# Temporal Knowledge Graph Reasoning with Uncertainty Modeling

**Wu Zijian**
A0237336A
E0767619
Team 12
e0767619@u.nus.edu

**Li Yuexin**
A0232292L
E0718954
Team 12
e0718954@u.nus.edu

**Luo Yang**
A0243677U
E0833524
Team 12
e0833524@u.nus.edu

**Lin Siyin**
A0159082B
E0046461
Team 12
linsiyin@u.nus.edu

## Abstract

Given the past facts stored in Temporal Knowledge Graphs (TKGs), how can we predict future facts in a reliable way? The huge number of entities and the sophisticated interactions between these entities pose great challenges to pinpoint logical clues for answering TKG extrapolation queries. To address these issues, we propose an improved reinforcement learning model - **Si**mplified and **S**tabilized **T**ime **T**ravel**er** (**SiSTTer**) for TKG reasoning. Our model applies Dirichlet-reshaped rewards to help the agent search more efficiently in the temporal environment, and an Actor-Critic model to stabilize the training process of reinforcement learning. Experiments on four TKG datasets demonstrate the effectiveness of our method compared to existing reasoning models.

## 1 Introduction

Temporal Knowledge Graph (TKG) refers to a knowledge base of real world events (or facts) with timestamps indicating the occurrences of these events [1] (see Figure 1 as an example). Due to the high cost of fact annotation, TKGs are often incomplete (i.e., a portion of events are missing in TKGs). This leads to a research direction that receives increasing attention, TKG reasoning, which targets at predicting never-before-seen facts based on a number of TKGs with different timestamps.

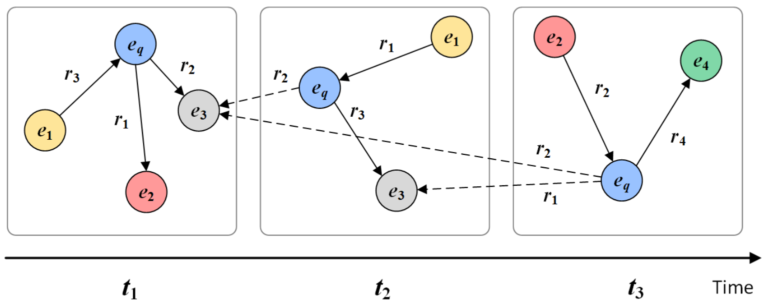

Figure 1: Illustration of a TKG with three timestamps. The dotted lines are temporal edges.

TKG reasoning can be categorized into two different settings based on its prediction objectives. One is interpolation [2] and the other is extrapolation [3]. Suppose we are given a set of TKGs with timestamps from $t_0$ to $t_N$. The interpolation setting tries to generate unseen but possibly happened events at time $t$ where $t_0 \leq t \leq t_N$. On the contrary, the extrapolation setting predicts new events at time $t$ such that $t > t_N$. In real world scenarios, we are more interested in TKG reasoning under extrapolation setting, since it helps make timely decisions, especially for those events requiring a high level of vigilance such as potential floods and upcoming wars.

The main procedure for humans to perform event reasoning is to search for relevant clues in their memories given a certain query [4]. By finding and analyzing these informative clue chains, one can obtain inference results with a certain degree of confidence. This naturally introduces uncertainty into the reasoning process. For example, clue-searching involves finding the next clue step-by-step. The search strategy should be prone to select more related clues with a higher chance, which can be modeled as a Markov Decision Process (MDP). Based on the above analysis, it is both reasonable and feasible to accomplish TKG reasoning with uncertainty modeling and deep learning techniques.

Motivated by several state-of-the-art works (e.g., RE-NET [3], MultihopKG [5], CluSTeR [6], TITer [7]), we aim to develop an improved reinforcement learning model to solve the extrapolative TKG reasoning problem from the perspective of uncertainty modeling. Specifically, we propose the **Si**mplified and **S**tabilized **T**ime **T**travel**er** (**SiSTTer**) with two highlighted features. To integrate prior knowledge of answer locations in temporal axis, we apply pre-estimated Dirichlet distributions for each relation to reshape the original reward so that the agent can be better guided temporally. Additionally, we further design an Actor-Critic model in order to enable efficient and stable clue searching for queries under complex TKG environment. Extensive experiments are conducted on four widely used TKG datasets (ICEWS14, ICEWS18, WIKI, YAGO), and the results show that, with the help of these uncertainty modeling approaches, our model achieves comparable or even superior performance to six existing TKG reasoning models.

