TimeTraveler: Reinforcement Learning for Temporal Knowledge Graph Forecasting

Haohai Sun[1]* **Jialun Zhong**[1]* **Yunpu Ma**[2]* **Zhen Han**[2,3]* **Kun He**[1]†
[1] School of Computer Science and Technology,
Huazhong University of Science and Technology
[2] Institute of Informatics, LMU Munich [3] Corporate Technology, Siemens AG
{haohais, zhongjl}@hust.edu.cn
cognitive.yunpu@gmail.com, zhen.han@campus.lmu.de
brooklet60@hust.edu.cn

Abstract

Temporal knowledge graph (TKG) reasoning is a crucial task that has gained increasing research interest in recent years. Most existing methods focus on reasoning at past timestamps to complete the missing facts, and there are only a few works of reasoning on known TKGs to forecast future facts. Compared with the completion task, the forecasting task is more difficult and faces two main challenges: (1) how to effectively model the time information to handle future timestamps? (2) how to make inductive inference to handle previously unseen entities that emerge over time? To address these challenges, we propose the first reinforcement learning method for forecasting. Specifically, the agent travels on historical knowledge graph snapshots to search for the answer. Our method defines a relative time encoding function to capture the timespan information, and we design a novel time-shaped reward based on Dirichlet distribution to guide the model learning. Furthermore, we propose a novel representation method for unseen entities to improve the inductive inference ability of the model. We evaluate our method for this link prediction task at future timestamps. Extensive experiments on four benchmark datasets demonstrate substantial performance improvement meanwhile with higher explainability, less calculation, and fewer parameters when compared with existing state-of-the-art methods.

1 Introduction

Storing a wealth of human knowledge and facts, Knowledge Graphs (KGs) are widely used for many downstream Artificial Intelligence (AI) applications, such as recommendation systems (Guo et al., 2020), dialogue generation (Moon et al., 2019), and question answering (Zhang et al., 2018). KGs store facts in the form of triples (e_s, r, e_o), i.e. (subject entity, predicate/relation, object entity), such as *(LeBron_James, plays_for, Cleveland_Cavaliers)*. Each triple corresponds to a labeled edge of a multi-relational directed graph. However, facts constantly change over time. To reflect the timeliness of facts, Temporal Knowledge Graphs (TKGs) additionally associate each triple with a timestamp (e_s, r, e_o, t), e.g., *(LeBron_James, plays_for, Cleveland_Cavaliers, 2014-2018)*. Usually, we represent a TKG as a sequence of static KG snapshots.

TKG reasoning is a process of inferring new facts from known facts, which can be divided into two types, interpolation and extrapolation. Most existing methods (Jiang et al., 2016a; Dasgupta et al., 2018; Goel et al., 2020; Wu et al., 2020) focus on interpolated TKG reasoning to complete the missing facts at past timestamps. In contrast, extrapolated TKG reasoning focuses on forecasting future facts (events). In this work, we focus on extrapolated TKG reasoning by designing a model for the link prediction at future timestamps. E.g., "which team LeBron James will play for in 2022?" can be seen as a query of link prediction at a future timestamp: *(LeBron_James, plays_for, ?, 2022)*.

Compared with the interpolation task, there are two challenges for extrapolation. (1) Unseen timestamps: the timestamps of facts to be forecast do not exist in the training set. (2) Unseen entities: new entities may emerge over time, and the facts to be predicted may contain previously unseen entities. Hence, the interpolation methods can not treat the extrapolation task.

The recent extrapolation method RE-NET (Jin et al., 2020) uses Recurrent Neural Network (RNN) to capture temporally adjacent facts information to predict future facts. CyGNet (Zhu et al., 2021) focuses on the repeated pattern to count the frequency of similar facts in history. However, these methods only use the random vectors to represent the previously unseen entities and view the link prediction

*Equal Contribution.
†Corresponding author.

task as a multi-class classification task, causing it unable to handle the second challenge. Moreover, they cannot explicitly indicate the impact of historical facts on predicted facts.

Inspired by path-based methods (Das et al., 2018; Lin et al., 2018) for static KGs, we propose a new temporal-path-based reinforcement learning (RL) model for extrapolated TKG reasoning. We call our agent the "**TI**me **T**rave**l**er" (**TITer**), which travels on the historical KG snapshots to find answers for future queries. TITer starts from the query subject node, sequentially transfers to a new node based on temporal facts related to the current node, and is expected to stop at the answer node. To handle the unseen-timestamp challenge, TITer uses a relative time encoding function to capture the time information when making a decision. We further design a novel time-shaped reward based on Dirichlet distribution to guide the model to capture the time information. To tackle the unseen entities, we introduce a temporal-path-based framework and propose a new representation mechanism for unseen entities, termed the Inductive Mean (IM) representation, so as to improve the inductive reasoning ability of the model.

Our main contributions are as follows:

- This is the first temporal-path-based reinforcement learning model for extrapolated TKG reasoning, which is explainable and can handle unseen timestamps and unseen entities.
- We propose a new method to model the time information. We utilize a relative time encoding function for the agent to capture the time information and use a time-shaped reward to guide the model learning.
- We propose a novel representation mechanism for unseen entities, which leverages query and trained entity embeddings to represent untrained (unseen) entities. This can stably improve the performance for inductive inference without increasing the computational cost.
- Extensive experiments indicate that our model substantially outperforms existing methods with less calculation and fewer parameters.

2 Related Work

2.1 Static Knowledge Graph Reasoning

Embedding-based methods represent entities and relations as low-dimensional embeddings in different representation spaces, such as Euclidean space (Nickel et al., 2011; Bordes et al., 2013), complex vector space (Trouillon et al., 2016; Sun et al., 2019), and manifold space (Chami et al., 2020). These methods predict missing facts by scoring candidate facts based on entity and relation embeddings. Other works use deep learning models to encode the embeddings, such as Convolution Neural Network (CNN) (Dettmers et al., 2018; Vashishth et al., 2020) to obtain deeper semantics, or Graph Neural Network (GNN) (Schlichtkrull et al., 2018; Nathani et al., 2019; Zhang et al., 2020) to encode multi-hop structural information.

Besides, path-based methods are also widely used in KG reasoning. Lin et al. (2015) and Guo et al. (2019) use RNN to compose the implications of paths. Reinforcement learning methods (Xiong et al., 2017; Das et al., 2018; Lin et al., 2018) view the task as a Markov decision process (MDP) to find paths between entity pairs, which are more explanatory than embedding-based methods.

