Applied Soft Computing Journal 103 (2021) 107144

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Multi-hop reasoning over paths in temporal knowledge graphs using reinforcement learning

Luyi Bai *, Wenting Yu, Mingzhuo Chen, Xiangnan Ma

School of Computer and Communication Engineering, Northeastern University (Qinhuangdao), Qinhuangdao 066004, China

ARTICLE INFO

Article history:
Received 31 July 2020
Received in revised form 21 December 2020
Accepted 26 January 2021
Available online 28 January 2021

Keywords:
Temporal knowledge graph
Multi-hop reasoning
Reinforcement learning
Long Short-Term Memory Networks

ABSTRACT

Knowledge graphs (KGs) are usually incomplete—many new facts can be inferred from KGs with existing information. In some traditional reasoning methods, temporal information is not taken into consideration, meaning that only triplets (*head*, *relation*, *tail*) are trained. In current dynamic knowledge graphs, it is a challenge to consider the temporal aspects of facts. Recent temporal reasoning methods embed temporal information into low-dimensional spaces. These methods mainly support implicitly reasoning, which means they cannot get the specific reasoning paths. These methods limit the accuracy of reasoning paths and ignore multiple explainable reasoning paths in temporal knowledge graphs (TKGs). To overcome this limitation, we propose a multi-hop reasoning model TPath in this paper. It is a reinforcement learning (RL) framework which can learn multi-hop reasoning paths and continuously adjust the reasoning paths in TKGs. More importantly, we add time vectors in reasoning paths, which further improve the accuracy of reasoning paths. Meanwhile, considering the diversity of temporal reasoning paths, we propose a new reward function. In TPath, the agent employs the Long Short-Term Memory networks (LSTM) to capture current observations from the environment, and it outputs action vectors (relation vectors and time vectors) to the environment through activation functions. Experimentally, our model outperforms other state-of-the-art reasoning methods in several aspects over two public temporal knowledge graph datasets.

1. Introduction

Knowledge graph is a significant concept in Artificial Intelligence, which plays an important role in many applications such as question answering [1], social networks [2], and semantic parsing [3], etc. Generally, a knowledge graph is the collection of relational facts that are represented as triplets (*head, relation, tail*), e.g., (*Donald Trump*, *Consult*, *Vladimir Putin*). Although the knowledge graphs consist of large amounts of triplets, they are far from complete. In reasoning task, it is a great challenge to infer the facts (which are not stored directly in the knowledge graphs) by observed information. From its early days, reasoning methods [4–6] embed entities and relations of triplets into low-dimensional vector spaces and score the reasoning vectors by score function. Meanwhile, these methods mainly support implicitly reasoning, which ignore the explainable reasoning paths and limit their usage in more complex reasoning tasks. Yang et al. [7] propose a novel model, which learns the logical rules and trains fixed-length of explainable reasoning paths. Some reasoning methods [8–10] take the reinforcement learning framework into consideration, and further learn the multi-hop reasoning paths in KGs. However, a large amount of event data contains complex temporal information, and traditional reasoning methods are hard to learn temporal information in form of triplets. Knowledge graphs need to be extended to the temporal knowledge graphs (TKGs) that represented as 4-tuple (e_s, r, e_o, tim). It is the observed information, and each edge expresses that two entities connected by relation at the time point.

Recently, temporal knowledge graph reasoning has gradually received attention. Some researchers [11–13] apply the embedding technologies to TKGs, which embed time information expressions into low-dimensional spaces. However, they ignore the dynamic changes of events over time. Some reasoning methods [14–16] consider the temporal scopes of facts to learn the complex events. Another similar work is the time-aware progress: Jiang et al. [17] propose a time-aware KG embedding model through using temporal order information among facts. Han et al. [18] utilize a deep learning architecture to capture temporal dependencies. They only reason at implicitly reasoning edges of different entities, and cannot handle specific reasoning paths. Implicitly reasoning path means that the models only get the final reasoning results, but cannot reason the true and specific reasoning paths. Specific reasoning paths means that our model

* Corresponding author.
E-mail address: baily@neuq.edu.cn (L. Bai).

https://doi.org/10.1016/j.asoc.2021.107144
1568-4946/

can get all reasoning paths, including the head entity, relation, tail entity and time information.

In knowledge graph shown in Fig. 1, given unobserved fact (*Donald Trump, Consult, ?*), there are two results: '*Elizabeth II*' and '*Vladimir Putin*'. We cannot get a precise target entity directly, and it will lead to a decrease in the accuracy of reasoning. However, in the temporal knowledge graph, given unobserved fact (*Donald Trump, Consult, ?, 2018-10-11*), we can easily get the accurate answer '*Elizabeth II*' in temporal knowledge graph reasoning. For the temporal knowledge graph reasoning task, current methods mainly support implicitly reasoning paths (cannot get the specific reasoning paths). For example, given unobserved fact (*Donald Trump, Sign formal agreement, ?, 2019-05-07*), current approaches only return an uncertain result (maybe a correct result or an incorrect result), and it will limit the accuracy of reasoning in temporal knowledge graphs. In our model, we can infer the unobserved facts (*Donald Trump, Sign formal agreement, ?, 2019-05-07*) from the following temporal specific multi-hop reasoning path:*Donald Trump→Consult/2018-10 24→Vladimir Putin→Make optimistic comment/2018-10-29→United States→Host a visit/2018-11-05→China*. It means that our model can get specific multi-hop reasoning path and improve the accuracy of reasoning in temporal knowledge graph.

In this paper, we propose a novel multi-hop reasoning model for complex temporal knowledge graph. Firstly, we model the path reasoning process through reinforcement learning (RL) framework. What is more, we add the time information as a separate vector to participate in the iteration of environment and agent. More importantly, we propose a policy network that can train the agent to learn temporal multi-hop reasoning paths. In addition, we propose a novel reward function that has a good performance in dealing with the diversity of temporal knowledge graphs reasoning. In temporal knowledge graph, given a missing 4-tuple ($entity_1$, *relation*, *?*, *time*), the agent learns the policy network, which can learn to arrive target entity by choosing the actions at each step from $entity_1$. At the same time, the agent will adjust the entire path through the reward function. We call the model TPath for "Temporal Multi-hop Reasoning Paths". We evaluate TPath on two benchmark datasets, experiment results show significant and consistent improvements compared to state-of-the-art models in several aspects. Meanwhile, these results also mean that multi-hop paths can be suitable in temporal knowledge graphs and our model can scale up to large temporal knowledge graphs.

