Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Scalable Rule Learning via Learning Representation

Pouya Ghiasnezhad Omran, Kewen Wang, Zhe Wang
Griffith University
pouya.ghiasnezhadomran@griffithuni.edu.au, {k.wang, zhe.wang}@griffith.edu.au

Abstract

We study the problem of learning first-order rules from large Knowledge Graphs (KGs). With recent advancement in information extraction, vast data repositories in the KG format have been obtained such as Freebase and YAGO. However, traditional techniques for rule learning are not scalable for KGs. This paper presents a new approach RLvLR to learning rules from KGs by using the technique of embedding in representation learning together with a new sampling method. Experimental results show that our system outperforms some state-of-the-art systems. Specifically, for massive KGs with hundreds of predicates and over 10M facts, RLvLR is much faster and can learn much more quality rules than major systems for rule learning in KGs such as AMIE+. We also used the RLvLR-mined rules in an inference module to carry out the link prediction task. In this task, RLvLR outperformed Neural LP, a state-of-the-art link prediction system, in both runtime and accuracy.

1 Introduction

Much attention has recently been given to the creation of large knowledge bases that contain millions of facts about various entities in the world, such as people, universities, movies, animals, etc. These knowledge bases have proven to be incredibly useful for intelligent Web search, question understanding, in-context advertising, social media mining, and biomedicine. Due to their new features, such modern knowledge bases are often referred to as *knowledge graphs* or just *KGs*. Major examples of KGs include YAGO [Suchanek *et al.*, 2007], DBpedia [Auer *et al.*, 2007], Wikidata [Vrandečić and Krötzsch, 2014] and Freebase [Bollacker *et al.*, 2008].

Due to their large data volume, it is impossible to construct large KGs manually. Thus, a major task in KG construction is to develop scalable methods for automated learning of new entities, their properties and relationships. As some researchers have pointed out, a KG is not just a graph database [Nickel *et al.*, 2016a]. In particular, it should have a layer of conceptual knowledge, which is usually represented as a set of rules like $\mathsf{BornIn}(x, y) \wedge \mathsf{Country}(y, z) \rightarrow \mathsf{Nationality}(x, z)$, meaning that if person x was born in city y and y is in country z, then x is a citizen of z. Rules are explicit knowledge (compared to a neural network) and can provide human understandable explanations to learning results (e.g., link prediction) based on them. Thus, it is useful and important to extract rules for KGs automatically.

Traditional methods of rule learning cannot be directly employed in rule construction over KGs for several reasons. First, those methods are not scalable enough to handle the huge amount of data contained in common KGs. For example, DBpedia 3.8 has more than 11M facts, which poses a challenge to most existing methods. Moreover, KGs do not explicitly express negative examples, which are essential for many data mining tools.

In the literature, many new approaches have been proposed to learn rules of various forms from a variety of databases. The problem of mining rules via exploring the space of all possible hypothesizes (rules) has been investigated in the paradigm of Inductive Logic Programming (ILP) [Muggleton, 1996]. Some efficient rule miners for KGs have been developed lately, including SWARM [Barati *et al.*, 2016], RDF2rules [Wang and Li, 2015], ScaleKB [Chen *et al.*, 2016] and AMIE+ [Galárraga *et al.*, 2015]. They are much more efficient than their predecessors and are able to learn rules from large datasets. However, scalability is still a major challenge for existing systems.

On the other hand, in the paradigm of representation learning, statistical predictive models have been widely applied in learning facts that are absent in a KG. Tensor factorization [Nickel *et al.*, 2016b] and the translation-based approach [Lin *et al.*, 2015] are two major approaches in this category. The basic idea is to encode relational information by using low-dimensional representations (embeddings) of entities and predicates. Such representation learning techniques have been applied in learning rules in KGs [Yang *et al.*, 2015; Neelakantan *et al.*, 2015]. This is a promising research direction for rule learning in large KGs but the resulting systems are still not very efficient compared to some major rule miners such as AMIE+ [Galárraga *et al.*, 2015].

In this paper, we tackle this challenge by providing a scalable method for learning rules in KGs. The main idea is to define new embeddings (of arguments) and new scoring functions, and then use them to guide the extraction of rules and thus reduce the search space. Besides, we proposed a novel sampling method and efficient rule evaluating mecha-

nism that allows our system to handle massive benchmarks efficiently.

We have implemented a system prototype RLvLR[1] based on our new methods and compared it with the state of art system AMIE+ on major benchmarks such as YAGO, DBpedia and Wikidata for the rule mining task. Our experimental results show that RLvLR outperformed AMIE+ in both time efficiency and the number of mined quality rules. For example, in a learning task for 20 specific predicates from Wikidata (which has over 8M facts), RLvLR mined more than 56 rules on average in 2.41 hours but AMIE+ mined 1 rule on average in 10 hours (with the same rule quality thresholds). We have also implemented an inference method to predict new facts. RLvLR also outperformed Neural LP [Yang *et al.*, 2017] on link prediction regarding scalability and accuracy.

