples (s^a, r^a, o^a) inferred by any type of axioms a:

$$\pi(s^a, r^a, o^a) = s_a \tag{19}$$

After axiom injection, a set of new triples $\mathcal{T}_{axiom} = \{(s^a, r^a, o^a) | s^a \in \mathcal{E}_{sparse} \text{ or } o^a \in \mathcal{E}_{sparse}\}$ are inferred via quality axioms and each new triple (s^a, r^a, o^a) is labeled with $l_{sro} = s_a$. Thus the input of embedding learning was updated, $\mathcal{I} = \{((s, r, o), l_{sro}) | (s, r, o) \in \mathcal{T} \wedge \mathcal{T}_{axiom}\}$. Then the process goes back to embedding learning.

4 EXPERIMENT

During experiments, we want to explore following questions: 1) whether axioms really help sparse entity embedding learning? To do this, we evaluate the quality of embeddings on link prediction task which is widely applied in previous knowledge graph embedding works; 2) whether embeddings really help rule learning overcome the huge search space and improve the quality of learned rules? To do this, we evaluate the efficiency of axiom learning with learning time and the quality with number and percentage of high quality axioms the method generate; 3) how does the iterative manner affect embedding learning and rule learning process? To this end, we show the changes of link prediction results, rule qualities and the number of triples injected along with different number of iterations.

4.1 Dataset

Four datasets are used in experiment, including WN18-sparse, WN18RR-sparse, FB15k-sparse, and FB15k-237-sparse. They are generated from WN18[3], WN18RR[7], FB15k[3] and FB15-237[34], four datasets that are commonly used in previous knowledge graph embedding work. WN18 and WN18RR are subsets of WordNet, a lexical knowledge graph for English. FB15k and FB15k-237 are subsets of a large collaborative knowledge base Freebase. The statistics of datasets are listed in Table 3.

We generate the sparse version of these datasets via changing the valid and test datasets which will be used for link prediction evaluation. In link prediction experiments, we want to explore whether axioms really contribute to sparse entity embeddings. Therefore, we only keep triples in test and valid dataset which involve at least one sparsity entity. In other words, for a test or valid triple (s, r, o), if either s or o is or both are sparse entity (Equation (12)), the triple will be kept in sparse dataset, otherwise, it will be filtered. When deciding sparse entity hyperparameter θ, intuitively a principle is considered: the percentage of left triples in valid and test set should not be larger than 80% or less than 20% of original valid and test set for all datasets. Thus we choose $\theta = 0.995$ and regard entities e

Table 3: Statistics of datasets.

Dataset	#E	#R	#Train	#Valid	#Test
WN18-sparse	40,943	18	141,442	3624(72.48%)	3590(71.8%)
WN18RR-sparse	40943	11	86835	1609(53.03)	1661(52.9%)
FB15k-sparse	14,951	1,345	483,142	18544(37.08%)	22013(37.26%)
FB15k-237-sparse	14541	237	272115	10671(60.8%)	12454(60.85%)

with $sparsity(e) > 0.995$ as sparse entity in this paper. Table 3 also shows the percent of triples left.

4.2 Training Details

For embedding learning, the number of negative samples is set to 6 and the number of scalars on the diagonal of each relation matrix is set to $\frac{d}{2}$ where d is embedding dimension. We initialize embeddings with uniform distribution $\mathbf{U}(-0.1, 0.1)$.

For axiom induction, we set the minimum axiom probability $p = 0.5$ and the including probability $t = 0.95$ for axiom pool generation. Based on these settings, the number of related triples selected for each relation is set as $k = 6$ according to Equation(9). Details of axiom pools generated for all datasets are shown in Table 4, where we can see that the number of possible axioms in FB15k and FB15k-237 are much larger than WN18 and WN18RR because the number of axioms is highly related to the diversity of relations.

Table 4: Axiom pools details.

	ref.	sym.	tra.	equ.	inv.	sub.	sub.(Cha.)
WN18-sparse	0	3	1	2	19	2	72
WN18RR-sparse	0	3	1	1	3	1	22
FB15k-sparse	41	94	52	1872	2743	1872	59709
FB15k-237-sparse	5	30	27	197	192	197	5017

For axiom injection, considering that axioms with high scores are more reliable and less possible to introduce noise, we set a threshold θ for each dataset and regard axioms with scores $s_{axiom} > \theta$ as high quality axioms. We also set a maximum inferred triples m for axioms in each dataset, if one axiom will infer a lot of triples we ignore it because a huge number of triples inferred by one axiom will change the data distribution significantly and make the embedding training unstable.