## 2 Related Work

**Static KG Reasoning.** Embedding-based methods [8, 9] which represent entities and relations as low-dimensional dense embeddings in different vector spaces like complex space [10] and relational space [11, 12] have drawn broad attention recently. These methods predict missing facts by scoring candidate facts based on the learned distributed representations. Among then, some works use Graph Neural Network (GNN) [13–15] to encode multi-hop and multi-relational information for the KGs [16, 17]. Besides, rule and path based methods [18, 19, 5] are also widely employed to make the reasoning process explainable. However, theses static methods underestimate the temporal information in KGs, which limits their usage in more complex reasoning tasks [20].

**Temporal KG Reasoning.** A considerable amount of works [21, 22] include temporal information to deal with the temporal dependencies among facts in TKGs. Some works design message-passing and aggregation networks [17, 23, 24] to jointly capture graph local and global information. These works are designed for interpolation. For extrapolation tasks, Know-Evolve [21] use temporal point process to capture entity and relation representation evolved in the continuous time domain. Additionally, RE-NET [3] uses a subgraph aggregator to capture graph information at different timestamps and a GRU [25] to model the subgraph sequence information. These methods use heuristic strategies in the clue searching state and use all the historical information, which can engage some noise. Besides, these two models are both computationally expensive [24] in the temporal reasoning stage, and they cannot provide intractability for the results since they take all historical events into consideration.

**Reinforcement Learning.** Recently, RL [26, 27] has witnessed a large amount of applications in various domains include knowledge graph reasoning [18, 19, 5]. Compared to the domain of games and many other applications, RL formulations in knowledge graph often have a large action space (e.g., in knowledge graph, the space of possible actions of one entity is the entire outgoing edges, and some entities may have thousands of neighbors such as USA). For the reason that there is no guaranteed golden path for a KG reasoning problem, we cannot use supervised learning to give the path a higher score. RL is a promising methods under this occasion given a well designed environment and reward setting.

## 3 Methodology

In this section, we first define the problem in section 3.1. Then we introduce the MDP framework for on-policy reinforcement learning and the Dirichlet distribution-based reward reshaping in section 3.2. In section 3.3 and 3.4, we describe our proposed solutions to the slow convergence problem: learnable embedding modules and Actor-Critic algorithm. Figure 2 is an overview of our SiSTTer model. The source code of our project can be found here [1].

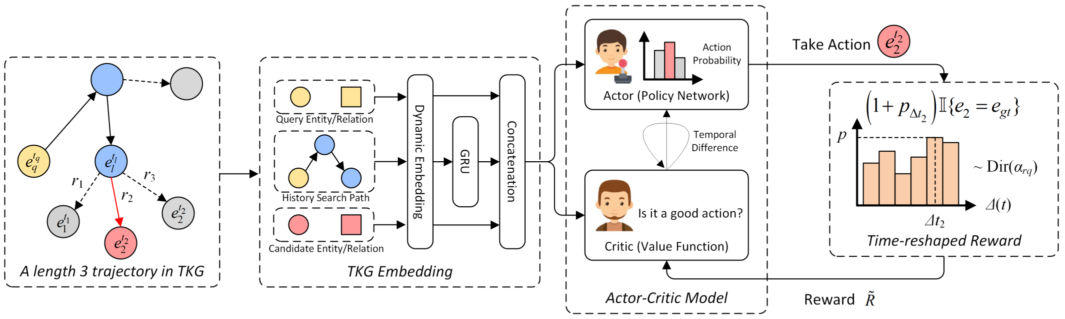

Figure 2: An overview of the SiSTTer model. The figure is an example of one search step at current state $s_l = (e_q, t_q, r_q, e_l, t_l)$ given a query $(e_q, r_q, ?, t_q)$. SiSTTer receives the query, the history search path and candidate actions as input, and feed them into the TKG embedding layer. The embedded state is used in the actor to obtain a list of action probabilities. The agent then takes a sampled action $a_l = (r_2, e_2, t_2)$ and receives a reshaped reward $\tilde{R}$. The critic leverages the reward $\tilde{R}$ and the embedded state to generate a temporal difference score that helps update both the Actor and the Critic itself.

### 3.1 Problem Formulation

For extrapolated TKG reasoning, given a known TKG, we do link prediction at future periods to forecast the evolution of TKGs through time.

