An Embedding-based Approach to Rule Learning in Knowledge Graphs

Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang

Abstract—It is natural and effective to use rules for representing explicit knowledge in knowledge graphs. However, it is challenging to learn rules automatically from very large knowledge graphs such as Freebase and YAGO. This paper presents a new approach, RLvLR (Rule Learning via Learning Representations), to learning rules from large knowledge graphs by using the technique of embedding in representation learning together with a new sampling method. Based on RLvLR, a new method RLvLR-Stream is developed for learning rules from streams of knowledge graphs. Both RLvLR and RLvLR-Stream have been implemented and experiments conducted to validate the proposed methods regarding the tasks of rule learning and link prediction. Experimental results show that our systems are able to handle the task of rule learning from large knowledge graphs with high accuracy and outperform some state-of-the-art systems. Specifically, for massive knowledge graphs 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 knowledge graphs such as AMIE+. In the setting of knowledge graph streams, RLvLR-Stream significantly improved RLvLR for both rule learning and link prediction.

Index Terms—Rule learning, knowledge graphs.

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 [1], DBpedia [2], Wikidata [3] and Freebase [4].

As some researchers have pointed out, a KG is not just a graph database [5]. 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 a person x was born in a city y and y is in a country z, then x is a citizen of z. A rule may not be true for all instances and thus a confidence degree is associated with the rule. Rules are explicit knowledge (compared to a neural network) and can provide human understandable explanations to learning results (e.g., link prediction). Thus, it is useful and important to extract rules for KGs automatically.

Rule learning has been studied for long [6]. In particular, numerous approaches to learning first-order rules have been proposed in Inductive Logic Programming (ILP) [6], [7], [8], [9]. However, traditional ILP techniques face at least two new challenges when they are applied to learn first-order rules from large KGs. First, a practical KG can be much larger than datasets typically considered in the ILP literature. For instance, as of 2012, YAGO contains 10 million entities and more than 120 million facts about these entities. Moreover, in the setting of rule learning with KGs, no negative examples are specified, which are required by most ILP rule learners.

In the past few years, some relatively efficient rule learners for KGs have been developed, including AMIE+ [10], RDF2rules [11], SWARM [12], ScaLeKB [13], RuDiK [14], and RuLES [15]. Among these approaches, AMIE+ [10] and ScaLeKB [13] demonstrate outstanding performance and are capable of handling rule learning from some large KGs like Wikidata, with more than 4M entities. AMIE+ gains its scalability by empowering traditional ILP methods with optimized query writing techniques and improved estimation on rule quality, whereas ScaLeKB further improves the scalability (in successfully processing Freebase) by deploying a novel pruning strategy and data partitioning techniques. While they are much more efficient than their ILP predecessors and are able to learn rules from large datasets, learning efficiency is still a major challenge.

Another stream of research on large KGs is representation learning [5], which builds statistical models of KGs by embedding entities and relations in KGs into low-dimensional latent spaces (called embeddings), and has achieved favourable accuracy in KG completion tasks such as link prediction [16], [17], [18], [19]. Recently, efforts have been made by employing such embedding techniques in rule learning [20], [21], [22]. EmbedRule [20] learns a fragment of first-order rules by exploring the space of predicate embeddings over KGs. Yet this approach is still limited to relatively small datasets like FB15K-237 (even with a powerful GPU). Neural LP [21] utilizes the Tensor Networks framework and learns essentially first-order rules as the underlying structure of its models. Yet does not output rules but directly applies the learned model for link prediction. Employing representation learning for rule learning shows promising results, but the scalability of existing (embedding-based) systems still cannot compete with AMIE+ or ScaLeKB; especially, they cannot handle KGs

- *The authors are with the School of Information and Communication Technology, Griffith University, Queensland, Australia.*

like Wikidata or Freebase.

In this paper, we propose an efficient rule learning approach for large-scale KGs, RLvLR (i.e., Rule Learning via Learning Representations), by exploiting the embedding techniques. Technically, the much enhanced learning efficiency is achieved due to a combination of three novel techniques. First, we develop a novel sampling method to significantly reduce the sizes of input KGs by pruning loosely relevant facts, which makes the computation of embeddings highly scalable. Second, we introduce a refined measure for rule-quality estimation through a novel notion of argument embeddings. Third, we develop an efficient algorithms based on matrix operations for rule-quality evaluation, in order to avoid expensive query answering operations (as in the implementation of EmbedRule).

We have implemented a prototype system RLvLR[1] and compared it with the state-of-the-art systems like AMIE+ [10] and Neural LP [21], on common benchmarks such as YAGO, DBpedia and Wikidata. The experimental results show that RLvLR outperforms AMIE+ in both time efficiency and the number of quality rules learned. For example, in a learning task on Wikidata (with over 8M facts), RLvLR learned more than 56 rules in less than 2.5 hours (on average) whereas AMIE+ learned only 1 rule using 10 hours (under the same rule quality thresholds). We have also implemented a rule-inference method for link prediction, and compared RLvLR with Neural LP. RLvLR also outperforms Neural LP on link prediction regarding both scalability and accuracy.

An increasing number of KGs are emerging that model events taking places over time besides static relations among entities. Based on RLvLR, we develop a new method RLvLR-Stream for learning temporal rules from KG streams. It can be used to complete temporal KGs and to predict events in the setting of KG streams. Our experiments show that RLvLR-Stream demonstrates clearly superior performance over its static counterpart RLvLR.

This paper is a significant extension of our conference paper [23], with our rule learning method and evaluation for KG streams. It is organised as follows. After introducing some preliminaries in Section 2, we describe our (static) rule learning approach, RLvLR, in Section 3, and present its extension for KG streams, RLvLR-Stream, in Section 4. We show results on experimental evaluation for both RLvLR and RLvLR-Stream in Section 5. Finally, we conclude our work in Section 6.

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* (or *link*) 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 a set of entities and F is a set of facts.

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

We focus on closed-path rules (or CP rules), which is a language bias that is widely adopted in the rule learning literature for knowledge graphs; for instance, CP rules are the underlying formalism of Path Ranking Algorithms [24], RuleEmbedding [20], RDF2Rules [11] and ScaLeKB [13]. Focusing on CP rules allows us to develop scalable methods, by reducing the search for candidate rules to the problem of path finding (and ranking). Also, CP rules are expressive enough to be used for link prediction as we will show in the experiments.

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

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

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 $P_1(x, z_1), P_2(z_1, z_2), \ldots,$ and $P_n(z_{n-1}, y)$ hold, then $P(x, y)$ holds too. The atom $P(x, y)$ is the head of 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. 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 [13] and [10].

Let r be a CP rule of the form (1). A pair of entities (e, e') satisfies the body of r, denoted $\mathit{body}(r, e, e')$, 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 $\mathit{head}(r, e, e')$, if $P(e, e')$ is a fact in the KG. Then the support degree of r is defined as

$$\mathit{supp}(r) = \#(e, e') : \mathit{body}(r, e, e') \wedge \mathit{head}(r, e, e')$$

To normalize support degree, the notions of standard confidence and head coverage have been introduced, which correspond to the standard accuracy and recall, respectively. The standard confidence (SC) is the ratio between support degree and the number of entity pairs satisfying the body, whereas head coverage (HC) is the ratio between support degree and the number of entity pairs satisfying the head.

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

$$HC(r) = \frac{\mathit{supp}(r)}{\#(e, e') : \mathit{head}(r, e, e')}$$

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 model based on the obtained embeddings to predict new facts.

