RENN: A Rule Embedding Enhanced Neural Network Framework for Temporal Knowledge Graph Completion

Linlin Zong[1], **Zhenrong Xie**[1], **Chi Ma**[1], **Xinyue Liu**[1], **Xianchao Zhang**[1], **Bo Xu**[2,3*]
[1]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province,
School of Software, Dalian University of Technology, China
[2]School of Computer Science and Technology, Dalian University of Technology, China
[3] Key Laboratory of Social Computing and Cognitive Intelligence (Dalian University of Technology),
Ministry of Education, China
{llzong, xyliu, xczhang, xubo}@dlut.edu.cn
xiezr1207@163.com, dlut.marx@gmail.com

Abstract

Existing approaches encompass deep neural network-based methods for temporal knowledge graph embedding and rule-based logical symbolic reasoning. However, the former may not adequately account for structural dependencies between relations.Conversely, the latter methods relies heavily on strict logical rule reasoning and lacks robustness in the face of fuzzy or noisy data. In response to these challenges, we present RENN, a groundbreaking framework that enhances temporal knowledge graph completion through rule embedding. RENN employs a three-step approach. First, it utilizes temporary random walk to extract temporal logic rules. Then, it pre-trains by learning embeddings for each logical rule and its associated relations, thereby enhancing the likelihood of existing quadruples and logical rules. Finally, it incorporates the embeddings of logical rules into the deep neural network. Our methodology has been validated through experiments conducted on various temporal knowledge graph models and datasets, consistently demonstrating its effectiveness and potential in improving temporal knowledge graph completion.

Keywords: Temporal knowledge graph, graph completion, rule embedding

1. Introduction

A Knowledge Graph is a graph-based data structure that represents knowledge using triples (s, r, o). Knowledge within a Knowledge Graph often contains significant metatemporal aspects, leading to dynamic changes as time progresses. The concept of a Temporal Knowledge Graph (TKG) (Cai et al., 2023) was introduced based on a static Knowledge Graph, incorporating timestamps t to express knowledge in the form of quadruples (s, r, o, t), such as (Italy, Electing_Prime_Minister, Berlusconi, 1994). However, since many TKGs are constructed manually or through a semi-manual process, they often suffer from incompleteness, as collecting and verifying facts can be expensive and time-consuming. The incompleteness of TKGs presents limitations to their potential applications. As a result, a critical research task is to develop methods that can predict missing facts in TKGs based on the existing quadruples. These methods aim to enhance the completeness and usefulness of TKGs, making them more valuable for various AI applications.

In the context of a TKG spanning from timestamp 0 to timestamp T, there are two primary TKG completion tasks: extrapolation and interpolation. The goal of the extrapolation task is to use historical knowledge (knowledge from timestamps 0 to T) to predict and infer future facts for timestamps greater than T. The interpolation task, on the other hand, uses both historical information (knowledge from timestamps 0 to t) and future information (knowledge from timestamps t to T) to reason and complete the facts for timestamps t. In this paper, we mainly focus on the extrapolation task.

To predict future facts based on the TKGs, global structural inference must be performed as time progresses, necessitating the ability to integrate temporal and structural information. Two major methods are used in the extrapolation task of TKG completion: deep neural network-based knowledge representation learning (Zhu et al., 2021), which embeds all entities and relations into vectors, and rule-based logical symbolic reasoning (Liu et al., 2022), which infers new facts based on logical rules.

However, the former may overlook the structural dependencies of relations. For example, Figure 1 shows a series of $Rafael\ Leão$ related subgraphs from TKG. For the query $(Rafael\ Leão, Transfer\_To, ?, 2019)$, the knowledge representation learning method may tend to choose $Lille\ F.C$, which is more similar to the subject entity $Rafael\ Leão$ in the embedding space. But the result ignores the structural information related to the subject entity $RafaelLeo$, i.e., two obvious logical paths $(Rafael\ Leão, Youth\_Train\_At, Portugese\ Sports$

*Corresponding author

LREC-COLING 2024, pages 13919–13928

$F.C, 2008$) ($Portugese\ Sports\ F.C, Contact, Lille\ F.C, 2015$) ($Lille\ F.C, Coordinate, AC\ Milan\ F.C, 2018$), and ($Rafael\ Leão, Transfer\_To, Lille\ F.C, 2017$) ($Lille\ F.C, Coordinate, AC\ Milan\ F.C, 2018$), through which can infer the correct answer $AC\ Milan\ F.C$.