Let $\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{F}$ denote the sets of entities, relations, timestamps, and facts. The graph snapshots through time can be used to depict a TKG by: $\mathcal{G}(1,T) = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T\}$, where $\mathcal{G}_t = \{\mathcal{E}_t, \mathcal{R}, \mathcal{F}_t\}$ is a multi-relational directed TKG snapshot. For each fact in $\mathcal{F}$, it can be represented in quadruple $(e_s, r, e_o, t)$, where $e_s \in \mathcal{E}$ and $e_o \in \mathcal{E}$ are subject entity and object entity respectively, and $r \in \mathcal{R}$ is the directed link between $e_s$ and $e_o$ at time $t \in \mathcal{T}$.

Overall, we are given a set of known facts $\{(e_{si}, r_i, e_{oi}, t_i) | t_i < t_q\}$ which constitute the known TKGs, and we need to predict the missing object or subject entity in the query $(e_q, r_q, ?, t_q)$ or $(?, r_q, e_q, t_q)$ at the future time $t_q$.

### 3.2 MDP Framework for Reinforcement Learning

We establish a MDP framework for the reinforcement learning process of the SiSTTer model. We follow the standard MDP formulation that mainly consists of the following four components.

**States.** The state is represented by a quintuple $s_l = (e_q, t_q, r_q, e_l, t_l) \in \mathcal{S}$. $e_q, t_q, r_q$ refer to the query entity, the query time and the query relation, which remain unchanged during the whole search process for a given query. $e_l$, and $r_l$ are the current entity and relation in search step $l$, respectively. $\mathcal{S}$ is the state space of the TKG reasoning problem. By combining the query and current information together, the agent can both be aware of its goal and take actions heuristically.

**Actions.** The action space $\mathcal{A}_l$ ($\mathcal{A}_l \in \mathcal{A}$, $\mathcal{A}$ is the complete action space) for a state $s_l$ is defined by $\mathcal{A}_l = \{(r', e', t') | (e_l, r', e', t') \in \mathcal{F}, t' \leq t_l, t' < t_q\}$. Based on previous studies[7], the original action space $\mathcal{A}_l$ can be very large, which leads to a huge search space. Hence, we sample a set of outgoing relation-entity-time triplets from the original $\mathcal{A}_l$ as the real action space.

[1]Code Repository: `https://github.com/zjwu0522/CS5340`.

**Transitions.** The transition function $\xi$ is defined by $s_{l+1} = \xi(s_l, a_l) = (e_q, t_q, r_q, e_{l+1}, t_{l+1})$, where $a_l \in \mathcal{A}_l$. In other words, the transition is deterministic because the selected outgoing relation-entity-time triplet $a_l$ leads to one entity node only.

**Rewards.** The agent receives a reward $R(s_L) = 1$ if and only if it arrives at the correct target entity $e_{gt}$ (i.e., ending up with terminal state $s_L = (e_q, t_q, r_q, e_L, t_L)$ where $e_L = e_{gt}$ and $(e_q, r_q, e_{gt}, t_q) \in \mathcal{F}$), while receiving 0 reward in other cases. However, due to the variability and sparsity of the temporal distribution of the target entities[5, 28], the agent may not learn efficiently with a fixed value of reward. Thus, a reward-reshaping mechanism is derived in the following subsection.

### 3.2.1 Dirichlet Distribution for Reward-reshaping

The temporal variability and sparsity are the challenges that the model can access *variable* amounts of reference temporal information in near TKG snapshots when answering different queries, and only *a small fraction* of entities are active within each TKG snapshot[5, 28]. This results in a temporal distribution for the answer entity of each query. By reshaping the reward based on this prior knowledge, the agent can know which TKG snapshot has a higher chance to possess the answer[7]. Here, a Dirichlet distribution that is pre-estimated for relation $r_q$ from the previous $K$ TKG snapshots is used to reshape the reward:

$$\tilde{R}(s_L) = (1 + p_{\Delta_{t_L}})R(s_L) \tag{1}$$

where $\Delta_{t_L} = t_q - t_L$, $(p_1, p_2, ..., p_K) \sim \text{Dir}(\boldsymbol{\alpha}_{r_q})$, and $\boldsymbol{\alpha}_{r_q} \in \mathbb{R}^K$ is the parameter of Dirichlet distribution.

