Learning Temporal Rules from Knowledge Graph Streams

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

Abstract

Knowledge Graphs (KGs) are a prevailing data management approach and have found extensive applications in recent years. While several methods have been proposed for learning schema information for KGs in the form of logical rules, they are not suitable for KGs with constantly evolving data. This paper makes the first attempt to address the problem by presenting an approach to learning temporal rules from KG streams. The learned temporal rules can be applied in link prediction and event prediction over KG streams. Based on the proposed method, a system StreamLearner has been implemented. Our experimental results show that StreamLearner is effective and efficient in learning temporal rules on real-life datasets and significantly outperforms some state-of-the-art systems that do not account for temporal knowledge or evolving data.

Introduction

Knowledge graphs (KGs) have proven to be a flexible and powerful data management approach that underlies a new generation of information systems. In many approaches, a KG is represented as a set of RDF triples and thus is also referred to as an RDF graph. A KG is usually very large and thus automated and scalable methods are needed for the creation, maintenance and use of KGs. On the other hand, it would be essential to extract schema information from KGs. As a result, in recent years, a number of systems have been developed for learning rules from KGs (Galárraga et al. 2015; Omran, Wang, and Wang 2018). For instance, they are able to learn rules of the form $\mathsf{playsFor}(x, y) \rightarrow \mathsf{isAffiliatedTo}(x, y)$ from the KG YAGO2(Suchanek, Kasneci, and Weikum 2007).

An increasing number of KGs are emerging that model events taking places over time besides static relations among entities. However, the existing rule learning systems customarily assume KGs to be static and are not suitable for KGs with constantly evolving data. A KG containing 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 four entities and some events occurred among them at various time points. Entities e_0, e_1, and e_2 are three countries, and e_3 is the missile. Three events occurred in the past: Country e_0 test missile at time point $\tau - 4$, and then it established military cooperation with country e_1 and imposed sanction on country e_2 at time point $\tau - 3$. Besides classical link prediction questions such as

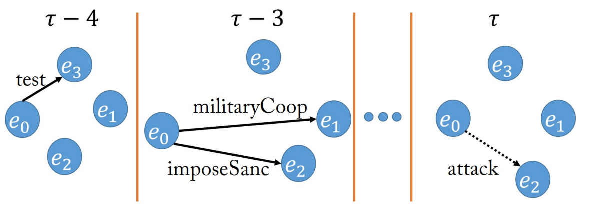

Figure 1: Example of KG Stream.

"Which countries besides e_0 tested missile?" or link prediction questions with temporal knowledge such as "Which counties may attack e_2 at the time point τ?", there are also schema level questions that of much interest, such as "After a country x tested missile and then imposed sanction on another country y on the next day, in how many days is it most likely that x will attack y?"

This brings in some research challenges. First, this requires temporal knowledge to be expressed in the schema information. For example, the above question can be expressed as a temporal rule $\mathsf{test}(x, missile, t) \land \mathsf{imposeSanc}(x, y, t + 1) \rightarrow \mathsf{attack}(x, y, t + k)$, where k ranges over integers. Learning a temporal rule with a specific k and a highest confidence degree indicates a answer to the above question. In fact, our experiments show that using temporal rules also improves the accuracy of classical link prediction over KG Streams (see Section). Moreover, existing rule learners for KGs cannot be directly used to learn temporal rules. Finally, a framework for stream learning and reasoning of temporal rules is still missing. Regarding the application of such framework, in the real-world problems, we face to this kind of dynamic data which ignoring their time feature make the obtained model inaccurate or incapable. For example, ICEWS (Trivedi et al. 2017) is a system for early warning designed to assist US policy experts. By this system, the analysts predict a range of international

Copyright held by the author(s). In A. Martin, K. Hinkelmann, A. Gerber, D. Lenat, F. van Harmelen, P. Clark (Eds.), Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019). Stanford University, Palo Alto, California, USA, March 25-27, 2019.

crises.