2.2 Temporal Knowledge Graph Reasoning

A considerable amount of works extend static KG models to the temporal domain. These models redesign embedding modules and score functions related to time (Jiang et al., 2016b; Dasgupta et al., 2018; Goel et al., 2020; Lacroix et al., 2020; Han et al., 2020a). Some works leverage message-passing networks to capture graph snapshot neighborhood information (Wu et al., 2020; Jung et al., 2020). These works are designed for interpolation.

For extrapolation, Know-Evolve (Trivedi et al., 2017) and GHNN (Han et al., 2020b) use temporal point process to model facts evolved in the continuous time domain. Additionally, TANGO (Ding et al., 2021) explores the neural ordinary differential equation to build a continuous-time model. RE-NET (Jin et al., 2020) considers the multi-hop structural information of snapshot graphs and uses RNN to model entity interactions at different times. CyGNet (Zhu et al., 2021) finds that many facts often show a repeated pattern and make reference to known facts in history. These approaches lack to explain their predictions and cannot handle the previously unseen entities. Explanatory model xERTE (Han et al., 2021) uses a subgraph sampling technique to build an inference graph. Although the representation method that refers to GraphSAGE (Hamilton et al., 2017) makes it possible to deal with unseen nodes, the continuous expansion of inference graphs also severely restricts the inference speed.

3 Methodology

Analogizing to the previous work on KGs (Das et al., 2018), we frame the RL formulation as "walk-based query-answering" on a temporal graph: the agent starts from the source node (subject entity of the query) and sequentially selects outgoing edges to traverse to new nodes until reaching a target. In this section, we first define our task, and then describe the reinforcement learning framework and how we incorporate the time information into the on-policy reinforcement learning model. The optimization strategy and the inductive mean representation method for previously unseen entities are provided in the end. Figure 2 is the overview of our model.

3.1 Task Definition

Here we formally define the task of extrapolation in a TKG. Let $\mathcal{E}$, $\mathcal{R}$, $\mathcal{T}$, and $\mathcal{F}$ denote the sets of entities, relations, timestamps, and facts, respectively. A fact in a TKG can be represented in the form of a quadruple (e_s, r, e_o, t), where $r \in \mathcal{R}$ is a directed labeled edge between a subject entity $e_s \in \mathcal{E}$ and an object entity $e_o \in \mathcal{E}$ at time $t \in \mathcal{T}$. We can represent a TKG by the graph snapshots over time. A TKG can be described as $\mathcal{G}_{(1,T)} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T\}$, where $\mathcal{G}_t = \{\mathcal{E}_t, \mathcal{R}, \mathcal{F}_t\}$ is a multi-relational directed TKG snapshot, and $\mathcal{E}_t$ and $\mathcal{F}_t$ denote entities and facts that exist at time t. In order to distinguish the graph nodes at different times, we let a node be a two-tuple with entity and timestamp: $e_i^t = (e_i, t)$. Thus, a fact (or event) (e_s, r, e_o, t) can also be seen as an edge from source node e_s^t to destination node e_o^t with type r.

Extrapolated TKG reasoning is the task of predicting the evolution of KGs over time, and we perform link prediction at future times. It is also forecasting of events occurring in the near future. Given a query $(e_q, r_q, ?, t_q)$ or $(?, r_q, e_q, t_q)$, we have a set of known facts $\{(e_{s_i}, r_i, e_{o_i}, t_i) | t_i < t_q\}$. These known facts constitute the known TKG, and our goal is to predict the missing object or subject entity in the query.

3.2 Reinforcement Learning Framework

Because there is no edge among the typical TKG snapshots, the agent cannot transfer from one snapshot to another. Hence, we sequentially add three types of edges. (i) *Reversed Edges*. For each quadruple (e_s, r, e_o, t), we add (e_o, r^{-1}, e_s, t) to the TKG, where r^{-1} indicates the reciprocal relation of r. Thus, we can predict the subject entity by converting $(?, r, e_q, t)$ to $(e_q, r^{-1}, ?, t)$ without loss of generality. (ii) *Self-loop Edges*. Self-loop edges can allow the agent to stay in a place and work as a stop action when the agent search unrolled for a fixed number of steps. (iii) *Temporal Edges*. The agent can walk from node $e_s^{t_j}$ to node $e_o^{t_i}$ through edge r, if (e_s, r, e_o, t_i) exits and $t_i < t_j \leq t_q$. Temporal edges indicate the impact of the past fact on the entity and help the agent find the answer in historical facts. Figure 1 shows the graph with temporal edges.

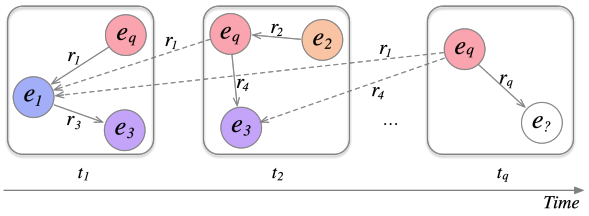

Figure 1: Illustration of the TKG with temporal edges. To ensure the figure be clear enough, we omit the self-loop edges and reversed edges. The dotted lines are temporal edges.

Our method can be formulated as a Markov Decision Process (MDP), and the components of which are elaborated as follows.

States. Let $\mathcal{S}$ denote the state space, and a state is represented by a quintuple $s_l = (e_l, t_l, e_q, t_q, r_q) \in \mathcal{S}$, where (e_l, t_l) is the node visited at step l and (e_q, t_q, r_q) is the elements in the query. (e_q, t_q, r_q) can be viewed as the global information while (e_l, t_l) is the local information. The agent starts from the source node of the query, so the initial state is $s_0 = (e_q, t_q, e_q, t_q, r_q)$.

Actions. Let $\mathcal{A}$ denote the action space, and $\mathcal{A}_l$ denote the set of optional actions at step l, $\mathcal{A}_l \subset \mathcal{A}$ consists of outgoing edges of node $e_l^{t_l}$. Concretely, $\mathcal{A}_l$ should be $\{(r', e', t') | (e_l, r', e', t') \in \mathcal{F}, t' \leq t_l, t' < t_q\}$, but an entity usually has many related historical facts, leading to a large number of optional actions. Thus, the final set of optional actions $\mathcal{A}_l$ is sampled from the set of above outgoing edges.