**Estimation.** For each relation $r$, we count the number of times the object entity $s_o$ appears in each of the previous $K$ historical snapshots to form a multinomial sample $\mathbf{p}_{r,i} \in \mathbb{R}^K$. By repeating the process through time axis in the whole training set, we can obtain a set of multinomial samples $\mathcal{D}_r = \{\mathbf{p}_{r,1}, \mathbf{p}_{r,2}, ..., \mathbf{p}_{r,N}\}$. $\mathcal{D}_r$ can be used to estimate a Dirichlet distribution via Maximum Likelihood Estimation (MLE) since it is the conjugate prior of multinomial distribution. Specifically, the goal of MLE is to obtain $\boldsymbol{\alpha}_r \in \mathbb{R}^K$ for the following distribution:

$$\text{Dir}(\boldsymbol{\alpha}_r) = \text{Dir}(\alpha_{r,1}, \alpha_{r,2}, ..., \alpha_{r,K}) = \frac{\Gamma(\sum_{k=1}^K \alpha_{r,k})}{\prod_{k=1}^K \Gamma(\alpha_{r,k})} \prod_{k=1}^K p_{r,k}^{\alpha_{r,k}-1} \tag{2}$$

MLE tries to maximize the likelihood $p(\mathcal{D}_r|\boldsymbol{\alpha}_r) = \prod_{i=1}^N p(\mathbf{p}_{r,i}|\boldsymbol{\alpha}_r)$. Its corresponding logarithm is

$$\log p(\mathcal{D}_r|\boldsymbol{\alpha}_r) = N \log \Gamma(\sum_{k=1}^K \alpha_{r,k}) - N \sum_{k=1}^K \log \Gamma(\alpha_{r,k}) + N \sum_{k=1}^K (\alpha_{r,k} - 1) \log \bar{p}_{r,k} \tag{3}$$

where $\log \bar{p}_{r,k} = \frac{1}{N} \sum_{i=1}^N p_{r,i,k}$. This log-likelihood is convex w.r.t $\boldsymbol{\alpha}_r$ since Dirichlet belongs to the exponential family, which implies that the gradient of log-likelihood can be utilized for optimization. The gradient w.r.t $\alpha_{r,k}$ is

$$\nabla_{\alpha_{r,k}} \log p(\mathcal{D}_r|\boldsymbol{\alpha}_r) = \frac{\partial \log p(\mathcal{D}_r|\boldsymbol{\alpha}_r)}{\partial \alpha_{r,k}} = N\Psi(\sum_{k=1}^K \alpha_{r,k}) - N\Psi(\alpha_k) + N \log \bar{p}_{r,k} \tag{4}$$

where $\Psi(x) = \frac{d \log \Gamma(x)}{dx} = \frac{\Gamma'(x)}{\Gamma(x)}$ is the digamma function. The exponential family offers a property that when setting this gradient to zero, the expected sufficient statistics are equal to the observed sufficient statistics (i.e., $\mathbb{E}(\log \bar{p}_{r,k}) = \Psi(\alpha_{r,k}) - \Psi(\sum_{k=1}^K \alpha_{r,k})$). Then we can derive a fixed-point iteration according to [29] to obtain the estimated $\boldsymbol{\alpha}_r$.

$$\alpha_{r,k}^{\text{new}} = \Psi^{-1}(\Psi(\sum_{k=1}^K \alpha_{r,k}^{\text{old}}) - \log \bar{p}_{r,k}) \tag{5}$$

## 3.3 Policy Network

We design a temporal difference Actor-Critic policy network $\pi_\theta(a_l|s_l) = P(a_l|s_l;\theta)$ to model the action of the agent in a continuous space. Specifically, the designed policy network consists of the following three embedding modules.

**Dynamic embedding.** For each relation $r \in \mathcal{R}$ we learn a dense vector representation $\mathbf{r} \in \mathbb{R}^{d_r}$. To make the entity representation aware of temporal information and evolve over time, we concatenate a relative time vector representation after entity representation. For this part, we use a dynamic embedding module to represent each node $e_i^t = (e_i, t)$ in $\mathcal{G}_t$, where using $\mathbf{e} \in \mathbb{R}^{d_e}$ to represent the local invariant representation of entities. The relative time encoding function is defined as $\mathbf{\Phi}(\Delta t) \in \mathbb{R}^{d_t}$, where $\Delta t = t_q - t$ and $\mathbf{\Phi}(\Delta t)$ is formulated as follows:

$$\mathbf{\Phi}(\Delta t) = \sigma(\mathbf{w}\Delta t + \mathbf{b}) \tag{6}$$

where $\mathbf{b}, \mathbf{w} \in \mathbb{R}^{d_t}$ are vectors with learnable parameters and $\sigma$ is an activation function. The final temporal-aware representation of a node $e_i^t$ is denoted as: $\mathbf{e}_i^t = [\mathbf{e}_i; \mathbf{\Phi}(\Delta t)]$.

