Neural Networks 160 (2023) 192–201

Contents lists available at ScienceDirect

Neural Networks

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

Adaptive pseudo-Siamese policy network for temporal knowledge prediction

Pengpeng Shao [a], Tong Liu [a], Feihu Che [a], Dawei Zhang [a], Jianhua Tao [b,c,*]

[a] *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China*
[b] *Department of Automation, Tsinghua University, China*
[c] *School of Artificial Intelligence, University of Chinese Academy of Sciences, China*

A R T I C L E I N F O

Article history:
Received 21 April 2022
Received in revised form 10 December 2022
Accepted 4 January 2023
Available online 14 January 2023

Keywords:
Temporal knowledge graphs
Prediction
Reinforcement learning

A B S T R A C T

Temporal knowledge prediction is a crucial task for early event warning, which has gained increasing attention recently. It aims to predict future facts based on relevant historical facts using temporal knowledge graphs. There are two main difficulties associated with the prediction task: from the perspective of historical facts, modeling the evolutionary patterns of facts to accurately predict the query and from the query perspective, handling the two cases where the query contains seen and unseen entities in a unified framework. Driven by these two problems, we propose a novel adaptive pseudo-Siamese policy network for temporal knowledge prediction based on reinforcement learning. Specifically, we design the policy network in our model as a pseudo-Siamese network consisting of two sub-policy networks. In the sub-policy network I, the agent searches for the answer to the query along the entity-relation paths to capture static evolutionary patterns. In sub-policy network II, the agent searches for the answer to the query along relation-time paths to deal with unseen entities. Moreover, we develop a temporal relation encoder to capture the temporal evolutionary patterns. Finally, we design a gating mechanism to adaptively integrate the results of the two sub-policy networks to help the agent focus on the destination answer. To assess the performance of our model, we conduct link prediction on four benchmark datasets, and extensive experimental results demonstrate that our method achieves considerable performance compared with existing methods.

1. Introduction

Knowledge graphs (KGs), which store numerous static triple facts in the form of (*subject*, *relation*, *object*), have been widely used in many natural language processing applications, such as question answering (Dong, Wei, Zhou, & Xu, 2015), recommendation systems (Koren, Bell, & Volinsky, 2009), and retrieval systems (Xiong & Callan, 2015). However, owing to the increased popularity of triple facts, many facts present dynamic attributes which only hold in a specific period or at a certain point in time. Therefore, temporal knowledge graphs (TKGs) and corresponding temporal knowledge reasoning tasks have received increasing academic attention recently.

TKG reasoning has two forms: interpolation and extrapolation. On a TKG with timestamps varying between $[t_0, t_T]$, interpolation TKG reasoning aims to infer the answer to a query within time $t \in [t_0, t_T]$. While the extrapolation setting employs only historical knowledge to predict future facts for time t ($t > t_T$), we adopt the extrapolation setting temporal knowledge prediction here. Furthermore, this method is a crucial task for early event warning. In this study, we focus on addressing problems in temporal knowledge prediction.

Recently, many temporal knowledge prediction methods (Jin, Qu, Jin, & Ren, 2020; Trivedi, Farajtabar, Biswal, & Zha, 2019; Zhu, Chen, Fan, Cheng, & Zhang, 2021) based on embedding have been presented. Moreover, only a few studies (Li et al., 2021; Sun, Zhong, Ma, Han, & He, 2021) have employed models based on reinforcement learning (RL) to predict temporal knowledge. Although embedding-based approaches are convenient for modeling knowledge with considerable performance, they fail to consider the symbolic compositionality of KG relations, which limits their application to more complex reasoning tasks. RL-based approaches can allow the agent to obtain the answer to the query by traversing the path it interacts with a complex environment. Thus, they possess powerful adaptability. However, there are two main difficulties experienced by an agent in the traversing process. The first is modeling the evolutionary patterns of historical facts to accurately predict future facts. Queries are usually uncertain, containing seen or unseen entities. Consequently,

* Corresponding author at: Department of Automation, Tsinghua University, China.
E-mail addresses: shaopengpeng2019@ia.ac.cn (P. Shao), jhtao@tsinghua.edu.cn (J. Tao).

https://doi.org/10.1016/j.neunet.2023.01.004
0893-6080/

the second difficulty is addressing the two cases where the query contains seen and unseen entities in a unified framework. Many related studies have been conducted to address these two problems. In embedding-based approaches, to model the temporal pattern of facts, Know-Evolve (Trivedi, Dai, Wang, & Song, 2017) and its extension DyRep (Trivedi et al., 2019) model the occurrence of facts as a temporal point process, Re-NET (Jin et al., 2020) models the occurrence of a fact as a probability distribution conditioned on temporal sequences of past knowledge graphs. To cope with the second obstacle, XERTE (Han, Chen, Ma and Tresp, 2021) develops a temporal relational attention mechanism and predicted the relevant substructure of TKGs by propagating attention. In addition, TITer (Sun et al., 2021) uses the inductive mean representation of the trained entities with the same co-occurrence query relation to represent unseen entities, obtaining a better entity embedding distribution and answer score distribution. However, the unseen entity does not exist in historical facts and has no historical neighbors. Thus, for a query containing an unseen entity, XERTE (Han, Chen et al., 2021) cannot propagate attention to entities of previous facts and TITer (Sun et al., 2021) fails to select candidate temporal actions to search for the answer.

In this study, we propose a novel RL-based temporal knowledge prediction model to address these two problems. Specifically, we design the policy network of the proposed model as a pseudo-Siamese network consisting of two sub-policy networks. Sub-policy network I is designed to capture the static evolutionary patterns so that agent I can search for the answer to the query along the entity-relation paths instead of the time information. In sub-policy network II, to deal with unseen entities, we add semantic edges to the TKGs so that agent II can search for the answer to the query along the relation-time paths instead of the entity information. Furthermore, we develop a temporal relation encoder to capture the temporal evolutionary patterns. Finally, to ensure that the two sub-policy networks compensate for each other, we employ a gating mechanism to adaptively integrate the results of the two sub-policy networks to help the agent focus on the destination entity. Extensive experimental results indicate that our method performs considerably better than the existing approaches, which underlines the effectiveness and superiority of our method.

The contributions of this study are listed as follows:

- We develop a novel RL-based temporal knowledge prediction approach to address the two cases where the query contains seen and unseen entities in a unified framework.
- We advocate the importance of developing a more comprehensive modeling framework regarding the evolutionary patterns of the facts. Therefore, we develop one sub-policy network to capture static evolutionary patterns, while designing another sub-policy network and temporal relation encoder to model temporal evolutionary patterns. Finally, we employ a gating mechanism to adaptively integrate the results of two sub-policy networks.
- We propose a new edge type to establish the relationship between the query containing the unseen entity and historical facts. Moreover, we design a novel method to handle the special type of query.
- The experimental studies on four TKG datasets demonstrate that our method achieves state-of-the-art performance.