Various approaches have been proposed to construct embeddings [18], which include translation based embeddings [17] and matrix factorization based embeddings [16],

[19]. The translation based approaches use vectors to embed predicates and employ additive calculus on such vectors, whereas the matrix factorization based approaches use matrices to embed predicates and dot calculus on matrices. Since our approach requires a relatively expressive form of embeddings (to guide the search for candidate rules), we adopt matrix factorization based embeddings. In particular, we employ the state-of-the-art RESCAL system [16], [19] to construct such embeddings, as other similar systems are either unavailable or difficult to use. It is worth noting that other matrix factorization based systems can be applied in our approach too.

RESCAL embeds each entity e to a vector $\mathbf{e}$ and each predicate P to a matrix $\mathbf{P}$. For each given fact $P(e_1, e_2)$, the following scoring function is used $f(e_1, P, e_2) = \mathbf{e}_1^T \cdot \mathbf{P} \cdot \mathbf{e}_2$ to measure the plausibility of the fact. It is applied to the task of link prediction, which is to identify for each predicate P and each entity e, an entity e' such that fact $P(e, e')$ is plausible; or alternatively, to identify for each predicate P and each entity e', an entity e such that $P(e, e')$ is plausible. While existing representation learners, such as RESCAL and TransE, use entity embeddings and predicate embeddings to predict the plausibility of new facts, we use these embeddings to measure the plausibility of new rules.

3 Rule Learning via Learning Represen-taions

In this section, we present our approach, Rule Learning via Learning Representations (RLvLR). We consider the problem of discriminative rule learning, that is, for specified target predicates P, to learn rules whose heads are about P. Discriminative rule learning is useful for link prediction, as for a link prediction task, the link P is often pre-specified.

In contrast to traditional ILP approaches, instead of using a refinement operator to search the rule space, we use embeddings to explore plausible paths. We achieve this by a combination of three novel techniques. First, existing methods for embedding construction cannot scale over massive KGs; for instance, RESCAL is unable to handle YAGO2 [19]. To overcome this challenge, we introduce a novel sampling algorithm to reduce the sizes of embeddings by restricting to only entities (and predicates) that are relevant to the target predicates. Second, we need a more efficient and accurate method to rank the plausibility of potential paths. To this purpose, we define a notion of embeddings for the arguments of predicates, and then introduce scoring functions for paths, based on embeddings of predicates and arguments (instead of only predicates). Third, we develop efficient matrix-based algorithms for the evaluation (and ranking) of candidate rules, i.e., efficient computation of standard confidence and head coverage.

Our method for rule learning is summarised in the following algorithm, while some major components in the algorithm will be explained later.

Algorithm 1 takes as input a KG K, a target predicate P, as well as an integer l as the maximum rule length, i.e., the number of atoms in the rules to be learned, and outputs a set of CP rules who heads are about P. To reduce the sizes of embeddings, a sampling method Sampling is first applied to obtain an (often much) smaller KG K' that contains only those entities and facts that are relevant to P. After sampling, the method Embeddings computes the embeddings $\mathcal{P}, \mathcal{A}$ for respectively the predicates and arguments in K'. Then, the method RuleSearch search for plausible paths that correspond to candidate CP rules to be stored in R'. Finally, the candidates rules R' are evaluated through the method Evaluate and only quality ones are returned.

Algorithm 1 KG Rule Learning (RLvLR)

```
input: a KG $K$, a target predicate $P$, a max rule length $l$
output: a set of CP rules $R$
1: $K' := \mathsf{Sampling}(K, P)$
2: $(\mathcal{P}, \mathcal{A}) := \mathsf{Embeddings}(K')$
3: $R' := \emptyset$
4: for $2 \leq k \leq l$ do
5:     Add $\mathsf{PathFinding}(K', P, \mathcal{P}, \mathcal{A}, k)$ to $R'$
6: end for
7: $R := \mathsf{Evaluation}(R', K)$
8: return $R$
```

3.1 Sampling

Intuitively, an instance of the CP rule (1) corresponds to a path $P_1(e, e_1), P_2(e_1, e_2), \ldots, P_n(e_{n-1}, e')$ in the KG K such that $P(e, e')$ is also in K. Clearly, entities e, e' are directly related to P in K, and entities $e_1, e_2, \ldots, e_{n-1}$ as well as predicates $P_1, P_2, \ldots, P_n$ are indirectly related to P, since they support (the plausibility of) the rule about P. On the other hand, if an entity is not connected (directly or indirectly, with no more than l links) to another entity that is related to P, then the entity has no use to the discovery of rules about P and can be disregarded in the path finding. Similarly, a predicate is no use if it does not connect entities that are (directly or indirectly) related to P.

Based on such an intuition, we present our sampling method as follows. For $K = (E, F)$ with E being the set of all entities and F being the set of all facts, to learn rules of maximum length l ($l \geq 2$), we sample entities in an incremental manner $E_0, \ldots, E_{l-2}$ as follows:

- E_0 consists of the entities that are connected to other entities by P, i.e., $E_0 = \{e \mid \exists e' \in E \text{ s.t. } P(e, e') \in F \text{ or } P(e', e) \in F\}$;
- E_i ($1 \leq i \leq l-2$) consists of the entities that are connected to some entities in E_{i-1} by any predicate P', i.e., $E_i = \{e \mid \exists e' \in E_{i-1} \text{ s.t. } P'(e, e') \in F \text{ or } P'(e', e) \in F \text{ for some } P'\}$;

The subset $E' = \bigcup_{i=0}^{l-2} E_i$ covers all relevant entities needed for learning rules about P with maximum length l.

Based on the sampled entities, we generate a sampled KG $K' = (E', F')$ with facts restricted to those entities in E', i.e., $F' = \{P'(e, e') \mid e, e' \in E', P'(e, e') \in F \text{ for some } P'\}$.

By the definition, E_0 always exists but can be empty. E_0 is empty only when the target predicate does not occur in any fact, that is, the predicate does not occur in the KG. In such a case, the output set of rules R is also empty. In our experiments (and in practice), the target predicates are from the KG and thus E_0 is not empty.

Given $K = (E, F)$, computing E' requires $2l \cdot |F| \cdot |E|$ or $2l \cdot |K|^2$ comparisons in the worst case. That is, the complexity of the sampling algorithm is square in terms of $|K|$ when l is fixed.

3.2 Embeddings and Path Finding

As explained above, the task of searching for CP rules can be reduced to that of finding plausible paths represented as sequences of predicates. This requires scoring functions to measure the plausibility of possible paths. Existing (standard) approaches to embedding, such as RESCAL, provides embeddings to only entities and their relations but not for some complex structures such as rules. In order to capture co-occurrence similarity of variables in a rule, we introduce the notion of argument embedding first and then define the co-occurrence scoring function.

To define a suitable scoring function for a CP rule of the form (1), note that each instance of the rule corresponds to two paths $p = P_1, P_2, \ldots, P_n$ and $p' = P$, where p' contains a single link. Moreover, the two paths connect the same pair of entities (in place of x and y respectively). Thus, a path is plausible if the pairs of entities connected by it are similar to those connected directly by the target predicate P. Such a similarity between p and P is referred to as *synonymy*, where two paths associate similar pairs of entities. For instance, the two paths (both with single predicates) BornIn and LiveIn connect similar pairs (e, e') with e being a person and e' a place.