2 Preliminaries

In this section, we briefly recall some basics of knowledge graphs and representation learning as well as fixing some notations to be used later.

2.1 Knowledge Graphs and Rules

An entity e is an object such as a place, a person, etc., and a fact is an RDF triple (e, P, e'), which means that the entity e is related to another entity e' via the binary predicate P. Following the convention in knowledge representation, we denote such a fact as $P(e, e')$. A *knowledge graph* (*KG*) is a pair $K = (E, F)$, where E is the set of entities and F is the set of facts.

We are interested in closed path rules (or CP rules) as the syntax provides a balance between the expressive power of mined rules and the efficiency of rule mining. Such a syntactic restriction is a standard approach in the rule mining literature. CP rules are the underlying formalism of Path Ranking Algorithms [Gardner and Mitchell, 2015], RuleEmbedding [Yang *et al.*, 2015], [Wang and Li, 2015] and ScaleKB [Chen *et al.*, 2016].

A *CP rule* (or simply a *rule*) r is of the form

$$P_1(x, z_1) \wedge P_2(z_1, z_2) \wedge ... \wedge P_n(z_{n-1}, y) \rightarrow P_t(x, y). \quad (1)$$

Here x, y and z_i's are variables, each $P(u, v)$ is called an atom, and u and v are called respectively, the subject and object argument for P. Intuitively, the rule r reads that if $P_1(x, z_1), P_2(z_1, z_2), ..., P_n(z_{n-1}, y)$ hold, then $P_t(x, y)$ holds too. The atom $P_t(x, y)$ is the head of r, denoted $head(r)$, and the set of atoms $P_1(x, z_1), P_2(z_1, z_2), ..., P_n(z_{n-1}, y)$ is the body of r, denoted $body(r)$. The rule r is called closed-path as the sequence of predicates in the rule body forms a path from the subject argument to the object argument of the head predicate. Note that CP rules allow recursion, i.e., the head predicate can occur in the body.

To assess the quality of mined rules, we recall measures that are used in some major approaches to rule learning [Chen *et al.*, 2016] and [Galárraga *et al.*, 2015].

Let r be a CP rule of the form (1). A pair of entities (e, e') satisfies the body of r, denoted $body(r)(e, e')$, if there exist entities $e_1, ..., e_{n-1}$ in the KG such that $P_1(e, e_1), P_2(e_1, e_2), ..., P_n(e_{n-1}, e')$ are facts in the KG. And (e, e') satisfies the head of r, denoted $P_t(e, e')$, if $P_t(e, e')$ is a fact in the KG. Then the support degree of r is defined as

$$supp(r) = \#(e, e') : body(r)(e, e') \wedge P_t(e, e')$$

To normalize this degree, the degrees of standard confidence (SC) and head coverage (HC) are defined as follows:

$$SC(r) = \frac{supp(r)}{\#(e, e') : body(r)(e, e')}, HC(r) = \frac{supp(r)}{\#(e, e) : P_t(e, e')}$$

2.2 Representation Learning

A method for representation learning from KGs often consists of two major steps: (1) to embed the entities and predicates of the given KG into a latent space, and (2) to construct a learning model based on the obtained embeddings to predict new facts.

Various approaches have been proposed to construct embeddings (e.g. [Shen, 2016]), which include translation based embeddings [Bordes *et al.*, 2013] and matrix factorization based embeddings [Nickel *et al.*, 2011; 2016b]. The translation based embeddings use additive calculus and use vectors to represent the embeddings of predicates. The matrix factorization based embeddings use dot calculus and the embedding of a predicate is a matrix. Since our rule mining approach requires a relatively expressive form of embeddings, we adopt matrix factorization based embeddings. In particular, we employ the state-of-the-art RESCAL system [Nickel *et al.*, 2011; 2016b] to construct embeddings. This choice is based on some reasons. First of all, since predicate embeddings are used to guide the candidate rule search, we need expressive predicate embeddings (i.e., matrices instead of vectors) produced by RESCAL. In addition, some other systems that compute embeddings as matrices were unavailable. As we will see later, RESCAL is sufficient for our purposes.

RESCAL embeds each entity e to a vector $\mathbf{E}$ and each predicate P to a matrix $\mathbf{P}$. For each given fact $P_0(e_1, e_2)$, the following scoring function is computed:

$$f(e_1, P_0, e_2) = \mathbf{E}_1^T \cdot \mathbf{P_0} \cdot \mathbf{E}_2$$

The scoring function indicates the plausibility of the fact $P_0(e_1, e_2)$. While existing representation learners, such as RESCAL and TransE, use entity embeddings and predicate embeddings for computing the plausibility score of the missing facts, we will use these embedding to find the plausibility of the desired rules later.