2. Related research

2.1. Static KG reasoning

Static KG reasoning methods can be roughly grouped into three categories: embedding-based, RL-based, and logic rule-based. Embedding-based approaches are the most popular owing to their high efficiency and outstanding effectiveness, which aim to project the entities and relations in KGs into a vector space and represent them as low-dimensional embeddings. This type of method is broadly classified into three paradigms: (i) Translational distance-based models (Bordes, Usunier, Garcia-Durán, Weston, & Yakhnenko, 2013; Ji, He, Xu, Liu, & Zhao, 2015; Wang, Zhang, Feng, & Chen, 2014). (ii) Tensor factorization-based models (Balažević, Allen, & Hospedales, 2019; Nickel, Tresp, & Kriegel, 2011; Trouillon, Welbl, Riedel, Gaussier, & Bouchard, 2016; Yang, Yih, He, Gao, & Deng, 2015). (iii) Neural network-based models (Dettmers, Pasquale, Pontus, & Riedel, 2018; Schlichtkrull et al., 2017; Vashishth, Sanyal, Nitin, & Talukdar, 2020). Although embedding-based approaches are simple and convenient for modeling knowledge with considerable performance, they are less sensitive to reasoning distance and ignore the logical rules between relations and paths, which limits their application in more complex reasoning tasks and limits their interpretability. RL-based approaches allow the agent to find the answer to the query by traversing the path on the KGs, which enables them to learn reasoning rules from relation paths. DeepPath (Xiong, Hoang, & Wang, 2017) is the first multi-hop reasoning work based on RL, which aims to search for generic representative paths between pairs of entities. MINERVA (Das et al., 2018) utilizes the history path to help the agent searching for the answer entities of a particular KG query in an end-to-end fashion. Based on MINERVA, M-Walk (Shen, Chen, Huang, Guo, & Gao, 2018) and Multi-HopKG (Lin, Socher, & Xiong, 2018) adopt a Monte Carlo tree search and pre-trained embedding model to overcome the problem of sparse rewards, respectively. However, none of these methods can model the evolutionary patterns of the facts in TKGs.

2.2. Temporal KG reasoning

According to the relationship between query time points and training temporal scope, TKG reasoning can be broadly classified into two forms: interpolation and extrapolation. Interpolation reasoning (Dasgupta, Ray, & Talukdar, 2018; García-Durán, Dumancic, & Niepert, 2018; Goel, Kazemi, Brubaker, & Poupart, 2020; Jiang et al., 2016; Lacroix, Obozinski, & Usunier, 2020) aims to infer new facts at historical timestamps by employing historical and future information. Corresponding extrapolation reasoning aims to predict future facts based only on historical information, accordingly, we call this reasoning task temporal knowledge prediction. Considering its great practical value, especially in early event warning, many temporal knowledge prediction studies have been presented recently. Know-Evolve (Trivedi et al., 2017) and its extension DyRep (Trivedi et al., 2019) employ a temporal point process to model the occurrence of temporal facts in TKGs. Re-NET (Jin et al., 2020) models the occurrence of temporal facts as probability distributions conditioned on the temporal sequences of past knowledge graphs. To model repetitive facts, CyGNet (Zhu et al., 2021) designs a copy mode to learn from the known facts that appeared in history. XERTE (Han, Chen et al., 2021) develops a temporal relational attention mechanism and predicts future facts by focusing on the relevant substructure of TKGs. CluSTeR (Li et al., 2021) and TITer (Sun et al., 2021) are two prominent temporal knowledge prediction approaches based on RL. CluSTeR (Li et al., 2021) first employs an agent to induce relevant clues from historical facts and then adopts an embedding model to deduce answers from the obtained clues. TITer (Sun et al., 2021) forces the agent to travel on historical knowledge graph snapshots and directly find the answer to the query. Furthermore, to deal with unseen entities in the query, TITer presents an inductive mean representation method to obtain a more reasonable initial embedding for unseen entities, implicitly improving the inductive inference ability of the model. However, it fails to search for an answer to a query with an unseen entity intuitively through self-loop edges. In this study, we propose a novel method to address this problem.

2.3. Pseudo-Siamese network

Siamese network is a class of neural network architectures containing two or more subnetworks with the same configuration, i.e., the same parameters and architectures, and it is also a type of metric learning model used to measure two or more input degrees of similarity. A Siamese network (Bromley, Guyon, LeCun, Säckinger, & Shah, 1993) is first proposed to verify whether the signature on a check is consistent with the reserved signature of the bank. Recently, Siamese networks have been widely used in computer vision, such as in object tracking (Cen & Jung, 2018; Li et al., 2019) and face recognition (Wu, Xu, Zhang, Yan, & Ma, 2017) tasks. Owing to their weight sharing capabilities, Siamese networks limit the difference between their inputs to a certain extent. In general, for inputs with large differences, such as images and text, we must use a pseudo-Siamese network. A pseudo-Siamese network has a similar network architecture to a Siamese network. However, they differ in that their subnetworks do not share weights, that is, the subnetworks are different in each case. For example, TPSN (TPSN, 2022) introduces a novel triple pseudo-Siamese network to encode the welding pool image, sound, current, and voltage to detect different welding defects or actions. PSGMN (Wu, Chen, He, & Jiang, 2022) employs a pseudo-Siamese network structure to learn the similarity between the 2-D image features and the 3-D mesh model of on object. In contrast, in this study, we design a novel pseudo-Siamese network using the RL framework to encode similar inputs for temporal knowledge prediction.

3. The proposed model

3.1. Notations and problem formulation

The temporal facts in TKGs are defined by quadruples $(s, r, o, t) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$, where $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{T}$ denote finite sets of entities, relations, and timestamps, respectively. TKGs can also be represented as graph snapshots over time, namely $\mathcal{G} = \{\mathcal{G}_{t_1}, \mathcal{G}_{t_2}, \ldots, \mathcal{G}_{t_T}\} = \{(s, r, o, t_l)\}_{t_l=1}^{|\mathcal{T}|}$, where $s, o \in \mathcal{E}$, $r \in \mathcal{R}$. Each $\mathcal{G}_{t_l}$ is static multi-relational KGs at time t_l. Thus, TKG is a collection of event facts at different timestamps, where the timestamps are arranged in ascending order.

The task of temporal knowledge prediction is to conduct a link prediction for future queries (i.e., $(s_q, r_q, ?, t_q)$) based on the historical facts $\mathcal{G}_{0:t_q-1}$. However, not all historical facts are relevant to the query in $\mathcal{G}_{0:t_q-1}$. Thus, the historical facts that are directly relevant to the query are represented as $E_s = \{(s_q, r, o, t_l) \in \mathcal{G}_{t_l} | t_l < t_q\}$, which have the same subject entity s_q as the query. In addition, we denote the historical facts that are semantically relevant to the query as $E_r = \{(s, r_q, o, t_l) \in \mathcal{G}_{t_l} | t_l < t_q\}$, that is, the facts in E_r have the same relation r_q as the query. The relation of an entity reflects the roles of the entity. Thus, they have similar semantic meanings. More specifically, our task is to employ historical facts related to the query to answer a given query.

3.2. Reinforcement learning system

The temporal knowledge prediction task is formulated as a finite-horizon sequential decision-making problem, and the reinforcement learning system is described as a deterministic Partially Observed Markov Decision Process (POMDP). The system consists of an agent and environment. The agent starts from the query entity s_q, follows a relation path in $\mathcal{G}$ according to its policy, and stops at the entity regarded as the correct answer to the query. The environment is described in detail as follows:

States. Each state can be represented as $s_k = (e_k, t_k, s_q, r_q, o_q, t_q) \in \mathcal{S}$, where $\mathcal{S}$ denotes the state space, e_k denotes the entity visited at step k, t_k denotes the timestamp of the action taken in the previous step, (s_q, r_q, t_q) can be seen as the global context shared by all states for a given query, and o_q is the answer. As the agent starts from the query node s_q and the default initial action is a self-loop action, the initial state is $s_0 = (s_q, t_q, s_q, r_q, o_q, t_q)$.

Observations. When searching for answers, the agent cannot observe all the states of the environment, except for the current location and query information, the answer o_q is invisible. Therefore, the observation function is defined as: $\mathcal{O}((e_k, t_k, s_q, r_q, o_q, t_q)) = (e_k, t_k, s_q, r_q, t_q)$.