**Path encoding.** The search history $h_l = (e_q, r_q, e_1, ..., r_l, e_l)$ consists of the sequence of observations and actions taken up to time step $t$. The agent encodes the history $h_l$ with a GRU:

$$\mathbf{h}_l = \text{GRU}(\mathbf{h}_{l-1}, [\mathbf{r}_{l-1}; \mathbf{e}_{i-1}^{t_{l-1}}]), \tag{7}$$

$$\mathbf{h}_0 = \text{GRU}(\mathbf{0}, [\mathbf{r}_0; \mathbf{e}_q^{t_q}]). \tag{8}$$

where $\mathbf{r}_0$ is a special start relation introduced to form a start action with $e_q$ and we keep the GRU state unchanged when the last action is self-loop.

**Action scoring.** To evaluate each possible action, we score each option and calculate the probability of state transition. We denote every action $a_n = (e_n, t_n, r_n) \in \mathcal{A}_l$ as an possible outgoing action at step $l$. For the reason that future unseen events are usually uncertain, and strong causal logic chain for some queries is hard to find, so more attention should be payed to the correlation between the entity and query. We adopt 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 $\tilde{e}$ and outgoing edge $\tilde{r}$ representations. The agent will then calculate the destination node score and outgoing edge score of the optional candidate action by calculating there normalised cosine similarity. Added the weighted term to control the importance between these two scores, the agent obtains the final possible candidate action score $\phi(a_n, s_l)$:

$$\phi(a_n, s_l) = \beta_n \langle \tilde{\mathbf{e}}, \mathbf{e}_n^{t_n} \rangle + (1 - \beta_n) \langle \tilde{\mathbf{r}}, \mathbf{r}_n \rangle, \tag{9}$$

$$\tilde{\mathbf{e}} = \mathbf{W}_e \text{ReLU}(\mathbf{W}_1[\mathbf{h}_l; \mathbf{e}_q^{t_q}; \mathbf{r}_q]), \tag{10}$$

$$\tilde{\mathbf{r}} = \mathbf{W}_r \text{ReLU}(\mathbf{W}_1[\mathbf{h}_l; \mathbf{e}_q^{t_q}; \mathbf{r}_q]), \tag{11}$$

$$\beta_n = \text{sigmoid}(\mathbf{W}_\beta[\mathbf{h}_l; \mathbf{e}_q^{t_q}; \mathbf{r}_q; \mathbf{e}_n^{t_n}; \mathbf{r}_n]), \tag{12}$$

where $\mathbf{W}_1$, $\mathbf{W}_e$, $\mathbf{W}_r$ and $\mathbf{W}_\beta$ are learnable matrices. In brief, the parameters of GRU, MLP and $\Phi$, the embedding matrices of relation and entity form the learnable parameters in $\theta$.

## 3.4 Actor-Critic Optimization and Training

In the training process, the search path length is fixed to $L$, and an $L$-lenghth trajectory will be generated according to the learned policy network $\pi_\theta : a_1, a_2, ..., a_L$. Previous works usually use the REINFORCE [30] algorithm, which iterates through all $(e_s, r, e_o, t)$ quadruples in $\mathcal{G}$ and updates $\theta$ with the following stochastic gradient:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_t \tilde{R}(s_T|e_s, r, t) \log \pi_\theta(a_t|s_t). \tag{13}$$

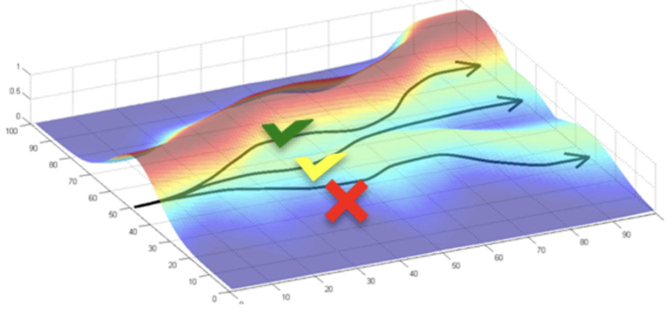

Figure 3: Slow convergence: monte carlo sampling.