3 An Overview of Our Approach

In this section, we present our embedding-based approach to rule learning in a nutshell. We focus on discriminative rule mining, that is, specifying a target predicate P_t in a KG, to mine quality rules whose head has the predicate P_t.

In contrast to ILP approaches, instead of using a refinement operator to search the rule space, we use embedding models to effectively prune the search. However, this is not

[1] https://www.ict.griffith.edu.au/aist/RLvLR/

straightforward. In order to develop an efficient algorithm for rule learning problem using embedding models, we need to resolve two issues. First, existing embedding models do not work directly for vast KGs. For instance, RESCAL is unable to handle YAGO2 [Nickel *et al.*, 2016b]. We address this issue by introducing a new sampling algorithm to restrict the range of entities to be considered so that embeddings are computed only for those entities that are relevant to the target predicate. Moreover, we need a more efficient and refined approach to rank candidate rules. To this purpose, we first define the embeddings of arguments of predicates and then introduce scoring functions that allow fast ranking of rules, based on embeddings of entities, predicates and arguments (instead of only entities and predicates). In addition, efficient numerical algorithms for matrices are also employed to speed up our method.

The above two aspects are also major sources where the efficiency of RLvLR is achieved. Our method for rule learning is summarised in the following algorithm, while some major components in the algorithm will be explained later.

Algorithm 1 Learn rules for a KG and a target predicate

Input: a KG K, a predicate P_t, an integer $len \geq 2$, and two real numbers $MinSC, MinHC \in [0, 1]$
Output: a set $Rule$ of CP rules

1: $K' := \mathsf{Sampling}(K, P_t, len)$
2: $(\mathcal{P}, \mathcal{A}) := \mathsf{Embeddings}(K')$
3: $Candidates := \emptyset$
4: **for** $2 \leq l \leq len$ **do**
5: Add $\mathsf{RuleSearch}(K', P_t, \mathcal{P}, \mathcal{A}, l)$ to $Candidates$
6: **end for**
7: $Rules := \mathsf{Evaluate}(Candidates, K)$
8: $Rules := \mathsf{Filter}(Candidates, MinSC, MinHC)$
9: **return** $Rules$

In Algorithm 1, the integer len is for the maximum length of rules to be learned; $MinSC$ and $MinHC$ set the minimum values of standard confidence and head coverage for learned rules, respectively. Due to the vast size of the input KG $K = (E, F)$, it is necessary to sample the data first. For this purpose, we use a sampling method $\mathsf{Sampling}()$ to obtain an (often much) smaller KG $K' = (E', F')$ that contains only those entities and facts that are relevant to the target predicate P_t. To learn rules of maximum body length $len = n$, we generate sample entities in an incremental manner $E_0, \ldots, E_{n-1}$ as follows:

- E_0 consists of the entities that are connected to another entity in E by P_t, i.e., $E_0 = \{e \mid$ there exists an entity e' in E s.t. $P_t(e, e') \in F$ or $P_t(e', e) \in F\}$;
- E_i $(0 < i < n)$ consists of the entities that are connected to another entity in E_{i-1} by any predicate P, i.e., $E_i = \{e \mid$ there exists an entity e' in E_{i-1} s.t. $P(e, e') \in F$ or $P(e', e) \in F\}$;

Since we are interested only in CP rules with no more than n body atoms, the subset $E' = \bigcup_{i=0}^{n-1} E_i$ of all the entities in K covers almost all information needed for mining such rules. We thus generate a sample set F' of facts as follows:

$$F' = \{P(e_1, e_2) \mid e_1, e_2 \in E', P(e_1, e_2) \in F\}. \quad (2)$$

After sampling, the next step, i.e., $\mathsf{Embeddings}()$, is to obtain the embeddings $\mathcal{P}, \mathcal{A}$ for respectively predicates and arguments in K'. Existing systems such as RESCAL usually compute embeddings only for entities and predicates. We will define the notion of argument embeddings later based on entity embeddings, which together with embeddings of predicates provide more refined scoring functions for candidate rules.

The search for candidate rules of the form (1) is actually reduced to the search for plausible paths, that is, sequences of predicates $P_1, P_2, \ldots, P_n$ and their inverses. This is achieved through the method $\mathsf{RuleSearch}()$. We use the proposed scoring function to guide and prune the search, which turns out to be rather effective for extracting rules. Then the selected candidates are kept for the final evaluation.

Finally, the candidates are evaluated according to their SC and HC before being returned. This is achieved through the method $\mathsf{Evaluate}()$. To compute the SC and HC of candidate rules, we utilise the efficient matrix multiplication. The embedding of arguments, the new scoring function and the employment of matrix algorithms greatly contributed to the scalability of our approach. More technical details will be explained in Sections 4 and 5.