During one iteration learning, we first train embedding learning for 10 epochs, and then conduct axiom induction and injection once. The maximum training iteration is set to 10 for WN18RR-sparse and FB15k-237-sparse and 50 for WN18-sparse and FB15k-sparse. We use Adam algorithm [19] during optimization with learning rate $lr = 0.001$. We apply grid search for the best hyperparameters based on the filter MRR on the validation dataset, with combinations from embedding dimension $d \in \{100, 200\}$ and l_1 regularizer weigh $\lambda \in \{10^{-4}, 10^{-5}, 10^{-6}\}$.

The final parameters are $d = 200, \lambda = 10^{-5}, \theta = 0.95, m = 20000$ for WN18-sparse, $d = 200, \lambda = 10^{-5}, \theta = 0.9, m = 10000$ for WN18RR-sparse, $d = 200, \lambda = 10^{-4}, \theta = 0.95, m = 100$ are for FB15k-sparse, and $d = 200, \lambda = 10^{-5}, \theta = 0.9, m = 1000$ are for FB15k-237-sparse.

4.3 Embedding Evaluation

We evaluate the quality of embeddings on link prediction tasks. Link prediction aims to predict the missing entity when given the other entity and relation in a triple, including subject entity prediction $(?, r, o)$ and object entity prediction $(s, r, ?)$.

4.3.1 Evaluation metrics. Given subject entity prediction task $(?, r, o)$ with right answer s, we first fit subject entity position with each entity $e \in \mathcal{E}$ and thus get a set of triples $\mathcal{T}_{subjectprediction} = \{(e, r, o) | e \in \mathcal{E}\}$. Then we calculate the score for each triple in $\mathcal{T}_{subjectprediction}$ according to Equation (6) and rank their scores in descending order. Thus the entity e in (e, r, o) with a higher rank is a more possible prediction. To evaluate how good the prediction is, we use the rank of (s, r, o) among all triples in $\mathcal{T}_{subjectprediction}$ as subject entity prediction evaluation result for (s, r, o), namely subject entity prediction rank $rank_s(?, r, o)$. The object entity prediction task is done in the same way and will get objection entity rank $rank_o(s, r, ?)$. The final prediction rank $rank(s, r, o)$ for (s, r, o) is the average of subject and object prediction rank:

$$rank(s, r, o) = \frac{1}{2}(rank_s(?, r, o) + rank_o(s, r, ?))$$

Aggregating prediction rank for all test triples, we applied mean reciprocal predicting rank *(MRR)* and the percentage of predicting ranks within n *(Hit@n)* to evaluate the whole prediction result. These two metrics are widely used in previous work [35][23]. Generally, a higher *MRR* or *Hit@n* indicates a better prediction result.

We also apply $filter$ and raw setting. As mentioned in [41], when fitting subject or object entity position with other entities, we may generate triples existing in knowledge graph which are known to be true. It is not wrong if these triples are ranked higher than current test triple. With $filter$ setting, we filter the generated triples that exist in both train/valid/test dataset before ranking but not current test triple. raw means the setting of NO filtering.

4.3.2 Baselines. For baselines, one method from translation-based assumption TransE [3] and three methods based on linear map assumption, DistMult, ComplEx, and ANALOGY [4] are considered.

4.3.3 Results and analysis. To show how axiom helps sparse entity embeddings, we adopt two strategies: (1) Firstly, we evaluate how axioms directly improve entity embedding quality and compare link prediction results from our method, denoted as **IterE** in Table 5, with other baselines directly. (2) Secondly, we evaluate how axioms can help improve sparse entity link prediction utilizing its deductive ability. Thus we compare prediction results with embeddings and axioms, denoted as **IterE+axioms** in Table 5 with IterE and other methods. In IterE+axioms, if the test triple (s, r, o) are inferred by axioms during axiom injection, which means $(s, r, o) \in \mathcal{T}_{axiom}$, we regard it as correct and mark its prediction rank as 1.