The main contributions of this paper:

- We propose a novel temporal reasoning model using reinforcement learning framework in temporal knowledge graphs.
- We construct a new policy network which takes the time vectors into consideration and selects the specific multi-hop reasoning paths in the temporal knowledge graphs.
- We propose a new reward function that considers diversity of temporal reasoning paths and offers better control in entire temporal multi-hop paths.
- Experiments show that our model can scale up to large temporal knowledge graphs, outperforming some state-of-the-art reasoning methods in several aspects.

The rest of the paper is organized as follows. In Section 2, related works are introduced briefly. We propose our model and detail its training process in Section 3. Experimental evaluations are given in Section 4. Section 5 concludes the paper.

Table 1
The properties of traditional knowledge graph reasoning models.

Year	Model	Temporal feature	Path reasoning	Explicitly path
2013	TransE			
2014	TransH			
2014	NeuralLP			
2015	TransR			
2015	DisMult			
2016	Transpare			
2016	ProJE			
2016	ComplEx			
2017	Deeppath		✓	✓
2018	MINERVA		✓	✓
2019	AttnPath		✓	
2020	R2D2			
	TPath	✓	✓	✓

2. Related work

2.1. Traditional knowledge graph reasoning

Embedding technology, as a main technology of traditional knowledge graph reasoning, has been extensively studied. Bordes et al. [4] propose TransE, which embeds entities and relations into the same low-dimensional vector spaces and defines a scoring function on each fact to measure its accuracy. Since TransE has issues when modeling 1-to-N, N-to −1 and N-to-N relations, TransH [5] is proposed to enable an entity having different representations when involved in various relations. Next, Lin et al. [6] present embedding approach for entities and relations in separate entity spaces and relation spaces, which learns embeddings via translation between projected entities. Yang et al. [19] constrain the relation matrix as a diagonal matrix to reduce the computing cost. There are other improved models considering additional information like heterogeneity [20], neural network [21] and tensor factorization [22]. Wang et al. [23] make a comprehensive review of these models. However, these methods cannot get specific and explainable reasoning paths in knowledge graphs. Yang et al. [7] present NeuralLP that learns the first-order logical rules in an end-to-end differentiable model to solve the problem of learning probabilistic first-order logical rules for knowledge base reasoning. Further, Xiong et al. [8] consider reinforcement learning (RL) framework for learning relational paths in knowledge graphs. Das et al. [9] explore a new way of automated reasoning on large knowledge bases which trains the agent to walk to the answer node conditioned on the input query. It addresses the much more difficult and practical task of answering questions where the relation is known, but only one entity. However, these methods receive false negative supervision and incidentally lead to correct reasoning paths. Lin et al. [24] adopt a pretrained one-hop embedding model to accurate environment representation and counter the spurious path to explore the search space. Wang et al. [25] explore a new mechanism of reinforcement learning called AttnPath, which incorporates Long Short-Term Memory networks and Graph Attention Mechanism to alleviate the model from pretraining. Hildebrandt et al. [10] propose Reveal Relations using Debate Dynamics, a method for knowledge graph reasoning based on a debate between two opposing reinforcement learning agents. However, they do not consider dynamic complex temporal changes in temporal knowledge graphs. We can summarize the properties of traditional knowledge graph reasoning models in Table 1. Our model TPath takes the time information into consideration, and we propose a new policy network to train multiple explainable reasoning paths in temporal knowledge graphs.

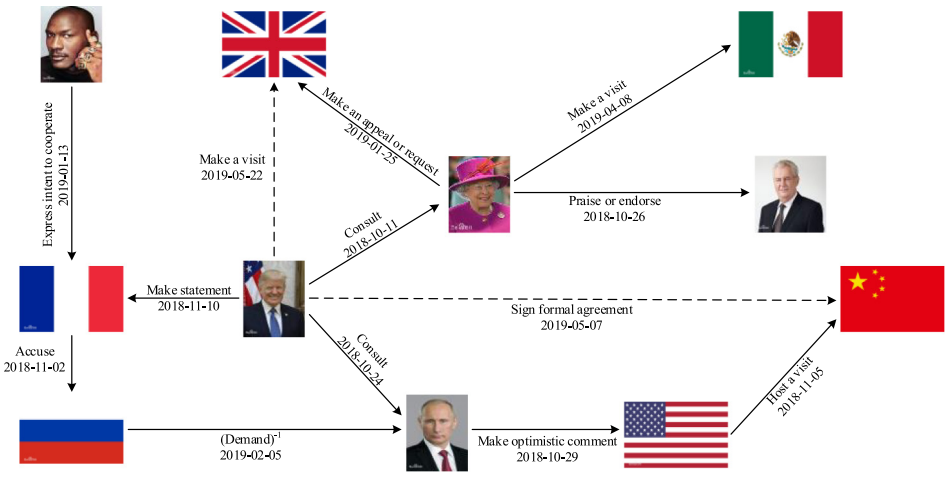

Fig. 1. A small fragment of a temporal knowledge graph. Solid arrows are observed and existing relation, and time links in the TKG and dotted arrows show relations and temporal information by the reasoning paths. r^{-1} denotes the inverse of relation r. Note the bold relation can be reasoned by reasoning paths between start entity *Donald Trump* and the corresponding target entity.

2.2. Temporal knowledge graph reasoning

In the field of temporal RDF (Resource Description Framework), Gutiérrez et al. [26] present definitions of temporal RDF and prove that entailment of temporal graphs does not yield extra complexity than RDF entailment. Pugliese et al. [27] present a notion of normalized temporal RDF graph, and propose a query language for these graphs based on SPARQL. Zimmermann et al. [28] present a generic framework for representing and reasoning with annotated Semantic Web data and a non-approximate reasoning algorithm for a subset of temporal RDF Schemas.