Based on such an intuition, a scoring function is defined using predicate embeddings in [20]. For a CP rule of form (1), let matrices $\mathbf{P}, \mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n$ be the embeddings of predicates $P, P_1, P_2, \ldots, P_n$ respectively. The embedding of path $p = P_1, P_2, \ldots, P_n$ is $\mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \ldots \cdot \mathbf{P}_n$, which captures the pairs of entities connected by the path and should be similar to those by P, denoted as $\mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \ldots \cdot \mathbf{P}_n \approx \mathbf{P}$. Hence, the synonymy scoring function is

$$f_{syn}(r) = sim(\mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \ldots \cdot \mathbf{P}_n, \mathbf{P}) \tag{2}$$

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

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

The synonymy scoring function provides a overall measure for the plausibility of the path. Yet for long paths, the computation involves nesting of matrix manipulation. Thus, we propose a "local" scoring function based on *co-occurrence*. Besides synonymy, co-occurrence is also widely studied in natural language processing [25]. In our context, it refers to the co-occurrence of arguments in the positions of $x, y, z_1, \ldots, z_{n-1}$ in rule (1). For example, LiveIn connects pairs of entities (e, e') with e a person and e' a city, and LocatedIn connects pairs (e', e'') where e' is a city and e'' is a geographical region. LiveIn and LocatedIn are adjacent in a path only if they share the same cities e' in the KG.

To define a co-occurrence scoring function, we first introduce the notion of argument embeddings. Recall that each predicate has a subject argument and an object argument. For an argument, its embedding is defined as a vector obtained by averaging the embeddings of all the entities appearing in the position of this argument. Formally, for a KG $K = (E, F)$ and a predicate P, the embeddings of the subject and object argument of a predicate P are defined as:

$$\mathbf{p}^{(1)} = \frac{1}{n} \sum_{e \in S_P} s_e.\mathbf{e} \quad \text{and} \quad \mathbf{p}^{(2)} = \frac{1}{n} \sum_{e \in O_P} o_e.\mathbf{e}$$

where n is the number of facts in the KG, S_P and O_P are the sets of entities occurring as respectively subjects and objects of P (more precisely, $S_P = \{e \mid \exists e' \text{ s.t. } P(e, e') \in F\}$ and $O_P = \{e' \mid \exists e \text{ s.t. } P(e, e') \in F\}$), and s_e and o_e are the numbers of occasions for entity e to occur as respectively a subject and a object in $\mathcal{K}$ (more precisely, $s_e = \#\{P(e, e') \in F\}$ and $o_e = \#\{P(e', e) \in F\}$).

In a CP rule (1), the co-occurrences of x as the subject arguments of both P_1 and P, y as the object arguments of both P_n and P, and z_i ($1 \leq i \leq n-1$) as the object argument of P_i and subject argument of P_{i+1}, can be represented by the argument embeddings as follows: $\mathbf{p}_1^{(1)} \approx \mathbf{p}^{(1)}$, $\mathbf{p}_n^{(2)} \approx \mathbf{p}^{(2)}$, and $\mathbf{p}_i^{(2)} \approx \mathbf{p}_{i+1}^{(1)}$ ($1 \leq i \leq n-1$). The local scoring functions are defined accordingly

$$f_{loc}^{(1)}(P_1, P) = sim(\mathbf{p}_1^{(1)}, \mathbf{p}^{(1)}),$$
$$f_{loc}^{(2)}(P_n, P) = sim(\mathbf{p}_n^{(2)}, \mathbf{p}^{(2)}),$$
$$f_{loc}^{(2,1)}(P_i, P_{i+1}) = sim(\mathbf{p}_i^{(2)}, \mathbf{p}_{i+1}^{(1)}).$$

An overall co-occurrence scoring function can be obtained by aggregating the above local ones as follows.

$$\begin{aligned} f_{coo}(r) = & f_{loc}^{(1)}(P_1, P) + f_{loc}^{(2)}(P_n, P) + \\ & f_{loc}^{(2,1)}(P_1, P_2) + \cdots + f_{loc}^{(2,1)}(P_{n-1}, P_n). \end{aligned} \tag{3}$$

While the scoring function f_{syn} concerns the predicates in the paths, f_{coo} considers the shared variables that are involved in the paths. Consequently, we use both of these two scoring functions, which complement each other.

3.3 Rule Evaluation

For efficiency, we first evaluate the candidate rules based on the sampled KG K' and select rules with $supp(r) \geq 1$. These rules may still contain a large number of redundant and low quality rules and thus it is necessary to do a further selection based on the two measures SC and HC over the original KG K.

Let $K = (E, F)$ with $E = \{e_1, \ldots, e_n\}$ 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 $\mathcal{A}$ of adjacency matrices like RESCAL, where each $n \times n$ matrix $\mathcal{A}(P_k)$ corresponds to a predicate P_k in the KG ($1 \leq k \leq m$). Specifically, the $[i, j]$ entry $\mathcal{A}(P_k)[i, j]$ of the adjacency matrix $\mathcal{A}(P_k)$ is 1 if the fact $P_k(e_i, e_j)$ is in the KG; and 0 otherwise. Thus, $\mathcal{A}(P_k)$ is a matrix of binary values (i.e., 0 or 1).

We use adjacency matrices to compute the SC and HC of CP rules, and illustrate the idea through an example. Consider the rule r: $P_1(x, z) \wedge P_2(z, y) \rightarrow P(x, y)$. To compute SC and HC of r in K, we need to calculate the numbers of entity pairs satisfying the head of r (i.e., $\#(e, e') : head(r, e, e')$), satisfying the body of r (i.e., $\#(e, e') : body(r, e, e')$), and satisfying both the head and body of r (i.e., $supp(r)$), respectively.

The pairs (e_i, e_j) satisfying the head can be directly read from the matrix $\mathcal{A}(P)$, that is, where $\mathcal{A}(P)[i, j] = 1$. For the pairs (e_i, e_j) satisfying the body, they are connected by the path $p = P_1, P_2$, and can be obtained from the product $\mathcal{A}(P_1) \cdot \mathcal{A}(P_2)$ of the two matrices $\mathcal{A}(P_1)$ and $\mathcal{A}(P_2)$. In particular, the $[i, j]$ entry of $\mathcal{A}(P_1) \cdot \mathcal{A}(P_2)$ represents the number of rule paths that start from e_i, traverse via P_1 to another entity and finally go to e_j via P_2. Now, $\mathcal{A}(P_1) \cdot \mathcal{A}(P_2)$ may have non-binary values. Let $\mathcal{A}(P_1, P_2) = \mathsf{binary}(\mathcal{A}(P_1) \cdot \mathcal{A}(P_2))$ where $\mathsf{binary}(M)$ is the matrix obtained from matrix M by setting all non-zero entries to 1. Thus, the pairs (e_i, e_j) satisfying the body can be seen from $\mathcal{A}(P_1, P_2)$, that is, where $\mathcal{A}(P_1, P_2)[i, j] = 1$.

Continue with our example, let $E = \{e_1, e_2, e_3\}$ and

$$F = \{P_1(e_1, e_2), P_1(e_2, e_1), P_1(e_1, e_3), P_2(e_2, e_3), P_2(e_2, e_1), P_2(e_3, e_3), P(e_1, e_3)\}$$

The adjacency matrices for the predicates P_1, P_2 and P are:

$$\mathcal{A}(P_1) : \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathcal{A}(P_2) : \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \mathcal{A}(P) : \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Then,

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

The matrix $\mathcal{A}(P_1, P_2)$ shows the pairs that are connected by the path $p = P_1, P_2$, i.e., $\{(e_1, e_1), (e_1, e_3)\}$, and they are exactly the pairs satisfying the body of r. And from $\mathcal{A}(P)$, there is one pair (e_1, e_3) satisfying the head of r. From these we can easily obtain $HC(r) = 1$ and $SC(r) = 0.5$.

Note that although our discussion here is based on rules of length three, it is straightforward to generalize this calculation to rules of any length.

3.4 Rule Applicaiton

In summary, the combination of the following techniques enables RLvLR to handle large-scale KGs and learn a sufficient amount of rules within reasonable time, which in turn provides superior prediction power compared to other counterparts as we will show in the experiments.