Actions. Given the current state $(e_{k-1}, t_{k-1}, s_q, r_q, o_q, t_q)$, the set of possible actions $\mathcal{A}_k \in \mathcal{A}$ at step k consists of the outgoing edges of e_{k-1}, where $\mathcal{A}$ is the action space. Here, we describe the three types of outgoing edges of entity e_{k-1} at step k as follows: (i) Self-loop Edges: The self-loop edges (i.e., $(e_{k-1}, r_{self}, e_{k-1}, t_{k-1})$) not only allow the agent to stay in a place but also allow it to stop adaptively when it searches for a certain number of steps or states that it has found the final answer. (ii) Temporal Edges: If there are related historical facts (e_k, r, o, t_l) for the current entity e_k at time t_k, we build the temporal edge between $e_k^{t_k}$ and o^{t_l} using the relation r. For example, as shown on the left of Fig. 1, in the initial state, when the query contains the seen entity s_q and the related historical facts (s_q, r_1, o_1, t_0) are observed, we build the temporal edges $(s_q^{t_q}, r_1, o_1, t_0)$. (iii) Semantic Edges: When the query contains the unseen entity s_q, we cannot find the related historical facts (s_q, r, o, t_l) for the entity s_q. Thus, in this case, the agent can only stay in a place through the self-loop edges. Fortunately, in the initial state, there exist historical facts (s_v, r_q, o, t_l) that are related to the query relation r_q, as is (s_v, r_q, o_8, t_1) shown on the right of Fig. 1. Therefore, we build the semantic edges (s_q, r_q, o_8, t_1). Note that semantic edges only exist in the first step to bridge the gap between the query and historical facts. In summary, when the query contains a seen entity, we can also build semantic edges through the relation so that the agent can walk along the above three types of edges in the first step, and then it walks along temporal edges or self-loop edges to find answers. When the query contains an unseen entity, the agent can only walk along the semantic edges in the first step and arrive at the seen entity, and then it walks along the temporal edges or self-loop edges to find answers. Therefore, $\mathcal{A}_0 = \{(r_{self}, s_q, t_q)\} \cup \{(r', e', t') | (s_q, r', e', t') \in \mathcal{G}_{0:t_q-1}, t' < t_q\} \cup \{(r_q, e', t') | (s_u, r_q, e', t') \in \mathcal{G}_{0:t_q-1}, (s_q, r', e', t') \notin \mathcal{G}_{0:t_q-1}, t' < t_q\}$. $\mathcal{A}_k = \{(r_{self}, e_k, t_k)\} \cup \{(r', e', t') | (e_k, r', e', t') \in \mathcal{G}_{0:t_k-1}, t' < t_k\}$, $k > 0$. Moreover, given action (r, e, t), we denote (r, e) and (r, t) as static and temporal actions, respectively.

Transition. Given the current location s_{k-1}, once action a_k is determined, the current location s_{k-1} is transferred to the next location s_k. Here, the transition function δ: $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is defined by $\delta(s_{k-1}, a_k) = s_k = (e_k, t_k, s_q, r_q, o_q, t_q)$. Therefore, the query information and answer remain unchanged, and the action becomes $a_k = (r_k, e_k, t_k)$.

Rewards. After K-hop navigation, the agent reaches its final state $s_K = (e_K, t_K, s_q, r_q, o_q, t_q)$. Then, the agent receives a terminal reward of 1 if $e_K = o_q$ and 0 otherwise. The reward mechanism is formally defined as : $R(s_K) = \mathbb{I}(e_K == o_q)$.

3.3. Policy network

The policy network π_θ is parameterized using the action path history and global context (query information). Moreover, it models the action of the agent in a continuous space, thus, we need to calculate the similarity between the continuous value output by the policy network and the candidate actions to determine the probability of selecting each candidate action. To deal with

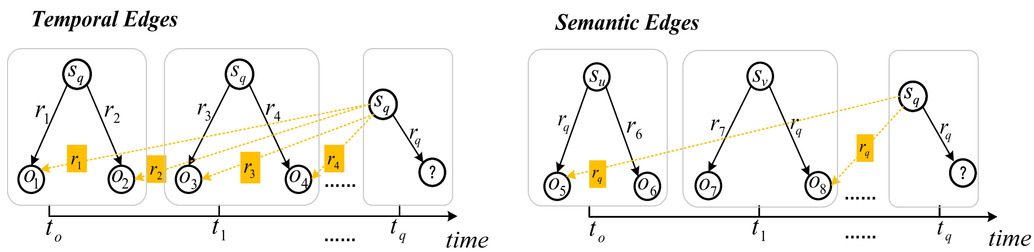

Fig. 1. Illustration of the temporal edges and semantic edges.

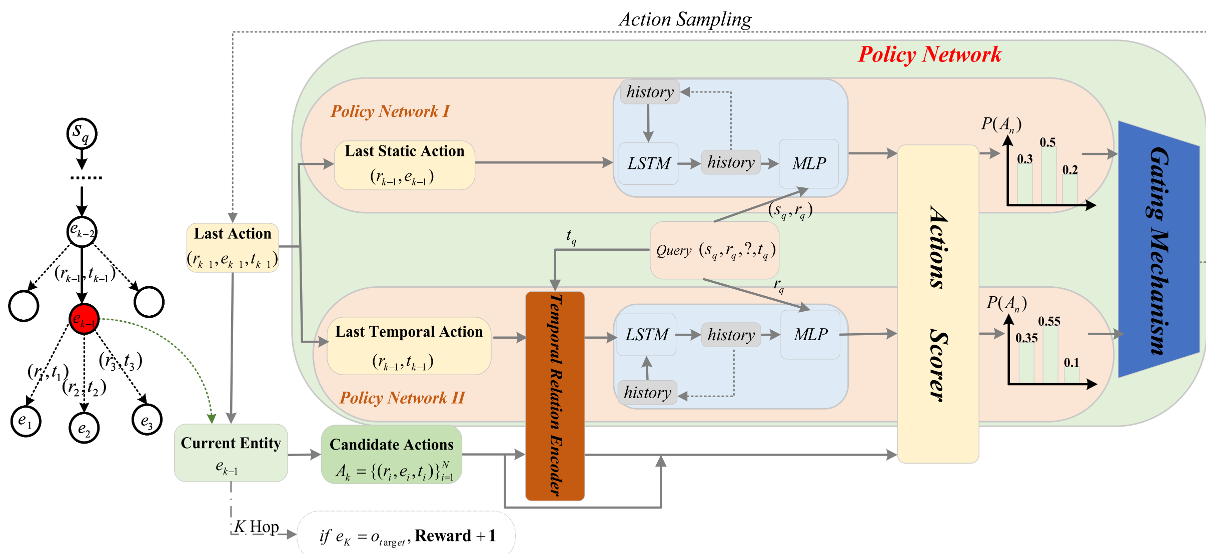

Fig. 2. An overview of our model. Given the query $(s_q, r_q, ?, t_q)$, the agent starts from query subject s_q, and then samples an outgoing edge (action) to walk and reach the next entity through the policy network π_θ. For example, after k-1 hop navigation, the agent arrives at e_{k-1} (red node). Subsequently, we can obtain a historical path consisting of k-1 historical actions. Then, we use two sub-policy networks to encode the two variant paths of this historical path and obtain the scores of each candidate action, respectively. Finally, we employ a gating mechanism to obtain the final scores of each candidate action adaptively and sample the next action. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the cases of seen and unseen entities simultaneously in a unified framework and model the evolutionary patterns of the facts, we designed two similar sub-policy networks to encode the path and score the candidate actions, respectively. Subsequently, we employ a gating mechanism to obtain a final score for each candidate action. An overview of our model is presented in Fig. 2.