On the other hand, the latter methods relies on hard logical rule reasoning, making it less tolerant of fuzzy and noisy data. As shown in Figure 1, when noise and fuzziness appear in TKG as the object entity $LilleF.C$ in the quad $(Rafael\ Leão, Transfer\_To, Lille\ F.C, 2017)$ is blurred to $Lisbon\ F.C$, traditional logical reasoning is prone to making incorrect inferences as $Paris\ Saint\text{-}Germain\ F.C$.

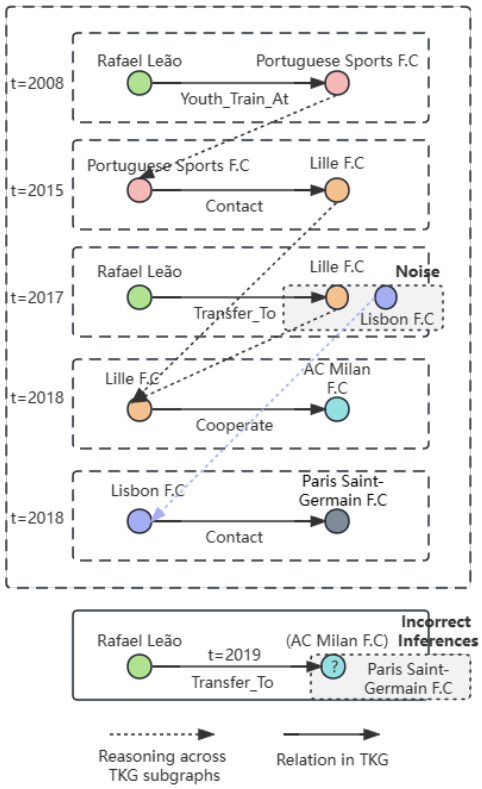

Figure 1: A series of $Rafael\ Leão$ related subgraphs from TKG. The black dashed line represents the traditional logical reasoning route, while the blue dashed line represents incorrect reasoning.

In response to the shortcomings of the two kinds of methods, we propose a Rule embedding Enhanced Neural Network framework for temporal knowledge graph completion (RENN). Firstly, we extract logical rules from temporal knowledge graphs in the form of first-order Horn clauses based on the Tlogic (Liu et al., 2022) method. Secondly, through joint training of relation and rule embeddings within the same vector space during pretraining, we maximize the likelihood of existing quads quadruples logical rules. Finally, we employ traditional deep neural network-based knowledge representation learning to effectively learn entity and relation embeddings. By injecting logical rule embeddings into the entity and relation embeddings, we enhance our ability to infer the global structure of future timestamp execution orders, leading to more efficient reasoning and improved predictions of new events. Experimental results across multiple datasets demonstrate that the logic injected through this pretraining approach enhances the effectiveness of existing deep neural network-based methods for completing temporal knowledge graphs. Furthermore, our method is universal and flexible, as pretraining can accommodate various rules with differing confidence levels and can enhance multiple TKG completion models.

2. Related Work

In this section, the paper provides an overview of existing approaches for TKG completion, specifically focusing on the extrapolation task. These approaches fall into two broad categories: deep neural network based knowledge representation learning and rule-based logical symbolic reasoning.

2.1. Deep Neural Network-based Knowledge Representation Learning

Deep neural network based knowledge representation learning aim to represent entities and relations as vectors in a continuous vector space, allowing for efficient inferences and predictions. Specifically, CyGNet incorporates a time-aware replication mechanism. This model comprises two distinct inference modes: the copy mode and the generation mode (Zhu et al., 2021). The copy mode is applied to forecast entity probabilities from the known entity vocabulary, while the generation mode is leveraged to deduce entity probabilities from the complete entity vocabulary. The RE-NET model (Jin et al., 2020) establishes the joint probability distribution for all events within a TKG through the utilization of autoregressive techniques. This model employs a circular event encoder to amalgamate data from previous event sequences and a neighborhood aggregator to collect data from simultaneous events occurring within the same time window. Subsequently, the decoder employs this synthesized information to define the joint probability of the current event. TITer (Sun et al., 2021) operates by navigating a TKG snapshot to retrieve answers for future queries. To tackle the challenges posed by future timestamps, TITer employs relative time encoding capabilities to incorporate temporal information when making decisions. The HIPNet model