- The target-oriented sampling: It reduces the sizes of embeddings by focusing only on relevant entities and predicates. In our experimental evaluation, our sampling method typically reduce the number of entities to 800. Thus, the sampling method is proven to be quite effective and is critical for our embedding-based approach.
- The novel path finding method: It reduces the search for candidate rules to embedding-based path finding. In comparison with [20], which only use predicate embeddings as a heuristic, we use argument embeddings to define a local scoring function which enhances both efficiency and accuracy in rule learning.
- The tensor-based rule evaluation: The evaluation of candidate rules, through the computation of standard confidence and head coverage, is often expensive, and much research effort has be dedicated to optimise such computation. By using tensor-based calculation, the rule evaluation against massive KGs can be done efficiently.

To apply the learned rules for link prediction, we also implemented an inference module that predicts new facts based on the given facts and the learned rules. For a CP rule of the form (1), if an instance of the rule body $P_1(e, e_1), P_2(e_1, e_2), \ldots, P_n(e_{n-1}, e')$ is found to exist in the KG, then the instance of the head $P(e, e')$ can be inferred with a confidence degree.

Our inference module can be used for link prediction. Given predicate P and an entity e, assuming the task is to predict all entities e' such that $P(e, e')$ is plausible, our approach first learns a set of rules about P and then apply the learned rules to infer missing facts of the form $P(e, e')$.

To obtain the confidence degree (CD) of a fact, we adapt the $score^*(\cdot)$ function from [10] 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 [10], we use SC as it is easier to compute. Formally, for a fact $f = P(e, e')$ and the set of rules R that can infer f from the given KG, the CD of f is defined as follows:

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

With this module, RLvLR goes beyond a rule learner and is capable of handling the link prediction task. The favourable performance in link prediction also demonstrates the quality of the rules learned by RLvLR.

4 Rule Learning from KG Streams

A KG containing temporal data or constantly evolving data can be viewed as a stream of snapshots of the KG over a sequence of time points. Figure 1 illustrates such a KG stream, which involve three entities (three countries). Country e_0 provided economic aid to country e_1 at time point $\tau - 4$, and then it imposed sanction on country at time point $\tau - 3$.

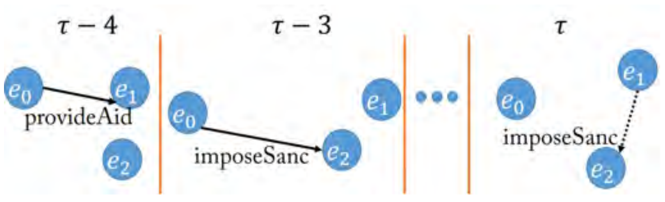

Fig. 1: Example of KG Stream.

Besides classical link prediction questions such as "Which countries besides e_0 provided aid to e_1?" and link prediction questions involving temporal knowledge such as "Which counties may impose sanction on e_2 at the time point τ?", there are also schema level questions that of much interest, such as "After a country x provided aid to country z and then imposed sanction on another country y on the next day, in how many days will z most likely impose sanction on y?"

This brings in some research challenges. First, this requires temporal knowledge to be expressed in the schema information. Several formalisms have been investigated for representing temporal knowledge over data streams [28], [29], [30]. In particular, [30] shows how temporal rules can be applied for efficient data stream processing.

A second challenge is how to learn such temporal rules over KG streams. Several works have proposed on embedding-based methods for link prediction over dynamic KGs [26], [31], where temporal information is embedded in statistical models. Learning logical models from data streams have also been studied in [32], which extracts event definitions in a logical form and refines the definitions using operators as in ILP (and thus cannot scale over large KGs). Yet none of the aforementioned approaches is capable to learn temporal rules as above. And existing rule learners for KGs cannot be directly used to learn temporal rules. A final challenge is how to develop a framework that combines stream learning and reasoning of temporal rules.

Based on RLvLR, we present a method for learning temporal rules from KG streams. Such temporal rules are useful for link prediction and event prediction in the setting of KG streams. We first adapt some basic definitions of rule systems for (static) KGs to dynamic KGs and then describe technical details of the proposed method.

4.1 KG Streams and Temporal Rules

A *KG stream* consists of a (possibly infinite) set of quadruples of the form (e, P, e', k), each of which expresses that the relation P associates entity e to entity e' at time point k. For convenience, such a fact with time stamp is also called an *event*. Following the convention in knowledge representation, we denote such an event as $P(e, e', k)$, where P is a ternary predicate, and k is a time point constant.

Consider an KG stream S and two integers i, j with $0 \leq i \leq j$, the $[i, j]$-*segment* $S[i, j]$ of S is the subset of S consisting of all the events with time points between and including i and j. That is, $S[i, j] = \{P(e, e', k) \mid P(e, e', k) \in S, i \leq k \leq j\}$. When $i = j$, it can be simplified as $S[i]$. Note that S can be seen as a sequence of KGs $S[0], S[1], \ldots$, and thus we also call S a *KG stream*. Sometimes we want to consider the facts via omitting the time points in the events, and S^* denotes the static KG obtained from S by replacing each event $P(e, e', k)$ with the fact $P(e, e')$.

In what follows, we consider an extension of CP rules, which we call temporal CP rules (or simply temporal rules) of the following form:

$$P_1(x, z_1, t) \wedge P_2(z_1, z_2, t) \wedge ... \wedge P_n(z_{n-1}, y, t) \rightarrow P(x, y, t + k). \quad (4)$$

Here t is a time point variable and k is an integer. The rule reads that if the rule body holds at time point t, then the rule head holds at time point $t + k$. Obviously, the class of temporal CP rules could be more general, for instance, different time points could be allowed for different atoms in the rule. Our definition of temporal CP rules is a balance of expressive power and efficiency of rule learning algorithms. Especially, the applicability of a temporal CP rule can be verified at a single time point t (as in the rule body).

4.2 Quality Measures for Temporal Rules

Consider a KG stream S, to learn the structure of the rules, our method uses facts in an initial segment of the stream $S^*[0, n]$ $(n \geq 0)$, which consists of the facts from time points 0 up to n, as *structure training data*. We use RLvLR to learn a set of CP rules R. Such a CP rule is referred to as a *structure rule*. For each structure rule r of the form (1) and each integer $k \geq 0$, we can obtain a temporal rule $r^{(k)}$ of the form (4).

For a temporal rule $r^{(k)}$, the support degree of $r^{(k)}$ at time point τ is naturally defined as the number of entity pairs for which the head of $r^{(k)}$ has instantiations at time point τ and the body of $r^{(k)}$ has instantiation at time point $\tau - k$. Formally, a pair of entities (e, e') satisfies the body of r at time point τ with $\tau \geq 0$, denoted $body(r, e, e', \tau)$, if there exist entities $e_1, ..., e_{n-1}$ in the KG stream S such that $P_1(e, e_1, \tau), P_2(e_1, e_2, \tau), ..., P_n(e_{n-1}, e', \tau)$ are events in $S[\tau]$. And (e, e') satisfies the head of r at time point τ, denoted $head(r, e, e', \tau)$, if $P(e, e', \tau)$ is an event in $S[\tau]$. Then the support degree of $r^{(k)}$ at time point τ is defined as

$$supp(r^{(k)}, \tau) = \begin{cases} 0, & \text{if } \tau < k \\ \#(e, e') : head(r, e, e', \tau) & \\ \wedge\, body(r, e, e', \tau - k), & \text{otherwise} \end{cases}$$

Note that, since in the static case there is only one time point 0, the standard notion of support is a special case of the above definition where $\tau = k = 0$.