3.3.1. Policy network I

The search history consisting of k actions is stated as $h_k = \big((s_q, t_q), (r_{self}, s_q, t_q), (r_1, e_1, t_1), \ldots, (r_k, e_k, t_k)\big)$. To address the general case in which the query contains a seen entity, we use the policy network to encode the action path history h_k as usual. However, to model the static and temporal evolutionary patterns, here, we use one policy network (policy network I) to encode the static action path history $h_k^s = \big((s_q, t_q), (r_{self}, s_q), \ldots, (r_k, e_k)\big)$ and employ another policy network (policy network II) to encode the temporal action path $h_k^t = \big((s_q, t_q), (r_{self}, t_q), (r_1, t_1), \ldots, (r_k, t_k)\big)$. With this design, we can model both static and temporal evolutionary patterns in the general case that the query contains a seen entity. The static action path h_k^s is encoded by policy network I as follows:

$$\begin{aligned}
\mathbf{h}_0^s &= \mathrm{LSTM}(0, [\mathbf{r}_{self}, \mathbf{s}_q]) \\
\mathbf{h}_1^s &= \mathrm{LSTM}(\mathbf{h}_0^s, [\mathbf{r}_1, \mathbf{e}_1]) \\
&\cdots \\
\mathbf{h}_k^s &= \mathrm{LSTM}(\mathbf{h}_{k-1}^s, [\mathbf{r}_k, \mathbf{e}_k])
\end{aligned} \tag{1}$$

Then, we calculate the probability of each candidate action in policy network I. First, we denote N candidate actions at step k as $A_k = \{(r_i, e_i, t_i)\}_{i=1}^N$. Similarly, the static candidate actions are $A_k^s = \{(r_i, e_i)\}_{i=1}^N$ and temporal candidate actions are represented by $A_k^t = \{(r_i, t_i)\}_{i=1}^N$. Thus, the probability distribution $\phi_\theta^s(a_k|s_{k-1})$ of the candidate actions in the state s_{k-1} is calculated as follows:

$$\phi_\theta^s(a_k|s_{k-1}) = \mathbf{A}_k^s \mathbf{W}_2^s \mathrm{ReLU}(\mathbf{W}_1^s[\mathbf{h}_{k-1}^s, \mathbf{s}_q, \mathbf{r}_q]) \tag{2}$$

where $\mathbf{W}_1^s$ and $\mathbf{W}_2^s$ are the training parameters of MLP.

3.3.2. Policy network II

As mentioned above, policy network II is used to encode the temporal action path history $h_k^t = \big((s_q, t_q), (r_{self}, t_q), (r_1, t_1), \ldots, (r_k, t_k)\big)$ to model the temporal evolutionary patterns in the case that the query contains a seen entity. Here, note that when the query contains an unseen entity, we bridge the gap between the query and historical facts through the semantic edges defined by the query relation. Thus, we need to design a policy network to encode the temporal action path history $h_k^t = \big((s_q, t_q), (r_{self}, t_q), (r_1, t_1), \ldots, (r_k, t_k)\big)$ instead of entity information. Therefore, encoding the temporal action path h_k^t can not only model the evolutionary patterns in the general case but also address the special case in which the query contains an unseen entity.

Here, as the time difference between the facts can better capture the temporal evolutionary patterns and express the dynamics of the facts, we first employ a temporal relation encoder to encode h_k^t to obtain the temporal relation path $\big((\mathbf{s}_q, \mathbf{t}_q), \mathbf{r}_{self}^{t_q},$

Table 1
Dataset statistics.

Datasets	N_{entity}	$N_{relation}$	$N_{timestamps}$	Time granularity	N_{train}	$N_{validation}$	N_{test}
ICEWS2014	7128	230	365	24 h	63 685	13 823	13 222
ICEWS2018	23 033	256	304	24 h	373 018	45 995	49 545
WIKI	12 554	24	232	1 year	539 286	67 538	63 110
YAGO	10 623	10	189	1 year	161 540	19 523	20 026

Table 2
Number of unseen entities and quadruples containing unseen entities in the test set.

Datasets	N_{entity}^{unseen}	$N_{quad}^{unseen\ entity}$	$N_{quad}^{unseen\ object}$	$N_{quad}^{unseen\ subject}$	$N_{quad}^{unseen\ subject\&object}$
ICEWS2014	496	862 (6.52%)	438	497	73
ICEWS2018	1140	1948 (3.93%)	975	1050	77
WIKI	2968	27 079 (42.91%)	11 086	22 967	6974
YAGO	540	1609 (8.03%)	1102	873	366

$\mathbf{r}_1^{t_1}, \ldots, \mathbf{r}_k^{t_k}$). Subsequently, we use policy network II to encode the obtained temporal relation path. Formally, the temporal action $\mathbf{r}_k^t$ is encoded by the temporal relation encoder, as follows:

$$\mathbf{r}_k^t = (\sigma(\mathbf{w}\Delta_t) + \mathbf{b}) * \mathbf{r}_k \tag{3}$$

where $\mathbf{w}$ and $\mathbf{b}$ are the training parameters and σ is an activation function. Δ_t is the timestamp difference between the temporal facts, and $\Delta_t = t_q - t_k$. Likewise, we can learn the temporal relation $\mathbf{A}_k^{tr} = \{\mathbf{r}_i^{t_i}\}_{i=1}^N$ of the candidate temporal actions A_k^t through Eq. (3) to score the candidate temporal actions.

Then, we use policy network II to encode the temporal relation path as follows:

$$\begin{aligned}
\mathbf{h}_0^t &= \mathrm{LSTM}(0, \mathbf{r}_{self}^{t_q}) \\
\mathbf{h}_1^t &= \mathrm{LSTM}(\mathbf{h}_0^t, \mathbf{r}_1^{t_1}) \\
&\cdots \\
\mathbf{h}_k^t &= \mathrm{LSTM}(\mathbf{h}_{k-1}^t, \mathbf{r}_k^{t_k})
\end{aligned} \tag{4}$$

Afterward, the probability distribution $\phi_\theta^t(a_k|s_{k-1})$ of the candidate actions in the state s_{k-1} is calculated as follows:

$$\phi_\theta^t(a_k|s_{k-1}) = \mathbf{A}_k^{tr}\mathbf{W}_2^t\mathrm{ReLU}(\mathbf{W}_1^t[\mathbf{h}_{k-1}^t, \mathbf{r}_q]) \tag{5}$$

where $\mathbf{W}_1^t$ and $\mathbf{W}_2^t$ are the training parameters of MLP.

3.3.3. Action scorer with gating mechanism

Future query events are usually uncertain and usually contain seen or unseen entities. To address these two cases in a unified RL framework and capture the evolutionary patterns of the facts more fully in the general case that the query contains a seen entity to predict the query accurately, we design a novel model with two sub-policy networks. Therefore, given two probability distributions of the candidate actions output by two sub-policy networks, we design a gating mechanism to adaptively integrate them to help the agent focus on the destination entity and obtain the final probability distribution $\pi_\theta(a_k|s_{k-1})$ of the candidate actions. This is expressed as follows:

$$\pi_\theta(a_k|s_{k-1}) = \mathrm{softmax}\big((1 - \mathbf{g}_t) * \phi_\theta^s(a_k|s_{k-1}) + \mathbf{g}_t * \phi_\theta^t(a_k|s_{k-1})\big) \tag{6}$$

$$\mathbf{g}_t = \mathrm{sigmoid}(\mathbf{W}_g[\mathbf{h}_{k-1}^t, \mathbf{r}_k^{t_k}, \mathbf{r}_q]) \tag{7}$$

where the gate function is parameterized by the temporal history, candidate temporal actions, and query relations. $\mathbf{W}_g$ is a learnable parameter.