The link prediction results are shown in Table 5. We analyze the results as follows: Firstly, the link prediction results of IterE competitive to ANALOGY, which means most of the triples injected into embedding learning are not noise, indicating that learning axioms from embeddings works well. Secondly, IterE outperforms baselines on WN18RR-sparse and FB15k-237-sparse, while are slightly improved on WN18-sparse and FB15k-sparse. These indicate that IterE helps

[3] the code for TransE is from https://github.com/thunlp/OpenKE. We train all dataset with learning rate 0.01, margin 1.0 and dimension 100 for maximum 3500 iterations.
[4] The code for DistMult, ComplEx and ANALOGY is from https://github.com/quark0/ANALOGY. We train them with the same parameter setting as [23] for maximum 500 iterations. For WN18-sparse, the parameters are dimension $d = 200$, regularizer weigh $\lambda = 10^{-1}$, negative weight $n = 6$ for DistMult and $d = 200$, $\lambda = 10^{-2}$, $n = 6$ for both ComplEx and ANALOGY. For WN18RR-sparse, the parameters are $d = 200$, $\lambda = 1-^{-1}$, $n = 3$ for DistMult, Complex and ANALOGY. For FB15k-sparse, the parameters are $d = 200$, $\lambda = 10^{-2}$, $n = 6$ for DistMult, Complex and ANALOGY. For FB15k-237-sparse, the parameters are dimension $d = 100$, $\lambda = 0.1$, $n = 6$ for ANALOGY and Complex and $d = 200$, $\lambda = 10^{-1}$, $n = 3$ for DistMult.

Table 5: Link prediction results with MRR and Hit@n on WN18RR-sparse and FB15k-237-sparse. Underlined scores are the better ones between ANALOGY and IterE(ANALOGY). Boldface scores are the best results among all methods.

	WN18-sparse					FB15k-sparse				
	MRR (filter)	MRR (raw)	Hit@1 (filter)	Hit@3 (filter)	Hit10 (filter)	MRR (filter)	MRR (raw)	Hit@1 (filter)	Hit@3 (filter)	Hit10 (filter)
TransE[3]	41.8	33.5	10.2	71.1	84.7	39.8	25.5	25.8	48.6	64.5
DistMult[47]	73.8	55.8	59.3	87.5	93.1	60.0	32.4	**61.8**	65.1	75.9
ComplEx[35]	91.1	67.7	89.0	93.3	94.4	61.6	32.7	54.0	65.7	76.1
ANALOGY[23]	**91.3**	67.5	89.0	93.4	94.4	62.0	33.1	54.3	66.1	76.3
IterE(ANALOGY)	90.1	67.5	87.0	93.1	**94.8**	61.3	35.9	52.9	66.2	76.7
IterE(ANALOGY) + **axioms**	**91.3**	**78.9**	**89.1**	**93.5**	**94.8**	**62.8**	**38.8**	55.1	**67.3**	**77.1**
	WN18RR-sparse					**FB15k-237-sparse**				
	MRR (filter)	MRR (raw)	Hit@1 (filter)	Hit@3 (filter)	Hit10 (filter)	MRR (filter)	MRR (raw)	Hit@1 (filter)	Hit@3 (filter)	Hit10 (filter)
TransE[3]	14.6	12.4	3.4	24.7	28.8	23.8	15.6	16.4	26.1	38.5
DistMult[47]	25.5	20.8	23.8	26.0	22.5	20.4	12.9	12.8	22.6	36.2
ComplEx[35]	25.9	21.4	24.6	26.2	28.6	19.7	13.3	12.0	21.7	35.4
ANALOGY[23]	19.8	13.3	24.6	27.5	28.7	19.8	13.9	12.3	21.4	34.9
IterE(ANALOGY)	27.2	22.7	25.0	**28.1**	**31.4**	20.7	14.0	13.1	22.8	36.2
IterE(ANALOGY) + **axioms**	**27.4**	**25.7**	**25.4**	**28.1**	31.4	**24.7**	**18.6**	**17.9**	**26.2**	**39.2**

Table 6: Rule evaluation results.