Transition. The environment state is transferred to a new node through the edge selected by the agent. The transition function $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ defined by $\delta(s_l, \mathcal{A}_l) = s_{l+1} = (e_{l+1}, t_{l+1}, e_q, t_q, r_q)$, where $\mathcal{A}_l$ is the sampled outgoing edges of $e_l^{t_l}$.

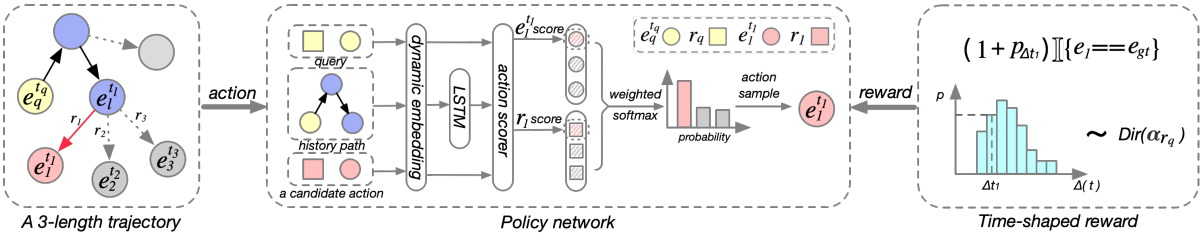

Figure 2: Overview of TITer. Given a query $(e_q, r_q, ?\ (e_{gt}), t_q)$, TITer starts from node $e_q^{t_q}$. At each step, TITer samples an outgoing edge and traverses to a new node according to π_θ (policy network). We use the last step of the search as an example. $e_l^{t_l}$ is the current node. Illustration of policy network provides the process for scoring one of the candidate actions (r_1, e_1, t_1). TITer samples an action based on the transition probability calculated from all candidate scores. When the search is completed, the time-shaped reward function will give the agent a reward based on the estimated Dirichlet distribution $Dir(\boldsymbol{\alpha}_{r_q})$.

Rewards with shaping. The agent receives a terminal reward of 1 if it arrives at a correct target entity at the end of the search and 0 otherwise. If $s_L = (e_L, t_L, e_q, t_q, r_q)$ is the final state and (e_q, r_q, e_{gt}, t_q) is the ground truth fact, the reward formulation is:

$$R(s_L) = \mathbb{I}\{e_L == e_{gt}\}. \tag{1}$$

Usually, the quadruples with the same entity are concentrated in specific periods, which causes temporal variability and temporal sparsity (Wu et al., 2020). Due to such property, the answer entity of the query has a distribution over time, and we can introduce this prior knowledge into the reward function to guide the agent learning. The time-shaped reward can let the agent know which snapshot is more likely to find the answer. Based on the training set, we estimate a Dirichlet distribution for each relation. Then, we shape the original reward with Dirichlet distributions:

$$\begin{aligned} \widetilde{R}(s_L) &= (1 + p_{\Delta t_L}) R(s_L), \\ \Delta t_L &= t_q - t_L, \\ (p_1, ..., p_K) &\sim Dirichlet(\boldsymbol{\alpha}_{r_q}), \end{aligned} \tag{2}$$

where $\boldsymbol{\alpha}_{r_q} \in \mathbb{R}^K$ is a vector of parameters of the Dirichlet distribution for a relation r_q. We can estimate $\boldsymbol{\alpha}_{r_q}$ from the training set. For each quadruple with relation r_q in the training set, we count the number of times the object entity appears in each of the most recent K historical snapshots. Then, we obtain a multinomial sample x_i and $D = \{x_1, ..., x_N\}$. To maximize the likelihood:

$$p(D|\boldsymbol{\alpha}_{r_q}) = \prod_i p(x_i|\boldsymbol{\alpha}_{r_q}), \tag{3}$$

we can estimate $\boldsymbol{\alpha}_{r_q}$. The maximum can be computed via the fixed-point iteration, and more calculation formulas are provided in Appendix A.5. Because Dirichlet distribution has a conjugate prior, we can update it easily when we have more observed facts to train the model.

3.3 Policy Network

We design a policy network $\pi_\theta(a_l|s_l) = P(a_l|s_l; \theta)$ to model the agent in a continuous space, where $a_l \in \mathcal{A}_l$, and θ are the model parameters. The policy network consists of the following three modules.

Dynamic embedding. We assign each relation $r \in \mathcal{R}$ a dense vector embedding $\mathbf{r} \in \mathbb{R}^{d_r}$. As the characteristic of entities may change over time, we adopt a relative time representation method for entities. We use a dynamic embedding to represent each node $e_i^t = (e_i, t)$ in $\mathcal{G}_t$, and use $\mathbf{e} \in \mathbb{R}^{d_e}$ to represent the latent invariant features of entities. We then define a relative time encoding function $\mathbf{\Phi}(\Delta t) \in \mathbb{R}^{d_t}$ to represent the time information. $\Delta t = t_q - t$ and $\mathbf{\Phi}(\Delta t)$ is formulated as follows:

$$\mathbf{\Phi}(\Delta t) = \sigma(\mathbf{w}\Delta t + \mathbf{b}), \tag{4}$$

where $\mathbf{w}, \mathbf{b} \in \mathbb{R}^{d_t}$ are vectors with learnable parameters and σ is an activation function. d_r, d_e and d_t represent the dimensions of the embedding. Then, we can get the representation of a node e_i^t: $\mathbf{e}_i^t = [\mathbf{e}_i; \mathbf{\Phi}(\Delta t)]$.

Path encoding. The search history $h_l = ((e_q, t_q), r_1, (e_1, t_1), ..., r_l, (e_l, t_l))$ is the sequence of actions taken. The agent encodes the history h_l with a LSTM:

$$\begin{aligned} \mathbf{h}_l &= \text{LSTM}(\mathbf{h}_{l-1}, [\mathbf{r}_{l-1}; \mathbf{e}_{l-1}^{t_{l-1}}]), \\ \mathbf{h}_0 &= \text{LSTM}(\mathbf{0}, [\mathbf{r}_0; \mathbf{e}_q^{t_q}]). \end{aligned} \tag{5}$$

Here, $\mathbf{r}_0$ is a start relation, and we keep the LSTM state unchanged when the last action is self-loop.