As knowledge graphs become more and more popular, many researchers shift their research focus from temporal RDF to temporal knowledge graphs. There are some recent methods on handling temporal information in TKGs. Dasgupta et al. [11] propose HyTE, which explicitly incorporates time into the entity-relation space by associating each timestamp with a corresponding hyperplane. It temporally exploits scoped facts of KG to perform KG inference, and can also predict time scopes for unannotated temporal facts. In the same year, García-Durán et al. [13] utilize recurrent neural network to learn time-aware representations of relation types by incorporating temporal information. Leblay et al. [12] embed time information into low-dimensional spaces. These methods are all based on embedding technology and cannot capture temporal dependency. Next, Trivedi et al. [14] present Know-Evolve, a novel deep evolutionary knowledge network, which learns non-linearly evolving entity representations over time. Ma et al. [16] generalize leading approaches for traditional knowledge graphs (i.e., ComplEx, DistMult) to temporal knowledge graphs and propose a novel tensor model denoted as ConT, which is an effective method to store episodic data and to be able to generalize to new facts. Further, Jin et al. [15] find a problematic formulation in Know-Evolve and propose a novel neural architecture for modeling complex event sequences. Another closely work is the time-aware progress. Jiang et al. [17] predict links in a TKG using both the existing facts and the temporal information of the facts. Han et al. [18] propose the Graph Hawkes Neural Network, which is a deep learning architecture to capture temporal dependencies on temporal knowledge graphs. We can summarize the properties of temporal knowledge graph reasoning models in Table 2. Our model TPath can select multiple specific temporal reasoning paths and further improve the accuracy of reasoning paths.

Table 2
The properties of temporal knowledge graph reasoning models.

Year	Model	Temporal feature	Path reasoning	Explicitly path
2017	KnowEvolve	✓		
2018	Hyte	✓		
2018	TTransE	✓		
2018	TA-TransE	✓		
2018	TA-DisMult	✓		
2018	ConT	✓		
2019	ReNet	✓		
2020	GHNN	✓		
	TPath	✓	✓	✓

3. Model

In this section, we propose our model in detail. Firstly, we introduce the environment and an agent-aimed policy network. The agent learns to select the best reasoning paths through interacting with the environment. Then, we will introduce the training process of TPath and the algorithm of interaction between environment and agent. Our overall model is illustrated in Fig. 2.

3.1. Reinforcement learning for reasoning paths

Our reinforcement learning framework is divided into two parts. In the first part, we will introduce the environment as a partially observed Markov decision process (POMDP) on a temporal knowledge graph. We define a 5-tuple (S, O, A, P, R) to represent POMDP, where S is the valid continuous state space, O is the unobserved complete state space, A is the set of all possible actions, P is the transition probability matrix (distinguish from symbol T in the graph), and R is the reward function of taking action from state space. In the second part, we will introduce the policy network π_θ which reasons the target entity from given missing 4-tuple (e_s, r, ?, tim). From start entitye_s, the agent learns to seek target entity e_o. Then, the agent will stay at the target entity when reaching it. At the same time, the environment will return the whole reward of reasoning paths.

We formally define the task of multi-relational and multi-hop reasoning in the TKGs. The TKGs are collection of the facts stored as 4-tuple (e_s, r, e_o, tim), e_s, $e_o \in \epsilon$, $r \in R$, $tim \in T$, where ϵ denotes the set of entities, R and T are the sets of binary relations and time points. Our model reasons the target entity of the form (e_s, r, ?, tim), e.g. (*Donald Trump*, *Sign formal agreement*, *?*, *2019-05-07*).

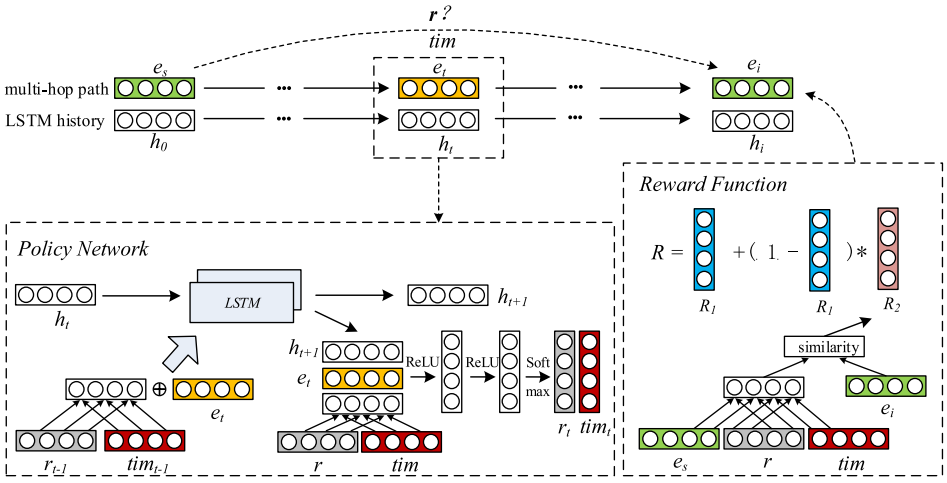

Fig. 2. The framework of our model TPath. Here $\{e_s, \ldots, e_t, \ldots, e_o\}$ denote the multi-hop path that found by the agent. The left part is the architecture of policy work at time point t. The agent employs LSTM that captures previous relation r_{t-1}, previous time tim_{t-1}, current entity e_t and the hidden state of LSTM h_t, outputs current relation vector r_t and time vector tim_t through two-layer feed-forward network (ReLU) and softmax function. The right part is the whole reward function of the reasoning path, which consist of basic reward R_1 and path reward R_2. The agent will receive the whole reward of reasoning path by reward function.

It also returns the whole rewards to predict if the fact is true, e.g. (*Donald Trump*, *Sign formal agreement*, *China*, *2018-12-04*)?.

3.1.1. Environment

Our environment is a POMDP. Formally, a TKG is a directed and labeled multi-graph $G = (V, E, R, T)$, where V and E denote the vertices and edges of the graph respectively. Note $V = \epsilon$ and $E \subseteq V \times R \times V \times T$. So we define a POMDP as a 5-tuple (S, O, A, P, R), where each component of POMDP is illustrated as follows.

S (***State***). State space S, each state $s \in S$ is represented by $s = (e_t, e_s, r, e_o, tim)$, where e_t is the current entity, e_s and e_o are start entity and target entity respectively, and r denotes the relation between e_s and e_o at time tim. S consists of all observed and valid combinations.

O (***Observations***). The environment is unobservable, and only transfer its current location and the task, i.e. (e_t, e_s, r, tim) is observed. The observation function O: $S \rightarrow V \times V \times R \times T$ is defined as $O(s = (e_t, e_s, r, e_o, tim)) = (e_t, e_s, r, tim)$.