	WN18-sparse			WN18RR-sparse			FB15k-sparse			FB15k-237-sparse		
	time	HQr	%	time	HQr	%	time	HQr	%	time	HQr	%
AMIE+	4.98s	16	11.4%	3s	2	5.71%	428s	1820	4.4%	66s	470	1.9%
IterE	**1.63**s	**20**	**20.2**%	**0.75**s	**6**	**19.3**%	**26.49**s	**11375**	**17.6**%	**4.72**s	**653**	**11.8**%

sparse entities much more in WN18RR-sparse and FB15k-237-sparse than in WN18RR-sparse and FB15k-sparse. This is quite reasonable in our opinion, because among these four datasets, WN18RR-sparse and FB15k-237-sparse are more challenging and more sparse. They are created from WN18-sparse and FB15k-sparse via removing one relation of all inverse relation pair in the dataset and also their related triples because inverse relation pair is a significant pattern in these two datasets as first noted in [34]. Thirdly, the results of IterE+axioms are improved compared with IterE in all datasets, especially on the most complex and sparse dataset FB15k-237-sparse. It indicates that the deductive capability of axioms can help the prediction of sparse entities further.

From the evaluation of link prediction, we can conclude that: (1) by injecting new triples for sparse entities, axioms help improve the quality of sparse entity embeddings and are more helpful in sparse KGs. (2) Combining axioms and embeddings together to predict missing links works better than using embeddings only. Both the deductive capability of axioms and the inductive capability of embeddings contribute to prediction and complement each other.

4.4 Rule Evaluation

We evaluate the learned rules/axioms from two perspectives: efficiency and quality. We compare our method with AMIE+[9][5] which is an improved rule mining system of widely used AMIE[10].

4.4.1 Evaluation metrics. The efficiency of rule learning is evaluated by learning time. The quality of rule learning is evaluated with the number of high quality rule (HQr) and their percentage. The quality of rules are evaluated by head coverage(HC) which is commonly used in pervious work, such as [10] and [29]. Head coverage for rule *rul* is defined as follows:

$$HC(rul) = \frac{\#(e, e') : support(rul) \wedge head(rul)(e, e')}{\#(e, e') : head(rul)(e, e')}$$

in which $head(rul)(e, e') = \{(e, r, e') \in \mathcal{T}\}$ if the head atom of *rul* is (X, r, Y). And $support(rul)$ is the supports for *rul*. We regard high quality rules as the rules with $HC > 0.7$ during test.

4.4.2 Results and analysis. Rule evaluation results are shown in Table 6. According to the time used to generate rules among all datasets, we can see that IterE learns rules more efficiently. For example, with FB15k-sparse and FB15k-237-sparse datasets, IterE costs 10 times less than AMIE+. This shows our pruning strategy works well. Among 4.72 seconds IterE cost for FB15k-237-sparse dataset, there is 4.55 seconds used for axiom pool generation and 0.17 seconds for axiom score calculation, namely axiom score calculation only cost 3.6% of the time. This means calculating axiom scores via embeddings is super efficient. We didn't include the time of embedding learning during this evaluation, as embedding learning is not devised mainly for rule learning, but for link prediction. The number of high quality rules shows that IterE generates more high quality rules than AMIE+ for each dataset and also achieves a higher percentage. This indicates that our axiom pool generation can filter meaningless axioms(rules) and achieve a good balance between small search space and coverage of highly possible axioms.

Further more, Figure 4 shows the changes of high quality axioms coverage and axiom percentage with different axiom score thresholds from IterE. For example, in 4(c), with axiom threshold 0.9, which means we select axioms with $s_a > 0.9$ from IterE for axiom injection, there are 23.4% axioms selected among all axioms and 46.4% high quality axioms included. And in 4(d), with axiom threshold 0.9, there are 53.9% axioms selected and 76.5% high quality axioms included. It illustrates that axiom scores calculated from

[5] we run AMIE+ code from https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie/