The standard confidence (SC) of a temporal rule $r^{(k)}$ at time point τ is the ratio between support degree at τ and the number of entity pairs satisfying the body at time point $\tau - k$:

$$SC(r^{(k)}, \tau) = \frac{supp(r^{(k)}, \tau)}{\#(e, e') : body(r, e, e', \tau - k)}$$

Similarly, we define head coverage (HC) of a temporal rule $r^{(k)}$ at time point τ to be the ratio between support degree and the number of entity pairs satisfying the head at time point τ:

$$HC(r^{(k)}, \tau) = \frac{supp(r^{(k)}, \tau)}{\#(e, e') : head(r, e, e', \tau)}$$

In the following example, we illustrate the process of temporal rule generation and the quality measures defined above.

Example 1. For a structure rule $r : P_1(x, z) \wedge P_2(z, y) \rightarrow P(x, y)$ and $0 \leq k \leq 2$, the following potential temporal rules can be generated from r:

$$\begin{aligned} r^{(0)} &: P_1(x, z, t) \wedge P_2(z, y, t) \rightarrow P(x, y, t), \\ r^{(1)} &: P_1(x, z, t) \wedge P_2(z, y, t) \rightarrow P(x, y, t + 1), \\ r^{(2)} &: P_1(x, z, t) \wedge P_2(z, y, t) \rightarrow P(x, y, t + 2). \end{aligned}$$

Assume the first three time points in a KG stream S are:

$$\begin{aligned} S[0] = \{&P_1(e_3, e_2, 0), P_1(e_2, e_1, 0), P_1(e_1, e_3, 0), \\ &P_2(e_3, e_1, 0)\} \\ S[1] = \{&P_1(e_2, e_2, 1), P_1(e_2, e_1, 1), P_2(e_1, e_3, 1), \\ &P_2(e_2, e_1, 1), P_2(e_2, e_3, 1), P_2(e_3, e_3, 1), \\ &P(e_1, e_1, 1), P(e_1, e_3, 1)\} \\ S[2] = \{&P_1(e_1, e_3, 2), P_2(e_2, e_2, 2), P_2(e_3, e_1, 2), \\ &P_2(e_2, e_1, 2), P_2(e_3, e_3, 2), P(e_1, e_3, 2), \\ &P(e_1, e_1, 2)\} \end{aligned}$$

Then, at time point $\tau = 2$ for example, the quality measures of these rules are:

$$\begin{array}{lll} supp(r^{(0)}, 2) = 2, & supp(r^{(1)}, 2) = 0, & supp(r^{(2)}, 2) = 1 \\ SC(r^{(0)}, 2) = 1, & SC(r^{(1)}, 2) = 0, & SC(r^{(2)}, 2) = 0.5 \\ HC(r^{(0)}, 2) = 1, & HC(r^{(1)}, 2) = 0, & HC(r^{(2)}, 2) = 0.5 \end{array}$$

The SC and HC of temporal rules generalises those standard notions with a temporal flavour, yet they haven't taken into consideration of the streaming nature of KG stream. In particular, the SC and HC of a temporal rule at one time point should aggregate the corresponding (SC and HC) values at previous time points. Hence, the dynamic standard confidence (DSC) of a temporal rule γ at time point τ is defined as follows:

$$DSC(\gamma, \tau) = \begin{cases} SC(\gamma, \tau), & \text{if } \tau = 0 \\ (1-\alpha) \times DSC(\gamma, \tau - 1) & \\ +\alpha \times SC(\gamma, \tau), & \text{otherwise} \end{cases}$$

where $0 < \alpha < 1$ is the learning rate to adjust the weights of previously aggregated DSC and that of the SC of current time point. The dynamic head coverage (DHC) is defined in a similar way.

We use the DSC and DHC scores to select quality temporal rules at each time point. Note that while the set of structure rules are always the same, the set of selected temporal rules at each point are often different, due to varying DSC and DHC scores of the temporal rules over time.

4.3 Temporal Rule Learning

In this section, we present our algorithm that combines the learning and reasoning of temporal rules in a dynamic manner over KG streams. Our algorithm takes as input a KG stream S (i.e., a stream of quadruples), for which all the facts in an initial segment up to time point n, i.e., $S^*[0, n]$, is stored, and two integers $l, m \geq 0$ as minimum and maximum *prediction distances*. And it produces as outputs a stream of temporal rule sets and a stream of derived events. In particular, we use the method from the above section to obtain a set of candidate temporal rules: $r^{(l)}, r^{(l+1)}, \dots, r^{(m)}$ for each structure rule r. Then, at each time point, we select quality temporal rules using their DSC and DHC scores and apply the selected rules to derive events about current and future time points.

For both rule quality measure computation and rule application over KG stream S, the notion of *shifting windows* is required. We assume at each time point τ, only a segment of S of size w, $S[\tau - w + 1, \tau]$, is used for computation. Here, $w \geq 1$ is an integer called the *window size*, which is specified by users and may vary over time points. The shifting windows are needed not only for memory space concerns but also due to efficiency requirement of stream processing. Note that the structure rules can be learned offline, whereas the (temporal) rule filtering and rule application need to be performed online.

At each time point τ, the method needs to assess the quality of candidate temporal rules $r^{(k)}$. If $\tau < k$, then $supp(r^{(k)}, \tau) = 0$ and thus $DSC(r^{(k)}, \tau) = DHC(r^{(k)}, \tau) = 0$; otherwise, we assume $DSC(r^{(k)}, \tau - 1)$ has been obtained from the previous time point. To compute DSC and DHC of the current time point, we would need to access the events at time points τ and $\tau - k$. Yet we can only access the events in the shifting window, i.e., $S[\tau - w + 1, \tau]$. In this case, if $k \leq w - 1$ then we have all the required events for computing the current DSC and DHC as defined above; otherwise, we set $DSC(r^{(k)}, \tau) = \beta \times DSC(r^{(k)}, \tau - 1)$ and $HC(r^{(k)}, \tau) = \beta \times HC(r^{(k)}, \tau - 1)$, where $0 < \beta < 1$ is used to adjust the weights of previously aggregated DSC due to the rule's quality not assessable at the current time point.

At each time point τ, the method also applies selected temporal rules $r^{(k)}$ to the current events in $S[\tau]$ to derive new events at time point $\tau + k$. The derived events are coupled with confidence degrees (CDs). For an event $\xi = P(e, e', \tau)$ and the set of temporal rules Γ that can derive ξ from the KG stream, the CD of ξ is defined as follows:

$$CD(\xi) = 1 - \prod_{\gamma \in \Gamma} (1 - DSC(\gamma, \tau)).$$

In Figure 2, we illustrate a simple case with one snapshot of the stream and one structure rule r, where the current point is τ, shifting-window size is $w = 4$, minimum prediction distance is 0 and maximum prediction distance is $m = 2$.

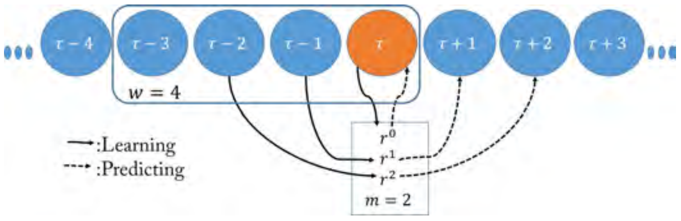

Fig. 2: Stream Learning and Reasoning.

Algorithm 2 shows the data flow and major components of our system for temporal rule learning and reasoning over KG streams.

In line 2, we obtain a set of structure rules R using static rule learner RLvLR over the structure training data $S^*[0, n]$. This is performed offline. Then, in line 3, a set Γ of candidate temporal rules of the form $r^{(k)}$ is obtained with $1 \leq k \leq m$. The online stream learning and reasoning starts from line 4. In line 5, current events at time point τ are read in, and in line 7, past events outside of the shifting window are forgotten. Thus, W consists of all the events in the current shifting window. In line 9, a selection on temporal rules is performed based on their DSC and DHC scores at time point τ. As discussed before, this involves aggregating previous DSC and DHC scores, and is restricted by W, the events available in the shifting windows. After filtering, the set of selected rules Γ_τ can be streamed out.