To show the usefulness of extracted rules, we implemented an inference method that predicts new facts with a degree of confidence based on the given facts and first-order rules that are augmented with SC scores. To obtain the confidence degree (CD) of a fact, we adapt the $score^*(\cdot)$ function from [Galárraga *et al.*, 2015] by aggregating the SC of all the rules inferring the facts in a Noisy-OR manner. The intuition is that facts inferred by more rules should have a higher confidence degree. Instead of using the PCA scores as in [Galárraga *et al.*, 2015], we use SC as it is easier to compute. Formally, for a fact $f = P(e, e')$ and the set of rules $\mathcal{R}$ that can infer f from the given KG, the CD of f is defined as follows:

$$CD(f) = 1 - \prod_{r \in \mathcal{R}} (1 - SC(r))$$

With this module, RLvLR becomes an end-to-end learner that is able to handle the link prediction task.

4 Scoring Functions

As explained above, the task of searching for CP rules can be reduced to that of searching for plausible paths of predicates. This is done by introducing scoring functions over all possible paths. To define such a scoring function, note that a path $p = P_1, P_2, \ldots, P_n$ can be seen as a binary predicate between the starting entity and the ending entity, and p is plausible if the pairs of entities associated by the path are similar to those associated by the target predicate P_t. Such a similarity between p and P_t is referred to as *synonymy*, where two predicates associate similar pairs of entities. For instance, the predicates $\texttt{BornIn}(u, v)$ and $\texttt{LiveIn}(u, v)$ associate similar pairs of a person u and a place v.

Based on this intuition, a scoring function is defined using an embedding model in [Yang *et al.*, 2015]. We use the following rule r of length 3 to illustrate the method:

$$P_1(x,z) \wedge P_2(z,y) \rightarrow P_t(x,y).$$

The embedding of path $p = P_1, P_2$ is $\mathbf{P}_1 \cdot \mathbf{P}_2$, which should be similar to the embedding of P_t, denoted $\mathbf{P}_1 \cdot \mathbf{P}_2 \approx \mathbf{P}_t$. Hence, the synonymy scoring function is

$$f_1(r) = sim(\mathbf{P}_1 \cdot \mathbf{P}_2, \mathbf{P}_t) \quad (3)$$

where sim is defined by the Frobenius norm as follows, for two matrices $\mathbf{M}_1$ and $\mathbf{M}_2$,

$$sim(\mathbf{M}_1, \mathbf{M}_2) = exp(-\|\mathbf{M}_1 - \mathbf{M}_2\|_F).$$

Yet for rules gets longer than 2, the computation of synonymy scoring function involves nesting of matrix manipulation and become less effective for exploring the search space as it involves whole paths. Thus, we propose a local scoring function based on *co-occurrence*. Besides synonymy, co-occurrence is also widely studied in natural language processing [Jones *et al.*, 2015]. In our context, P_1 and P_2 are adjacent in a path only if they share many entities in the KG. For example, $\texttt{LiveIn}(u,v)$ and $\texttt{LocatedIn}(v,w)$ share cities v.

To define a co-occurrence scoring function, we first introduce the notion of argument embeddings. For an argument of a predicate, its embedding is defined as the average value of the embeddings of all the entities appearing in the position of this argument. Formally, the embeddings of the subject and object argument of a predicate P are defined as:

$$\mathbf{P}^s = \frac{1}{n}\sum_{e \in E_P^s} k_e.\mathbf{E} \quad \text{and} \quad \mathbf{P}^o = \frac{1}{n}\sum_{e \in E_P^o} l_e.\mathbf{E}$$

where $n = \#\{P(e,e') \in F'\}$, $E_P^s = \{e \mid \exists e' \text{ s.t. } P(e,e') \in F'\}$, $E_P^o = \{e' \mid \exists e \text{ s.t. } P(e,e') \in F'\}$, F' is defined by Eq.(2), , $k_e = \#\{P(e,e') \mid \exists e' \text{ s.t. } P(e,e') \in F'\}$ and $l_e = \#\{P(e',e) \mid \exists e \text{ s.t. } P(e,e') \in F'\}$. In path p, the embedding of the object argument of P_1 should be similar to the embedding of the subject argument of P_2, denoted $\mathbf{P}_1^o \approx \mathbf{P}_2^s$. Hence, the co-occurrence scoring function is

$$f_{loc}(P_1, P_2) = sim(\mathbf{P}_1^o, \mathbf{P}_2^s).$$

Similarly, we should have $\mathbf{P}_1^s \approx \mathbf{P}_t^s$ and $\mathbf{P}_2^o \approx \mathbf{P}_t^o$, as well as the corresponding co-occurrence scoring functions

$$f_{loc}(P_1, P_t) = sim(\mathbf{P}_1^s, \mathbf{P}_t^s), f_{loc}(P_2, P_t) = sim(\mathbf{P}_2^o, \mathbf{P}_t^o)$$

We aggregate the local scoring functions as follows.

$$f_2(r) = f_{loc}(P_1, P_2) + f_{loc}(P_1, P_t) + f_{loc}(P_2, P_t). \quad (4)$$

While the scoring function f_1 considers the predicates in the paths, with f_2, we consider the shared variables that are involved in the paths. Consequently, we use both of these two scoring functions, which complement each other.

