Two-Stage Temporal Knowledge Graph Completion Based on Reinforcement Learning

Dong Li[1], Yong Wei[1], Xinyi Dong[1], Jingyou Sun[1], LinLin Ding[1](✉), and Yue Kou[2]

[1] Liaoning University, Shenyang, China
dinglinlin@lnu.edu.cn
[2] Northeastern University, Shenyang, China

Abstract. Knowledge graphs have become an indispensable technology for organizing and processing vast amounts of information, with applications spanning intelligent robotics, risk management, recommender systems, and healthcare analytics. However, the effectiveness of knowledge graphs in downstream tasks is often limited by inherent structural incompleteness. To address this issue, we propose TS-TKGC (Two-Stage Temporal Knowledge Graph Completion), a novel reinforcement learning-based method. Our TS-TKGC method consists of two key stages: clue search and temporal reasoning. In the first stage, reinforcement learning is employed to identify informative clues. In the second stage, the method integrates Gated Recurrent Units (GRU) for temporal reasoning, alongside a multi-dimensional reward mechanism to optimize the training strategy. Finally, experimental results validate the feasibility and effectiveness of the proposed key technique, demonstrating the model's capability to enhance temporal knowledge graph completion.

Keywords: Temporal Knowledge Graphs · Two-Stage Knowledge Completion · Reinforcement Learning · Clue Search · Gated Recurrent Units.

1 Introduction

In the digital era, the information explosion has provided abundant data resources for intelligent services. However, such data are often characterized by high redundancy, structural heterogeneity, and low utilization efficiency, necessitating the construction of domain-specific knowledge bases with refined management. As an extension of traditional knowledge graphs, temporal knowledge graphs incorporate timestamps to evolve fact representations from static triples (h, r, t) to dynamic quadruples (h, r, t, τ). By not only modeling entity-relationship structures but also capturing their temporal dynamics, temporal knowledge graphs enable advanced capabilities such as trend analysis, temporal querying, and event prediction. These enhancements significantly empower applications in decision support, risk assessment, and cybersecurity, among other domains.

Temporal knowledge graph completion enhances dynamic knowledge representation by modeling facts as timestamped quadruples (h, r, t, τ). This approach

captures both explicit event occurrences and implicit temporal patterns, enabling sophisticated temporal reasoning capabilities. Despite its practical value, current temporal knowledge graph completion approaches exhibit significant limitations: rule-based methods are hindered by poor scalability from rule explosion and ineffective temporal modeling; GNN-based techniques struggle with frequent entity/relation changes and long-term dependency capture; while reinforcement learning frameworks face challenges in reward mechanism design and suffer from sparse feedback-induced learning inefficiency. Crucially, existing methods fail to effectively bridge temporal evolution with structural semantics, creating a disconnect between dynamic patterns and graph topology. Although reinforcement learning offers interpretable temporal reasoning, fundamental challenges remain unresolved, particularly the joint modeling of temporal dynamics and semantic relationships, coupled with severe training inefficiencies stemming from inadequate reward signals in sparse environments.

In this paper, we propose a temporal knowledge graph completion model that incorporates reinforcement learning to address these challenges. The main contributions of this paper include: (1) We propose a novel temporal knowledge graph completion framework integrating clue searching and temporal reasoning to jointly model temporal information and structural characteristics; (2) We propose an interpretable reinforcement learning mechanism for enhanced performance in complex temporal reasoning; (3) We design a collaborative training mechanism that synchronizes clue searching and temporal reasoning, allowing adaptive optimization under varying temporal constraints; (4) Extensive experiments on multiple benchmark datasets validate the model's effectiveness, achieving better performance on temporal knowledge graph completion tasks.

2 Related Work

In this section, we review related work on temporal knowledge graph completion and highlight our technical contributions. Existing temporal knowledge graph completion approaches fall into three main categories: rule-based approaches, graph neural network (GNN)-based approaches, and reinforcement learning-based approaches.

2.1 Rule-based Temporal Knowledge Graph Completion

In static environments, rule-based temporal knowledge graph completion demonstrates strong interpretability and reliability. Early approaches relied on manually defined temporal logic rules for reasoning. For instance, the approach proposed in [1] achieved 89.6% prediction precision in biomedical event forecasting by designing 14 temporal clause rules through multi-path dependency analysis. Subsequent work by Bai et al. [2] enhanced this framework using Bayesian pruning strategies, maintaining temporal confidence scores above 0.85. Recent advances have focused on automated rule extraction. For example, the ALREIR model [3] leverages temporal attention mechanisms to autonomously discover

logical patterns, achieving a 23.7% improvement in F1-score over manual rule-engineering methods while preserving model interpretability.

2.2 GNN-based Temporal Knowledge Graph Completion

GNN-based approaches have significantly advanced temporal knowledge graph completion by jointly learning structural and temporal entity representations. RENET [4] pioneered this direction with multi-relational graph aggregation for structural feature extraction. Subsequent innovations in variants of GNN-based approaches have further enhanced temporal modeling. For example, the method proposed in [5] incorporates GRU gates conditioned on event frequency to capture temporally inactive dependencies. REGCN [6] employs autoregressive GRU networks to model temporal event dynamics. DACHA [7] introduces architectural innovation through dual-graph convolutional networks, synchronizing information flow between primal graphs and their edge-graph counterparts. Notably, Zhu et al. [8] propose an advanced framework with implicit knowledge propagation, enabling concurrent future event prediction via three synergistic modules: entity vocabulary pattern recognition, temporal transaction recurrence detection, and known fact reference mechanisms.