However, the steps taken during policy update do not guarantee an improvement in the performance and since the policy gradient depends on the set of trajectories collected, the complete algorithm gets trapped in a bad loop and is not always able to recover. This means that just one bad step can completely destroy hours of training and force the algorithm to learn from the start. Additionally, REINFORCE algorithm improve probability of trajectories with positive reward and vice versa. In a knowledge graph environment with the above reward design, all trajectories have positive rewards with some large than others. Figure 3 shows that under the condition of monte carlo sampling, it is unsatisfactory since at the earlier state of training, the agent almost random selects outgoing action and receives extremely sparse rewards, which leads to unacceptable slow convergence.

To alleviate this slow convergence problem, we adopt Temporal Difference Actor-Critic [31] for optimization and training. In a simple term, Actor-Critic is a Temporal Difference version of Policy gradient. It has two networks: Actor and Critic. The actor decided which action should be taken and critic inform the actor how good was the action and how it should adjust. The learning of the actor is based on policy gradient approach. In comparison, critics evaluate the action produced by the Actor by computing the value function. The new modified advantage function for Actor-Critic is:

$$\tilde{A}_{\pi_\theta}(s_T|e_s, r, t) = \tilde{R}(s_T|e_s, r, t) + V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t). \tag{14}$$

Alternatively, advantage function is called as TD error as shown in the Actor-Critic framework. As mentioned above, the learning of the actor is based on policy-gradient. The policy gradient expression of the actor as shown below:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{m\in[1,L]} \tilde{A}_{\pi_\theta}(s_T|e_s, r, t) \log \pi_\theta(a_l|s_l). \tag{15}$$

## 4 Experiment

### 4.1 Setup

**Datasets.** The evaluation uses four public TKG datasets: ICEWS14, ICEWS18 [32], YAGO[33] and WIKI [34]. The ICEWS (Integrated Crisis Early Warning System) is an event dataset. ICEWS14 and ICEWS18 are subsets of ICEWS events occurring in 2014 and 2018 with a time granularity of days. In terms of the two knowledge bases, WIKI and YAGO, which collect facts with time information, we use subsets of the fact sets with a time granularity of years.

**Evaluation Metrics.** The model is evaluated with TKG forecasting, a task that predicts links at future timestamps. Performance metrics include Mean Reciprocal Rank (MRR) and Hits@1/3/10. We evaluate two queries for each quadruple $(e_s, r, e_o, t)$ in the test set, $(e_s, r, ?, t)$ and $(?, r, e_o, t)$. With time-aware filtering scheme, we only filter out quadruples with query times $t$. It is more reasonable than the method used in [3] [35]. **MRR** is a measure to evaluate systems that return a ranked list of answers to queries. 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 $q_s = (r, r, e_o, t)$. Our model ranks entities according to their transition probability for each query. In cases where the ground truth entity does not appear in the final set of

Table 1: Results on two ICEWS datasets. The results of MRR and Hits@1/3/10 are multiplied by 100. The best results are in bold.

| | **ICEWS14** | | | | **ICEWS18** | | | |
|---|---|---|---|---|---|---|---|---|
| **Methods** | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 |
| TNTComplEx | 32.12 | 23.35 | 36.03 | 49.13 | 27.54 | 19.52 | 30.80 | 42.86 |
| CyGNet | 32.73 | 23.69 | 36.31 | 50.67 | 24.93 | 15.90 | 28.28 | 42.61 |
| RE-NET | 38.28 | 28.68 | 41.34 | 54.52 | 28.81 | 19.05 | 32.44 | **47.51** |
| xERTE | 40.79 | **32.70** | 45.67 | 57.30 | 29.31 | 21.03 | **33.51** | 46.48 |
| TANGO-Tucker | - | - | - | - | 28.68 | 19.35 | 32.17 | 47.04 |
| TANGO-DistMult | - | - | - | - | 26.75 | 17.92 | 30.08 | 44.09 |
| Our Model | **41.34** | 31.81 | **46.52** | **58.72** | **29.60** | **21.63** | 33.11 | 44.75 |

Table 2: Results on WIKI and YAGO. The results of MRR and Hits@1/3/10 are multiplied by 100. The best results are in bold.