5 Rule Evaluation

In the Evaluate() method, for efficiency, we first evaluate the picked candidate rules based on the sampled KG K', which provides us an estimated quality of rules. We return the set of rules with $supp \geq 1$. These rules may still contain a large number of redundant and low quality rules and thus it is necessary to further clean the returned rules based on two measures SC and HC.

To compute SC and HC degrees, a method is required to check the satisfiability of body atoms of all candidate rules in the last phase. In other words, we need to figure out all relevant atoms that can trigger a candidate rule. In this task, we have a KG (e.g. K or K') and a CP rule as input. Let $E = \{e_1, \ldots, e_n\}$ be the set of all entities and $\mathbb{P} = \{P_1, \ldots, P_m\}$ be the set of all predicates. We can represent the input KG as a set of S matrices like RESCAL, where each $n \times n$ matrix $S(P_k)$ corresponds to a predicate P_k in the KG ($1 \leq k \leq m$). Specifically, the (i,j) entry $S(P_k)[ij]$ of the adjacency matrix $S(P_k)$ is 1 if the fact $P_k(e_i, e_j)$ is in the input KG; 0 otherwise. Thus, $S(P_k)$ is a binary one.

There is a close connection between the product of adjacency matrices for predicates and the closed-path rules. We use an example to explain the idea. Consider the rule r: $P_1(x,z) \wedge P_2(z,y) \rightarrow P_t(x,y)$. We say a fact $P_t(e,e')$ is inferred by the rule r in a KG $K = (E,F)$ if $P_1(e,e'') \in F$ and $P_2(e'',e') \in F$ for some $e'' \in E$. Then the product $S(P_1) \cdot S(P_2)$ of two matrices $S(P_1)$ and $S(P_2)$ is the adjacency matrix of the set of facts that are inferred by r.

The $[i,j]$ entry of $S(P_1) \cdot S(P_2)$ represents the number of rule paths that start from e_i, traverse via P_1 to another entity and so on but finally go to e_j via P_2. From a matrix S, we can obtain a binary matrix $\mathsf{binary}(S)$ by setting all non-zero entries of S as 1. Let $S(P_1, P_2) = \mathsf{binary}(S(P_1) \cdot S(P_2))$. Note that although we explain the case where rule length is three, it is straightforward to generalize this relation to rules of any length.

Let r be $P_1(x,z) \wedge P_2(z,y) \rightarrow P_t(x,y)$ and $K = (E,F)$ where $E = \{e_1, e_2, e_3\}$ and

$$K = \{P_1(e_1,e_2), P_1(e_2,e_1), P_1(e_1,e_3), P_2(e_2,e_3),$$
$$P_2(e_2,e_1), P_2(e_3,e_3), P_t(e1,e3)\}$$

The matrices for the predicates P_1, P_2 and P_t are:

$$S(P_1): \begin{bmatrix}0&1&1\\1&0&0\\0&0&0\end{bmatrix}, S(P_2): \begin{bmatrix}0&0&0\\1&0&1\\0&0&1\end{bmatrix}, S(P_t): \begin{bmatrix}0&0&1\\0&0&0\\0&0&0\end{bmatrix}$$

Then

$$S(P_1) \cdot S(P_2) = \begin{bmatrix}1&0&2\\0&0&0\\0&0&0\end{bmatrix} \xrightarrow{\mathsf{binary}()} S(P_1, P_2) = \begin{bmatrix}1&0&1\\0&0&0\\0&0&0\end{bmatrix}$$

The last matrix represents the pairs that are inferred by the rule: $\{(e_1,e_1), (e_1,e_3)\}$. Since we have just one fact for the target predicate, $P_t(e_1,e_3)$, the quality degrees of rule r are $HC = 1$ and $SC = 0.5$, respectively.

6 Experiments

Based on the methods presented in previous sections, we have implemented a system RLvLR (Rule Learner via Learning Representation) and conducted two sets of experiments to evaluate the new system. The executable codes, benchmark datasets and experimental results are publicly available at https://www.ict.griffith.edu.au/aist/RLvLR/.