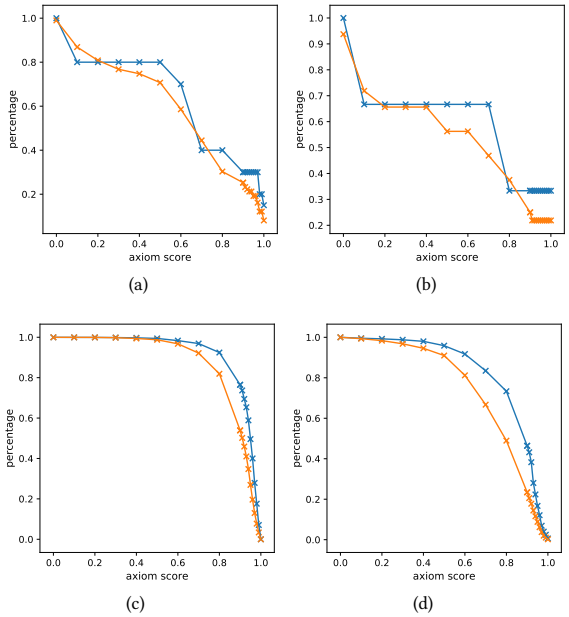

Figure 4: The coverage of high quality axioms (blue) and the axiom percentage (orange) with different axiom score thresholds. 4(a) is from WN18-sparse. 4(b) is from WN18RR-sparse. 4(c) is from FB15k-sparse. 4(d) is from FB15k-237-sparse.

embeddings are reliable because it is consistent with rule evaluation results. The results in 4(a) and 4(b) for WN18-sparse and WN18RR-sparse are not obvious because a few relations are contained in these two datasets and the number of learned rules is limited.

From rule evaluation results, we can conclude that (1) embeddings together with axiom pool generation help rule learning overcome large search space problem and improve rule learning efficiency, and (2) they also improve rule learning qualities and rules' reliable scores generated based on calculation with embeddings.

4.5 Iterative learning

To explore how iterative training improves embedding and rule learning during training, we show link prediction results on FB15k-237-sparse and number of injected triples at different iterations in Figure 5.

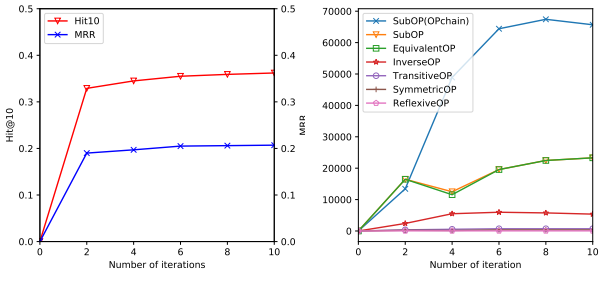

Figure 5: 5(a) shows link prediction results, MRR and Hit@10, in different iterations. 5(b) shows the number of triples injected into embedding learning in different iterations.

Figure 5 shows that the prediction results including Hit@10 and MRR become better as the training iteration increases and generally the number of injected triples increases during training and finally gets stable. The number of injected triples for `subOP` and `equivalentOP` in iteration 4 is less than iteration 3 which means axioms with high scores learned from previous iteration might get a lower score in the next iteration. This indicates that embedding learning effects rule learning. Thus from Figure 5, we can conclude that: (1) Iterative learning benefits embedding learning as the quality of embeddings gets better during training. (2) Iteratively learning benefits axiom learning because more axioms are learned and more triples are injected during training. (3) Axioms and embeddings influence and constrain each other during training.

4.6 Case study

In Table 7, we give a case study with 3 test triple examples which are inferred by axioms during training. Using the third one as an example, the test triple is (`Jenny_McCarthy`, /*type_of_union*, `Marriage`). `Jenny_McCarthy` is a person with only a few triples in KG and is a sparse entity with sparsity 0.997. `Marriage` is a type of union for people which has many links with individual person and its sparsity is 0.607. The subject prediction rank (?, /type_of_union, `Marriage`) in ANALOGY is 4199 and the object prediction (`Jenny_McCarthy`, /*type_of_union*, ?) is 2, from which we can see that the prediction for a sparse entity is much worse than a frequent entity. In IterE, this triple is predicted by rule (`X`, /*type_of_union*, `Z`) ⟵ (`X`, *spouse*, `Y`), (`Y`, /*type_of_union*, `Z`) because there are (`Jenny_McCarthy`, *spouse*, `Jim_Carrey`) and (`Jim_Carrey`, /*type_of_union*, `Marriage`) in KG which can compose a grounding for previous rules. Thus during training, the test triple is inferred by IterE and injected into embedding learning. Thus the subject prediction is improved from 4199 to 33 compared with ANALOGY and the object prediction keeps the same. When we predict with both embeddings and axioms (IterE + axioms), the subject prediction is improved from 33 to 1 and the object prediction from 2 to 1 compared with IterE. Other two cases in Table 7 perform similar to the third one.