3.4. Optimization

We set the length of the search path to K, and the K-hop action path generated by the policy network π_θ is defined as $\{a_1, a_2, \ldots, a_K\}$. Thus, the policy network is optimized by maximizing the expected reward over all queries in the training set $\mathcal{D}_{train}$:

$$J(\theta) = \mathbb{E}_{(s_q,r_q,o_q,t_q)\sim\mathcal{D}_{train}}[\mathbb{E}_{a_1,\ldots,a_K\sim\pi_\theta}[R(s_K|s_q, r_q, t_q)]] \tag{8}$$

Then, we employ the policy gradient method to optimize the policy network. Specifically, Eq. (8) is optimized by the REINFORCE algorithm (Williams, 1992), which iterates through all quadruples in $\mathcal{D}_{train}$ and updates θ with the following stochastic gradient:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_k R(s_K|s_q, r_q, t_q)\log\pi_\theta(a_k|s_{k-1}) \tag{9}$$

4. Experimental results

4.1. Datasets

We assess the performance of our model using four public TKG datasets: ICEWS14 (Boschee et al., 2015), ICEWS18 (Boschee et al., 2015), WIKI (Leblay & Chekol, 2018), and YAGO (Mahdisoltani, Biega, & Suchanek, 2015). ICEWS14 and ICEWS18 are the two subsets of the Integrated Crisis Early Warning System (ICEWS) dataset that contain the events in ICEWS that occurred in 2014 and 2018, respectively. WIKI primarily extracts temporal events from the Wikipedia dataset. Moreover, the temporal facts in YAGO mainly come from Wikipedias, WordNet, and GeoNames. The dataset is divided by timestamps in the form *train time* < *validation time* < *test time*, and more data information is summarized in Tables 1 and 2. In Table 1, N_{entity}, $N_{relation}$, and $N_{timestamps}$ denote the total number of entities, relations, and timestamps in the dataset, respectively. N_{train}, $N_{validation}$, and N_{test} denote the number of quadruples in the training, validation, and test sets, respectively. In Table 2, N_{entity}^{unseen} represents the number of new entities in the test set. $N_{quad}^{unseen\ entity}$ represents the number of quadruples where the entities are unseen. $N_{quad}^{unseen\ object}$ is the number of quadruples in which object entities are unseen. $N_{quad}^{unseen\ subject}$ denotes the number of quadruples in which the subject entities are unseen. $N_{quad}^{unseen\ subject\&object}$ represents the number of quadruples in which both the subject and object entities are unseen.

4.2. Evaluation metrics

We conduct a temporal knowledge-prediction task to evaluate the proposed model. Specifically, we predict two types of queries in the test set: $q_o = (s_q, r_q, ?, t_q)$ and $q_s = (?, r_q, o_q, t_q)$. We add inverse facts to the training dataset and transform the prediction of the subject entity for $(?, r_q, o_q, t_q)$ to predict the object entity for $(o_q, r_q^{-1}, ?, t_q)$ without loss of generality. Given the ground-truth o_q and s_q, we use Hits@1/3/10 and MRR (Nickel, Murphy,

Table 3
The results of link prediction on ICEWS14, ICEWS18, WIKI, and YAGO datasets. The compared metrics are filtered MRR and Hits@1/3/10. The best results are stated in bold.

Method	ICEWS2014				ICEWS2018				WIKI				YAGO			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
TransE	0.224	0.133	0.256	0.412	0.124	0.058	0.128	0.251	–	–	–	–	–	–	–	–
DistMult	0.276	0.181	0.311	0.469	0.101	0.045	0.103	0.212	0.496	0.461	0.528	0.541	0.548	0.473	0.598	0.685
ComplEx	0.308	0.215	0.344	0.495	0.210	0.118	0.234	0.398	–	–	–	–	–	–	–	–
MINERVA*	0.322	0.254	0.363	0.475	0.210	0.153	0.275	0.330	0.572	0.540	0.593	0.604	0.759	0.727	0.768	0.783
T-TransE	0.134	0.031	0.173	0.345	0.083	0.019	0.085	0.218	0.292	0.216	0.344	0.423	0.311	0.181	0.409	0.512
TA-DistMult	0.264	0.170	0.302	0.454	0.167	0.086	0.184	0.335	0.445	0.399	0.487	0.517	0.549	0.481	0.596	0.667
De-simplE	0.326	0.244	0.356	0.491	0.193	0.115	0.218	0.348	0.454	0.426	0.477	0.495	0.549	0.516	0.573	0.601
TNTComplEx	0.321	0.233	0.360	0.491	0.212	0.132	0.240	0.369	0.450	0.400	0.493	0.520	0.579	0.529	0.613	0.666
RE-NET	0.382	0.286	0.413	0.545	0.288	0.190	0.324	0.475	0.496	0.468	0.511	0.534	0.580	0.530	0.610	0.662
CyGNet	0.327	0.236	0.363	0.506	0.249	0.159	0.282	0.426	0.338	0.290	0.361	0.418	0.520	0.453	0.561	0.637
TANGO-Tucker	–	–	–	–	0.286	0.193	0.321	0.470	0.504	0.485	0.514	0.535	0.578	0.530	0.607	0.658
TANGO-DistMult	–	–	–	–	0.267	0.179	0.300	0.440	0.511	0.496	0.521	0.533	0.627	0.591	0.603	0.679
XERTE	0.407	0.327	0.456	0.573	0.293	0.210	0.335	0.464	0.711	0.680	0.761	0.790	0.841	0.800	0.880	0.897
TITer	0.417	0.327	0.464	0.584	0.299	0.220	0.334	0.448	0.755	0.729	0.774	0.790	0.874	0.848	0.899	0.902
Ours	**0.429**	**0.348**	**0.490**	**0.608**	**0.313**	**0.229**	**0.349**	**0.476**	**0.777**	**0.756**	**0.795**	**0.804**	**0.894**	**0.865**	**0.924**	**0.926**

Tresp, & Gabrilovich, 2016) to assess the performance of our model. MRR is the average of the reciprocal of the mean rank (MR) assigned to the true triple overall candidate triples, which is defined as follows:

$$MRR = \frac{1}{2 * |test|} \sum_{q \in test} \left(\frac{1}{rank(o_q|q_o)} + \frac{1}{rank(s_q|q_s)} \right) \tag{10}$$

Hits@n measures the percentage of test set rankings where a true triple is ranked within the top n candidate triples, which is defined as follows:

$$Hits@n = \frac{1}{2 * |test|} \sum_{q \in test} (\mathbb{I}\{rank(o_q|q_o)\} \le n + \mathbb{I}\{rank(s_q|q_s)\} \le n) \tag{11}$$

Furthermore, in this study, we employ the time-aware filtering (Han, Chen et al., 2021) scheme instead of the static filtering approach to filter out the quadruples that are genuine at the timestamp, obtaining more reasonable results.

4.3. Baselines

To comprehensively assess the performance of the proposed model, we compare it with static KG reasoning models, namely, TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), and MINERVA (Das et al., 2018), interpolation TKG reasoning models, namely, TTransE (Jiang et al., 2016), TA-DistMult (García-Durán et al., 2018), DE-SimplE (Goel et al., 2020), and TNTComplEx (Lacroix et al., 2020), and several extrapolation TKG reasoning models, namely, RE-NET (Jin et al., 2020), CyGNet (Zhu et al., 2021), TANGO (Ding, Han, Ma, & Tresp, 2021), XERTE (Han, Chen et al., 2021), and TITer (Sun et al., 2021).