In this paper, we make the first attempt to tackle this problem by developing a method for learning temporal rules from KG streams. In our proposed method, an existing rule learner for KG is used to extract static rules from an initial data set, which is the collection of facts at the first few time points. These static rules, referred to as *structure rules*, are used to construct a space of candidate temporal rules. Then we extract temporal rules from this rule space by generalising major quality measures for static rules to temporal rules. Based on this method, we have implemented a system called StreamLearner. Experiments have been conducted to evaluate the proposed method. Our experiments show that StreamLearner outperforms the state-of-the-art static KG learners including HOLE (Nickel, Rosasco, and Poggio 2016) and TransE (Bordes et al. 2013) regarding the link prediction in the streaming scenario. We also show that (temporal) rule learning from KG data streams and the corresponding link prediction can be performed effectively and efficient over various configurations of the stream. For example, it can forecast the events for different time points ahead with acceptable accuracy.

Background

Knowledge graphs (KGs) concerns about entities, such as persons and places, and binary relations among them. A KG consists of a set of RDF triples (e, P, e'), each of which expresses a fact that entity e is related to another entity e' via relation P. Such a KG is static in the sense that temporal information is not taken into account. In this paper, we consider streams of KGs. A *KG stream* consists of a (possibly infinite) set of quadruples of the form (e, P, e', k), each of which expresses an *event* 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 *event* . Following the convention in knowledge representation, we denote such an event as $P(e, e', k)$, where P is a ternary predicate, e and e' are entities, 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 $S[0], S[1], \ldots$. 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 $P(e, e')$.

The class of *closed path rules* (or *CP rules*) have attracted attention in the community of KGs as it provides a balance between the expressive power of mined rules for KGs and the efficiency of rule mining. Such a syntactic restriction is now a widely adopted language bias in the literature of rule mining for KGs. For instance, 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

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

Here x, y and z_i's are variables, each $P(u, v)$ is an atom, and u and v are called respectively, the subject and object argument for P. Intuitively, the rule r reads that if $P_1(x, z_1), P_2(z_1, z_2), \ldots, P_n(z_{n-1}, y)$ hold, then $P(x, y)$ holds too. Atom $P(x, y)$ is the head of r, denoted $head(r)$, and the set of atoms $P_1(x, z_1), P_2(z_1, z_2), \ldots, P_n(z_{n-1}, y)$ is the body of r, denoted $body(r)$. It 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.

In this paper, we consider temporal CP rules (or simply temporal rules) γ of the following form:

$$P_1(x, z_1, t) \wedge P_2(z_1, z_2, t) \wedge \ldots \wedge P_n(z_{n-1}, y, t) \\ \rightarrow P(x, y, t + k). \quad (2)$$

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 mining algorithms. Especially, in learning rules from data streams, we are interested in making predictions based on current data stream.

Temporal Rule Learning

Existing rule learning methods generally involve two components: one is to learn the structure of the rules (e.g., through search and refinement) and the other is to filter out low quality rules through certain quality measures. Our temporal rule learning also involve such two components.

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*. A static rule learner such as RLvLR (Omran, Wang, and Wang 2018) can be used to learn a set of CP rules R. Such a CP rule is referred to as a *structure rule*. That is, a structure rule is a static rule learned from the structure training data.

Given rule r of the form (1), for each integer $k \geq 0$, we obtain a temporal rule $r^{(k)}$ of the form (2):

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

Then to assess the quality of candidate temporal rules, we adapt standard measures, namely support, standard confidence, and head coverage, that are used in rule learning literature (Chen et al. 2016; Galárraga et al. 2015).

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

$$supp(r^{(k)}, \tau) = \begin{cases} 0, & \text{if } \tau < k \\ \#(e, e') : head(r, e, e', \tau) & \\ \wedge body(r, e, e', \tau - k), & \text{otherwise} \end{cases} \quad (3)$$

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 $\tau = k = 0$.