The selected temporal rules can then be used for reasoning. In line 12, each rule $r^{(k)}$ is applied to the current events $S[\tau]$ to derive events in future time point $\tau + k$. Note that events at a time point τ is derived incrementally from a sequence of past time points. Once all the learned temporal rules at time point τ have been applied, the derived events at τ, Ξ_τ, will not change and can be streamed out.

Algorithm 2 KG Stream Learning and Reasoning

```
input: a KG stream $S$ and 3 integers $l, m, n \geq 0$ with $l \leq m$
output: a set of temporal rules $\Gamma_\tau$ and a set of events $\Xi_\tau$ at
    each time point $\tau \geq 0$
 1: $W := \emptyset, \tau := 0, \Gamma_\tau := \emptyset, \Xi_\tau := \emptyset$
 2: $R := \mathsf{RLvLR}(S^*[0, n])$
 3: $\Gamma := \mathsf{TemporalRules}(R, l, m)$
 4: loop
 5:     $W := W \cup S[\tau]$
 6:     if $\tau \geq w$ then
 7:         $W := W \setminus S[0, \tau - w]$
 8:     end if
 9:     $\Gamma_\tau := \mathsf{Select}(\Gamma, W)$
10:     stream out $\Gamma_\tau$
11:     for each $l \leq k \leq m$ and each $r^{(k)} \in \Gamma_\tau$ do
12:         $\Xi_{\tau+k} := \Xi_{\tau+k} \cup \mathsf{Apply}(r^{(k)}, S[\tau])$
13:     end for
14:     stream out $\Xi_\tau$
15:     $\tau := \tau + 1$
16: end loop
```

Overall, our approach can achieve pleasant scalability and predict power due to the following aspects. First, it benefits from our powerful static-rule learner RLvLR in learning sufficiently many quality structure rules in a reasonable time. Note that the learning of structure rules in performed only once, before the stream learning and reasoning, whereas the stream learning focuses on parameter learning (and dynamically adapting the rule-based model). Hence, the quality and quantity of structure rules have a critical impact on those of the learned temporal rules. Secondly, the proposed language bias, i.e., temporal rules of the form (4), helps to achieve a good balance between expressive power and learning efficiency. As discussed before, if arbitrary time points are allowed in the rule body, the rule space would quickly blow up. Meanwhile, the current form of temporal rules still provide satisfactory predict power in event prediction, as we will show in Section 5.2. Finally, the evaluation of candidate temporal rules can be done efficiently utilising our tensor-based calculation (ref. Section 3.3), which reduces the computation time to an acceptable level over streams.

5 Experiments

Based on the methods presented in previous sections, we have implemented our system RLvLR (Rule Learner via Learning Representations) and its extension RLvLR-Stream (RLvLR for KG Streams). Experiments were conducted to evaluate the systems. The compiled codes, benchmark datasets and experimental results are publicly available at https://www.ict.griffith.edu.au/aist/RLvLR/.

5.1 The Evaluation of RLvLR

We conducted two sets of experiments on well known (static) large-scale KGs. 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 in link prediction. The benchmark datasets adopted in our experiments include various versions of Freebase, YAGO, DBpedia and Wikidata that are widely used in benchmarking rule learning and link prediction systems. RLvLR was compared with state-of-the-art rule learners such as AMIE+ [10] and ScaLeKB [13], as well as state-of-the-art system for link prediction, Neural LP [21]. Our experiments were designed to validate the following statements:

1) RLvLR is much faster than major rule learning 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.

TABLE 1: Benchmark KG specifications

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

The five benchmark datasets are specified in Table 1, where the last three have been often used in rule learning [10], [13]. FB15K-237 [33] (aka. FB15KSelected) and FB75K (from NIPS'13 dataset) are obtained from Freebase and widely adopted for link prediction benchmarking [21].

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.

5.1.1 Rule Learning

This set of experiments concerns learning quality rules. The rule quality was measured by standard confidence (SC) and head coverage (HC) [10]. Note that while we used sampling for rule learning, HC and SC degrees of learned 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, respectively. A 10 hour limit was set for each target predicate. Table 2 shows the average numbers of rules (#R, with SC$\geq$ 0.1 and HC$\geq$ 0.01 as in [10]), the percentage of high quality rules (%QR, ratios of rules with SC$\geq$ 0.7 over all learned rules), and the running times (in hours, averaged over the targets) of RLvLR and AMIE+.

TABLE 2: Rule learning by RLvLR and AMIE+

KG	RLvLR			AMIE+		
	#R	%QR	Time	#R	%QR	Time
YAGO2s	**6.3**	**29%**	**0.96**	5.65	9%	10.00
DBpedia 3.8	**42.7**	**22%**	**3.88**	9.05	5%	4.59
Wikidata	**56.8**	**45%**	**2.41**	0.95	32%	10.00

Compared to AMIE+, RLvLR showed significantly better performance in terms of both the runtime and the numbers of learned rules. Note that RLvLR deployed the same redundancy elimination as AMIE+, and the numbers were obtained after the redundancy elimination. The superiority of RLvLR is more obvious in learning high quality rules, that is, among all the rules learned, RLvLR has a consistently higher percentage of high quality rules.

To estimate the predictive power of the corpus of learned 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 learned rules on the remaining facts. Table 3 shows the numbers of predicted facts (# Facts) and those predictions with CD≥ 0.9 (#QFacts). In this part, we consider five target predicates in all three benchmarks. Note that while AMIE+ learned 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.

TABLE 3: New facts prediction by 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

The numbers of quality rules learned by RLvRL on YAGOs, DBpedia 3.8 and Wikidata are 1.1, 4.7 and 59.7 times of those learned by AMIE+ (from Table 2), whereas 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 learned by RLvLR in link prediction.

Since we were unable to run ScaLeKB system or obtain the Freebase benchmark used in [13], we could only compare the rules learned by RLvLR on YAGO2s with those reported in [13]. 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 three body atoms:

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

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 To demonstrate the usefulness of our co-occurrence scoring function, we compared the performance of RLvLR under two configurations, with synonymy scoring function alone as in [20], denoted RLvLR*, and with both the synonymy and co-occurrence scoring functions. We set a 5 hours time limit, SC≥ 0.01 and HC≥ 0.001. As FB15k-401 KG was used in [20], our experiment was conducted on the similar benchmark, FB15K-237. The numbers of learned rules under the two settings on some target predicates are listed in Table 4, which show that with co-occurrence scoring function, RLvLR was capable to learn (up to 2.4 times) more rules than RLvLR*.

TABLE 4: Rule learning with different scoring funcions

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

5.1.2 Link Prediction

The second set of experiments aim to evaluate the predictive power of learned 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 link prediction systems. So, the major advantage of RLvLR is in its capability of handling massive KGs with an accuracy that is comparable to other major systems. Note that our goal is not to compete with link prediction systems on accuracy over relatively small KGs.

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 link prediction systems on accuracy.

Following the experiments of Neural LP [21], 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 [21] on two benchmark datasets FB75K and Wikidata, with 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 handled by Neural LP in [21] is FB15K-237, which is still much smaller than FB75K regarding the number of entities. 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, where RLvLR was configured to learn rules with SC≥ 0.005 and HC≥ 0.001, and the times were in hours with a 5-hour time limit for each target predicate.

TABLE 5: Link prediction by RLvLR and Neural LP

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	-	-	-

Experiment 2 In the literature, three benchmarks WN18, FB15K and FB15K-237 are usually used for evaluating link

prediction [19], [21], [34], among which FB15K-237 is shown to be most challenging in [21]. This is because the test entities in FB15K-237 are loosely linked and to process them, a link prediction system 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 adopted FB15K-237 for our comparison.