In addition, to evaluate the importance of the different components of our model, we propose several variants of our model by adjusting the use of its components. (1) Policy network I: We use only policy network I to obtain the final candidate action score. (2) Policy network II: We use only policy network II. (3) w/o TRE: 'w/o' refers to 'without', and TRE denotes the temporal relation encoder. (4) w/o GM: GM represents the gating mechanism. Here, we use the mean results of policy networks I and II as the final candidate action scores. (5) w/o Semantic Edges: This means removing the proposed semantic edges in TKGs.

4.4. Implementation details

In our experiments, we conduct the proposed model in the Pytorch (Paszke et al., 2017) framework. The model is optimized using the Adam (Duchi, Hazan, & Singer, 2011) algorithm with a batch size of 512 and a learning rate of 0.001. We set the entity embedding dimension to 100, the relation embedding dimension to 100, the timestamp embedding dimension to 100, and the discount factor of the REINFORCE algorithm to 0.95. In addition, at each hop, for all candidate actions of the given state, we sample the N latest candidate actions to score. Here, N was 100 for the ICEWS14 and ICEWS18 datasets, 90 for WIKI, and 100 for YAGO. The length of the search path K is 3. In the temporal relation encoder, the activation function σ was *tanh*. Note that we reproduce the results of MINERVA with $N = 50$ for all datasets, and the results of some baseline models are taken from those reported in TANGO* (Han, Ding, Ma, Gu and Tresp, 2021), XERTE (Han, Chen et al., 2021), and TITer (Sun et al., 2021).

4.5. Results and analysis

4.5.1. Comparative study

Table 3 presents the temporal knowledge prediction results of the proposed model and baseline models on four TKG datasets. We have the following observations and analyses from the results. First, intuitively, our model outperforms all baseline models on the four datasets, which justifies the feasibility of our method based on the considerable results it achieves. Second, compared with the related baseline model TITer, our model achieves 2.2% and 2.7% improvements in MRR and Hits@1 on the WIKI dataset, respectively, and obtains different degrees of improvement on the other three datasets. Third, the static reasoning model TransE is superior to its extended version T-TransE on the ICEWS2014 dataset. The main reason for this phenomenon is that TransE and T-TransE are designed for static and interpolation TKG reasoning, respectively, and they cannot model the unseen future timestamps in the query. Therefore, the training-time information may affect the prediction performance of T-TransE. Fourth, the performance of MINERVA is much higher than that of other static models, all interpolation TKGs models, and part of extrapolation TKG models on WIKI and YAGO datasets. This is mainly because most entities in WIKI and YAGO have a small number of neighboring entities, which allows reinforcement learning-based neighbor search algorithms to find answer entities quickly and accurately. This is also one of the reasons why our method achieves a considerable performance on these two datasets.

4.5.2. Ablation study

To verify the effectiveness of the different components of the proposed model, we present the results of the five variants of

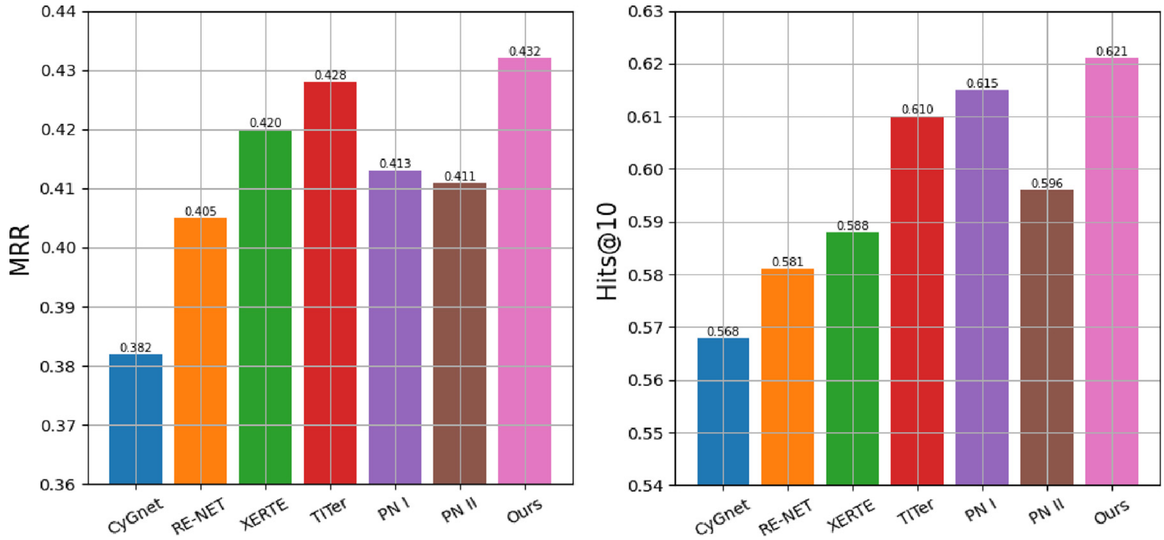

Fig. 3. Temporal knowledge prediction results on the subset of ICEWS14 that contains seen entities. PN I and PN II represent policy network I and policy network II, respectively.

Table 4
Ablation study on the ICEWS14 dataset.

Model variant	MRR	Hits@1	Hits@3	Hits@10
Policy network I	0.391	0.290	0.441	0.590
Policy network II	0.415	0.328	0.460	0.578
w/o TRE	0.390	0.288	0.441	0.593
w/o GM	0.394	0.295	0.444	0.588
w/o semantic edges	0.418	0.336	0.475	0.592
Ours	**0.429**	**0.348**	**0.490**	**0.608**

ICEWS14 in Table 4. Policy networks I and II perform different functions in our model, each of them can calculate the probability distribution of candidate actions and predict the answer to the query. From these results, we can see that our model outperforms the component models policy networks I and II, which demonstrates that our model can integrate these two component models adaptively using the gating mechanism and achieve remarkable results. In addition, we also observe that summing the outputs of the two sub-policy networks directly instead of using the gating mechanism leads to a drop of 3.5% in MRR. This indicates that the gating mechanism can help our model adaptively focus on the destination answer for the query. The temporal relation encoder is a crucial component in our model, which is assessed by removing the temporal relation encoder module in policy network II and adopting the same encoding method as policy network I. This model performance dropped by 3.9% on MRR compared with that of the proposed model. Moreover, we propose integrating semantic edges into TKGs to help the agent find the actions in the case that the query contains an unseen entity. Accordingly, we remove the semantic edges here to evaluate their importance. From these results, we observe that this variant model performance dropped by 1.1% on MRR compared with that of the proposed model. This also demonstrates the effectiveness of the temporal relation encoder in temporal knowledge representation.

4.5.3. Ablation study II

Our motivation for designing such a novel model is to better handle the two cases in which the query contains seen and unseen entities in a unified framework. Specifically, when the query contains a seen entity, policy networks I and II can cooperatively capture the evolutionary patterns of the facts. When the query contains an unseen entity, policy network II can activate its particular function. Therefore, in this section, we justify our design from two perspectives.

We first verify our design from the first perspective that policy networks I and II can capture the evolutionary patterns of the facts cooperatively when the query contains a seen entity. Therefore, we conduct link prediction on the subset of ICEWS14 that contains seen entities, the results of which are presented in Fig. 3. From the results, we can conclude that the proposed model is superior to the baseline models. Most importantly, the proposed model outperforms two component models, namely, policy network I and policy network II.

To justify our design from the second perspective that policy network II can exert its particular function in the case where the query contains the unseen entity, we also conduct link prediction on the subset of ICEWS14 that contains unseen entities. The results of which are shown in Fig. 4. Intuitively, we can observe from the results that policy network II has explicit advantages in handling the case in which the query contains an unseen entity, and our model can absorb the advantages of policy network II to achieve considerable performance. These results demonstrate the effectiveness of our design and prove that our model can predict events in complex scenarios.