A (***Action***). The action space is defined as a set of possible relations and time points from a state s consisting of all outgoing edges of the vertex e_t in G, i.e. $A_s = \{(r, tim|s)\}$. Formally, $A = \{(e_t, r_c, v, tim_c) \in E: s = (e_t, e_s, r, e_o, tim), r_c \in R, v \in V, tim_c \in T\}$, where v is the provisional entity. It means that the agent at each state selects from multiple options outgoing edge r_c at timetim_c and reaches provisional vertex v. Meanwhile, the agent takes fewer steps of reasoning paths than given length. To make the agent stay at the target node for the given number of steps, a special self-loop action is added to the action space which goes from a node to itself.

P (***Transition***). The environment updates the state to new state vertex via the edge selected by the agent. Formally, the transition function is $P: S \times R \times T \times V \rightarrow S$ defined by $P(s, a) = (v, e_s, r, e_o, tim)$, where s is the state and a is the action space selected by the agent.

R (***Reward***). There are two factors that contribute to the quality of the temporal reasoning paths as follows.

Basic reward: In our environment, the agent needs to select particular action from a large number of actions. We define the basic reward function as follows to guide the agent make a choice so that it can reach the target entity e_o.

$$R_1 = \begin{cases} +1 & if\ path\ reaches\ e_o \\ +0 & otherwise \end{cases} \tag{1}$$

Path reward: In temporal reasoning paths, the agent tends to choose repeating efficient paths, which will limit the scope of reasoning paths. To determine the location of current entity in the reasoning paths and encourage the agent to explore as positive paths as possible, we define the second reward function that takes repetitive reasoning paths into consideration as follows:

$$R_2 = \sigma\left(\frac{f(e_s, e_t)}{\max_{(e_s, r, v, tim) \in D}(f(e_s, v))}\right) \tag{2}$$

where $\sigma(x)$ is the sigmoid function, D denotes all true and observed 4-tuples (e_s, r, v, tim) in temporal knowledge graphs, and $f(x, y)$ represents the reasoning paths from entity x to current entity y, usually dot product between the entities. The path reward scores the reasoning paths, which will properly reduce the whole rewards of the reasoning paths and further promote the probability of exploring the new entities.

Reward: Reward is the linear combination of the defined reward functions.

$$R = \lambda_1 R_1 + \lambda_2 R_2 \tag{3}$$

where R_1 and R_2 are defined in Eq. (1) and Eq. (2), and linear parameters $\lambda_1, \lambda_2 \in [0, 1]$.

3.1.2. Policy network

There are amounts of entities, relations and time points in temporal knowledge graphs. Each entity usually connects multiple entities by different edges. In addition, there are multi-hop reasoning paths between two entities. In a temporal knowledge graph, given (e_s, r, e_o, tim), we define multiple reasoning paths between entity pairs (e_s, e_o): $P = \{p_1, p_2, \ldots, p_m\}$, where $P \in \Re^m$, and $p = \{e_s, r_0{:}tim_0, e_1, R_1{:}tim_1, \ldots, r_{k-1}{:}tim_{k-1}, e_o\}$, $p \in P$, ':' denotes vector concatenation. The length of reasoning paths generally does not exceed 6 (≤ 6).

The agent based on Long Short-Term Memory (LSTM) encodes the history H_t as a continuous vector $h_t \in \Re^{2d}$. The history $H_t = (A_{t-1}, O_t, H_{t-1}) = (r_{t-1}{:}\ tim_{t-1}, O_t, H_{t-1})$ is the sequence of past actions and observations. We also define $\boldsymbol{e} \in \Re^{|\varepsilon| \times d}$, $\boldsymbol{tim} \in \Re^{|T| \times d}$ and $\boldsymbol{r} \in \Re^{|R| \times d}$ as the embedding matrix for the entities and binary relations. Update the LSTM dynamics as follows:

$$\boldsymbol{h_0} = LSTM(\boldsymbol{a_s}; \boldsymbol{o_s}, \boldsymbol{0}) \tag{4}$$

$$\boldsymbol{h_t} = LSTM(\boldsymbol{a_{t-1}}; \boldsymbol{o_t}, \boldsymbol{h_{t-1}}), t > 0 \tag{5}$$

where A_s denotes a special virtual action vector from the start entity e_s, o_s is the start observation, $a_{t-1} = (r_{t-1}, \boldsymbol{tim}_{t-1})$ denotes the vector representation for relation and event time at time *t-1*, O_t denotes the observation at time *t*, and ';' denotes vector concatenation.

Based on the current observation O_t and current history vector h_t, the network chooses the relation and event time (actions) as outputs from all conditional actions. The neural network is two-layer feed-forward network, and each layer is followed by a rectified nonlinearity layer (ReLU). The output is an action vector from all possible action vectors over current entity, and is normalized using a softmax function. The policy network π is defined as follows:

$$\pi\theta(a_t|o_t) = \text{softmax}(W_1\text{ReLU}(W_2[o_t; h_t])) \tag{6}$$

where W_1 and W_2 are weight matrices of a two-layer feed-forward neural network, O_t and H_t are the observation vector and history vector at time *t*, and ';' denotes vector concatenation.

3.2. Training

Fig. 3 shows the training process with iterative guidance through the environment and the agent. The environment stores all vectors, including entity vectors, relation vectors, time vectors, etc. Given a missing 4-tuple (*e*, *r*, *?*, *tim*), initialize state S_1 with the current vectors, and transmit the observation O_1 that obtained by observation function to the agent. The agent will compute the action A_1 through the policy network (Fig. 2) and transmit action A_1 to the environment. After converting from state S_1 to S_2 through the action A_1 in the environment, the agent will return the reward of this reasoning path (computed by Eq. (3)). This forms the iteration with the environment and the agent. Finally, the training process will get the reward of entire reasoning path by Eqs. (7) and (8) as follows.

For the policy network (π_θ) described above, we update the parameters θ to maximize the expected cumulative reward using Monte-Carlo Policy Gradient (REINFORCE) [29]:

$$J(\theta) = E_{A\sim\pi_\theta(r,tim|s;\theta)}E_{(e_c,r_c,v,tim_c)\in D}(\sum_t R_{s_t,A_t}) \tag{7}$$

where $J(\theta)$ represents the reward of one episode, *A* denotes the action space, the first *E* (expectation) is replaced with empirical average over the training dataset. For the second *E* (expectation), we approximate it by running multiple rollouts for each training example. *D* denotes all true and observed 4-tuples (e_t, r_c, *v*, tim_c) in temporal knowledge graphs, and R_{S_t,A_t} denotes the whole rewards of taking action *A* in state *S* at time *t*.