These examples show that: (1) adding triples that can be inferred by axioms back into training improves the prediction results of related sparse entities without hurting the results for non-sparse entities. (2) the deductive ability of axioms helps ensure truth value of triples with sparse entities, which, in our opinion, overcomes uncertainties and noises that effect embedding prediction.

5 RELATED WORK

In this paper, we focus on iteratively learning embeddings and rules from knowledge graphs. Thus the related work includes two parts: (1) embedding learning, (2) rule learning.

5.1 Embedding Learning

Knowledge graph embedding learns latent representations for entities and relations in continuous vector space. Normally entities are represented as vectors [3][27][35] and relations are represented as vectors [3][41][27] or matrices [47][28][23]. These embeddings are assumed to preserve the semantics in a knowledge graph such as the similarity between entities[30], the truth of triples[3]. For most embedding methods, the input are triples existing in knowledge graph and embeddings are trained based on different vector space assumptions, such as translation-based assumption in [3] [41] [22] [18], and linear map assumption in [28] [47] [35] [23]. Some methods do not follow specific vector space assumption for embeddings, and special neural networks are applied to train embeddings and

Table 7: Examples from FB15k-237-sparse about the link prediction changes of test triples involving sparse entity. Following details are shown: the sparsity of subject and object entity in test triple, number within the square bracket beside the entity; the axiom that infers the test triple; the change of link prediction rank from ANALOGY to IterE; the change of link prediction rank from IterE to IterE+axiom.

triple	(Groundhog_Day[0.996]), $/film/currency$, United_States_Dollar[0.615])			
predicted by axiom	(X, $/film/currency$, Z) ← (X, $film\_release\_region$, Y), (Y, $/location/currency$, Z)			
rank change(ANALOGY → IterE)	**subject prediction rank:**	1409 → 356 (+1053)	**object prediction rank:**	1 → 1(+0)
rank change(IterE → IterE+axiom)	**subject prediction rank:**	356 → 1 (+355)	**object prediction rank:**	1 → 1(+0)
triple	(USA[0.0]), $/second\_level\_divisions$, Champaign_County[0.999])			
predicted by axiom	(X, $/second\_level\_divisions$, Z) ← (Y, $/bibs\_location$, X), (Z, $/county\_seat$, Y)			
rank change(ANALOGY → IterE)	**subject prediction rank:**	1 → 1 (+0)	**object prediction rank:**	416 → 214(+202)
rank change(IterE → IterE+axiom)	**subject prediction rank:**	1 → 1 (+0)	**object prediction rank:**	214 → 1(+213)
triple	(Jenny_McCarthy[0.997]), $/type\_of\_union$, Marriage[0.607])			
predicted by axiom	(X, $/type\_of\_union$, Z) ← (X, $spouse$, Y), (Y, $/type\_of\_union$, Z)			
rank change(ANALOGY → IterE)	**subject prediction rank:**	4199 → 33 (+4166)	**object prediction rank:**	2 → 2(+0)
rank change(IterE → IterE+axiom)	**subject prediction rank:**	33 → 1 (+32)	**object prediction rank:**	2 → 1(+1)

assign reasonable scores for different triples. For example, neural tensor networks in [32], convolution neural network in [7] and shared memory neural network in [31]. Linear map assumption is adopted for embedding learning in this paper because of its simplicity and good property for rule learning, while other assumptions or neural network models don't have.

Besides triples in KGs, some embedding methods also utilize other information during learning, for examples, entity descriptions in [40][43][39][42], entity types in [12][45], entity images in [44], and paths in [11][21][25], which can be regarded as a kind of rules.