Action scoring. We score each optional action and calculate the probability of state transition. Let $a_n = (e_n, t_n, r_n) \in \mathcal{A}_l$ represent an optional action at step l. Future events are usually uncertain, and there is usually no strong causal logic chain for some queries, so the correlation between the entity and query is sometimes more important. Thus, we use a weighted action scoring mechanism to help the agent pay more attention to attributes of the destination nodes or types of edges. Two Multi-Layer Perceptrons (MLPs) are used to encode the state information and output expected destination node $\widetilde{e}$ and outgoing edge $\widetilde{r}$ representations. Then, the agent obtains the destination node score and outgoing edge score of the candidate action by calculating the similarity. With the weighted sum of the two scores, the agent obtains the final candidate action score $\phi(a_n, s_l)$:

$$\phi(a_n, s_l) = \beta_n \left\langle \widetilde{\mathbf{e}}, \mathbf{e}_n^{t_n} \right\rangle + (1 - \beta_n) \left\langle \widetilde{\mathbf{r}}, \mathbf{r}_n \right\rangle, \quad (6)$$

$$\begin{aligned} \widetilde{\mathbf{e}} &= \mathbf{W}_e \mathrm{ReLU}(\mathbf{W}_1[\mathbf{h}_l; \mathbf{e}_q^{t_q}; \mathbf{r}_q]), \\ \widetilde{\mathbf{r}} &= \mathbf{W}_r \mathrm{ReLU}(\mathbf{W}_1[\mathbf{h}_l; \mathbf{e}_q^{t_q}; \mathbf{r}_q]), \end{aligned} \quad (7)$$

$$\beta_n = \mathrm{sigmoid}(\mathbf{W}_\beta[\mathbf{h}_l; \mathbf{e}_q^{t_q}; \mathbf{r}_q; \mathbf{e}_n^{t_n}; \mathbf{r}_n]), \quad (8)$$

where $\mathbf{W}_1$, $\mathbf{W}_e$, $\mathbf{W}_r$ and $\mathbf{W}_\beta$ are learnable matrices. After scoring all candidate actions in $\mathcal{A}_l$, $\pi_\theta(a_l|s_l)$ can be obtained through softmax.

To summarize, the parameters of the LSTM, MLP and $\mathbf{\Phi}$, the embedding matrices of relation and entity form the parameters in θ.

3.4 Optimization and Training

We fix the search path length to L, and an L-length trajectory will be generated from the policy network π_θ: $\{a_1, a_2, ..., a_L\}$. The policy network is trained by maximizing the expected reward over all training samples $\mathcal{F}_{train}$:

$$\begin{aligned} J(\theta) = \mathbb{E}_{(e_s, r, e_o, t) \sim \mathcal{F}_{train}} [\mathbb{E}_{a_1, ..., a_L \sim \pi_\theta} \\ [\widetilde{R}(s_L|e_s, r, t)]]. \end{aligned} \quad (9)$$

Then, we use the policy gradient method to optimize the policy. The REINFORCE algorithm (Williams, 1992) will iterate through all quadruple in $\mathcal{F}_{train}$ and update θ with the following stochastic gradient:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{m \in [1, L]} \widetilde{R}(s_L|e_s, r, t) log \pi_\theta(a_l|s_l) \quad (10)$$

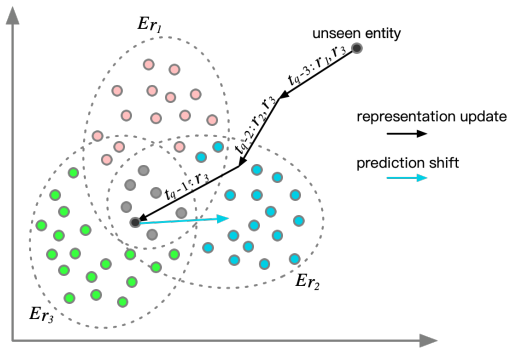

Figure 3: Illustration of the IM mechanism. For an unseen entity e_q, "$t_q - 3 : r_1, r_3$" indicates e_q has co-occurrence relations r_1, r_3 at $t_q - 3$, and updates its representation based on E_{r_1}, E_{r_3}, and finally gets the IM representation at $t_q - 1$. Then to answer a query $(e_q, r_2, ?, t_q)$, we do a prediction shift based on E_{r_2}.

3.5 Inductive Mean Representation

As new entities always emerge over time, we propose a new entity representation method for previously unseen entities. Previous works (Bhowmik and de Melo, 2020; Han et al., 2021) can represent unseen entities through neighbor information aggregation. However, newly emerging entities usually have very few links, which means that only limited information is available. E.g., for a query $(Evan\_Mobley, plays\_for, ?, 2022)$, entity "Evan_Mobley" does not exist in previous times, but we could infer this entity is a player through relation "plays_for", and assign "Evan_Mobley" a more reasonable initial embedding that facilitates the inference. Here we provide another approach to represent unseen entities by leveraging the query information and embeddings of the trained entities, named Inductive Mean (IM), as illustrated in Figure 3.

Let $\mathcal{G}_{(t_j, t_q-1)}$ represent the snapshots of the TKG in the test set. The query entity e_q first appears in $\mathcal{G}_{t_j}$ and gets a randomly initialized representation vector. We regard r as the *co-occurrence relation* of e_q, if there exists a quadruple which contains (e_q, r). Note that entity e_q may have different co-occurrence relations over time. We denote $R_t(e_q)$ as the co-occurrence relation set of entity e_q at time t. Let E_r represent the set that comprises all the trained entities having the co-occurrence relation r. Then, we can obtain the inductive mean representation of the entities with the same co-occurrence relation r:

$$\overline{\mathbf{e}^r} = \frac{\sum_{e \in E_r} \mathbf{e}}{|E_r|}. \quad (11)$$

Entities with the same co-occurrence relation r have similar characteristics, so IM can utilize $\overline{\mathbf{e}^r}$ to gradually update the representation of e_q based on the time flow. $0 \leq \mu \leq 1$ is a hyperparameter:

$$\mathbf{e}_{q,t} = \mu \mathbf{e}_{q,t-1} + (1-\mu)\frac{\sum_{r \in R_t(e_q)} \overline{\mathbf{e}^r}}{|R_t(e_q)|}. \quad (12)$$

For relation r_q, we do a prediction shift based on $\overline{\mathbf{e}^{r_q}}$ to make the entity representation more suitable for the current query. To answer a query $(e_q, r_q, ?, t_q)$, we use the combination of e_q's representation at time $t_q - 1$ and the inductive mean representation $\overline{\mathbf{e}^{r_q}}$:

$$\mathbf{e}_{q,t_q,r_q} = \mu \mathbf{e}_{q,t_q-1} + (1-\mu)\overline{\mathbf{e}^{r_q}}. \quad (13)$$

4 Experiments

4.1 Experimental Setup

Datasets. We use four public TKG datasets for evaluation: ICEWS14, ICEWS18 (Boschee et al., 2015), WIKI (Leblay and Chekol, 2018a), and YAGO (Mahdisoltani et al., 2015). Integrated Crisis Early Warning System (ICEWS) is an event dataset. ICEWS14 and ICEWS18 are two subsets of events in ICEWS that occurred in 2014 and 2018 with a time granularity of days. WIKI and YAGO are two knowledge base that contains facts with time information, and we use the subsets with a time granularity of years. We adopt the same dataset split strategy as in (Jin et al., 2020) and split the dataset into train/valid/test by timestamps such that:

time_of_train < time_of_valid < time_of_test. Appendix A.2 summarizes more statistics on the datasets.