4.5.4. Case study

Table 5 presents the specific reasoning paths of four queries in the test set of ICEWS2014, the first two of which are queries with a seen entity and the last two are queries with an unseen entity. For a query with a seen entity (*Ed Royce*, *Make pessimistic comment*, ?, *2014-11-23*), the agent arrives at entity *Nuri al-Maliki* through the temporal edge (*Ed Royce*, *Criticize or denounce*, *Nuri al-Maliki*, *2014-2-6*) starting from the entity *Ed Royce* in the first step. After three steps, the agent reaches the answer entity *Iran*. However, for a query with an unseen entity (*Mehdi Hasan*, *Make an appeal or request*, ?, *2014-11-12*), the entity *Mehdi Hasan* is unseen in the historical facts. Accordingly, the agent cannot find temporal edges to traverse, except for the self-loop edge. In this case, the agent first reaches the seen entity *India Citizen* through the semantic edge (*Shivraj*, *Make an appeal or request*, *India Citizen*, *2014-6-22*), and then arrives at the answer through the temporal edges or self-loop edges in the following two steps. From these results, we can observe that the semantic edge bridges the gap between the query with an unseen entity and historical facts, which exerts an important function in our method.

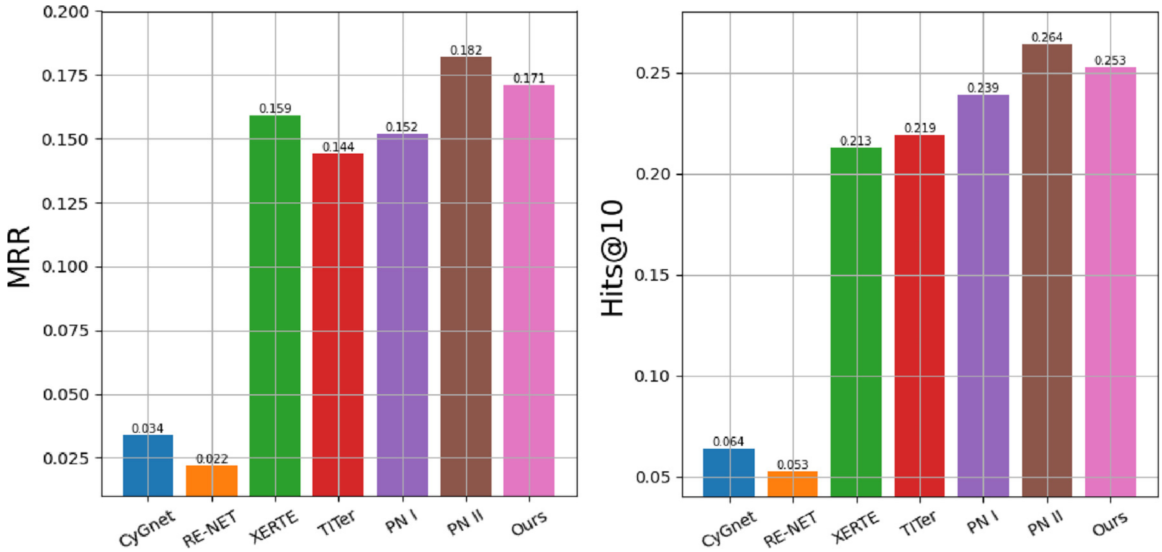

Fig. 4. Temporal knowledge prediction results on the subset of ICEWS14 that contains unseen entities. PN I and PN II represent policy network I and policy network II, respectively.

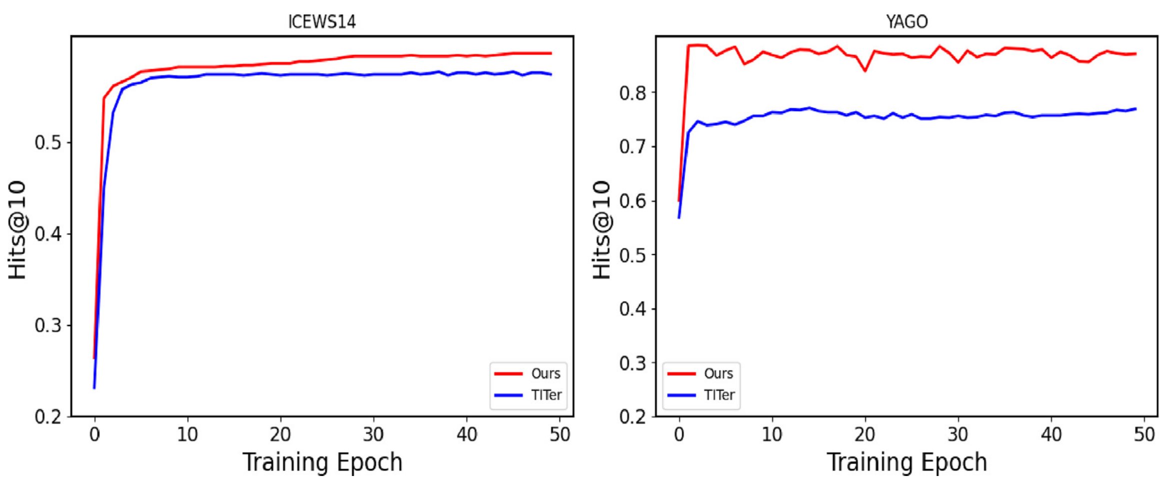

Fig. 5. Hits@10 on the validation set versus training epoch.

Table 5
Case study for two kinds of queries on ICEWS2014 dataset. $(\cdot)^{\dagger}$ represents the query with seen entity, and $(\cdot)^{\ddagger}$ indicates the query with unseen entity.

Query	Path	Answer
(*Ed Royce*, *Make pessimistic comment*, ?, *2014-11-23*)†	(*Ed Royce*, *Criticize or denounce*, *Nuri al-Maliki*, *2014-2-6*) ⇒ (*Nuri al-Maliki*, *Make statement*, *Iran*, *2014-1-25*) ⇒ (*Iran*, *self-loop*, ***Iran***, *2014-1-25*)	***Iran***
(*Cabinet USA*, *Consult*, ?, *2014-11-11*)†	(*Cabinet USA*, *Deny responsibility*$^{-1}$, *Sergey*, *2014-11-9*) ⇒ (*Sergey*, *self-loop*, *Sergey*, *2014-11-9*) ⇒ (*Sergey*, *self-loop*, ***Sergey***, *2014-11-9*)	***Sergey***
(*Mehdi Hasan*, *Make an appeal or request*, ?, *2014-11-12*)‡	(*Shivraj*, *Make an appeal or request*, *India Citizen*, *2014-6-22*) ⇒ (*India Citizen*, *self-loop*, *India Citizen*, *2014-4-7*) ⇒ (*India Citizen*, *self-loop*, ***India Citizen***, *2014-6-22*)	***India Citizen***
(*India Chief*, *Threaten*, ?, *2014-11-19*)‡	(*Australia Citizen*, *Threaten*, *India Citizen*, *2014-7-30*) ⇒ (*India Citizen*, *self-loop*, *India Citizen*, *2014-8-20*) ⇒ (*India Citizen*, *self-loop*, ***India Citizen***, *2014-6-30*)	***India Citizen***

4.5.5. Convergence study

Fig. 5 plots the Hits@10 scores on the validation set against the number of training epochs comparing our model with TITer. Notice that our model converges to a higher score than TITer does, which implies that our model exerts a positive function in the proposed reasoning problems and justifies its effectiveness. Furthermore, note that at the initial stage of training (epoch 0), our model has a much higher performance than that of TITer. This suggests that, in the initial stages, while our model and TITer would simply be performing random walks in the neighboring facts, our model can provide more temporal edges for the agent to find answers. Thus, our model has a much smarter strategy than that of TITer when searching for the answer to the query.