Rules, as useful information for reasoning, are also considered to assist embedding learning. Among these methods different types of rules are considered, for examples, [37] introduces one type of logical rule and three physical rules, [14] utilizes horn soft rules learned from [9], [24] incorporates equivalence and inversion Axioms, and [13] considers two types of logical rules. To incorporate rules into embedding learning, some methods try to improve the embedding method's capability of modeling rules by various ways, such as encoding relations as convex regions in [15] and non-negativity constraints on entity representations in [8]. Some methods regard rules as guidance for embedding learning. For examples, [37] forms the whole learning process as a integer linear programming problem with conditions defined based on rules. [14] infers unlabeled triples according to input rules. [21] adds constrains for relation embeddings participating in path rules. [6] maps entity-tuple embedding into an approximately Boolean space and encourages a partial ordering over relation embedding based on implication rules. [8] adds approximate entailment constraints on relation representations. Some methods jointly embed rules and triples during learning, such as [13] which first learns rules based on embeddings from [3] and then retrains the embeddings incorporating with rules.

Aforementioned embedding methods with rules normally make rule learning detached from embedding learning and unrelated with embeddings. While our goal in this paper is not only an embedding learning system, but also a rule learning system in which rules are learned from embeddings and help improve embedding quality.

5.2 Rule learning

Rule is an important component for reasoning and has been studied in many previous works. Among rule learning works, different kinds of rules are adopted, for example, Horn rules in [9], closed path rules in [20] and general rules with both variables and constants and atoms occurring either positively or negatively in [16], frequent predicate cycles in [38], semantic association rules in [1]. These methods all consider the types of rules to learn from their structures, while in this paper, we consider rules based on their semantics because we think semantics are important for the development of knowledge graphs. Thus we adopt OWL2 object property expression axioms to form the types of rules to learn.

To calculate the confidence of rules, standard confidence or PCA confidence are usually used [9][16] which are based on searching for supports of rules in the whole knowledge graph. To reduce the search space, many rule learning systems have a pruning strategy. Apart from graph search, there are also some works trying to learn both structures and scores of rules based on deep neural models [48] or reinforcement learning [46][5].

Embeddings are also used to help rule learning and are used for different purposes. Some utilize embeddings to guide and prune the search for candidate rules[29]. Some make embeddings to complete the knowledge graph during rule learning[16]. Some use embeddings to assign scores for rules[47][13]. They all rely on calculations among embeddings and the way of calculations depends on specific embedding methods.

Different from these methods which mainly proposed to learning rules, we devote to learn embeddings and rules at the same time and make their advantages contribute to each other's learning.

6 CONCLUSION AND FUTURE WORK

In this paper, we discuss the advantages and disadvantages of two common knowledge graph reasoning methods, embedding-based method and rule-based method, and propose IterE that iteratively learns embeddings and rules in one model and enjoys the mutual benefit between them. In the future, we will continuously investigate combining inductive and deductive reasoning together and develop models that could unify different kinds of reasoning.

ACKNOWLEDGMENTS

This work is funded by NSFC91846204/61473260, national key research program YS2018YFB140004, Alibaba CangJingGe(Knowledge Engine) Research Plan and SNF Sino Swiss Science and Technology Cooperation Programme program under contract RiC 01-032014.

REFERENCES

[1] Molood Barati, Quan Bai, and Qing Liu. 2016. SWARM: An Approach for Mining Semantic Association Rules from Semantic Web Data. In *PRICAI (Lecture Notes in Computer Science)*, Vol. 9810. Springer, 30–43.

[2] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of SIGMOD* (2008), 1247–1250.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. *Proceedings of NIPS* (2013), 2787–2795.

[4] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. KBQA: Learning Question Answering over QA Corpora and Knowledge Bases. *PVLDB* 10, 5 (2017), 565–576.

[5] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alexander J. Smola, and Andrew McCallum. 2017. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. *CoRR* abs/1711.05851 (2017).

[6] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted Rule Injection for Relation Embeddings. In *EMNLP*. The Association for Computational Linguistics, 1389–1399.

[7] Tim Dettmers. 2018. Convolutional 2D Knowledge Graph Embeddings. *Proceddings of AAAI* (2018).

[8] Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving Knowledge Graph Embedding Using Simple Constraints. In *ACL (1)*. Association for Computational Linguistics, 110–121.

[9] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.* 24, 6 (2015), 707–730.

[10] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. 413–422.

[11] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2015. Composing Relationships with Translations. *Proceedings of EMNLP* (2015), 286–290.

[12] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically Smooth Knowledge Graph Embedding. *Proceedings of ACL* (2015), 84–94.