Evaluation metrics. We evaluate our model on TKG forecasting, a link prediction task at the future timestamps. Mean Reciprocal Rank (MRR) and Hits@1/3/10 are performance metrics. For each quadruple (e_s, r, e_o, t) in the test set, we evaluate two queries, $(e_s, r, ?, t)$ and $(?, r, e_o, t)$. We use the time-aware filtering scheme (Han et al., 2020b) that only filters out quadruples with query time t. The time-aware scheme is more reasonable than the filtering scheme used in (Jin et al., 2020; Zhu et al., 2021). Appendix A.1 provides detailed definitions.

Baseline. As lots of previous works have verified that the static methods underperform compared with the temporal methods on this task, we do not compare TITer with them. We compare our model with existing interpolated TKG reasoning methods, including TTransE (Leblay and Chekol, 2018b), TA-DistMult (García-Durán et al., 2018), DE-SimplE (Goel et al., 2020), and TNTComplEx (Lacroix et al., 2020), and state-of-the-art extrapolated TKG reasoning approaches, including RE-NET (Jin et al., 2020), CyGNet (Zhu et al., 2021), TANGO (Ding et al., 2021), and xERTE (Han et al., 2021). An overview of these methods is in Section 2.

4.2 Implementation Details

Our model is implemented in PyTorch[1]. We set the entity embedding dimension to 80, the relation embedding dimension to 100, and the relative time encoding dimension to 20. We choose the latest N outgoing edges as candidate actions for TITer at each step. N is 50 for ICEWS14 and ICEWS18, 60 for WIKI, and 30 for YAGO. The reasoning path length is 3. The discount factor γ of REINFORCE is 0.95. We use Adam optimizer to optimize the parameters, and the learning rate is 0.001. The batch size is set to 512 during training. We use beam search for inference, and the beam size is 100. For the IM, μ is 0.1. The activation function of $\mathbf{\Phi}$ is $cosine$. For full details, please refer to Appendix A.3.

4.3 Results and Discussion

Performance on the TKG datasets. Our method can search multiple candidate answers via beam search. Table 1 reports the TKG forecasting performance of TITer and the baselines on four TKG datasets. TITer outperforms all baselines on all datasets when evaluated by MRR and Hits@1 metrics, and in most cases (except ICEWS18), TITer exhibits the best performance when evaluated by the other two metrics. TTransE, TA-DistMult, DE-SimplE, and TNTComplEx cannot deal with unseen timestamps in the test set, so they perform worse than others. The performance of TITer is much higher than RE-NET, CyGNet, and TANGO on WIKI and YAGO. Two reasons cause this phenomenon: (1) the characteristic of WIKI and YAGO that nodes usually have a small number of neighbors gives the neighbor search algorithm an advantage. (2) for WIKI and YAGO, a large number of quadruples contain unseen entities in the test set(See Table 2), but these

[1] https://github.com/JHL-HUST/TITer/

Table 1: Comparison on future link prediction. The results of MRR and Hits@1/3/10 are multiplied by 100. The best results are in bold. We average the metrics over five runs.

Method	ICEWS14				ICEWS18				WIKI				YAGO			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TTransE	13.43	3.11	17.32	34.55	8.31	1.92	8.56	21.89	29.27	21.67	34.43	42.39	31.19	18.12	40.91	51.21
TA-DistMult	26.47	17.09	30.22	45.41	16.75	8.61	18.41	33.59	44.53	39.92	48.73	51.71	54.92	48.15	59.61	66.71
DE-SimplE	32.67	24.43	35.69	49.11	19.30	11.53	21.86	34.80	45.43	42.6	47.71	49.55	54.91	51.64	57.30	60.17
TNTComplEx	32.12	23.35	36.03	49.13	27.54	19.52	30.80	42.86	45.03	40.04	49.31	52.03	57.98	52.92	61.33	66.69
CyGNet	32.73	23.69	36.31	50.67	24.93	15.90	28.28	42.61	33.89	29.06	36.10	41.86	52.07	45.36	56.12	63.77
RE-NET	38.28	28.68	41.34	54.52	28.81	19.05	32.44	**47.51**	49.66	46.88	51.19	53.48	58.02	53.06	61.08	66.29
xERTE	40.79	32.70	45.67	57.30	29.31	21.03	**33.51**	46.48	71.14	68.05	76.11	79.01	84.19	80.09	88.02	89.78
TANGO-Tucker	–	–	–	–	28.68	19.35	32.17	47.04	50.43	48.52	51.47	53.58	57.83	53.05	60.78	65.85
TANGO-DistMult	–	–	–	–	26.75	17.92	30.08	44.09	51.15	49.66	52.16	53.35	62.70	59.18	60.31	67.90
TITer	**41.73**	**32.74**	**46.46**	**58.44**	**29.98**	**22.05**	33.46	44.83	**75.50**	**72.96**	**77.49**	**79.02**	**87.47**	**84.89**	**89.96**	**90.27**

Table 2: The percentage of quaduples containing unseen entities of used test datasets.

Datasets	ICEWS14	ICEWS18	WIKI	YAGO
Proportion	6.52	3.93	42.91	8.03

Table 3: Comparison on the number of parameters and calculation. (M means million.)