The approximated gradient used to update the policy network is:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_t \log \pi_\theta\,(a_t|s_t)\, R(s_t, a_t) \tag{8}$$

The detail of the training process is described in Algorithm 1. Firstly, we input the 4-tuple vectors (e_s, *r*, *?*, *tim*) of the temporal knowledge graph, including entity vectors, relation vectors and time vectors. Next, the training process of TPath is the iteration between the agent and the environment (Line 01–13). For each episode (Line 01), we initialize the state vector and the episode length (Line 02–03). Next, we get observation vector through current state vector (Line 04). We set previous history h_{t-1} and previous action a_{t-1} (Line 05). Then, we limit the episode length with an upper bound *max_length* (Line 06). Within the length of reasoning paths *max_length*, the agent will train the policy network and get next action vector (Line 07–08). The environment will arrive at the next state and calculate the reward of reasoning paths by the reward function to the agent (Line 09). The agent will update the observation vector through current state (Line 10), and increment the *steps* to loop within *max_length* (Line 11). The agent will update the parameter θ (Line 12–13), and the algorithm will finally output the entire path and maximize the expected cumulative reward.

Table 3
Statistics of various datasets.

Dataset	#Ent	#Rel	#Time	#Train	#Test	#Valid
ICEWS	7531	253	287	391936	48266	48690
GDELT	6172	236	15	839590	104638	105010

Algorithm 1: Training Process of TPath

Input: 4-tuple vectors (e_s, *r*, *?*, *tim*) consists of entity vectors {e_1, e_2,...}, relation vectors {r_1, r_2,...}, and time vectors {tim_1, tim_2,...}

Output: the entire path p:{e_s, r_0:tim_0, e_1, r_1:tim_1,..., r_{k-1}:tim_{k-1}, e_o} and maximize the expected cumulative reward *g*

01 **for** *episode*←*1* **to** *N* **do**
02 **Initialize:** state vector *S*: s_t←s_0
03 **Initialize:** episode length *steps*←*0*
04 Get observation vector *O*: o_t←$o(s_t)$
05 Set history vector h_{t-1}←0, action vector *A*: a_{t-1}←0
06 **While** *steps*<*max_length* **do**
07 Update history vector h_t←LSTM(a_{t-1}; o_t, h_{t-1})
08 Get action vector: $a_t(r_t; tim_t)$← $\pi_\theta(a_t|\, o_t)$
09 Observe reward *R*: R_t← $\lambda_1 R_1 + \lambda_2 R_2$, next state s_{t+1}
10 Get observation vector o_{t+1}←$o(s_{t+1})$
11 Increment *steps*
12 Update parameter θ using:
 $g \propto \nabla_\theta \sum_t log\pi(r_t; tim_t|\, s_t, \theta) R_t$
13 **end for**

4. Experiments

4.1. Experimental setup

4.1.1. Datasets

We evaluate our model on two event datasets: Global Database of Events, Language, and Tone (GDELT) [30] and Integrated Crisis Early Warning System (ICEWS) [31]. Meanwhile, GDELT dataset is collected from January 1, 2018 to January 15, 2018. ICEWS dataset is collected from October 7, 2018 to June 25, 2019. Both datasets contain records of events that include two entities, one relation and one time point. Table 3 reports the various statistics of the datasets. #Ent, #Rel and #Time denote the number of entities, relations and time points. Besides, original datasets are randomly divided into train datasets, test datasets and valid datasets by (80%)/(10%)/(10%).

For each dataset, we build "graph" that adds the inverse 4-tuples in train datasets. For example, there is a 4-tuple (e_1, *r*, e_2, *tim*) in train dataset and its inverse 4-tuple is (e_2, r^{-1}, e_1, *tim*), where r^{-1} denotes the inverse relation of *r*. In training process, the agent can go back by inverse 4-tuples in the TKG.

For evaluating performance of reasoning path, we separately rank against the relations in two original datasets and build subsets of the top-n relations as the reasoning tasks. For each top-n relation r_n, we extract all 4-tuples with r_n from original datasets and split them into train, valid, test datasets of the relation r_n. In addition, we generate some false 4-tuples, and rank them with test datasets for evaluation of reasoning tasks.

4.1.2. Baseline methods

We evaluate our model with some reasoning methods: (1) *traditional knowledge graph reasoning*. We have summarized some

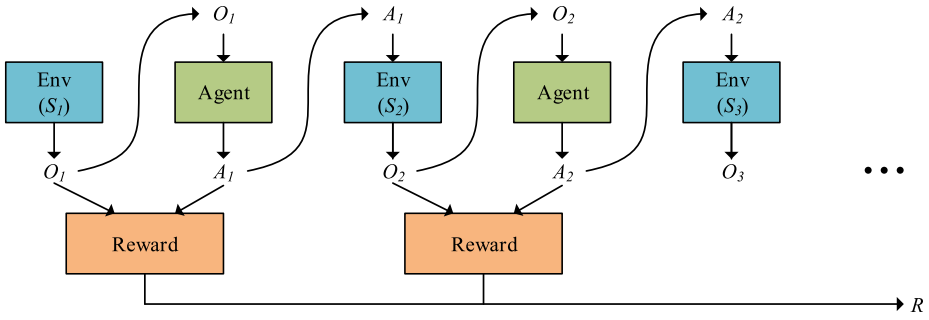

Fig. 3. Training process of our model TPath. Env, Agent and Reward denote the environment, agent (policy network) and reward function of RL framework. The environment stores all vectors, including entity vectors, relation vectors, time vectors, vector groups State S, etc. $\{O_1, O_2, \ldots\}$ denote a group of partially observable vectors. $\{A_1, A_2, \ldots\}$ $\{A_1, A_2, \ldots\}$ represent the action vectors selected by the agent through the policy network. R represents the reward of the entire path, calculated by the reward function.