| | **WIKI** | | | | **YAGO** | | | |
|---|---|---|---|---|---|---|---|---|
| **Methods** | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 |
| TNTComplEx | 45.03 | 40.04 | 49.31 | 52.03 | 57.98 | 52.92 | 61.33 | 66.69 |
| CyGNet | 33.89 | 29.06 | 36.10 | 41.86 | 52.07 | 45.36 | 56.12 | 63.77 |
| RE-NET | 49.66 | 46.88 | 51.19 | 53.48 | 58.02 | 53.06 | 61.08 | 66.29 |
| xERTE | 71.14 | 68.05 | 76.11 | **79.01** | 84.19 | 80.09 | 88.02 | 89.78 |
| TANGO-Tucker | 50.43 | 48.52 | 51.47 | 53.58 | 57.83 | 53.05 | 60.78 | 65.85 |
| TANGO-DistMult | 51.15 | 49.66 | 52.16 | 53.35 | 62.70 | 59.18 | 60.31 | 67.90 |
| Our Model | **74.44** | **71.99** | **76.14** | 77.85 | **86.60** | **83.50** | **90.80** | **91.46** |

searched entities, we set the rank to the number of entities in the dataset. The equation for MRR is defined as:

$$MRR = \frac{1}{2 * |\mathcal{F}_{test}|} \sum_{q \in \mathcal{F}_{test}} \left(\frac{1}{rank(e_o|q_o)} + \frac{1}{rank(e_s|q_s)}\right) \tag{16}$$

Besides, **Hits@k** measures the percentage of cases in which the true triple appears in the top $k$ ranked triples.

**Baseline.** We compare our model with an existing interpolated TKG reasoning method named TNT-ComplEx [36], and state-of-the-art extrapolated TKG reasoning approaches, including RE-NET [3], CyGNet [35], TANGO [37], and xERTE [38].

### 4.2 Results

Our model and baselines on four TKG datasets are presented in the Table 1 and the Table 2. By using beam search, our model can search multiple candidates at once. The results in Table 1 show that our model outperforms all baselines on the ICEWS14 dataset when evaluated by Hits@3 and Hits@5 metrics, and that it achieves the most optimal performance on the ICEWS18 dataset when evaluated by MRR and H@1. Table 2 further illustrates our model's excellent performance. In the case of the YAGO dataset, our model outperforms all baselines when evaluated by all metrics.

### 4.3 Ablation Study

Figure 4 shows our model's performance on ICEWS18 and WIKI datasets compared with the variant without reward reshaping. We observe that reward reshaping improves the model's performance to some degree. The reason for the performance's promotion is that our model acquires knowledge

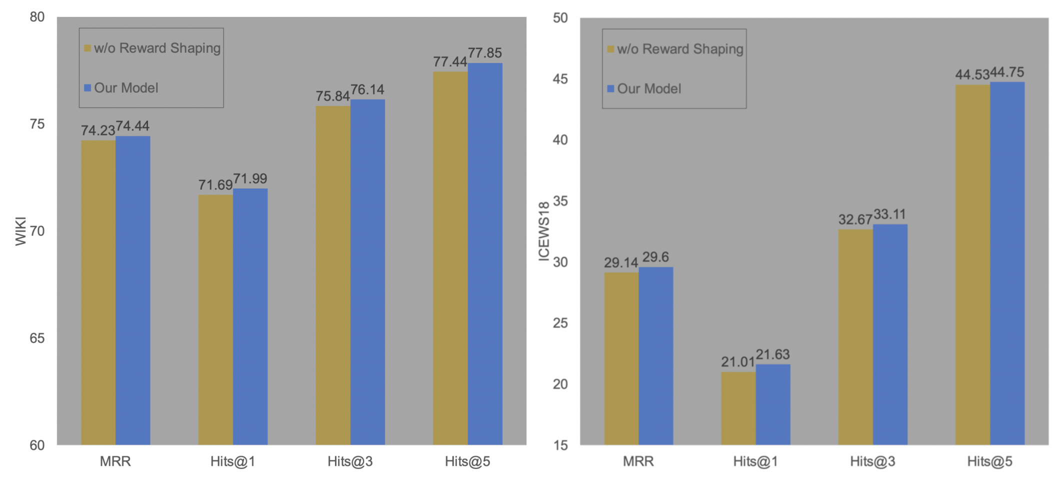

Figure 4: Ablation of Reward Reshaping.

about the probability distribution of the target's appearance over the entire period of time by utilizing the Dirichlet prior distribution to direct the decision process.