[13] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly Embedding Knowledge Graphs and Logical Rules. In *EMNLP*. The Association for Computational Linguistics, 192–202.

[14] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge Graph Embedding With Iterative Guidance From Soft Rules. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

[15] Víctor Gutiérrez-Basulto and Steven Schockaert. 2018. From Knowledge Graph Embedding to Ontology Embedding? An Analysis of the Compatibility between Vector Space Representations and Rules. In *KR*. AAAI Press, 379–388.

[16] Vinh Thinh Ho, Daria Stepanova, Mohamed H. Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. 2018. Rule Learning from Knowledge Graphs Guided by Embedding Models. In *International Semantic Web Conference (1) (Lecture Notes in Computer Science)*, Vol. 11136. Springer, 72–90.

[17] Petr HÃajek. 1998. The metamathematics of fuzzy logic. (1998).

[18] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. *Proceedings of ACL* (2015), 687–696.

[19] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[20] Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011. Random Walk Inference and Learning in A Large Scale Knowledge Base. In *EMNLP*. ACL, 529–539.

[21] Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. *Proceedings of EMNLP* (2015), 705–714.

[22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. *Proceedings of AAAI* (2015), 2181–2187.

[23] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-relational Embeddings. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2168–2178.

[24] Pasquale Minervini, Luca Costabello, Emir Muñoz, Vít Novácek, and Pierre-Yves Vandenbussche. 2017. Regularizing Knowledge Graph Embeddings via Equivalence and Inversion Axioms. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Vol. 10534. Springer, 668–683.

[25] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional Vector Space Models for Knowledge Base Completion. *Proceedings of ACL* (2015), 156–166.

[26] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33.

[27] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic Embeddings of Knowledge Graphs. *Proceedings of AAAI* (2016), 1955–1961.

[28] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. *Proceedings of ICML* (2011), 809–816.

[29] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2018. Scalable Rule Learning via Learning Representation. In *IJCAI*. ijcai.org, 2149–2155.

[30] Petar Ristoski and Heiko Paulheim. 2016. RDF2Vec: RDF Graph Embeddings for Data Mining. *Procedings of ISWC* (2016), 498–514.

[31] Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. 2017. Modeling Large-Scale Structured Relationships with Shared Memory for Knowledge Base Completion. In *Rep4NLP@ACL*. Association for Computational Linguistics, 57–68.

[32] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. *Prodeddings of NIPS* (2013), 926–934.

[33] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. *Proceedings of WWW* (2007), 697–706.

[34] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. 57–66.

[35] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. *Proceedings ICML* (2016), 2071–2080.

[36] Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Wang, Hai Zhu, and Huajun Chen. 2018. Label-Free Distant Supervision for Relation Extraction via Knowledge Graph Embedding. In *EMNLP*. The Association for Computational Linguistics.

[37] Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge Base Completion Using Embeddings and Rules. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 1859–1866.

[38] Zhichun Wang and Juan-Zi Li. 2015. RDF2Rules: Learning Rules from RDF Knowledge Bases by Mining Frequent Predicate Cycles. *CoRR* abs/1512.07734 (2015).

[39] Zhigang Wang and Juan-Zi Li. 2016. Text-Enhanced Representation Learning for Knowledge Graph. *Proceedings of IJCAI* (2016), 1293–1299.

[40] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph and Text Jointly Embedding. *Proceedings of EMNLP* (2014), 1591–1601.

[41] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. *Proceedings of AAAI* (2014), 1112–1119.

[42] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions. *Proceedings of AAAI* (2017), 3104–3110.

[43] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. *Proceedings of AAAI* (2016), 2659–2665.

[44] Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2017. Image-embodied Knowledge Representation Learning. In *IJCAI*. ijcai.org, 3140–3146.

[45] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types. *Proceedings of IJCAI* (2016), 2965–2971.

[46] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *EMNLP*. Association for Computational Linguistics, 564–573.

[47] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *Proceedings of ICLR* (2015).

[48] Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *NIPS*. 2316–2325.

[49] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. ACM, 353–362.

[50] Wen Zhang. 2017. Knowledge Graph Embedding with Diversity of Structures. *Proceedings WWW Companion* (2017), 747–753.