Method	# Params	# MACs
RE-NET	5.459M	4.370M
CyGNet	8.568M	8.554M
xERTE(3 steps)	2.927M	225.895M
TITer(3 steps)	1.455M	0.225M

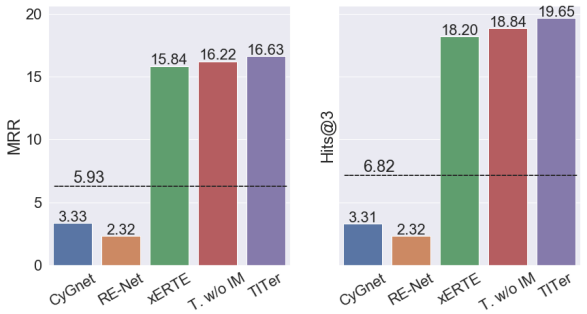

Figure 4: Inductive inference results on a subset of ICEWS14 that contain unseen entities. The dotted line corresponds to the score of a 3-steps random search. T. w/o IM: TITer without IM mechanism.

methods cannot handle these queries.

Inductive inference. When a query contains an unseen entity, models should infer the answer inductively. For all such queries that contain unseen entities in the ICEWS14 test set, we present experimental results in Figure 4 and Table 6. The performance of RE-NET and CyGNet decays significantly when compared with their result on the whole ICEWS14 test set (see Table 1). Due to the lack of training for unseen entities' embedding and the classification layer for all entities, RE-Net and CyGNet could not reach the performance of a 3-hop neighborhood random search baseline, as illustrated by the dotted line in Figure 4. In contrast, xERTE can tackle such queries by dynamically updating the new entities' representation based on temporal message aggregation, and TITer can tackle such queries by the temporal-path-based RL framework. We also observe that TITer outperforms xERTE, no matter TITer adopts the IM mechanism or not. The IM mechanism could further boost the performance, demonstrating its effectiveness in representing unseen entities.

Case study. Table 4 visualizes specific reasoning paths of several examples in the test set of ICEWS18. We notice that TITer tends to select a recent fact (outgoing edge) to search for the answer. Although the first two queries have the same subject entity and relation, TITer can reach the ground truths according to different timestamps. As shown in Eq. (6), β increases when TITer emphasizes neighbor nodes more than edges. After training, the representations of entities accumulate much semantic information, which helps TITer select the answer directly with less extra information for queries 3 and 4. In comparison, TITer needs more historical information when unseen entities appear. Query 5 is an example of multi-hop reasoning. It indicates that TITer can tackle combinational logic problems.

Efficiency analysis. Table 3 reflects the complexity of RE-NET, CyGNet, xERTE, and TITer. Due to the enormous linear classification layers, RE-

Table 4: Path visualization on ICEWS18. † indicates unseen entities not appeared in the training set. ‡ indicates entities along a path with the same country. β is defined in Eq. (6). For a test quadruple, we use the object prediction as an example.

ID	Test quadruple	Path	β
1	(Military (Comoros)†, Use conventional military force, Citizen (Comoros), 2018/10/17)	Military‡ $\xrightarrow[t=2018/10/16]{fight\ with\ small\ arms\ and\ light\ weapons}$ Citizen‡	0.46
2	(Military (Comoros)†, Use conventional military force, Armed Rebel (Comoros)†, 2018/10/26)	Military‡ $\xrightarrow[t=2018/10/19]{Investigate^{-1}}$ Armed Rebel‡	0.54
3	(Nigeria, Consult, Muhammadu Buhari, 2018/10/04)	Nigeria $\xrightarrow[t=2018/10/01]{Make\ an\ appeal\ or\ request^{-1}}$ Muhammadu Buhari	0.96
4	(Police (India), Physically assault, Citizen (India), 2018/10/08)	Police‡ $\xrightarrow[t=2018/10/06]{Investigate}$ Citizen‡	0.96
5	(Governor (Cote d'Ivoire)†, Make an appeal or request, Citizen (Cote d'Ivoire), 2018/10/14)	Governor‡ $\xrightarrow[t=2018/10/12]{Praise\ or\ endorse}$ Party Member‡ $\xrightarrow[t=2018/9/29]{Make\ an\ appeal\ or\ request}$ Citizen‡	–

Table 5: Ablation study on ICEWS18. w/o: without, ws: weighted action scoring mechanism, rs: reward shaping.

Method	MRR	H@1	H@3	H@10
TITer w/o ws and rs	29.13	20.73	32.74	45.04
TITer w/o ws	29.25	20.84	32.83	**45.06**
TITer w/o rs	29.17	21.00	32.72	44.51
TITer	**29.98**	**22.05**	**33.46**	44.83

Table 6: Results improvement with IM mechanism on subsets of ICEWS14 and WIKI that contain unseen entities.

Datasets	MRR	H@1	H@3	H@10
ICEWS14	+0.41	+0.46	+0.81	+0.01
WIKI	+1.52	+1.15	+2.15	+1.90

NET and CyGNet have much more parameters than other methods. To achieve the best results, xERTE adopts the graph expansion mechanism and the temporal relational graph attention layer to perform a local representation aggregation for each step, leading to a vast amount of calculation. Compared with xERTE, the number of parameters of TITer has reduced by at least a half, and the number of Multi-Adds operations (MACs) has greatly reduced to 0.225M, which is much less than the counterpart, indicating the high efficiency of the proposed model.

In summary, compared to the previous state-of-the-art models, TITer has saved at least 50.3% parameters and 94.9% MACs. Meanwhile, TITer still exhibits better performance.

4.4 Ablation Study

In this subsection, we study the effect of different components of TITer by ablation studies.The results are shown in Table 5 and Figure 5.

Relative time encoding. The relative time representation is a crucial component in our method. Figure 5 shows the notable change from temporal to static on ICEWS18 and WIKI. We remove the relative time encoding module to get the static model. For Hits@1, the temporal model improves 13.19% on ICEWS18 and 18.51% on WIKI, compared with the static model. It indicates that our relative time encoding function can help TITer choose the correct answer more precisely.

Weighted action scoring mechanism. We observe that setting β to a constant 0.5 can lead to a drop of 5.49% on Hits@1, indicating that TITer can better choose the source of evidence when making the inference. After training, TITer learns the latent relationship among entities. As expounded in Table 4, TITer prefers to pay more attention to the node for inferring when there exists more information to make a decision, and TITer chooses to focus on edges (relations in history) to assist the inferring for complex queries or unseen entities.

Reward shaping. We observe that TITer outperforms the variant without reward shaping, which means an improvement of 5% on Hits@1. By using Dirichlet prior distribution to direct the decision process, TITer acquires knowledge about the probability distribution of the target's appearance over the whole time span.