## 5 Conclusion

In this project, we propose an improved reinforcement learning model called Simplified and Stabilized Time Traveler (SiSTTer) for TKG reasoning with uncertainty modeling. SiSTTer forecasts the evolution of TKGs by relying on historical snapshots and exploring the temporal evidence chain. Our method models amounts of temporal information within near TKG snapshots by reshaping reward with Dirichlet prior distribution and applies Actor-Critic algorithm to optimize the search process in the temporal environment and stabilize the training process for reinforcement learning. We evaluate our model on four widely used TKG datasets (ICEWS14, ICEWS18, WIKI, YAGO), and the experiments demonstrate our model achieves competitive results compared with some existing methods.

## References

[1] Gottschalk, S., E. Demidova. Eventkg–the hub of event knowledge on the web–and biographical timeline generation. *Semantic Web*, 10(6):1039–1070, 2019.

[2] García-Durán, A., S. Dumančić, M. Niepert. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821. Association for Computational Linguistics, Brussels, Belgium, 2018.

[3] Jin, W., M. Qu, X. Jin, et al. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6669–6683. Association for Computational Linguistics, Online, 2020.

[4] Evans, J. S. B. T. Dual-processing accounts of reasoning, judgment, and social cognition. *Annual Review of Psychology*, 59(1):255–278, 2008. PMID: 18154502.

[5] Lin, X. V., R. Socher, C. Xiong. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253. Association for Computational Linguistics, Brussels, Belgium, 2018.

[6] Li, Z., X. Jin, S. Guan, et al. Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4732–4743. Association for Computational Linguistics, Online, 2021.

[7] Sun, H., J. Zhong, Y. Ma, et al. TimeTraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8306–8319. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021.

[8] Bordes, A., N. Usunier, A. Garcia-Duran, et al. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[9] Wang, Z., J. Zhang, J. Feng, et al. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28. 2014.

[10] Trouillon, T., J. Welbl, S. Riedel, et al. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[11] Ji, G., S. He, L. Xu, et al. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696. 2015.

[12] Lin, Y., Z. Liu, M. Sun, et al. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*. 2015.

[13] Hamilton, W., Z. Ying, J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] Kipf, T. N., M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Veličković, P., G. Cucurull, A. Casanova, et al. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[16] Schlichtkrull, M., T. N. Kipf, P. Bloem, et al. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[17] Shang, C., Q. Liu, K.-S. Chen, et al. Edge attention-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1802.04944*, 2018.

[18] Xiong, W., T. Hoang, W. Y. Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

[19] Das, R., S. Dhuliawala, M. Zaheer, et al. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.

[20] Chen, X., S. Jia, Y. Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

[21] Trivedi, R., H. Dai, Y. Wang, et al. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, pages 3462–3471. PMLR, 2017.

[22] Ji, S., S. Pan, E. Cambria, et al. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[23] Goel, R., S. M. Kazemi, M. Brubaker, et al. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pages 3988–3995. 2020.

[24] Li, Z., X. Jin, W. Li, et al. Temporal knowledge graph reasoning based on evolutional representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 408–417. 2021.

[25] Chung, J., C. Gulcehre, K. Cho, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[26] Mnih, V., K. Kavukcuoglu, D. Silver, et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[27] Sutton, R. S., D. McAllester, S. Singh, et al. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[28] Wu, J., M. Cao, J. C. K. Cheung, et al. TeMP: Temporal message passing for temporal knowledge graph completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5730–5746. Association for Computational Linguistics, Online, 2020.

[29] Minka, T. Estimating a dirichlet distribution, 2000.

[30] Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[31] Konda, V., J. Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[32] Boschee, E., J. Lautenschlager, S. O'Brien, et al. ICEWS Coded Event Data, 2015.

[33] Mahdisoltani, F., J. Biega, F. M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*. www.cidrdb.org, 2015.

[34] Leblay, J., M. W. Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1771–1776. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018.

[35] Zhu, C., M. Chen, C. Fan, et al. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *AAAI*. 2021.

[36] Lacroix, T., G. Obozinski, N. Usunier. Tensor decompositions for temporal knowledge base completion. *ArXiv*, abs/2004.04926, 2020.

[37] Han, Z., Z. Ding, Y. Ma, et al. Temporal knowledge graph forecasting with neural ode, 2021.

[38] Han, Z., P. Chen, Y. Ma, et al. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*. 2021.