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) 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 $\tau - k$:

$$SC(r^{(k)}, \tau) = \frac{supp(r^{(k)}, \tau)}{\#(e, e') : body(r, e, e', \tau - k)} \quad (4)$$

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:

$$HC(r^{(k)}, \tau) = \frac{supp(r^{(k)}, \tau)}{\#(e, e') : head(r, e, e', \tau)} \quad (5)$$

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

Example 1 *Consider the first three time points in a KG stream S as follows:*

$$S[0] = \{P_1(e_3, e_2, 0), P_1(e_2, e_1, 0), P_1(e_1, e_3, 0), P_2(e_3, e_1, 0)\}$$

$$S[1] = \{P_1(e_2, e_2, 1), P_1(e_2, e_1, 1), P_2(e_1, e_3, 1), P_2(e_2, e_1, 1), P_2(e_2, e_3, 1), P_2(e_3, e_3, 1), P(e_1, e_1, 1), P(e_1, e_3, 1)\}$$

$$S[2] = \{P_1(e_1, e_3, 2), P_2(e_2, e_2, 2), P_2(e_3, e_1, 2), P_2(e_2, e_1, 2), P_2(e_3, e_3, 2), P(e_1, e_3, 2), P(e_1, e_1, 2)\}$$

Let $r : P_1(x, z) \wedge P_2(z, y) \rightarrow P(x, y)$ be a CP rule. Consider $k = 0, 1, 2$ and we have the following candidate temporal rules based on r:

$$r^{(0)} : P_1(x, z, t) \wedge P_2(z, y, t) \rightarrow P(x, y, t)$$

$$r^{(1)} : P_1(x, z, t) \wedge P_2(z, y, t) \rightarrow P(x, y, t + 1)$$

$$r^{(2)} : P_1(x, z, t) \wedge P_2(z, y, t) \rightarrow P(x, y, t + 2)$$

At time point $\tau = 2$, the quality of the these rules can be assessed as follows:

$$supp(r^{(0)}, 2) = 2, \quad SC(r^{(0)}, 2) = 1, \quad HC(r^{(0)}, 2) = 1$$

$$supp(r^{(1)}, 2) = 0, \quad SC(r^{(1)}, 2) = 0, \quad HC(r^{(1)}, 2) = 0$$

$$supp(r^{(2)}, 2) = 1, \quad SC(r^{(2)}, 2) = 0.5, \quad HC(r^{(2)}, 2) = 0.5$$

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 is independent from that at other time points. On the other hand, the quality of temporal rules at a time point in a KG stream 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:

$$DSC(\gamma, \tau) = \begin{cases} SC(\gamma, \tau), & \text{if } \tau = 0 \\ (1 - \alpha) \times DSC(\gamma, \tau - 1) & \\ +\alpha \times SC(\gamma, \tau), & \text{otherwise} \end{cases} \quad (6)$$

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.

$$DHC(\gamma, \tau) = \begin{cases} HC(\gamma, \tau), & \text{if } \tau = 0 \\ (1 - \alpha) \times DHC(\gamma, \tau - 1) & \\ +\alpha \times HC(\gamma, \tau), & \text{otherwise} \end{cases} \quad (7)$$

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, and so are the candidate temporal rules, 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.

Combine Stream Learning and Reasoning

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, $S^*[0, n]$ is stored, and two integers $l, m \geq 0$ as minimum and maximum *prediction distances*. And it produces as output 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: $r^{(l)}$, $r^{(l+1)}$, ..., $r^{(m)}$ 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 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 time point τ, to assess the quality of candidate temporal rule $r^{(k)}$, it is clear that if $\tau < k$ then $DSC(r^{(k)}, \tau) = DHC(r^{(k)}, \tau) = 0$; otherwise, we assume $DSC(r^{(k)}, \tau -$