2.3 Reinforcement Learning-based Temporal Knowledge Graph Completion

Recent advances in reinforcement learning for temporal knowledge graph completion have highlighted the critical importance of sophisticated reward design [9], where early binary reward mechanisms [10] established performance baselines while subsequent innovations progressively addressed reward sparsity through temporal pattern encoding and exploration optimization. Notable developments include path diversity rewards [11] for multi-pattern exploration, Dirichlet temporal allocation [12] for time-aware reward distribution, and LSTM-enhanced beam rewards [13] for sequential decision optimization.

Compared with the above models, our proposed TS-TKGC model makes three significant advances: first, it adopts a unified framework integrating clue searching with temporal reasoning to jointly capture semantic dependencies and temporal evolution patterns; second, it utilizes a two-stage collaborative learning mechanism that effectively mitigates the persistent challenge of reward sparsity in reinforcement learning; third, it implements dynamic temporal constraint-aware path adjustment to simultaneously enhance exploration efficiency and task performance in complex scenarios.

3 Problem Definition

This section mainly introduces the concepts involved in temporal knowledge graph completion and then formally defines the problem of temporal knowledge graph completion.

Definition 1 (Quadruple). *A quadruple* $(h, r, t, timestamp)$ *consists of four parts: head entity, relation, tail entity, and timestamp. The quadruple represents a fact triplet with temporal information, indicating that at time* t*, the triplet* (h, r, t) *holds.*

Definition 2 (Temporal Knowledge Graph). *A temporal knowledge graph is an extension of traditional knowledge graph in the temporal dimension, aiming to record the relationships and evolution processes of things over time.*

In temporal knowledge graphs, each fact is represented by a quadruple consisting of a head entity, a relation, a tail entity, and a timestamp associated with the fact. By incorporating temporal attributes, temporal knowledge graphs can reflect the states and changes of things at different time points.

Temporal knowledge graph uses entities as nodes, relationships to represent connections between entities, and timestamps to indicate the temporal attributes of current facts. They can be represented by the quadruple $G = (E, R, T, F)$, where E denotes the set of entities (e.g., Xiaomi Company, Li Auto, China, Physics), R denotes the set of relationships between entities (e.g., father-son, bordering, possession), T denotes the set of timestamps (e.g., 2021, March 2nd, 8 a.m.), and F denotes the set of quadruples (e.g., (COVID-19, outbreak, Wuhan, Hubei Province, December 2019), (Alibaba Group, acquires, Ele.me, April 2, 2018), (Tencent stock price, is, 400 yuan, April 30, 2023)).

Definition 3 (Temporal Knowledge Graph Completion). *Temporal knowledge graph completion refers to the process of using algorithms and technical means to complete or supplement missing entities, relationships, or quadruples in temporal knowledge graph with temporal labels.*

Temporal knowledge graph completion addresses evolving entity-relationship dynamics by inferring valid missing facts. For example, in a corporate leadership knowledge graph tracking directors' positions across time intervals, temporal gaps may exist in role transition records. Temporal knowledge graph completion predicts such missing position changes and their precise timestamps by leveraging existing temporal patterns.

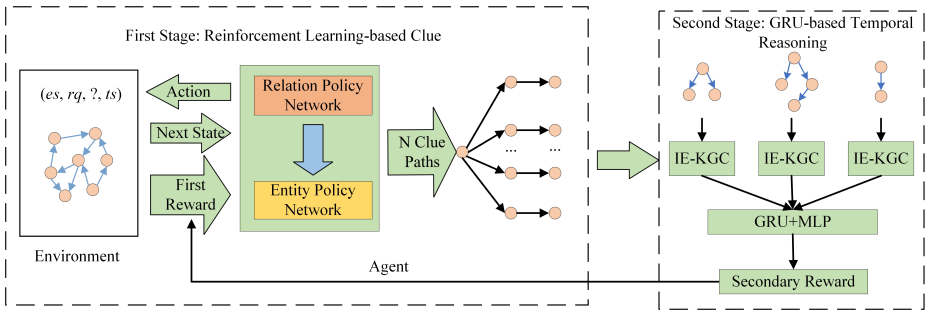


Fig. 1: Model Overview

It is to infer the missing components of quadruples in a given temporal knowledge graph G, based on the existing fact quadruples $\{(e_h, r, e_t, t) \in F\}$. Specifically, it aims to predict the missing tail entity in quadruples of the form $(e_h, r, ?, t)$, the missing head entity in quadruples of the form $(?, r, e_t, t)$, or the missing relation in quadruples of the form $(e_h, ?, e_t, t)$.

The proposed model operates in two stages: 1) clue searching: identifies temporally constrained clue paths relevant to the query, capturing rich semantic information for reasoning. 2) temporal reasoning: processes clue paths and candidate entities to capture temporal dependencies and generate final predictions. This integrated approach maintains information accuracy while handling temporal dynamics, improving prediction precision (Fig. 1).

4 Temporal Knowledge Graph Completion Model

4.1 Model Architecture

In the first stage, the clue searching process is formulated as a Markov Decision Process (MDP), incorporating two specialized reasoning agents: one focuses on relationship-based inference, and the other on entity-based reasoning. First, the relationship agent conducts preliminary reasoning on the relationships associated with the current entity to identify potential target entities. Subsequently, based on the candidate relationships provided by the relationship agent, the entity agent performs a more refined reasoning process to determine the next action. Ultimately, this process generates N informative clue paths.

The temporal reasoning stage begins by reorganizing the identified clue paths into timestamp-ordered quadruples, which are then chronologically sorted to construct a temporal graph sequence. The model initializes each graph structure through comprehensive integration of entities and relationships, followed by temporal feature extraction using GRU networks to capture evolutionary patterns. Multi-layer perceptrons then process these temporal representations to either generate final predictions or produce stage feedback rewards that optimize the clue searching process through a cross-stage reinforcement mechanism. This integrated approach not only enhances path discovery efficiency but also effectively mitigates reward sparsity, ultimately improving overall model performance through coordinated temporal-structural learning.