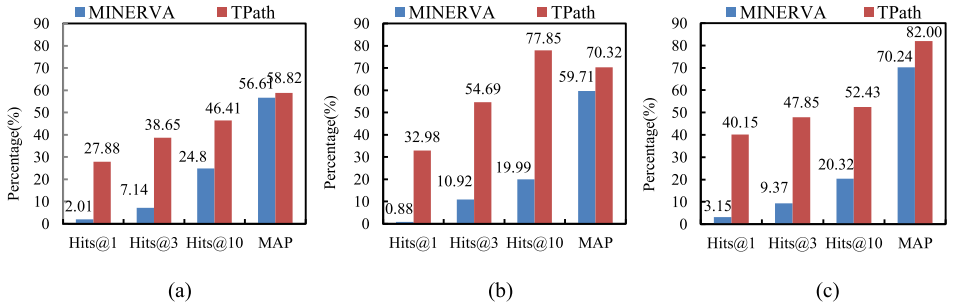

Fig. 4. Performance comparisons of TPath (Red) and MINERVA (Blue) on top-3 relations of ICEWS. The (a), (b) and (c) are the evaluation results of the relation '*Make statement, not specified below*', '*Arrest, detain*' and '*Consult, not specified below*'.

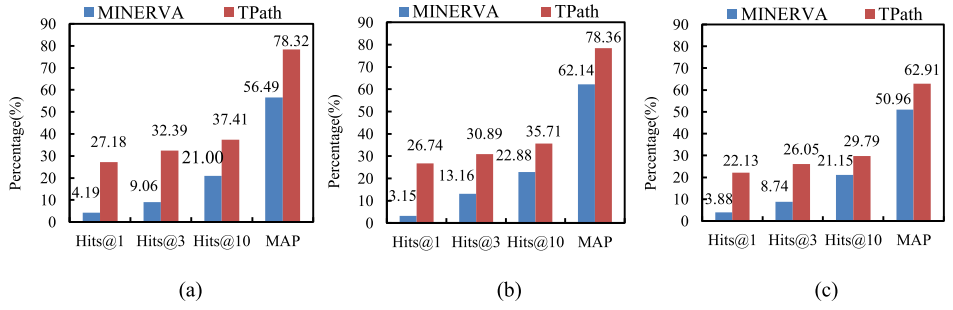

Fig. 5. Performance comparisons of TPath (Red) and MINERVA (Blue) on top-3 relations of GDELT. The (a), (b) and (c) are the evaluation results of the relations '*Make statement, not specified below*', '*Make a visit*' and '*Host a visit*'.

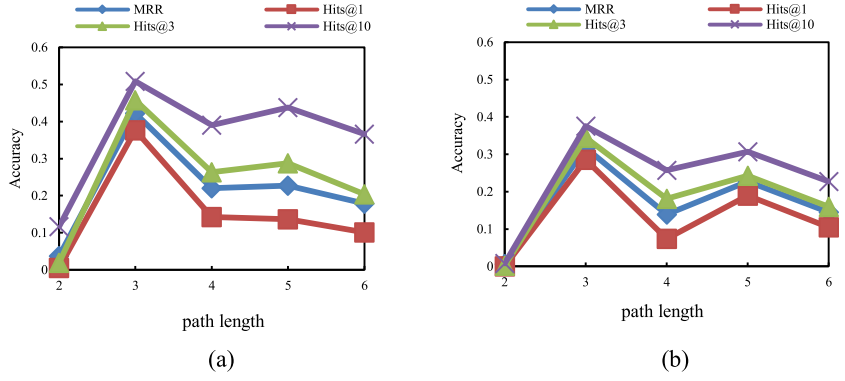

Fig. 6. Performance comparisons on different path lengths. The (a) is on the ICEWS, and the (b) is on the GDELT.

Table 4
Performance comparisons on two temporal datasets.

Method	ICEWS				GDELT			
	Hits@%			MRR%	Hits@%			MRR%
	1	3	10		1	3	10	
TransE	5.03	23.50	43.48	20.98	5.85	19.35	32.30	15.33
DistMult	18.31	29.12	43.19	26.68	24.04	31.43	41.70	30.12
NeuralLP	27.84	45.54	**64.29**	40.04	18.66	32.57	**50.49**	29.14
MINERVA	29.96	35.47	40.29	33.42	0.54	1.62	7.99	3.19
TTransE	4.78	14.18	31.19	13.22	4.83	10.81	21.93	10.56
TA-TransE	2.49	24.02	44.88	17.16	5.85	20.92	33.95	16.18
TA-DistMult	21.98	30.43	43.46	29.19	13.9	23.41	34.03	21.14
TPath	**37.61**	**45.74**	50.82	**42.19**	**28.99**	**34.49**	37.36	**32.08**

common and reasoning models to train event datasets by ignoring the time points, such as TransE [4], NeuralLP [7], DistMult [19], and MINERVA [9]. (2) *temporal knowledge graph reasoning*. We also evaluate with state-of-the-art temporal reasoning models on temporal knowledge graphs, including TTransE [12] and TA-TransE/DisMult [13].

4.1.3. Experimental details

We add the inverse relation of each relation, i.e. for each 4-tuple (e_s, r, e_o, tim), we append (e_o, r^{-1}, e_s, tim) to the graph. With these inverse tuples, the agent can return to the previous entity and undo wrong decision in the TKGs.

We set embedding dimension size as 50. The action embedding is formed by concatenating the entity vector and relation vector. We use one layer LSTM with dimension size of 200. The hidden layer size of MLP (weights W_1 and W_2) is set to 200. We use Adam Optimizer [32] as the stochastic optimizer with default hyper-parameter settings: we set β in the range of {0.05} and λ in the range of {0.02, 0.10, 0.15}, where β is entropy regularization constant and λ is the moving average constant for the REINFORCE baseline.

For the experiments, we set the maximum length $T = 3$. Following convention, we use Mean Reciprocal Ranks (the sum of the inverse of the correct answer ranking), Mean Average Precision (MAP) and Hits@1/3/10 (the percentage of case which the correct answer is in the top 1/3/10 results) as evaluation metrics.

4.2. Evaluation protocol

4.2.1. Performance comparisons

Table 4 summarizes the performances of our model by the baselines on two public datasets: ICEWS and GDELT. The top part presents traditional knowledge graph reasoning methods and bottom part presents temporal knowledge graph reasoning methods.