1) has been obtained from the previous time point. By (3) – (7), 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, that is only those in $S[\tau - w + 1, \tau]$. In this case, if $k \leq w - 1$ then we have required events and the quality measures can be computed as in (3) – (7); 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 time point τ, to derive new events at time point $\tau + k$ with $l \leq k \leq m$ (recall that m is the maximum prediction distance), we apply selected temporal rule of the form $r^{(k)}$ to the events in $S[\tau]$. To obtain the confidence degree (CD) of a derived event, we adapt the $score^*(\cdot)$ function from (Galárraga et al. 2015) by aggregating the DSC of all the temporal rules that can derive the event in a Noisy-OR manner. The intuition is that events derived by more rules should have a higher confidence degree. Formally, 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:

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

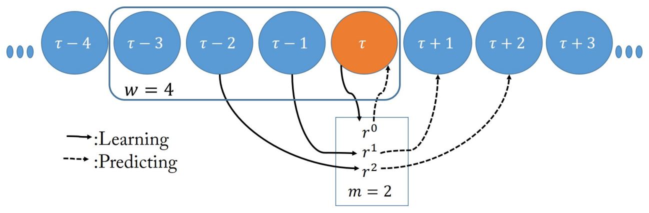

Figure 2: Stream Learning and Reasoning.

In Figure 2, we illustrate one snapshot of the stream where the current point is τ, $w = 4$, and $m = 2$. In this example, we consider one structure rule, r. In this time point, we learn the DSC from current and previous time points while we use this time point as the body of rules which get fired and predict the new events for current time point and the future ones.

Algorithm 1 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 StaticLearner, such as 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 maximum k being 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, candidate temporal rules are filtered based on their DSC and DHC scores at time point τ. As discussed

Algorithm 1 KG Stream Learning and Reasoning

```
Input: A KG stream $S$ and two integers $m, n \geq 0$
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: $R :=$ StaticLearner$(S^*[0, n])$
 3: $\Gamma :=$ CandidateConstruct$(R, m)$
 4: loop
 5:     $W := W \cup S[\tau]$
 6:     if $\tau \geq w$ then
 7:         $W := W \setminus S[\tau - w]$
 8:     end if
 9:     $\Gamma_\tau :=$ Filter$(\Gamma, W)$
10:     stream out $\Gamma_\tau$
11:     for each $l \leq k \leq m$ and each $r^{(k)} \in \Gamma_\tau$ do
12:         $\Xi_{\tau+k} := \Xi_{\tau+k} \cup$ Apply$(r^{(k)}, S[\tau])$
13:     end for
14:     stream out $\Xi_\tau$
15:     $\tau := \tau + 1$
16: end loop
```

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 learnt temporal rules at time point τ have been applied, the derived events at τ, Ξ_τ, will not change and can be streamed out.

Experiments

We have implemented a system, StreamLearner, based on the above algorithm and conducted several experiments to evaluate it. In our implementation, RLvLR (Omran, Wang, and Wang 2018) is deployed for learning structure rules.

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 some existing statistical models, such as TransE (Bordes et al. 2013) and HOLE (Nickel, Rosasco, and Poggio 2016), that do not account for temporal knowledge or evolving data. Our goal is not to compete with temporal statistical models like Know-Evolve (Trivedi et al. 2017). To analyse the benefit of having temporal rules over static rules in stream reasoning, we also used a "static" version of StreamLearner (StreamLearner-S), which only uses the static structure rules (not the temporal rules). Specifically, our experimental results aimed to validate the following observations:

1. StreamLearner significantly outperforms the baseline methods StreamLearner-S, TransE and HOLE 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 increased, the prediction accuracy also increases without significant sacrifice on the efficiency.

3. The accuracy of link prediction generally increases when the sizes of shifting window increases or when the minimum prediction distances reduces. StreamLearner still outperforms StreamLearner-S in the cases of small window sizes and long minimum prediction distances.