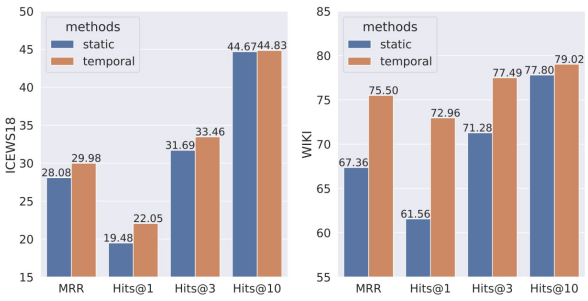

Figure 5: Ablation of time information.

5 Conclusion

In this work, we propose a temporal-path-based reinforcement learning model named TimeTraveler (TITer) for temporal knowledge graph forecasting. TITer travels on the TKG historical snapshots and searches for the temporal evidence chain to find the answer. TITer uses a relative time encoding function and time-shaped reward to model the time information, and the IM mechanism to update the unseen entities' representation in the process of testing. Extensive experimental results reveal that our model outperforms state-of-the-art baselines with less calculation and fewer parameters. Furthermore, the inference process of TITer is explainable, and TITer has good inductive reasoning ability.

Acknowledgements

This work is supported by National Natural Science Foundation (62076105).

References

Rajarshi Bhowmik and Gerard de Melo. 2020. Explainable link prediction for emerging entities in knowledge graphs. In *International Semantic Web Conference*, pages 39–55.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems*, pages 2787–2795.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Zifeng Ding, Zhen Han, Yunpu Ma, and Volker Tresp. 2021. Temporal knowledge graph forecasting with neural ode. *arXiv preprint arXiv:2101.05151*.

Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3988–3995.

Lingbing Guo, Zequn Sun, and Wei Hu. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2505–2514.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, pages 1025–1035.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020a. DyERNIE: Dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7301–7316.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*.

Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. 2020b. Graph hawkes neural network for forecasting on temporal knowledge graphs. In *Conference on Automated Knowledge Base Construction*.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016a. Towards time-aware knowledge graph completion. In *COLING 2016, 26th International Conference on Computational Linguistics*, pages 1715–1724.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016b. Towards time-aware knowledge graph completion. In *COLING 2016, 26th International Conference on Computational Linguistics*, pages 1715–1724.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6669–6683.

Jaehun Jung, Jinhong Jung, and U Kang. 2020. T-GAP: Learning to walk across time for temporal knowledge graph completion. *arXiv preprint arXiv:2012.10595*.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. In *International Conference on Learning Representations*.

Julien Leblay and Melisachew Wudage Chekol. 2018a. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference*, pages 1771–1776.

Julien Leblay and Melisachew Wudage Chekol. 2018b. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pages 1771–1776.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A knowledge base from multilingual wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research*.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-Evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3462–3471.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 2071–2080.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020. InteractE: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3009–3016.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L. Hamilton. 2020. TeMP: Temporal message passing for temporal knowledge graph completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5730–5746.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 6069–6076.

Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. 2020. Relational graph neural network with hierarchical attention for knowledge graph completion. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 9612–9619.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 4732–4740.

A Appendix

A.1 Definitions for Evaluation Metrics

We use two popular metrics, Mean Reciprocal Rank (MRR) and Hits@k (we let $k \in \{1, 3, 10\}$), to evaluate the models' performance. For each quadruple $q = (e_s, r, e_o, t)$ in the test fact set $\mathcal{F}_{test}$, we evaluate two queries: $q_o = (e_s, r, ?, t)$ and $q_s = (?, r, e_o, t)$. For each query, our model ranks the entities searched by beam search according to the transition probability. If the ground truth entity does not appear in the final searched entity set, we set the rank as the number of entities in the dataset. xERTE ranks the entities in the final inference graph, and others rank the total entities in the dataset.

MRR is defined as:

$$MRR = \frac{1}{2 * |\mathcal{F}_{test}|} \sum_{q \in \mathcal{F}_{test}} (\frac{1}{rank(e_o|q_o)} + \frac{1}{rank(e_s|q_s)}). \tag{14}$$

Hits@k is the percentage of times that the ground truth entity appears in the top k of the ranked candidates, defined as:

$$\text{Hits@}k = \frac{1}{2 * |\mathcal{F}_{test}|} \sum_{q \in \mathcal{F}_{test}} (\mathbb{I}\{rank(e_o|q_o\} \leq k + \mathbb{I}\{rank(e_s|q_s\} \leq k). \tag{15}$$

There are two filtering settings, static filtering and time-aware filtering. RE-NET (Jin et al., 2020) and CyGNet (Zhu et al., 2021) directly use static filtering setting to remove the entities from the candidate list according to the triples in the dataset. However, this filtering setting is not appropriate for temporal KGs. The facts of temporal KGs always change over time. For example, we evaluate the test quadruple (A, visit, B, 2018/04/03). There are two other facts (A, visit, C, 2018/04/03) and (A, visit, D, 2018/03/28). Static filtering setting will remove both C and D from the candidates even though the triple (A, visit, D) is not invalid in 2018/04/03. A more appropriate setting is only to remove C from the candidates. Therefore, we use the time-aware filtering setting that eliminates the entities according to the quadruple.

A.2 Dataset Statistics

Dataset statistics are provided in Table 8. N_{train}, N_{valid} and N_{test} are the numbers of quadruples in training set, valid set, and test set, respectively. N_{ent} and N_{rel} are the numbers of total entities and total relations. ICEWS14 and ICEWS18 are event-based knowledge graphs, and we use the same version as (Han et al., 2021)[2]. WIKI and YAGO datasets contain temporal facts with time span $(e_s, r, e_o, [t_s, t_e])$, and each fact is converted to $\{(e_s, r, e_o, t_s), (e_s, r, e_o, t_s+1_t), ..., (e_s, r, e_o, t_e)\}$ where 1_t is a unit time. Here, $1_t == 1year$. We use the same version of WIKI and YAGO as (Jin et al., 2020)[3].

Table 7: Number of unseen entities in the test set.

Dataset	N_{ent}^{unseen}	N_{oq}^{unseen}	N_{sq}^{unseen}	N_{soq}^{unseen}
ICEWS14	496	438	497	73
ICEWS18	1140	975	1050	77
WIKI	2968	11086	22967	6974
YAGO	540	1102	873	366

Since we split each dataset by timestamps, some entities in the test set do not exist in the training set. Table 7 describes the number of unseen entities and the number of quadruples containing these entities in the test set. N_{ent}^{unseen} is the number of new entities in the test set. N_{oq}^{unseen} is the number of quadruples that object entities are unseen. N_{sq}^{unseen} is the number of quadruples that subject entities are unseen. N_{soq}^{unseen} is the number of quadruples that both subject entities and object entities are unseen.