We find our model TPath outperforms other traditional knowledge graph reasoning methods except NeuralLP (Hits@10: 64.29% vs 50.82 on ICEWS, 50.49% vs 37.36% on GDELT). We find the datasets have a large amount of the same triples after ignoring temporal information in traditional knowledge graph reasoning, which makes them easy to learn. Meanwhile, NeuralLP is not limited by the length of logical rules, which is more adaptable than our model (fixed length of reasoning paths). TPath outperforms MINERVA method significantly and consistently on the two datasets. Especially, for GDELT, our results have increased overall by 30%. It means that temporal information will help the agent reach target entity in multi-hop paths. The results indicate that considering temporal information is effective in temporal reasoning. Next, in the bottom part, we find TPath outperforms all the other temporal knowledge graph reasoning methods on two datasets. Our results have increased overall by 5.94% on ICEWS and 3.33% on GDELT. TTransE, TA-TransE and TA-DistMult mainly support reasoning on implicitly reasoning paths, these results mean that multi-hop paths can be suitable in temporal knowledge graphs.

4.2.2. Analysis of temporal reasoning paths

To analyze the temporal reasoning paths, we separately show a few specific reasoning paths that found by the TPath in Table 5. The Edge denotes two entities are linked by the relation at a certain time. The Temporal Reasoning Path denotes the specific temporal reasoning paths trained by our model. Meanwhile, we take the first edge '*Make an appeal or request, not specified below*:*2018-01-01*' for example. The relation '*Make statement, not specified below*' at time '*2019-05-18*' is unobserved in a TKG. The model will select the temporal reasoning paths in the observed TKG. One of the specific temporal reasoning paths is shown as follows: '*Fight with small arms and light weapons*: *2018-01-11*'→'(*Consult, not specified below*)$^{-1}$: *2018-01-13*'→'*Engage in negotiation*: *2018-01-13*'. It means our model can get the specific temporal reasoning paths, instead of implicitly reasoning vectors.

Moreover, we separately extract top-3 relations and build the special relations subsets of two datasets. In ICEWS, we separately extract the 4-tuples with top-3 relations: '*Make statement, not specified below*', '*Arrest, detain*' and '*Consult, not specified below*'. Besides, we generate the negative examples from test datasets and rank them with positive examples. For a fair comparison, we use the same rank text in the experiments.

For this experiment, we set the length of paths as 3 and the number of iterations as 100. We compare the Hits@1, Hits@3, Hits@10 and mean average precision (MAP) scores for each reasoning task. Figs. 4 and 5 separately summarize the performance results of our model with MINERVA on ICEWS and GDELT. For the overall Hits@1/3/10 and MAP shown in Figs. 4 and 5, our model significantly outperforms MINERVA method, which validates the strong reasoning ability in TKGs. In Figs. 4 and 5, we can see that: (1) our results have increased overall by average 25% on the Hits@1/3/10. It indicates that our model can enhance the performance of reasoning paths in special relation datasets. (2) TPath outperforms baseline methods on MAP. It means that temporal information is the vital role in multi-hop path. Since MINERVA fails to use the temporal information in the reasoning paths, it generally performs worse than our model.

Our model can train different lengths of reasoning paths. For evaluating the performance of different length on experimental results, we examine the effectiveness of various lengths. Fig. 6 shows the performance of our model under various length of reasoning paths over two datasets. From Fig. 6, we observe that: (1) the results significantly outperform other lengths of temporal reasoning paths in our model, when the length of temporal reasoning paths is 3. (2) when the length of temporal reasoning paths is 2, the results do not achieve expected performance (not 0). We find that there are not enough temporal reasoning paths (length of paths: 2) between entities in two datasets, our model hardly achieves expected results. (3) when the path length is from 4 to 6, we train separately for each length and the results gradually decreases. As a result, we predict that the results will decrease when the path length exceeds 6.

Experimental results show that TPath outperforms current methods on Hits@1, Hits@3 and MRR over ICEWS and GDELT, but our results are slightly lower than NeuralLP on Hits@10 over two datasets and DistMult on Hits@10 over GDELT in Table 4. Then, Table 5 shows some specific temporal reasoning paths, which means that our model can get the specific temporal reasoning paths, instead of implicitly reasoning vectors. Next, we compare our model with MINERVA method for multi-hop reasoning in KG, our model significantly outperforms it at top-3 relations in datasets. In addition, Fig. 6 shows that out model can train multi-hop reasoning at different fixed lengths, and get the best performance, when the length is 3.

Table 5
Part examples of specific temporal reasoning paths found by our model. The first edge (relation and time) comes from the GDELT dataset. The others are from ICEWS. $^{-1}$ denotes the inverse relation.

Edges	Temporal Reasoning Paths
Make an appeal or request, not specified below: 2018-01-01	Fight with small arms and light weapons: 2018-01-11→(Consult, not specified below)$^{-1}$: 2018-01-13→Engage in negotiation: 2018-01-13
Make statement, not specified below: 2019-05–18	Accuse of human rights abuses: 2018-10-28→Accede to requests or demands for political reform, not specified below: 2019-06-01
Consult, not specified below: 2019-05-01	Host a visit: 2019-05-01→ Host a visit: 2018-11-27→Make a visit: 2018-11-27

5. Conclusion

In this paper, we propose TPath, which takes temporal information into consideration and selects specific multi-hop reasoning paths in the temporal knowledge graphs. What is more, we propose a policy network that can train the agent to learn temporal multi-hop reasoning paths. Considering the diversity of temporal reasoning paths, we propose a reward function. Moreover, we analyze the temporal reasoning paths, and achieve state-of-the-art results on multi-hop reasoning tasks. However, the fixed length of reasoning paths may be a limitation, which needs to be further improved.

For the future work, we expect to investigate the possibility of training temporal reasoning paths of various lengths simultaneously. Instead of fixed lengths in each experiment, the model can choose various length of reasoning paths according to designing rewards. In addition, multi-hop path reasoning has a wide range of applications in temporal knowledge graphs, including recommendation systems, intelligent retrieval, etc. We also consider utilizing the spatial information in TKGs to further improve the performance of reasoning paths.

CRediT authorship contribution statement

Luyi Bai: Conceptualization, Methodology, Formal analysis, Funding acquisition, Writing - review & editing. **Wenting Yu:** Investigation, Formal analysis, Writing - original draft. **Mingzhuo Chen:** Validation, Writing - original draft. **Xiangnan Ma:** Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (61402087), the Natural Science Foundation of Hebei Province, China (F2019501030), the Natural Science Foundation of Liaoning Province, China (2019-MS-130), the Key Project of Scientific Research Funds in Colleges and Universities of Hebei Education Department, China (ZD2020402), the Fundamental Research Funds for the Central Universities, China (N2023019), and in part by the Program for 333 Talents in Hebei Province, China (A202001066).

The authors would also like to express their gratitude to the anonymous reviewers for providing very helpful suggestions.