In our experiments, we used the Integrated Crisis Early Warning System (ICEWS) dataset (Trivedi et al. 2017), which is based on historical events including interactions between socio-political agents (i.e., 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). ICEWS is an ideal benchmark for our experiments because the behaviour of agents in this benchmark has complex interaction patterns. We are unaware of any other benchmarks of a similar type. We also adopted the ICEWS-500 as in (Trivedi et al. 2017), which contains a small subset of ICEWS with 500 entities. We included this small subset to examine the impact of data sizes on different systems. Some statistics about the two datasets are shown in Table 1, including the numbers of entities, numbers of events, numbers of predicates, and numbers of time points.

Table 1: Dataset statistics

Dataset	#Entity	#Event	#Pred.	#TPoint
ICEWS	12498	668080	260	365
ICEWS-500	500	445665	260	365

We have conducted two sets of experiments.

In the first set of experiments, we used the facts from the first 50 days (i.e., $S^*[0, 50]$) as the training data for learning structure rules. We note that the size of the data set is much smaller than the initial data sets in other approaches for learning from streams. Since static rule learner RLvLR learns static rules about specified predicates in rule heads, called target predicates, we randomly selected 20 target predicates.

We assessed the accuracy of link prediction fortnightly after the 50th day. That is, 23 time points after the rules structure training. For each of these time points k, we separated the datasets into 70% training and 30% testing as in most literature. The link prediction task is to identify for each target predicate P and each entity e, an entity e' 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. The prediction accuracy was measured by filtered Mean Reciprocal Rank (MRR) and Hits@10 as in the literature (Bordes et al. 2013).

In this experiment, the size of shifting window was fixed to 10 days and the maximal prediction distance was also 10 days. For each time point k, SteamLearner applied the learnt temporal rules to the training data in $S[k]$, as well as $S[k-10, k-1]$, to derive missing events at k, whereas StreamLearner-S applied all the structure rules to the training data in $S[k]$. TransE and HOLE obtained their respective statistical models from the training data in $S[k]$.

Table 2 shows the performance of StreamLearner, StreamLearner-S, TransE and HOLE. We report the numbers of temporal rules (#R), the numbers of structure rules (#SR), MRR and Hits@10 (H@10) scores, averaged over all selected time points and target predicates.

The experimental results show that StreamLearner significantly outperformed the baseline systems. In particular, StreamLearner was around 4 times more accurate than TransE and HOLE on Hits@10, and about 5 times more accurate on MRR. StreamLearner also showed superior performance to its static version, which clearly demonstrates the benefit of temporal rules over static rules on stream reasoning. Indeed, the average number of temporal rules is only around 3 times more than that of structure rules. Considering the number of candidate temporal rules (10 times of structure rules), it shows that the temporal rules can model much refined and more precise association among events.

Figure 3 shows the detailed performance of StreamLearner and the baseline systems, from which we can see that of StreamLearner showed consistently superior performance over the time.

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 ICWEC dataset are obfuscated and thus it is difficult to assess the meaning of learnt temporal rules, it is not hard to image 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.

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

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.

The quality of learnt temporal rules and their performance in link prediction are affected by several factors related to the KG streams, including the sizes of structure training data, the window sizes, and the (minimum and maximum) prediction distance. For the second set of experiments, we evaluate how these factors affect the quality and performance of temporal rules.

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 StreamLearner. Table 3 shows the performance of StreamLearner with different sizes n of structure training

Table 2: Temporal rules in link predition

Dataset	StreamLearner			StreamLearner-S			TransE		HOLE	
	#TR	MRR	H@10	#SR	MRR	H@10	MRR	H@10	MRR	H@10
ICWEC	1748	0.35	0.48	568	0.24	0.30	0.05	0.12	0.07	0.10
ICWEC-500	1728	0.37	0.54	535	0.22	0.31	0.05	0.14	0.05	0.09