KG	# Facts	# Entities	# Predicates
FB15K-237	310K	15k	237
FB75K	316K	75K	13
YAGO2s	4.12M	1.65M	37
Wikidata	8.40M	4.00M	430
DBpedia 3.8	11.02M	2.20M	650

Table 1: Benchmark specifications

The first set aims to evaluate the scalability of RLvLR and the number of quality rules learned by the system, while the second set is to evaluate the scalability and accuracy of RLvLR for link prediction. The benchmark datasets adopted in our experiments include various versions of Freebase, YAGO, DBpedia and Wikidata that are widely used in experimental evaluations by major systems for rule learning and link prediction in KGs. RLvLR was compared to state-of-the-art rule miners such as AMIE+ [Galárraga *et al.*, 2015] and ScaleKB [Chen *et al.*, 2016], as well as state-of-the-art system for link prediction, Neural LP [Yang *et al.*, 2017]. Our experiments were designed to validate the following statements:

1. RLvLR is much faster than major rule mining systems such as AMIE+ for large-scale KGs.
2. RLvLR is able to mine significantly more quality rules than AIME+, especially on vast datasets such as DBpedia 3.8 and Wikidata.
3. Regarding link prediction, RLvLR outperforms the state-of-the-art link prediction system Neural LP in terms of both scalability and accuracy.

The five benchmark datasets are specified in Table 1, where the last three have been often used in rule mining [Galárraga *et al.*, 2015; Chen *et al.*, 2016]. FB15K-237 [Toutanova and Chen, 2015] (aka. FB15KSelected) and FB75K (from NIPS'13 dataset) are obtained from Freebase and widely adopted for link prediction benchmarking [Yang *et al.*, 2017].

For benchmark KGs FB15K-237 and FB75K, we used a PC with Intel Core i5-4590 CPU at 3.30GHz $\times$ 4 and with 5GB of RAM, running Ubuntu 14.04. For other larger benchmark KGs we tested, the experiments were conducted on a server with Intel Xeon CPU at 2.67GHz (one thread) and with 40GB of RAM, running RedHat Linux 6.1.

6.1 Rule Mining

This set of experiments concerns mining quality rules. The rule quality was measured by standard confidence (SC) and head coverage (HC) [Galárraga *et al.*, 2015]. Note that while we used sampling for rule mining, HC and SC degrees of mined rules are computed over the whole datasets (i.e., not on the samples).

Experiment 1 We randomly selected 20 target predicates for YAGO2, Wikidata and DBPedia. A 10 hour limit was set for each target predicate. Table 2 shows the average numbers of quality rules (#R, SC$\geq$ 0.1 and HC$\geq$ 0.01 as in [Galárraga *et al.*, 2015]) and numbers of high quality rules (#QR, SC$\geq$ 0.7) mined for selected target predicates and the running times (in hours, averaged over the targets) of RLvLR and AMIE+.

KG	RLvLR			AMIE+		
	#R	#QR	Time	#R	#QR	Time
YAGO2s	**6.3**	**1.8**	**0.96**	5.65	0.5	10.00
DBpedia 3.8	**42.7**	**9.2**	**3.88**	9.05	0.5	4.59
Wikidata	**56.8**	**25.6**	**2.41**	0.95	0.3	10.00

Table 2: Rule mining comparison between RLvLR and AMIE+

KG	RLvLR		AMIE+	
	#Facts	#QFacts	#Facts	#QFacts
YAGO2s	**1.1M**	**7K**	0.27M	1
DBpedia 3.8	**16.6M**	**162K**	1.6M	1.8K
Wikidata	**2.1M**	**99K**	0.17M	4.6K

Table 3: The numbers of new facts predicted by RLvLR and AMIE+

Compared to AMIE+, RLvLR showed better performance in terms of both the runtime and the numbers of mined quality rules. The superiority of RLvLR is more obvious in mining high quality rules. Note that RLvLR deployed the same redundancy elimination as AMIE+, and the numbers were obtained after the redundancy elimination.

To estimate the predictive power of the corpus of mined rules, we eliminated from each benchmark 30% of its facts (up to 5K facts) involving the target predicates and checked how many facts (including the eliminated ones) can be predicated by applying mined rules on the remaining facts. Table 3 shows the numbers of predicted facts (# Facts) and those predictions with CD$\geq$ 0.9 (#QFacts). In this part, we consider five target predicates in all three benchmarks. Note that while AMIE+ mined some non-CP rules whose application cannot be implemented using our inference method, the majority of them can be applied: 77% for YAGO2s, 100% for DBpedia, and 100% for Wikidata.

The numbers of quality rules mined by RLvRL on YAGOs, DBpedia 3.8 and Wikidata are 1.1, 4.7 and 59.7 times of those mined by AMIE+ (from Table 2), where as the facts predicated by RLvRL are 4.1, 10.3 and 12.3 times of those predicated by AMIE+. This suggests the usefulness of the additional rules mined by RLvLR in link prediction.