Six major link prediction systems were evaluated in [21] but only three could successfully handle on FB15K-237, which are Neural LP, DISTMULT [20] and Node+LinkFeat [33]. We included the results from [21] for these three systems, and compare them with RLvLR in Table 6.

TABLE 6: Link prediction on FB15K-237

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

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

5.2 The Evaluation of RLvLR-Stream

We also evaluated RLvLR-Stream on dynamic KGs. Our experiments are designed to demonstrate that temporal rules are an effective model to capture temporal knowledge and thus can provide more accurate link prediction than static rules that do not account for temporal knowledge or evolving data. Our goal is not to compete with temporal statistical models like Know-Evolve [26], as the advantage of temporal rules are, compared to statistical models, in their simplicity and explainability. Specifically, our experimental results aimed to validate the following statements:

1) RLvLR-Stream significantly outperforms the baseline RLvLR in terms of accuracy in link prediction.
2) Stream learning of and link prediction through temporal rules can be performed efficiently. When the sizes of structure training data increase, the prediction accuracy also increases without significant sacrifice on the efficiency.
3) The accuracy of link prediction generally increases when the size of shifting window increases or when the minimum prediction distances reduces.

We used the Integrated Crisis Early Warning System (ICEWS) dataset [26] which contains a rich set of historical events including cooperative or antagonistic actions between individuals, associations, organizations, sectors and nation states. The events were automatically recognized and extracted from the news. It contains events from the year of 2014 with a 24hrs interval between each two adjunct time points (that is, 365 time points in total).We are unaware of any other benchmarks of a similar type. ICEWS has 12498 entities, 260 predicates, 365 time points and 668080 events in total.

To evaluate the performance of RLvLR-Stream in comparison with RLvLR, we set the default training data (for learning structure rules) to be the first 50 days (i.e., $S^*[0, 50]$). We note that this size is much smaller than the initial data sets in other approaches for learning from streams [26]. The default size of the shifting window was 10 days, and the default minimum and maximum prediction distances were respectively 0 and 10 days.

The link prediction task is to identify an entity e' for each predicate P, each entity e and each time point k such that $P(e, e', k)$ is an event occurred in the KG stream; or alternatively, to identify for each target predicate P and each entity e, an entity e' such that $P(e', e, k)$ is an event in the stream.For each of these time points k, we separated the datasets into 70% training and 30% test.

Since the quality of learned temporal rules and their performance in link prediction are affected by the sizes of structure training data, the window sizes, and the minimum and maximum prediction distances. In our evaluation, we compared RLvLR-Stream and RLvLR under various configuration, by varying one of the factors and fixing the others.

The size of structure training data affects the quantity and quality of the structure rules, which in turn affects the learning of temporal rules. In particular, it impacts the efficiency of RLvLR-Stream. Table 7 shows the performance of RLvLR-Stream and RLvLR with different sizes n of structure training data $S^*[0, n]$. We report the numbers of structure rules (#SR), rule learning and reasoning times (Time, in seconds), MRR and Hits@10 (H@10) scores, averaged over all selected time points and target predicates.

TABLE 7: Performance w.r.t. sizes of structure learning data

Size	#SR	RLvLR-Stream			RLvLR		
		Time	MRR	H@10	Time	MRR	H@10
10	27	0.8	0.19	0.25	0.1	0.02	0.02
50	123	3.3	0.22	0.30	0.2	0.06	0.11
100	262	7.0	0.27	0.37	0.4	0.10	0.19
150	355	9.6	0.26	0.38	0.6	0.12	0.21
250	551	14.1	0.30	0.44	0.8	0.14	0.24

RLvLR-Stream showed superior performance in the accuracy of link predictions compared to RLvLR in all cases, which clearly demonstrates the benefit of temporal rules over static rules on stream reasoning. Allowing larger structure learning data provides more structure rules and leads to better prediction accuracy on both RLvLR-Stream and RLvLR. Yet, there is trade-off between time efficiency and prediction accuracy, yet the processing time of RLvLR-Stream remains acceptable (14.1 seconds) even when a significantly large portion of the KG stream (250 out of 365) were used as structure training data.

The window sizes and the maximum prediction distances both determine the amount of historical data that can be utilised for learning and prediction, and thus affect the performance of our system. For simplicity, in the following experiments, we set the window sizes to be fixed over time and equivalent to the maximum prediction distances. Figure 3 shows the performance of RLvLR-Stream over window sizes ranging from 1 to 15. Again, we used RLvLR as a baseline, whose performance is not impacted by the window sizes, and MRR and Hits@10 scores are averaged over all selected time points and target predicates.

Again, RLvLR-Stream showed superior performance to RLvLR in nearly all cases. Generally speaking, the performance of RLvLR-Stream improves as the window size

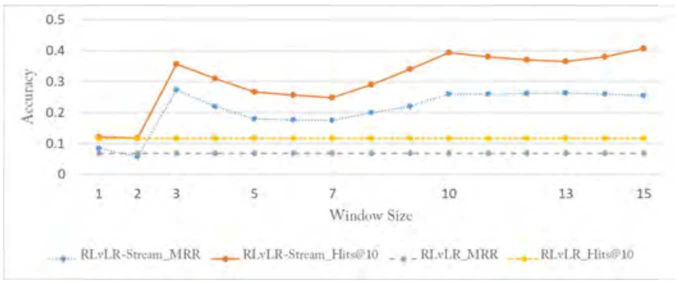

Fig. 3: Performance over varying window sizes.

increases. It also reveals some local optimal points, namely window sizes 3 and 10. This suggests that in real-life, events occurring further way in the stream tend to be more loosely associated with the current events, and in practice it could be effective to use small window sizes (like 3 or 10).

The minimum prediction distance also impacts the learning and prediction, and probably more on prediction. Clearly, as the minimum prediction distance increases, the system is challenged to make predictions only in distance. For instance, if the minimum prediction distance is set to 3 then the learned temporal rules are of the form $r^{(3)}, r^{(4)}, \ldots$. That means, events at time point τ (Ξ_τ) must be derived at time point $\tau - 3$. This effectively requires the prediction to happen at least 3 days ahead.

In the following experiment, we evaluated the impact of minimum prediction distance on the performance of RLvLR-Stream, using RLvLR as a baseline. Note that if the maximum prediction distance stays the same, the increase of minimum prediction distance will reduce the number of candidate temporal rules. To separate this factor from the challenged post by distant prediction, we assume the difference between the maximum and minimum prediction distance remains 10. Figure 4 shows the performance of RLvLR-Stream over minimum prediction distances ranging from 0 to 10. Again, the values are averaged.

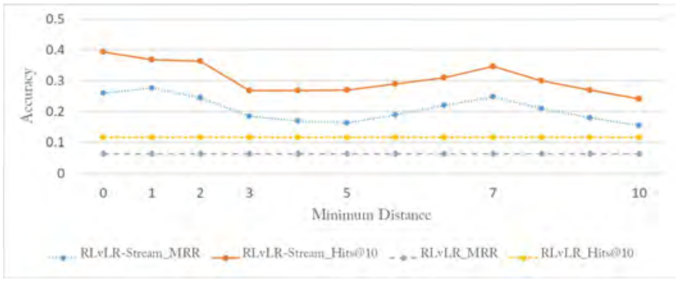

Fig. 4: Performance w.r.t. minimum prediction distances

RLvLR-Stream again outperformed RLvLR in all cases. The performance of RLvLR-Stream drops as the prediction distances increase. An interesting observation is that the accuracy decrease was not as dramatic as one would expect. In particular, predicting one day or two days ahead had comparable accuracy as "predicting" about today. Also, 7 turns out to be a local optimal point, which may suggest a weekly pattern in the event association.