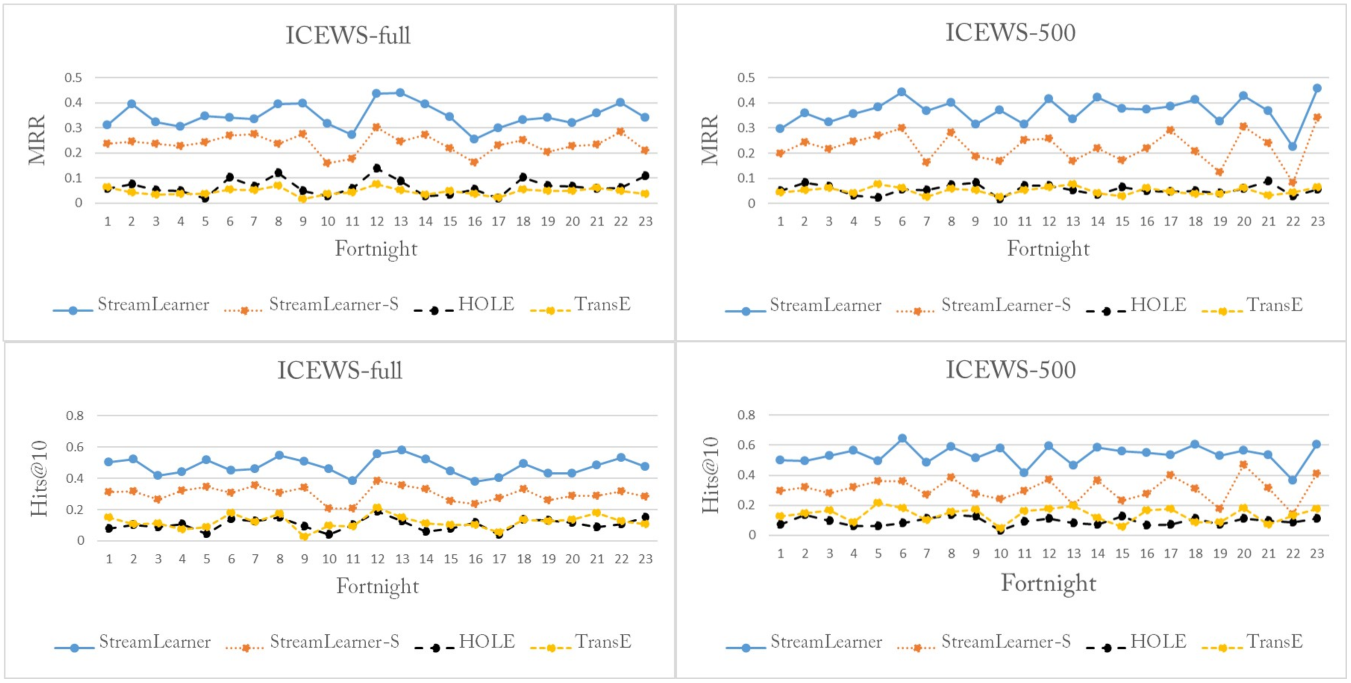

Figure 3: Performance comparison on link prediction over KG streams.

data $S^*[0, n]$, and we compare its performance with that of StreamLearner-S. We report the numbers of structure rules (#SR), stream learning and reasoning times (Time, in seconds), MRR and Hits@10 (H@10) scores, averaged over all selected time points and target predicates.

Table 3: Performance on varying sizes of structure learning data

Size	#SR	StreamLearner			StreamLearner-S		
		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

Overall, allowing larger structure learning data provides more structure rules and leads to better prediction accuracy on both StreamLearner and StreamLearner-S. Yet, there is trade-off between time efficiency and prediction accuracy, and as the size of structure learning data increases, the processing time of StreamLearner increases more significantly than that of StreamLearner-S, due to the number of candidate temporal rules. Yet the processing time of StreamLearner 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. While StreamLearner again significantly outperformed StreamLearner-S in all cases, an interesting observation is when the available data for rule structure learning is smaller, the benefit of using temporal rules over static rules is more obvious. This indicates when the available structural knowledge (i.e., association among facts, represented in structure rules and temporal rules) is relatively limited, the temporal knowledge (i.e., association among events, represented in temporal rules) plays a more significant role in prediction.