4.2 Clue Searching Based on Reinforcement Learning

The goal of the first stage of clue searching is to find and summarize clue paths related to the given query from historical information. For a given query $(e_s, r_q, ?, t_s)$, where e_s denotes the head entity, r_q denotes the query placeholder, ? represents the unknown tail entity placeholder, and t_s is the timestamp of the query, the first stage aims to obtain several candidate entities close to the target entity e_p and their paths.

The TS-TKGC model employs a reinforcement learning system with two agents in the first stage to reason over paths in historical knowledge. First, the

model constructs a candidate relationship set by aggregating all relationships associated with the current entity. Subsequently, the relationship policy network directs the relationship agent to select the optimal next-hop relationship. Finally, the entity policy network calculates and selects the next target entity.

This reasoning process continues until the target entity e_p is reached, marking the end of the reasoning task. This dual-agent strategy enables the model to conduct knowledge reasoning with higher accuracy and efficiency. The first stage aims to discover and aggregate clue paths relevant to the query quadruple $(e_s, r_q, ?, t_s)$ from historical knowledge. Thus, developing a learnable and efficient clue searching strategy is critical. The first stage is formulated as a sequential decision-making problem that is solved by the reinforcement learning system.

Static quadruples are transformed into entity nodes with timestamps to construct a temporal graph structure. Nodes (entity, timestamp) are connected via bidirectional edges: forward edges retain original relations while reverse edges use inverse relations for semantic symmetry, with an entity-to-temporal-nodes mapping dictionary established. Action spaces are dynamically generated for each current entity state and query time: only strictly preceding events are accessible when current time equals query time; otherwise, preceding or concurrent events are permitted. State-transition actions are (relation, entity, timestamp) triples including NO-OP (stay) and PAD (padding) actions. Reverse chronological sorting is used to prioritize the latest events. Batch action spaces are standardized by truncating earliest events when exceeding capacity, padding with PAD when undersized, and disabling NO-OP at initial steps using a first-step policy, yielding reinforcement learning tensors with consistent dimensions.

The reinforcement learning system in TS-TKGC consists of an agent and an environment, formalized as a Markov Decision Process (MDP). This standard RL framework models agent-environment interactions to discover N optimal clue paths that satisfy the query constraints. Starting from the initial state, the agent selects actions according to a policy to traverse to new entities, continuing until reaching the maximum step length I or the target entity. Formally, the MDP is defined by four components:

State. Each state $s_i = (e_i, t_i, e_s, r_q, t_s) \in ST$ is a tuple, where ST is the set of all available states;$e_i(e_0 = e_s)$ is the entity visited by the agent at step i;$t_i(t_0 = t_s)$ is the timestamp when the agent took the action in the previous step.

Action. The action is represented as the set of relationships, tail entities, and timestamps that the agent might reach at step i, including the next entity in the quadruple. Let $A_i \in A$,A be the set of all possible actions. The action definition is shown in Equation. 4.1:

$$A_i = \{(r', e', t') | (e_i, r', e', t') \in G_{t-1}\} \tag{4.1}$$

Among them, G_{t-1} denotes the set of possible quadruples that the entity e_i may reach in the next step, r' denotes the relation, e' denotes the tail entity, and t' denotes the timestamp.

Transition. The transition function $\delta : ST \times A \rightarrow ST$, and ST is deterministic in the context of temporal knowledge graph, simply updating the state to the new entity associated with the action chosen by the agent.

Reward. At the end of the search process, the agent receives a final reward consisting of primary and secondary components. Specifically, the primary reward is 1 for correct target entity e_p identification in the N candidate entities generated by the entity agent, 0 otherwise. If e_p is correctly identified, the agent also receives a secondary reward from the second stage. This reward is designed to drive the agent to locate the target entity more accurately and promote seamless collaboration between the two stages.

(1) Relationship Policy Network

When constructing the relationship policy network, a Gated Recurrent Unit (GRU) is employed to encode historical reasoning paths. This encoding process combines historical information with the current node state to serve as the input for preliminary relationship reasoning in the network. This design enables the proposed model to fully leverage historical information, thereby enhancing reasoning accuracy and efficiency. The detailed architecture of the relationship policy network is shown in Fig. 2(a).

The GRU encodes historical path-searching information by taking the previous historical state encoding H_{t-1}, current node information e_s, and query relation r_q as inputs to compute the current historical state H_t. Notably, the initial state H_0 is defined as the current node e_s. The specific calculation is detailed in Equation. 4.2. This strategy effectively integrates historical context, providing a richer basis for the reasoning process.

$$H_t = GRU([e_s \oplus r_q], H_{t-1}) \tag{4.2}$$

The historical state information H_t at the current time step t is concatenated with the current node information e_s and the given query r_q to form a fusion vector. This fusion vector is then passed through a ReLU function and multiplied with the set of candidate relationships, followed by a softmax activation function to obtain the probability distribution of the current node e_s over the set of candidate relationships. The relationship agent can select the relationship with the highest score according to this probability distribution as the next action for the agent. The calculation process of the relationship policy network in this method is shown in Equation. 4.3, where R_t represents the set of candidate relationships connected to the current entity, W_1 and W_2 are different weights.

$$\pi_\theta^h(r_t|s_t) = softmax(R_t W_2 ReLU(W_1[H_t, e_s, r_q])) \tag{4.3}$$

After constructing the relationship policy network π_θ^h, we apply a Dropout algorithm that randomly discards a subset of candidate relationships, yielding a filtered relationship policy network π_θ^h, as shown in Equation. 4.4.

$$\pi_\theta^{h'}(r_t|s_t) = \pi_\theta^h(r_t|s_t) b_i \tag{4.4}$$

Here, b_i represents a binary variable sampled from a Bernoulli distribution. After obtaining the final relationship policy network $\pi_\theta^{h'}$, the relationships connected to the current node are probabilistically assigned to guide the relationship agent in selecting the next relationship.