Since we were unable to run ScaleKB system or obtain the Freebase benchmark used in [Chen *et al.*, 2016], we could only compare the rules mined by RLvLR on YAGO2s with those reported in [Chen *et al.*, 2016]. As they only reported rules with length up to 3, we restricted ourselves to rules of length 3 too. For instance, we observed that the following rules that were learned by RLvLR but not by ScaleKB. The two numbers preceding a rule denote SC and HC degrees of the rule, respectively.

$$0.82, 1 : \mathsf{isAffiliatedTo}(x, y) \rightarrow \mathsf{playsFor}(x, y).$$
$$1, 0.82 : \mathsf{playsFor}(x, y) \rightarrow \mathsf{isAffiliatedTo}(x, y).$$

We also observed a large number of informative rules of length 4 learned by RLvLR but neither ScaleKB nor AMIE+ could learn them. For example, the following pattern with

Target Predicate	RLvLR	RLvLR*
formOfGovernment	**137**	97
awardWinner	**1024**	574
parentGenre	**52**	30
sameDirector	**691**	284
eventLocation	**229**	141

Table 4: Number of mined rules with different scoring funcions

three body atoms:

$$0.89, 0.13 : \mathsf{hasChild}(x,t) \wedge \mathsf{hasChild}^{-1}(t,z) \wedge \mathsf{isCitizenOf}(z,y) \rightarrow \mathsf{isCitizenOf}(x,y).$$

This rule means that if a person x has a common child t with somebody else z and the person z is a citizen of some place y, then the first person x is also the citizen of place y.

Experiment 2 We have conducted an experiment to demonstrate that our new scoring function is more informative than the scoring function defined in [Yang *et al.*, 2015]. For this purpose, we implemented a system, named RLvLR*, by replacing the scoring function in RLvLR with the scoring function in [Yang *et al.*, 2015]. We set a 5 hours time limit, SC$\geq$ 0.01 and HC$\geq$ 0.001. As FB15k-401 KG was used in [Yang *et al.*, 2015], our experiment was conducted on the similar benchmark, FB15K-237. The experimental results are summarised in Table 4, which show that our scoring function was capable to mine (up to 2.4 times) more rules than that reported in [Yang *et al.*, 2015].

6.2 Link Prediction

The second set of experiments aim to evaluate the predictive power of mined rules for link prediction in KGs. Specifically, our experiments show that, for the task of link prediction, RLvLR significantly outperforms Neural LP in terms of scalability, while the accuracy of RLvLR is comparable to that of other systems for link prediction. So, the major advantage of RLvLR is in its capability of handling vast KGs with a accuracy that is comparable to other major systems. Note that our goal is not to compete with them on accuracy of link prediction for relatively small KGs.

Given a KG, the task of link prediction is to identify for each predicate P and each entity e, an entity e' such that $P(e,e')$ is in the KG; or alternatively, to identify for each predicate P and each entity e', an entity e such that $P(e,e')$ is in the KG.

We conducted two experiments for link prediction. The first one is to demonstrate the scalability of RLvLR while the second one is to show that RLvLR is comparable to major systems of link prediction in terms of accuracy.

Following the experiments of Neural LP [Yang *et al.*, 2017], we used two metrics Mean Reciprocal Rank (MRR) and Hits@10. MRR is the average of the reciprocal ranks of the desired entities and Hits@10 is the percentage of desired entities being ranked among top ten.

Experiment 1 In this experiment, we compared RLvLR with Neural LP [Yang *et al.*, 2017] on two benchmark datasets FB75K and WikiData. The two datasets FB75K and WikiData have 75K and 4M entities, respectively. Each dataset is divided into training set (70%) and test set (30%). Neural LP is a state-of-the-art system for link prediction based on rule learning and it features for its scalability. It was able to handle FB75K while it could not handle Wikidata in our experiment. We note that the largest dataset for comparing Neural LP with other systems is FB15K-237, which is still much smaller than FB75K regarding the number of entities. We ran both RLvLR and Neural LP on these two datasets for link prediction. While all 13 predicates of FB75K were tested, 50 randomly selected predicates for Wikidata were tested as it has too many predicates (430). The experimental results are summarised in Table 5.

KG	RLvLR			Neural LP		
	MRR	Hits@10	Time	MRR	Hits@10	Time
FB75K	**0.34**	**43.4**	**0.09**	0.13	25.7	2.33
WikiData	0.29	38.9	3.14	-	-	-

Table 5: Link prediction of RLvLR and Neural LP on large KGs

Learner	MRR	Hits@10
DISTMULT	0.25	40.8
Node+LinkFeat	0.23	34.7
Neural LP	0.24	36.1
RLvLR	0.24	39.3

Table 6: The comparison on link prediction on FB15K-237

The parameters of RLvLR were set to SC$\geq$ 0.005 and HC$\geq$ 0.001 to achieve better accuracy. The time unit was hour and a 5-hour time limit was set for each target predicate.