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 4 shows the performance of StreamLearner over window sizes ranging from 1 to 15. Again, we used StreamLearner-S 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.

Generally speaking, the performance of StreamLearner improves as the window size increases. It also reveals some local optimal points, namely window sizes 3 and 10. Al-

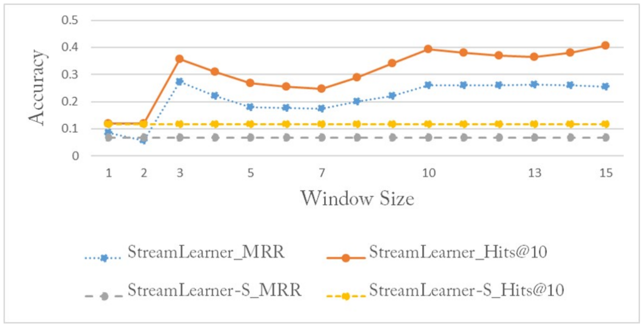

Figure 4: Performance over varying window sizes.

though slight improvement in Hits@10 can be observed for widow sizes larget than 10, MRR seems to be flattening. 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. In the previous experiments, we set the minimum prediction distance to be 0, which allowed events to be derived from the current data. Clearly, as the minimum prediction distance increases, the system is challenged to finalise the derived events in distance. For instance, if the minimum prediction distance is set to 3 then the learnt temporal rules are of the form $r^{(3)}, r^{(4)}, \ldots$. That means, all derived events at time point τ (Ξ_τ) are obtained at time point $\tau - 3$. This effectively requires the prediction to happen at least 3 days ahead.

In the following experiments, we evaluated the impact of minimum prediction distance on the performance of StreamLearner, using StreamLearner-S 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 5 shows the performance of StreamLearner over minimum prediction distances ranging from 0 to 10. Again, the values are averaged.

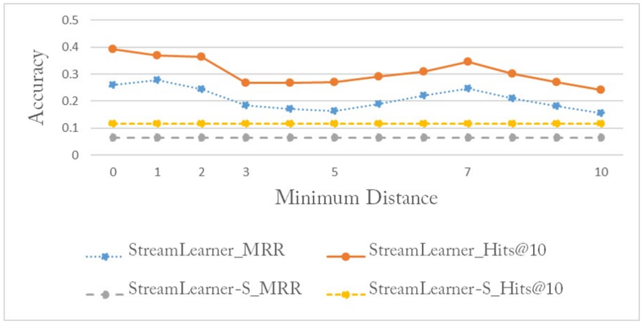

Figure 5: Performance over varying minimum prediction distances.

The performance of StreamLearner 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 weekly pattern in the event association.

Discussion and Conclusion

The topic of our work is in the intersection of stream reasoning and stream learning. There have been some approaches to address these two issues but separately. Knowledge representation and reasoning in the presence of data streams has been investigated in (Beck, Dao-Tran, and Eiter 2018; Marx, Krötzsch, and Thost 2017; Chekol and Pirr 2017). For instance, (Marx, Krötzsch, and Thost 2017) proposed a multi-attributed relational structure (MARS) to present the following rule:

$$\forall x, y, z_1, z_2, z_3 : spouse(x, y)@\{start : z_1, loc : z_2, end : z_3\} \rightarrow$$
$$spouse(y, x)@\{start : z_1, loc : z_2, end : z_3\}$$

But they did not investigate how to extract such rules from data streams.

In KG community, some methods have been proposed for learning rules from a given (static) KG. Recently, the method of embedding from representation learning is applied in rule learning for (static) KGs (Omran, Wang, and Wang 2018; Yang, Yang, and Cohen 2017), which has been proven quite effective.