(2) Entity Policy Network

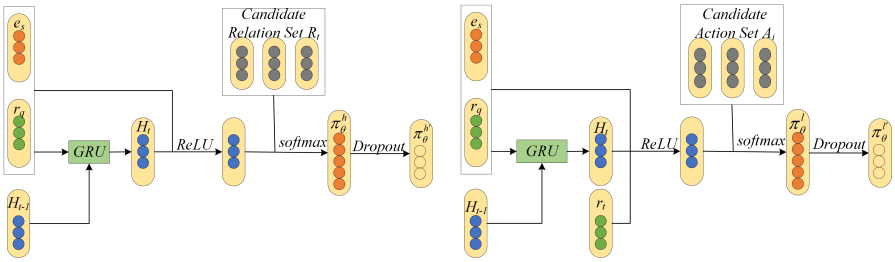
Similar to the relationship policy network, the entity policy network takes the current node e_s, the given query relationship r_q, the historical search encoding H_{t-1}, and the relationship r_t obtained from the relationship policy network as inputs. The entities connected to the selected relationship by the relationship agent in the current node are used as the candidate action set for the entity policy network to construct the preliminary entity policy network π_θ^l. The final entity policy network π_θ^l is obtained by randomly discarding some nodes in π_θ^l using the Dropout algorithm. The entity policy network is shown in Fig. 2(b).

The calculation process of the entity policy network in this paper is shown in Equation. 4.5 and Equation. 4.6, where A_i represents the action set of the current entity e_s, and W_3 and w_4 are different weights.

$$\pi_\theta^l(e_s|s_t) = softmax(A_i W_4 ReLU(W_3[H_t, e_s, r_q, r_t]) \tag{4.5}$$

$$\pi_\theta^{l'}(e_s|s_t) = \pi_\theta^l(e_s|s_t) b_i \tag{4.6}$$

After successfully constructing the entity policy network $\pi_\theta^{l'}(e_s|s_t)$, the entity agent can select the entity with the highest probability as the next action based on this probability distribution. This process ensures that the entity agent can make more accurate and effective entity selections based on the relationship chosen by the relationship agent, and continuously optimize its decision-making process through the reward mechanism.



(a) Relationship Policy Network (b) Entity Policy Network

Fig. 2: Relationship and Entity Policy Network

4.3 Temporal Reasoning Based on Gated Recurrent Units

To comprehensively model the temporal dependencies between clue facts at different timestamps and the structural correlations among simultaneously occurring clue facts, the second stage first reorganizes all clue facts in chronological order, then transforms them into a temporal graph sequence. Specifically, all

clue facts are reorganized into a set of quadruples, and quadruples sharing the same timestamp are aggregated to form multi-relational temporal graphs. Each graph is composed of clue facts with timestamps $j \in \{0, 1, \ldots, t_s - 1\}$, where t_s denotes the query timestamp.

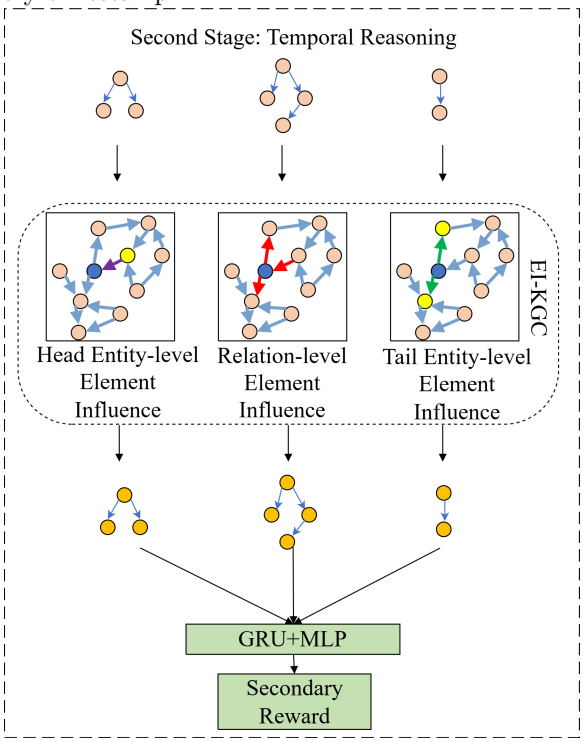


Fig. 3: The Basic Idea of Temporal Reasoning

After obtaining multiple reorganized graphs, we adopt this approach. We define intra-graph interaction vectors for head entities, relations, and tail entities. These vectors are computed using Graph Neural Networks (GNNs) to integrate structural information and node semantics from the knowledge graph. **Initialization Step**: For each temporal graph, the model computes initial node and relation embeddings using GNN message passing. **Iterative Update**: For each entity in the graph, the model computes three interaction vectors capturing: Head entity-to-tail entity influences, Relation-to-entity influences, and Tail entity-to-head entity influences. These vectors enable the model to capture semantic dependencies between triplets and extract implicit information within the graph. The updated entity embedding is obtained by merging these interaction vectors with the original entity embedding, as defined in Equation. 4.7:

$$e'_s = e_s + g_s^{head} + g_s^{rel} + g_s^{tail} \tag{4.7}$$

Here, e'_s represents the updated embedding of entity e_s in the multi-relational graph, while g_s^{head}, g_s^{rel} and g_s^{tail} represent the influences between head entity-level elements, relation-level elements, and tail entity-level elements, respectively.