5. Conclusion

This study proposes a novel reinforcement learning model for temporal knowledge prediction tasks for TKGs. The model is presented mainly to address two problems: the modeling of the evolutionary patterns of facts and addressing uncertain queries that contain seen or unseen entities. Therefore, we design a novel adaptive pseudo-Siamese policy network to address these two problems in a unified framework. Policy network I is used to capture the static evolutionary patterns of the facts, and policy network II, designed with a temporal relation encoder, is employed to capture the temporal evolutionary patterns of the facts and address the special case in which the query contains an unseen entity. Finally, we develop a gating mechanism to adaptively integrate the results of the two sub-policy networks to help the agent focus on the destination answer. The experimental results on four public datasets explicitly demonstrate the effectiveness and superiority of the proposed model on temporal knowledge prediction tasks.

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.

Data availability

No data was used for the research described in the article.

Acknowledgment

We gratefully acknowledge the help and assistance from Bin Liu, Xiao Wang, and Xinke Wang. Moreover, this work is supported by the National Natural Science Foundation of China (NSFC) (No. 61831022, No. 62206278, No. 62276259, No. U21B2010).

References

Balažević, I., Allen, C., & Hospedales, T. M. (2019). TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the conference on empirical methods in natural language processing and the international joint conference on natural language processing* (pp. 5185–5194).

Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Proceedings of the international conference on neural information processing systems* (pp. 2787—2795).

Boschee, E., Lautenschlager, J., O'Brien, S., Shellman, S., Starz, J., & Ward, M. (2015). ICEWS coded event data. In *Proceedings of harvard dataverse*.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a "Siamese" time delay neural network. In *Proceedings of the 6th international conference on neural information processing systems* (pp. 737—744).

Cen, M., & Jung, C. (2018). Fully convolutional siamese fusion networks for object tracking. In *Proceedings of 25th ieee international conference on image processing* (pp. 3718–3722).

Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., et al. (2018). Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *Proceedings of international conference on learning representations*.

Dasgupta, S. S., Ray, S. N., & Talukdar, P. (2018). HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 2001–2011).

Dettmers, T., Pasquale, M., Pontus, S., & Riedel, S. (2018). Convolutional 2D knowledge graph embeddings. In *Proceedings of AAAI conference on artificial intelligence* (pp. 1811–1818).

Ding, Z., Han, Z., Ma, Y., & Tresp, V. (2021). Temporal knowledge graph forecasting with neural ODE. In *Proceedings of CoRR*.

Dong, L., Wei, F., Zhou, M., & Xu, K. (2015). Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the annual meeting of the association for computational linguistics and the international joint conference on natural language processing* (pp. 260–269).

Duchi, J. C., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. In *Proceedings of machine learning research* (pp. 2121—2159).

García-Durán, A., Dumancic, S., & Niepert, M. (2018). Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 4816–4821).

Goel, R., Kazemi, S. M., Brubaker, M., & Poupart, P. (2020). Diachronic embedding for temporal knowledge graph completion. In *Proceedings of AAAI conference on artificial intelligence*.

Han, Z., Chen, P., Ma, Y., & Tresp, V. (2021). Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *Proceedings of international conference on learning representations*.

Han, Z., Ding, Z., Ma, Y., Gu, Y., & Tresp, V. (2021). Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 conference on empirical methods in natural language processing* (pp. 8352–8364).

Ji, G., He, S., Xu, L., Liu, K., & Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of annual meeting of the association for computational linguistics* (pp. 687—696).

Jiang, T., Liu, T., Ge, T., Sha, L., Li, S., Chang, B., et al. (2016). Encoding temporal information for time-aware link prediction. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 2350–2354).

Jin, W., Qu, M., Jin, X., & Ren, X. (2020). Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *Proceedings of the conference on empirical methods in natural language processing*.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. In *Proceedings of computer* (pp. 30–37).

Lacroix, T., Obozinski, G., & Usunier, N. (2020). Tensor decompositions for temporal knowledge base completion. In *Proceedings of international conference on learning representations* (pp. 253—256).

Leblay, J., & Chekol, M. W. (2018). Deriving validity time in knowledge graph. In *Proceedings of the the web conference 2018* (pp. 1771—1776).

Li, Z., Jin, X., Guan, S., Li, W., Guo, J., Wang, Y., et al. (2021). Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. In *Proceedings of the annual meeting of the association for computational linguistics and the international joint conference on natural language processing* (pp. 4732–4743).

Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., & Yan, J. (2019). SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition* (pp. 4277–4286).

Lin, X. V., Socher, R., & Xiong, C. (2018). Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 1—9).

Mahdisoltani, F., Biega, J., & Suchanek, F. M. (2015). YAGO3: A knowledge base from multilingual wikipedias. In *Proceedings of CIDR*.

Nickel, M., Murphy, K., Tresp, V., & Gabrilovich, E. (2016). A review of relational machine learning for knowledge graphs. In *Proceedings of the IEEE*.

Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the international conference on machine learning* (pp. 809—816).

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Devito, Z., et al. (2017). Automatic differentiation in pytorch. In *Proceedings of the international conference on neural information processing systems*.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., & Welling, M. (2017). Modeling relational data with graph convolutional networks. In *Proceedings of the semantic web* (pp. 593–607).

Shen, Y., Chen, J., Huang, P.-S., Guo, Y., & Gao, J. (2018). M-walk: Learning to walk over graphs using Monte Carlo tree search. In *Proceedings of the international conference on neural information processing systems* (pp. 6787—6798).

Sun, H., Zhong, J., Ma, Y., Han, Z., & He, K. (2021). TimeTraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 8306–8319).

(2022). Triple pseudo-siamese network with hybrid attention mechanism for welding defect detection. In *Proceedings of materials design*. Article 110645.

Trivedi, R., Dai, H., Wang, Y., & Song, L. (2017). Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the international conference on machine learning* (pp. 3462—3471).

Trivedi, R. S., Farajtabar, M., Biswal, P., & Zha, H. (2019). DyRep: Learning representations over dynamic graphs. In *Proceedings of international conference on learning representations*.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016). Complex embeddings for simple link prediction. In *Proceedings of the international conference on machine learning* (pp. 2071—2080).

Vashishth, S., Sanyal, S., Nitin, V., & Talukdar, P. (2020). Composition-based multi-relational graph convolutional networks. In *Proceedings of international conference on learning representations*.

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI conference on artificial intelligence* (pp. 1112—1119).

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Proceedings of machine learning* (pp. 229–256).

Wu, C., Chen, L., He, Z., & Jiang, J. (2022). Pseudo-siamese graph matching network for textureless objects' 6-D pose estimation. In *Proceedings of IEEE transactions on industrial electronics* (pp. 2718–2727).

Wu, H., Xu, Z., Zhang, J., Yan, W., & Ma, X. (2017). Face recognition based on convolution siamese networks. In *Proceedings of 10th international congress on image and signal processing, biomedical engineering and informatics (CISP-BMEI)* (pp. 1–5).

Xiong, C., & Callan, J. P. (2015). Query expansion with freebase. In *Proceedings of the international conference on the theory of information retrieval* (pp. 111—120).

Xiong, W., Hoang, T., & Wang, W. Y. (2017). DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 564–573).

Yang, B., Yih, S. W.-t., He, X., Gao, J., & Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the international conference on learning representations*.

Zhu, C., Chen, M., Fan, C., Cheng, G., & Zhang, Y. (2021). Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of AAAI conference on artificial intelligence* (pp. 4732–4740).