A.3 Detailed Implementation

Hyperparameters search. We use a grid search to choose the hyperparameters. The search space of μ is $[0.1, 0.3, 0.5, 0.7, 0.9]$. The search space of the outgoing edges number N is $[30, 50, 80, 100, 200]$. The search space of the path length is $[2, 3, 4]$. We also try using different activation functions for the $\mathbf{\Phi}$, such as $relu, sigmoid, tanh$.

Details of TITer. For the policy network, we set the entity embedding dimension to 80, and the function $\mathbf{\Phi}$ output dimension to 20. The node representation is the concatenation of the entity embedding and $\mathbf{\Phi}$ output, and its dimension is set to 100. we also set the relation embedding size to 100. The hidden state dimension of LSTM and the middle layer dimensions of the two two-layer MLPs are 100. We choose the latest N outgoing edges for the agent as the current state's candidate actions. N is 50 for ICEWS14 and ICEWS18, 60 for WIKI, and

[2] https://github.com/TemporalKGTeam/xERTE

[3] https://github.com/INK-USC/RE-Net

Table 8: Statistics on datasets.

Dataset	N_{train}	N_{valid}	N_{test}	N_{ent}	N_{rel}	Time granularity
ICEWS14	63685	13823	13222	7128	230	24 hours
ICEWS18	373018	45995	49545	23033	256	24 hours
WIKI	539286	67538	63110	12554	24	1 year
YAGO	161540	19523	20026	10623	10	1 year

30 for YAGO. The reasoning path length is set to 3. All the parameters are initialized with Xavier initialization.

Details of Training. Same as MINERVA, we use an additive control variate baseline to reduce the variance and add an entropy regularization term to the cost function scaled by a constant to encourage diversity in the paths sampled by the policy. The scaling constant is initialized to 0.01, and it will decay exponentially with the number of training epochs. The attenuation coefficient is 0.9. The discount factor γ of REINFORCE is 0.95. We use Adam optimizer to optimize the parameters, and the learning rate is 0.001. The weight decay of the optimizer is 0.000001. We clip gradients greater than 10 to avoid the gradient explosion. The batch size is set to 512.

Details of Testing. We use beam search to obtain a list of predicted entities with the corresponding scores. The beam size is set to 100. Multiple paths obtained through beam search may lead to the same target entity, and we keep the highest path score among them as the final entity score. For the IM module, we set the time decay factor μ to 0.1.

Details of other Methods. We use the released code to implement DE-SimplE[4], TNTComplEx[5], CyGNet[6], RE-NET[7], and xERTE[8]. We use the default parameters in the code. Partial results in Table 1 are from (Han et al., 2021; Ding et al., 2021). The authors of CyGNet only made object predictions when evaluating their model. We find that the subject prediction is more difficult than the object prediction for these four datasets. We use the code of CyGNet to train two models to predict object and subject, respectively. TANGO (Ding et al., 2021) does not release the code, so we use the results reported in their paper.

[4]https://github.com/BorealisAI/de-simple
[5]https://github.com/facebookresearch/tkbc
[6]https://github.com/CunchaoZ/CyGNet
[7]https://github.com/INK-USC/RE-Net
[8]https://github.com/TemporalKGTeam/xERTE

A.4 Model Robustness

We run TITer on all datasets five times by using five different random seeds with fixed hyperparameters. Table 9 reports the mean and standard deviation of TITer on these datasets. It shows that TITer demonstrates a small standard deviation, which indicates its robustness.

A.5 Estimating a Dirichlet Distribution

The Dirichlet density is:

$$p(\mathbf{p}) \sim Dir(\alpha_1, ..., \alpha_k) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k p_k^{\alpha_k - 1}, \tag{16}$$

where $p_k > 0$, $\sum_k p_k = 1$ and $\alpha_k > 0$. To estimate a Dirichlet distribution of order k with parameters $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_k\}$, we observe a set of samples $D = \{x_1, ..., x_N\}$ where x is a multinomial sample with length n with probability $\mathbf{p}$. n_k represents the count of corresponding category.

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\alpha}) &= \int_{\mathbf{p}} p(\mathbf{x}|\mathbf{p}) p(\mathbf{p}|\boldsymbol{\alpha}) \\ &= \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(n + \sum_k \alpha_k)} \prod_k \frac{\Gamma(n_k + \alpha_k)}{\Gamma(\alpha_k)}, \end{aligned} \tag{17}$$

$$\begin{aligned} p(D|\boldsymbol{\alpha}) &= \prod_i p(x_i|\boldsymbol{\alpha}) \\ &= \prod_i \left(\frac{\Gamma(\sum_k \alpha_k)}{\Gamma(n_i + \sum_k \alpha_k)} \prod_k \frac{\Gamma(n_{ik} + \alpha_k)}{\Gamma(\alpha_k)}\right). \end{aligned} \tag{18}$$

The gradient of the log-likelihood is:

$$\begin{aligned} \frac{d\, log(p(D|\boldsymbol{\alpha})}{d\, \alpha_k} = \sum_i \Psi\Big(\sum_k \alpha_k\Big) - \Psi\Big(n_i + \sum_k \alpha_k\Big) \\ + \Psi(n_{ik} + \alpha_k) - \Psi(\alpha_k), \end{aligned} \tag{19}$$

where Ψ is the digamma function. The maximum can be computed via the fixed-point iteration:

$$\alpha_k^{new} = \alpha_k \frac{\sum_i \Psi(n_{ik} + \alpha_k) - \Psi(\alpha_k)}{\sum_i \Psi(n_i + \sum_k \alpha_k) - \Psi(\sum_k \alpha_k)}. \tag{20}$$

Table 9: Mean and standard deviation of TITer across five runs on four datasets.

Datasets	MRR	H@1	H@3	H@10
ICEWS14	41.75±0.21	32.75±0.12	46.46±0.09	58.44±0.05
ICEWS18	29.98±0.15	22.05±0.07	33.46±0.06	44.83±0.03
WIKI	75.50±0.22	72.96±0.18	77.46±0.09	79.02±0.04
YAGO	87.47±0.08	84.89±0.07	89.96±0.03	90.27±0.04

We can also maximize the leave-one-out likelihood. The digamma function Ψ can also be inverted efficiently by using a Newton-Raphson method.