A secondary iterative refinement is applied to the current entity to effectively capture and aggregate interaction features from neighboring nodes. Taking the first-iteration updated entities as input, a second-round information aggregation is performed via iterative message passing, enabling each entity to fully integrate the semantic and structural information within its local neighborhood. The mathematical formulation for the secondary iteration is detailed in Equation. 4.8:

$$e_s^2 = e_s' + (g_s^{head})^1 + (g_s^{rel})^1 + (g_s^{tail})^1 \tag{4.8}$$

Here, e_s^2 represents the secondary embedding of entity e_s^2, while $(g_s^{head})^1$, $(g_s^{tail})^1$ and $(g_s^{tail})^1$ represent the influences between head entity-level elements, relation-level elements, and tail entity-level elements for e_s in the first layer, respectively.

The embeddings are fed into the GRU to obtain the final output of the GRU, denoted as H_j, as shown in Equation. 4.9. H_j and W_{mlp} are then passed into a multi-layer perceptron (MLP) to obtain the final scores for all entities, as shown in Equation. 4.10.

$$H_j = GRU([e_s \oplus g_j \oplus r_q], H_{j-1}) \tag{4.9}$$

$$p(e|e_s, r_q, t_s) = \sigma(H_j W_{mlp}) \tag{4.10}$$

Finally, the candidate entities are re-ranked based on the obtained scores. To provide positive feedback for the clue paths that lead to the answer, the second stage offers a secondary reward to the first stage, which is equal to the final score obtained. The architecture of the temporal reasoning process is shown in Fig. 3.

4.4 Training Strategy and Multi-dimensional Rewards

In the first stage, the search strategy network is trained by maximizing the expected return for all queries in the training set. Since the first and second stages are interrelated, they are trained jointly. Before the joint training process, the first stage is pre-trained using binary rewards. then, the second stage is trained while the parameters of the first stage are frozen. Finally, both stages are trained together. The final reward function is shown in Equation. 4.11, where $J(\theta)$ represents the final reward, $R(e_I|e_s, r_q, t_s)$ represents the reward obtained when reaching the maximum number of steps, and $E_{(a_0 \ldots a_{I-1})}$ represents the action chosen at the maximum step. The relationship policy network and entity policy network guide the relationship agent and entity agent, respectively, in selecting the next relationship and action. By using an interactive reward mechanism, the interaction between these two policies is strengthened, ensuring that the agents maximize their rewards. This approach enhances the model's reasoning accuracy.

$$J(\theta) = E_{(a_0 \ldots a_{I-1})}[R(e_I|e_s, r_q, t_s)] \tag{4.11}$$

Reinforcement learning is applied to the task of temporal knowledge graph completion. To avoid performance degradation caused by sparse rewards, a new reward mechanism is constructed to calculate reward weights from multiple dimensions. The rewards are mainly divided into two parts. In the first stage, an

interactive reward function is constructed for clue searching to give timely rewards for the choices made by the relationship and entity agents. In the second stage, a secondary reward is returned if the correct result is ultimately obtained

The rewards in the first stage are primarily obtained by calculating the similarity scores between the candidate relationships or entities selected by the two agents and the target quadruple using a scoring function score $f(e_t, r_{t+1}, e_{t+1}, t_s)$. These similarity scores are used as the reward scores for the agents' choices, facilitating interaction between the relationship and entity agents. When the agents find the target relationship or entity, a global reward is given, as $R_g^h = 1$ and $R_t^l = 1$; if the target is not found and the maximum reasoning step length is not reached, the similarity score calculated by the scoring function is used as the reward for the agents' choices. The calculation methods for the interactive reward functions are shown in Equation. 4.12 and Equation. 4.13.

$$R_t^h = R_g^h + (1 - R_g^h) f(e_t, r_{t+1}, e_{t+1}, t_s) \tag{4.12}$$

$$R_t^l = R_g^l + (1 - R_g^l) f(e_t, r_{t+1}, e_{t+1}, t_s) \tag{4.13}$$

Here, R_t^h and R_t^l are the interactive reward functions for the relationship agent and the entity agent, respectively, while R_g^h and R_g^l serve as the global reward. When the relationship agent and the entity agent find the target relationship and the target entity, respectively, a reward of $+1$ is given; otherwise, the reward is 0. If the reward is 0, the similarity between the choices made by the two agents can be calculated using a scoring function based on the embedding model, and this score is used as the reward for the agents' choices.

In the first stage, embedding-based models provide intermediate rewards by computing cosine similarity between candidate actions (i.e., relations and entities) and the target quadruple at each step. This mechanism offers progressive feedback during the search process—for instance, immediately evaluating a relation's semantic match with the target rather than delaying reward until termination. The second stage generates secondary rewards from final prediction scores, which are fed back to the first stage as evaluation signals. Correct path verification triggers additional rewards, forming an interstage reinforcement loop: effective clue searching improves temporal reasoning accuracy, while accurate reasoning enhances search capability. This cross-stage reward loop dynamically adapts the first stage's search strategy to the second stage's reasoning needs, propagating global feedback for end-to-end policy optimization.

5 Temporal Knowledge Graph Completion Algorithm

This section describes the algorithm for the policy network in the TS-TKGC model. The algorithm takes the current entity embedding e_s, the given query relation r_q, the set of candidate relationships R_t, and the set of candidate actions A_i as inputs. By computing the relationship policy network and the entity policy network, the algorithm ultimately generates clue paths, which are then outputted as the final result. The specific calculation process is shown in Algorithm 1.