There are numerous works on link prediction by employing the method of embedding such as (Trivedi et al. 2017; Sadeghian et al. 2018). Our method for rule learning from KGs is different from theirs, while our method can also be applied in link prediction. This is because we focus on learning structural information in the form of first order rules.

There are also some approaches to learning logic-based models form streams of relational data such as (Katzouris, Artikis, and Paliouras 2015; Mitra and Baral 2016). A method is proposed in (Katzouris, Artikis, and Paliouras 2015) for extracting event definitions automatically. Their system incrementally refine the logical definition of events based on temporal events information dataset. As the method is based on refinement operators in inductive logic programming, the resulting system is not scalable for handling KGs.

In this paper, we have proposed a method for learning temporal rules from data streams in the format of Knowledge Graphs (KGs). Such rules can be used for reasoning about event over different time points. Especially, temporal rules are useful for link prediction and event prediction in the setting of KG data streams. The proposed method is essentially a framework for rule learning from KG data streams by extending existing rule learners for KGs. Based on RLvLR, a state-of-the-art rule learner for KGs, we have implemented a system StreamLearner, which can learn temporal rules from KG data streams, completing dynamic KGs and predicting events for them. Our experiments show that StreamLearner significantly outperforms the base systems. It is able to handle large KGs and a promising approach to creating, maintaining and using data streams in the format of KGs.

There are still some interesting issues for future work. We plan to develop a framework to handle more complex events such as the event with the beginning and termination time or the event with the location of the happening label.

References

Beck, H.; Dao-Tran, M.; and Eiter, T. 2018. LARS: A Logic-based framework for Analytic Reasoning over Streams. *Artificial Intelligence* 261:16–70.

Bordes, A.; Usunier, N.; Weston, J.; Yakhnenko, O.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*.

Chekol, M. W., and Pirr, G. 2017. Marrying Uncertainty and Time in Knowledge Graphs. In *AAAI 2017 (Proceedings of the 31th Conference on Artificial Intelligence)*, 88–94.

Chen, Y.; Daisy, ·.; Wang, Z.; Goldberg, S.; Chen, B. Y.; Wang, D. Z. D.; and Goldberg, S. 2016. ScaLeKB: scalable learning and inference over large knowledge bases. *The International Journal on Very Large Data Bases* 25:893–918.

Galárraga, L.; Teflioudi, C.; Hose, K.; and Suchanek, F. M. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The International Journal on Very Large Data Bases* 24(6):707–730.

Gardner, M., and Mitchell, T. 2015. Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In *Conference on Empirical Methods on Natural Language Processing*, number September.

Katzouris, N.; Artikis, A.; and Paliouras, G. 2015. Incremental learning of event definitions with Inductive Logic Programming. *Machine Learning* 100(2-3):555–585.

Marx, M.; Krötzsch, M.; and Thost, V. 2017. Logic on MARS: Ontologies for Generalised Property Graphs. In *IJCAI*.

Mitra, A., and Baral, C. 2016. Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning. In *AAAI*.

Nickel, M.; Rosasco, L.; and Poggio, T. 2016. Holographic Embeddings of Knowledge Graphs. In *AAAI*.

Omran, P. G.; Wang, K.; and Wang, Z. 2018. Scalable Rule Learning via Learning Representation. In *IJCAI*.

Sadeghian, A.; Miguel, R.; Daisy, Z. W.; and Anthony, C. 2018. Temporal Reasoning Over Event Knowledge Graphs. In *1st Workshop on Knowledge Base Construction, Reasoning and Mining*, volume 2065, 54–57.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *International conference on World Wide Web*, 697–706.

Trivedi, R.; Dai, H.; Wang, Y.; and Song, L. 2017. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In *ICML*.

Wang, Z., and Li, J.-Z. 2015. RDF2Rules: Learning Rules from RDF Knowledge Bases by Mining Frequent Predicate Cycles. *Computer Research Repository*.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *NIPS*.