References

[1] C. Xiong, S. Merity, R. Socher, Dynamic memory networks for visual and textual question answering, in: Proceedings of the 33nd International Conference on Machine Learning, 2016, pp. 2397–2406, http://proceedings.mlr.press/v48/xiong16.html.

[2] G.H. Nguyen, J.B. Lee, R.A. Rossi, N.K. Ahmed, E. Koh, S. Kim, Continuous-time dynamic network embeddings, in: Proceedings of Companion of the The Web Conference 2018, 2018, pp. 969–976, https://doi.org/10.1145/3184558.3191526.

[3] L.P. Heck, D. Hakkani-Tür, G. Tür, Leveraging knowledge graphs for web-scale unsupervised semantic parsing, in: Proceedings of the 14th Annual Conference of the International Speech Communication Association, 2013, pp. 1594–1598, http://www.isca-speech.org/archive/interspeech_2013/i13_1594.html.

[4] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of Advances in Neural Information Processing Systems, 2013, pp . 2787–2795, http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.

[5] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the Twenty-Eighth Conference on Artificial Intelligence, 2014, pp. 1112–1119, http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[6] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Proceedings of the Twenty-Ninth Conference on Artificial Intelligence, 2015, pp. 2181–2187, http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[7] F. Yang, Z. Yang, W.W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in: Proceedings of Advances in Neural Information Processing Systems, 2017, pp. 2319–2328, http://papers.nips.cc/paper/6826-differentiable-learning-of-logical-rules-for-knowledge-base-reasoning.

[8] W. Xiong, T. Hoang, W.Y. Wang, Deeppath: a reinforcement learning method for knowledge graph reasoning, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 564–573, https://doi.org/10.18653/v1/d17-1060.

[9] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, A. McCallum, Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning, in: Proceedings of the 6th International Conference on Learning Representations, 2018, https://openreview.net/forum?id=Syg-YfWCW.

[10] M. Hildebrandt, J.A.Q. Serna, Y. Ma, M. Ringsquandl, M. Joblin, V. Tresp, Reasoning on knowledge graphs with debate dynamics, in: Proceedings of the Thirty-Fourth Conference on Artificial Intelligence, 2020, pp. 4123–4131, https://aaai.org/ojs/index.php/AAAI/article/view/6600.

[11] S.S. Dasgupta, S.N. Ray, P.P. Talukdar, Hyte: hyperplane-based temporally aware knowledge graph embedding, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2001–2011, https://doi.org/10.18653/v1/d18-1225.

[12] J. Leblay, M.W. Chekol, Deriving validity time in knowledge graph, in: Proceedings of Companion Proceedings of the The Web Conference 2018, 2018, pp. 1771–1776, https://doi.org/10.1145/3184558.3191639.

[13] A. García-Durán, S. Dumancic, M. Niepert, Learning sequence encoders for temporal knowledge graph completion, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 4816–4821, https://www.aclweb.org/anthology/D18-1516.

[14] R. Trivedi, H. Dai, Y. Wang, L. Song, Know-evolve: Deep temporal reasoning for dynamic knowledge graphs, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 3462–3471, http://proceedings.mlr.press/v70/trivedi17a.html.

[15] W. Jin, C. Zhang, P.A. Szekely, X. Ren, Recurrent event network for reasoning over temporal knowledge graphs, 2019, arXiv preprint arXiv:1904.05530.

[16] Y. Ma, V. Tresp, E.A. Daxberger, Embedding models for episodic knowledge graphs, J. Web. Semant. 59 (2019) http://dx.doi.org/10.1016/j.websem.2018.12.008.

[17] T. Jiang, T. Liu, T. Ge, L. Sha, B. Chang, S. Li, Z. Sui, Towards time-aware knowledge graph completion, in: Proceedings of the 26th International Conference on Computational Linguistics, 2016, pp. 1715–1724, https://www.aclweb.org/anthology/C16-1161.

[18] Z. Han, Y. Wang, Y. Ma, S. Günnemann, V. Tresp, Graph hawkes network for reasoning on temporal knowledge graphs, 2020, arXiv preprint arXiv:2003.13432.

[19] B. Yang, W. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: Proceedings of the 3rd International Conference on Learning Representations, 2014, http://arxiv.org/abs/1412.6575.

[20] G. Ji, K. Liu, S. He, J. Zhao, Knowledge graph completion with adaptive sparse transfer matrix, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 985–991, http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11982.

[21] B. Shi, T. Weninger, ProjE: embedding projection for knowledge graph completion, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 1236–1242, http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14279.

[22] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: Proceedings of International Conference on Machine Learning, 2016, pp. 2071–2080, http://proceedings.mlr.press/v48/trouillon16.html.

[23] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: a survey of approaches and applications, IEEE Trans. Knowl. Data Eng. 29 (2017) 2724–2743, http://dx.doi.org/10.1109/TKDE.2017.2754499.

[24] X.V. Lin, R. Socher, C. Xiong, Multi-hop knowledge graph reasoning with reward shaping, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 3243–3253, https://doi.org/10.18653/v1/d18-1362.

[25] H. Wang, S. Li, R. Pan, M. Mao, Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019, pp . 2623–2631, https://doi.org/10.18653/v1/D19-1264.

[26] C. Gutiérrez, C.A. Hurtado, A.A. Vaisman, Introducing.time.into. rdf, Introducing time into rdf, IEEE Trans. Knowl. Data Eng. 19 (2007) 207–218, http://dx.doi.org/10.1109/TKDE.2007.34.

[27] A. Pugliese, O. Udrea, V.S. Subrahmanian, Scaling RDF with time, in: Proceedings of the 17th International Conference on World Wide Web, 2008, pp. 605–614, https://doi.org/10.1145/1367497.1367579.

[28] A. Zimmermann, N. Lopes, A. Polleres, U. Straccia, A general framework for representing, reasoning and querying with annotated semantic web data, J. Web Semant. 11 (2012) 72–95, http://dx.doi.org/10.1016/j.websem.2011.08.006.

[29] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (1992) 229–256, http://dx.doi.org/10.1007/BF00992696.

[30] K. Leetaru, P.A. Schrodt, GDELT: Gobal Data on Events, Location, and Tone, ISA Annual Convention, 2013.

[31] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, M. Ward, ICEWS Coded event data, 2015.

[32] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proceedings of 3rd International Conference on Learning Representations, 2015, http://arxiv.org/abs/1412.6980.