Step 1: Obtain the candidate relationships for the current entity. First, acquire the historical state information H_t of the current entity e_s; then obtain the relationship policy network; filter the relationship policy network; and finally, get the candidate relationships r_t.

Step 2: Obtain the candidate entities for the current entity. First, acquire the entity policy network for the current entity e_s; then filter the entity policy network; and finally, get the candidate entities e_t.

Algorithm 1: Policy Network Construction Algorithm

Input: Current moment entity embedding e_s, given query relation r_q, relation candidate set R_t and candidate action set A_i

Output: N search paths

for $i < N$ **do**
  while *step* $< I$ *and not success* **do**
    $H_t = GRU([e_s \oplus r_q], H_{t-1})$ //Obtain the current state
    $\pi_\theta^h(r_t|s_t) = softmax(R_t W_2 ReLU(W_1[H_t, e_s, r_q]))$ //Obtain relation strategy network
    $\pi_\theta^l(e_s|s_t) = softmax(A_i W_4 ReLU(W_3\lceil H_t, e_s, r_q, r_t\rceil)$ //Obtain entity strategy network
    $\pi_\theta^{l'}(r_t|s_t) = \pi_\theta^l(e_t|s_t)b_i$ //Perform filtering
    select e_t from $\pi_\theta^{l'}$ //Obtain candidate entity
    Obtain a candidate triple for the current entity (e_s, r_t, e_t)
  end
  Obtain a candidate path $(e_s, r_t, e_t, r_{t+1}, e_{t+1}, ..., r_I, e_I)$
end

6 Experiment And Analysis

6.1 Datasets and Experimental Settings

Table 1: Statistical Summary of Temporal Knowledge Graph Datasets

Dataset	ICEWS14	ICEWS05-15	ICEWS18	GDELT	YAGO
entity	7128	10,488	23,033	7,691	10,623
relation	230	251	256	240	10
train triple	74,845	368,868	373,018	1,734,399	136,770
valid triple	8,514	46,302	45,995	238,765	10,000
test triple	7,371	46,159	49,545	305,241	14,770
time interval	24 hours	24 hours	24 hours	15 mins	1 year

This section introduces the five temporal KG datasets used in experiments: GDELT (Global Database of Events, Language, and Tone [14]), ICEWS14, ICEWS05-15, ICEWS18 (sourced from the Integrated Crisis Early Warning System [15]) and YAGO. Dataset statistics are provided in Table 1.

While static knowledge graph filtering effectively removes known facts from corrupted rankings during training/validation/testing, it is unsuitable for temporal knowledge graph reasoning under extrapolation. To enable more accurate

evaluation of temporal knowledge graph completion methods, this paper adopts a time-aware filtering approach. This method filters out only quadruples occurring at the specific query timestamp during metric calculation, ensuring evaluation accuracy and reliability. The hyperparameters used in the model affect both the training process and accuracy of the final model predictions. In the experiments, the model's accuracy was boosted by continuously tuning the hyperparameters. The main hyperparameter settings for the TS-TKGC model are shown in Table 2. **Note**: Except for the learning rate (α) and dimension of the GRU vectors, which may vary across the four datasets, all other parameters are the same across the datasets.

Table 2: TS-TKGC Model Parameters

Parameter	Value
I	2
α	0.5/0.5/0.4/0.3
learning rate	0.001
vector dimension	100
number of GRU layers	2
number of EI-KGC layers	2
maximum GRU sequence length	10
GRU vector dimension	100/100/100/200

We utilize Mean Reciprocal Rank (MRR), Hits@N, and Mean Absolute Error (MAE) for evaluation. MRR and Hits@N provide complementary performance assessment, while MAE specifically evaluates temporal sensitivity in time prediction experiments.

6.2 Comparative Experiment

For entity prediction experiments, we first replace the entity in the quadruple (e_h, r, e_t, t) participating in the test with each element in the entity set, and then calculate the scores of each quadruple according to the scoring function. After obtaining all scores, arrange the quadruple in ascending order in the score order to obtain the score table, and obtain the correct quadruple position.

Table 3: Results of Entity Prediction Experiment on ICEWS14 Dataset

Model	MRR	Hits@1	Hits@3	Hits@10
RE-NET	0.367	0.226	0.332	0.481
CyGNet	0.264	0.172	0.371	0.403
TiRGN	0.439	0.343	0.502	**0.586**
RE-GCN	0.253	0.162	0.478	0.532
TS-TKGC(ours)	**0.452**	**0.341**	**0.504**	0.572

We compare our model with previous classic temporal knowledge graph completion models, including RE-NET, CyGNet[16], TiRGN[17], RE-GCN, to verify the effectiveness of the TS-TKGC model. To evaluate the performance of these models, the TS-TKGC model conducted detailed comparative experiments on

the ICEWS14, ICEWS05-15, ICEWS18, GDELT and YAGO dataset. MRR, Hits@1, Hits@3 and Hits@10 are selected as evaluation indicators.

As shown in Tables Table 3 - 7: 1) On the ICEWS14 and YAGO datasets, TS-TKGC model obtains optimal performance in the MRR, Hits@1 and Hits@3 metrics compared to other baseline models. 2) On the ICEWS05-15 and ICEWS18, compared with other baseline models, the TS-TKGC model obtains the optimal performance in the MRR, Hits@1, Hits@3 and Hits@10 metrics. 3) On the GDELT dataset, compared with other baseline models, the TS-TKGC model obtains optimal performance in the Hits@1, Hits@3 and Hits@10 indicators compared with other baseline models.