A major benefit of temporal rules compared to statistical models is that their meanings are human understandable, and hence the learned temporal rules themselves contain valuable temporal knowledge about the domain. Although entities and predicates in the ICEWS dataset are obfuscated and thus it is difficult to assess the meaning of learned temporal rules, it is not hard to imagine the temporal knowledge behind some common patterns. For illustration purpose, we present the following two temporal rules whose predicate names are replaced with meaningful terms from the ICEWS dictionary.

$$\begin{aligned} 0.05 : \ & \mathsf{rejectMaterialCooperation}(y, x, t) \rightarrow \\ & \qquad \mathsf{threatenWithSanctions}(x, y, t). \\ 0.15 : \ & \mathsf{rejectMaterialCooperation}(y, x, t) \rightarrow \\ & \qquad \mathsf{threatenWithSanctions}(x, y, t+1). \end{aligned}$$

The number before each rule is the corresponding DSC. The two rules indicate that if country y rejects the material cooperation with country x, then it is unlikely that y threatens x with sanctions on the same day. But this is more likely to happen on the next day.

6 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 learning 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.

We have also proposed a method for learning temporal rules from KG streams by extending RLvLR. Our RLvLR-Stream can also complete dynamic KGs and predict events.

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. Our experiments also show that RLvLR-Stream significantly outperforms RLvLR when handling dynamic KGs.

As for future work, we plan to develop an efficient parallel algorithm for RLvLR. The RLvLR algorithm learns rules with different target predicates independently, which is useful for developing a parallel algorithm. Furthermore, we plan to develop a framework to handle more complex events such as events specified with duration but not specific beginning or ending times.

Acknowledgments

The authors are grateful to the anonymous reviewers for their valuable comments. This work was supported by the Australian Research Council (ARC) under DP130102302.

References

[1] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proc. Int'l World Wide Web Conf.*, pp. 697–706, 2007.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBPEDIA: A nucleus for a web of open data," in *Proc. Int'l Semantic Web Conf.*, pp. 722–735, 2007.

[3] D. Vrandečić and M. Krötzsch, "Wikidata," *Communications of the ACM*, pp. 78–85, 2014.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. Int'l Conf. Management of Data*, pp. 1247–1250, 2008.

[5] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," in *Proc. IEEE*, vol. 104, pp. 1–23, 2016.

[6] J. Fürnkranz, D. Gamberger, and N. Lavrač, *Foundations of Rule Learning*. Springer, 2012.

[7] S. Muggleton, "Learning from positive data," in *Proc. Int'l Conf. Inductive Logic Programming*, pp. 358–376, 1996.

[8] S. Muggleton, "Inverse entailment and progol," *New Generation Comput.*, vol. 13, no. 3&4, pp. 245–286, 1995.

[9] Q. Zeng, J. Patel, and D. Page, "QuickFOIL: Scalable inductive logic programming," *VLDB Journal*, vol. 8, pp. 197–208, 2014.

[10] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek, "Fast rule mining in ontological knowledge bases with AMIE+," *VLDB Journal*, vol. 26, pp. 707–730, 2015.

[11] Z. Wang and J.-Z. Li, "RDF2Rules: Learning rules from RDF knowledge bases by mining frequent predicate cycles," *Computer Research Repository*, 2015.

[12] M. Barati, Q. Bai, and Q. Liu, "SWARM: An approach for mining semantic association rules from semantic web data," in *Proc. Pacific Rim Int'l Conf. Artificial Intelligence*, pp. 30–43, 2016.

[13] Y. Chen, D. Z. Wang, and S. Goldberg, "ScaLeKB: scalable learning and inference over large knowledge bases," *VLDB Journal*, vol. 25, pp. 893–918, 2016.

[14] S. Ortona, V. V. Meduri, and P. Papotti, "Rudik: Rule discovery in knowledge bases," *PVLDB*, vol. 11, pp. 1946–1949, 2018.

[15] V. T. Ho, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, and G. Weikum, "Rule learning from knowledge graphs guided by embedding models," in *Proc. Int'l Semantic Web Conf.*, pp. 72–90, 2018.

[16] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proc. Int'l Conf. Machine Learning*, pp. 809–816, 2011.

[17] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems*, pp. 2787–2795, 2013.

[18] Y. Shen, P.-S. Huang, M.-W. Chang, and J. Gao, "Traversing knowledge graph in vector space without symbolic space guidance," *Computer Research Repository*, 2016.

[19] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proc. AAAI*, 2016, pp. 1955–1961.

[20] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proc. Int'l Conf. Learning Representation*, p. 12, 2015.

[21] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable learning of logical rules for knowledge base reasoning," in *Advances in Neural Information Processing Systems*, pp. 2316–2325, 2017.

[22] A. Neelakantan, B. Roth, and A. McCallum, "Compositional vector space models for knowledge base inference," in *Proc. AAAI Spring Symposium Series*, pp. 156–166, 2015.

[23] P. G. Omran, K. Wang, and Z. Wang, "Scalable rule learning via learning representation," in *Proc. IJCAI*, pp. 2149–2155, 2018.

[24] M. Gardner and T. Mitchell, "Efficient and expressive knowledge base completion using subgraph feature extraction," in *Proc. EMNLP*, pp. 1488–1498, 2015.

[25] M. N. Jones, J. Willits, and S. Dennis, "Models of semantic memory," *Oxford Handbook of Mathematical and Computational Psychology*, pp. 232–254, 2015.

[26] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-Evolve: Deep temporal reasoning for dynamic knowledge graphs," in *Proc. Int'l Conf. Machine Learning*, pp. 3462–3471, 2017.

[27] M. Marx, M. Krötzsch, and V. Thost, "Logic on MARS: Ontologies for generalised property graphs," in *Proc. IJCAI*, pp. 1188–1194, 2017.

[28] H. Beck, M. Dao-Tran, and T. Eiter, "LARS: A logic-based framework for analytic reasoning over streams," *Artificial Intelligence*, vol. 261, pp. 16–70, 2018.

[29] M. W. Chekol and G. Pirr, "Marrying uncertainty and time in knowledge graphs," in *Proc. AAAI*, pp. 88–94, 2017.

[30] A. Ronca, M. Kaminski, B. C. Grau, B. Motik, and I. Horrocks, "Stream reasoning in temporal datalog," in *Proc. AAAI*, pp. 1941-1948, 2018.

[31] A. Sadeghian, R. Miguel, Z. W. Daisy, and C. Anthony, "Temporal reasoning over event knowledge graphs," in *Proc. Workshop on Knowledge Base Construction, Reasoning and Mining*, pp. 54–57, 2018.

[32] N. Katzouris, A. Artikis, and G. Paliouras, "Incremental learning of event definitions with inductive logic programming," *Machine Learning*, vol. 100, pp. 555–585, 2015.

[33] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proc. Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

[34] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. AAAI*, pp. 2181–2187, 2015.

Pouya Ghiasnezhad Omran received PhD in computer science from Griffith University, Australia. He is currently a research fellow at Australian National University. His research interests include Knowledge Graphs, Machine Learning and Knowledge Representation.

Kewen Wang is a professor of computer science with Griffith University, Australia. His current research interests include reasoning and learning in knowledge-based systems. He has been publishing in top-ranked conferences and journals in AI. He is currently serving on the editorial board of the Journal of Web Semantics. He has been regularly on the program committees of major conferences in AI, such as IJCAI and AAAI.

Zhe Wang is a lecturer at Griffith University, Australia. His research interests include semantic technologies, knowledge representation and reasoning, and ontology-based data access. His research has led to over 40 publications in influential journals and conferences in AI.