Experiment 2 In the literature, three benchmarks WN18, FB15K and FB15K-237 are usually used for evaluating link prediction [Nickel *et al.*, 2016b; Lin *et al.*, 2015; Yang *et al.*, 2017]. As pointed out in [Yang *et al.*, 2017], the challenging benchmark for state-of-the-art systems is FB15K-237 as the test entities are rarely directly linked in the KG, a system for link prediction needs to reason explicitly about compositions of relations. The CP rules learned by RLvLR and Neural LP can naturally capture such relation compositions. So, we selected FB15K-237 as the benchmark for our experiment.

Yang et al. selected five major systems for comparing with their Neural LP but for the dataset FB15K-237, experimental results are only available for the two systems DISTMULT and Node+LinkFeat. As a result, we included these two systems and Neural LP for comparison. The test results for these three systems are extracted from the experimental results for Neural LP [Yang *et al.*, 2017]. We compare the performance of RLvLR with the three systems and the experimental results are summarized in Table 6.

From the results, RLvLR obtained better Hits@10 than Neural LP and Node+LinkFeat while DISMULT is slightly better. Regarding the MRR measure, all of these systems performed quite similar.

7 Conclusion

In this paper, we have proposed a system RLvLR for extracting closed-path rules, a special but useful class of first-order

rules, from RDF Knowledge Graphs (KGs). RLvLR mines in the space of closed-path rules (hypothesizes) by exploring the space of embeddings of predicates and arguments. This is achieved by a novel embedding model and new scoring functions. A target oriented sampling has also been proposed, which significantly contributes to the scalability of RLvLR in handling large KGs with over 10 million facts. In a rule mining system, a challenging and time-consuming task is about how to evaluate candidate rules, and we reduce its computation to a series of matrix operations.

Our experimental results demonstrate that RLvLR outperforms AMIE+ in terms of rule quality and efficiency. For link prediction, RLvLR outperforms Neural LP in terms of efficiency and accuracy.

We plan to develop an efficient parallel algorithm for RLvLR. The current algorithm learns rules with different target predicates independently, which could be useful for developing a parallel algorithm.

Acknowledgments

This work was partially supported by the Australian Research Council (ARC) under DP130102302.

References

[Auer *et al.*, 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proc. ISWC*, pages 722–735, 2007.

[Barati *et al.*, 2016] Molood Barati, Quan Bai, and Qing Liu. SWARM: An Approach for Mining Semantic Association Rules from Semantic Web Data. In *Proc. PRICAI*, pages 30–43, 2016.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. SIGMOD*, pages 1247–1250, 2008.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proc. NIPS*, pages 2787–2795, 2013.

[Chen *et al.*, 2016] Yang Chen, Daisy Zhe Wang, and Sean Goldberg. ScaLeKB: scalable learning and inference over large knowledge bases. *The VLDB Journal*, pages 893–918, 2016.

[Galárraga *et al.*, 2015] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal*, pages 707–730, 2015.

[Gardner and Mitchell, 2015] Matt Gardner and Tom Mitchell. Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In *Proc. EMNLP*, pages 1488–1498, 2015.

[Jones *et al.*, 2015] Michael N. Jones, Jon Willits, and Simon Dennis. Models of semantic memory. *Oxford Handbook of Mathematical and Computational Psychology*, pages 232–254, 2015.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proc. AAAI*, pages 2181–2187, 2015.

[Muggleton, 1996] Stephen Muggleton. Learning from positive data. In *Proc. ILP*, pages 358–376, 1996.

[Neelakantan *et al.*, 2015] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional Vector Space Models for Knowledge Base Inference. In *Proc. AAAI Spring Symposium Series*, pages 156–166, 2015.

[Nickel *et al.*, 2011] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proc. ICML*, pages 809–816, 2011.

[Nickel *et al.*, 2016a] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graph. *Proceedings of the IEEE*, 104(1): 1–23, 2016.

[Nickel *et al.*, 2016b] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic Embeddings of Knowledge Graphs. In *Proc. AAAI*, pages 1955–1961, 2016.

[Shen, 2016] Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. Traversing Knowledge Graph in Vector Space without Symbolic Space Guidance. arXiv preprint arXiv:1611.04642, 2016.

[Suchanek *et al.*, 2007] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proc. WWW*, pages 697–706, 2007.

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

[Vrandečić and Krötzsch, 2014] Denny Vrandečić and Markus Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10): 78–85, 2014.

[Wang and Li, 2015] Zhichun Wang and Juan-Zi Li. RDF2Rules: Learning Rules from RDF Knowledge Bases by Mining Frequent Predicate Cycles. *Computer Research Repository*, 2015.

[Yang *et al.*, 2015] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proc. ICLR*, page 12, 2015.

[Yang *et al.*, 2017] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *Proc. NIPS*, 2017.