Table 4: Results of Entity Prediction Experiment on ICEWS05-15 Dataset

Model	MRR	Hits@1	Hits@3	Hits@10
RE-NET	0.417	0.276	0.432	0.581
CyGNet	0.384	0.292	0.371	0.483
TiRGN	0.337	0.232	0.382	0.542
RE-GCN	0.351	0.312	0.478	0.539
TS-TKGC(ours)	**0.443**	**0.342**	**0.525**	**0.584**

Table 5: Results of Entity Prediction Experiment on ICEWS18 Dataset

Model	MRR	Hits@1	Hits@3	Hits@10
RE-NET	0.286	0.208	0.312	0.481
CyGNet	0.269	0.172	0.271	0.403
TiRGN	0.322	0.201	0.388	0.511
RE-GCN	0.251	0.162	0.288	0.432
TS-TKGC(ours)	**0.325**	**0.211**	**0.395**	**0.561**

Table 6: Results of Entity Prediction Experiment on GDELT Dataset

Model	MRR	Hits@1	Hits@3	Hits@10
RE-NET	0.177	0.111	0.332	0.342
CyGNet	0.154	0.102	0.271	0.331
TiRGN	**0.194**	0.113	0.212	0.343
RE-GCN	0.161	0.102	0.278	0.298
TS-TKGC(ours)	0.192	**0.117**	**0.344**	**0.389**

The five comparative experiments verify that the proposed TS-TKGC model, with its two-stage clue search and temporal reasoning modules, effectively mines the interplay between temporal information and entity relations in temporal knowledge graphs, achieving superior knowledge graph completion performance.

6.3 Ablation Study

To evaluate the effects of different stages and agents in first stage of the TS-TKGC model, an ablation study was conducted on ICEWS18 and YAGO dataset. The variants of the TS-TKGC model are: using only relation agent in first stage, using only entity agent in first stage, and using only first stage of the model.

As shown in Table 8 and Table 9, the most vital impact on experimental results was observed when the second stage was completely removed, with the greatest decrease in all indicators. For the first stage, both agents have a certain degree of impact on final results. Thus, experiments have proven that each part of two stages in the TS-TKGC model has an impact on the overall model, which validates the effectiveness of each component in the model.

Table 7: Results of Entity Prediction Experiment on YAGO Dataset

Model	MRR	Hits@1	Hits@3	Hits@10
RE-NET	0.177	0.111	0.332	0.342
CyGNet	0.759	0.730	0.779	0.798
TiRGN	0.871	0.113	0.212	0.343
RE-GCN	0.809	0.843	0.902	**0.929**
TS-TKGC(ours)	**0.877**	**0.853**	**0.912**	0.905

Table 8: Ablation Study Results on ICEWS18

Model	MRR	Hits@1	Hits@10
TS-TKGC(ours)	0.325	0.211	0.561
-First Stage (Relation Agent)	0.312	0.187	0.464
-First Stage (Entity Agent)	0.321	0.202	0.401
-Second Stage	0.302	0.182	0.352

Table 9: Ablation Study Results on YAGO

Model	MRR	Hits@1	Hits@10
TS-TKGC(ours)	0.877	0.853	0.905
-First Stage (Relation Agent)	0.855	0.833	0.794
-First Stage (Entity Agent)	0.861	0.829	0.783
-Second Stage	0.803	0.816	0.718

6.4 Relation Prediction Experiment

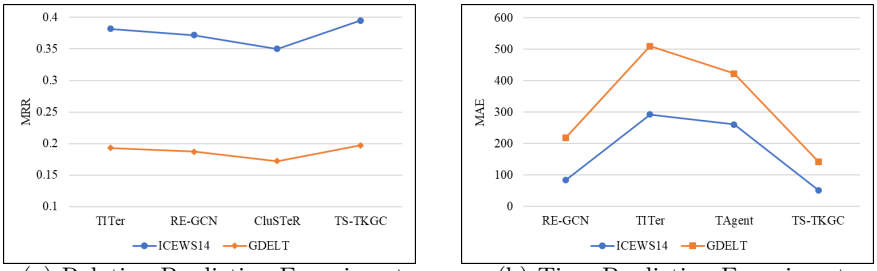
To assess the effectiveness of the TS-TKGC model in relation prediction, we conducted a relation prediction experiment on temporal knowledge graphs. The TITer[18], RE-GCN, and CluSTeR[19] models were selected as baseline comparisons. For evaluation, MRR was useed as the primary metric in this section to ensure a focused and consistent evaluation of predictive accuracy. The detailed results of the relation prediction experiment are visualized in Fig. 4(a).

TS-TKGC consistently outperforms baselines, exhibiting narrower performance margins in relation prediction owing to the limited candidate relation pool. Analysis of the GDELT dataset reveals inherent limitations: abstract concepts (e.g., "government") lack contextual anchors (e.g., national affiliation), which restricts temporal reasoning and leads to performance plateaus across models. TS-TKGC's two-stage temporal-graph integration architecture demonstrates superior noise resilience to such contextual ambiguities, achieving state-of-the-art results on both ICEWS14 and GDELT while significantly outperforming structure-centric baselines such as RE-GCN. This highlights the critical role

of joint temporal-structural modeling in learning precise relational embeddings under incomplete contextual constraints.

6.5 Time Prediction Experiment

For the time prediction experiment, given a test quadruple (h, r, t, t'), the goal is to predict the expected time of next occurrence of fact (h, r, t). The Mean Absolute Error(MAE) between predicted time and actual time is used as evaluation metric, measured in hours. The specific results are shown in Fig. 4(b).



(a) Relation Prediction Experiment (b) Time Prediction Experiment

Fig. 4: Results of the Relation Prediction and the Time Prediction Experiment

In contrast to RE-GCN, which employs relation-aware GCN to capture structural dependencies in knowledge graphs, TS-TKGC utilizes a two-stage temporal reasoning framework to better capture temporal dynamics. Unlike TAgent (binary terminal rewards) and TITer (Dirichlet distribution-based temporal rewards), TS-TKGC adopts a multi-dimensional reward mechanism to address the sparse reward problem and facilitate accurate temporal modeling. In the time prediction experiment, the proposed TS-TKGC achieved state-of-the-art performance, demonstrating its capability to effectively model dynamic temporal dependencies in temporal knowledge graphs.

6.6 Sensitivity Analysis Experiment

To optimize the model effectively, this paper investigates how TS-TKGC's performance varies with different hyperparameters. We designed experiments to analyze the impact of two key hyperparameters: the maximum reasoning steps$I$ and the reward balancing factor α. MRR served as the primary evaluation metric, with detailed results tabulated in Table 10 and 11.

As shown in Table 10, TS-TKGC exhibits significant performance degradation when the maximum reasoning steps exceed 3, aligning with findings in prior RL-based TKGC literature. This phenomenon can be attributed to the accumulation of noise in multi-step reasoning: while early steps capture relevant temporal dependencies, longer reasoning chains introduce compounding errors that ultimately impair prediction accuracy.

Table 11 lists the optimal reward balancing factors for the four datasets: 0.5 for ICEWS14, 0.5 for ICEWS05-15, 0.4 for ICEWS18, and 0.3 for GDELT. The

ICEWS subsets prioritize semantic-level learning rewards, indicating that when temporal KG datasets share similar temporal granularity, dataset scale shows a positive correlation with semantic-level adaptive rewards.

Table 10: Impact of Maximum Reasoning Steps I on Model Performance

I	ICEWS14	ICEWS05-15	ICEWS18	GDELT
2	**0.452**	**0.443**	**0.325**	**0.192**
3	0.442	0.431	0.320	0.187
4	0.431	0.419	0.315	0.172
5	0.413	0.402	0.295	0.163

Table 11: Impact of Reward Balancing Factor α on Model Performance

α	ICEWS14	ICEWS05-15	ICEWS18	GDELT
0.2	0.435	0.426	0.298	0.182
0.3	0.442	0.431	0.307	**0.192**
0.4	0.448	0.439	**0.325**	0.179
0.5	**0.452**	**0.443**	0.318	0.176
0.6	0.439	0.440	0.312	0.168

7 Conclusion

Current temporal knowledge graph completion models over-rely on timestamp correlations while neglecting joint modeling of temporal dynamics and semantic relationships. To address this limitation, we propose TS-TKGC, which features two synergistic stages: **Clue Search**: A reinforcement learning module identifies temporally constrained semantic pathways; **Temporal Reasoning**: GRU networks model temporal clue evolution patterns. Joint training enables synergistic integration of temporal and structural features. Experiments on four benchmark datasets demonstrate that TS-TKGC consistently outperforms state-of-the-art baselines. For future research, we suggest integrating large language models for temporal information extraction or developing more efficient mining strategies to address current limitations in existing approaches.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (62472204), the Social Science Planning Fund Program of Liaoning Province (L23BJY018), the Research Program of the Liaoning Liaohe Laboratory (LLL24KF-01-04).

Disclosure of Interests. We declare that we have no competing financial interests.

References

1. Li, N., E, H.: TR-rules: Rule-based model for link forecasting on temporal knowledge graph considering temporal redundancy. In: Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 7885–7894. Association for Computational Linguistics, Singapore (2023)

2. Bai, L.: Multi-hop temporal knowledge graph reasoning with temporal path rules guidance. Expert Systems with Applications **223**, 119804 (2023)
3. Mei, X., Yang, L.: An adaptive logical rule embedding model for inductive reasoning over temporal knowledge graphs. In: Proceedings of the 2022 conference onempirical methods in natural language processing. pp. 7304–7316 (2022)
4. Jin, W., Qu, M.: Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6669–6683. Association for Computational Linguistics, Online (2020)
5. Wu, J.: Temp: Temporal message passing for temporal knowledge graph completion. arXiv preprint arXiv:2010.03526 (2020)
6. Li, Z., Jin, X.: 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. pp. 408–417 (2021)
7. Chen, L.: Dacha: A dual graph convolution based temporal knowledge graph representation learning method using historical relation. ACM Transactions on Knowledge Discovery from Data (TKDD) 16(3), 1–18 (2021)
8. Zhu, C., Chen, M.: Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In: Proceedings of the AAAI conference on artificial intelligence. pp. 4732–4740 (2021)
9. Hu, J.: A blockchain-based reward mechanism for mobile crowdsensing. IEEE Transactions on Computational Social Systems **7**(1), 178–191 (2020)
10. Tao, Y., Li, Y.: Temporal link prediction via reinforcement learning. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3470–3474. IEEE (2021)
11. Bai, L.: Multi-hop reasoning over paths in temporal knowledge graphs using reinforcement learning. Applied Soft Computing 103, 107144 (2021)
12. Sun, H., Zhong, J., Ma, Y.: Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. arXiv preprint arXiv:2109.04101 (2021)
13. Li, Z., Jin, X., Guan, S.: Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. arXiv preprint arXiv:2106.00327 (2021)
14. Leetaru, K.: Gdelt: Global data on events, location and tone, 1979-2012 (2013)
15. Jäger, K.: The limits of studying networks via event data: Evidence from the icews dataset. Journal of Global Security Studies 3(4), 498–511 (2018)
16. Li, Y., Sun, S.: Tirgn: Time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022. pp. 2152–2158 (2022)
17. Zhu, C., Chen, M.: Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In: AAAI (2021)
18. Sun, H., Zhong, J.: Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In: EMNLP (2021)
19. Li, Z., Jin, X.: Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. arXiv preprint arXiv:2106.00327 (2021)