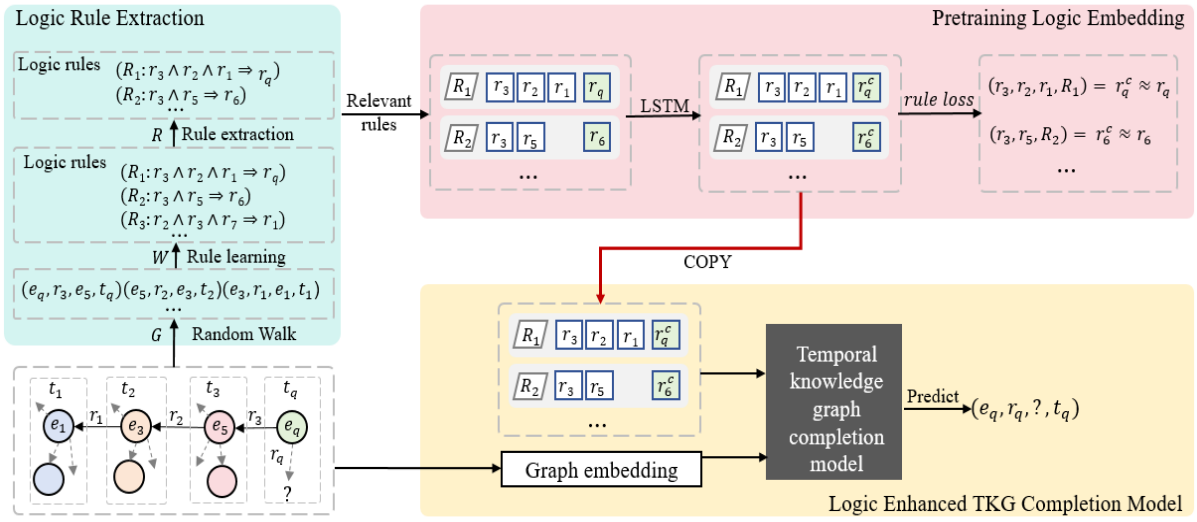

Figure 2: The framework of the RENN.

(He et al., 2021) leverages CompGCN to independently update structural representations. RTFE (Xu et al., 2021) distinguishes itself from static knowledge graph embedding models and temporal knowledge graph embedding models by modeling timestamp transformations as a Markov process. RTFE achieves this by recursively learning entity representations for various timestamps. In order to encode the global graph and capture the complex semantic relationships and long-term time dependencies between entities, HGLS (Zhang et al., 2023b) proposes a hierarchical relational graph neural network and designs a gated fusion module to model the dependencies of different types of entities and relationships on long-term and short-term time information. To explicitly discover and utilize latent relations, L2TKG (Zhang et al., 2023a) proposes a novel latent relations learning method for TKG reasoning.

The deep neural network-based knowledge representation learning is the prevailing approach for completing TKGs. Despite their strong performance, these methods could have a limitation: they might not adequately consider the structural dependencies between relations in the TKGs.

2.2. Rule-based Logical Symbolic Reasoning

Rule-based logical symbolic reasoning apply formal logic rules to infer new facts based on existing knowledge. Specifically, MLNs applies Markov logic networks and probabilistic soft logic to temporal reasoning in the temporal knowledge graph (Chekol et al., 2017). RLvLR Stream model considers temporal closed path rules and can learn the structure of rules from temporal changes in the knowledge graph for inference (Omran et al., 2021). TLogic provides an interpretable framework that revolves around the extraction of temporal logic rules via temporal random walks (Liu et al., 2022). Remarkably, it's the first symbolic framework to directly acquire temporal logic rules from temporal knowledge graphs and subsequently employ these rules for link prediction.

The rule-based logical symbolic reasoning has shown some promising results. However, these methods heavily depend on strict logical rule reasoning, making them less suitable for many real-world applications.

To address the limitations of the two aforementioned methods, we propose a rule embedding enhanced neural network framework for TKG completion, aiming to leverage the benefits of both methodologies while mitigating their individual drawbacks.

3. The RENN Framework

3.1. Preliminaries

Let $\mathcal{E}, \mathcal{R}, \mathcal{T}$, and $\mathcal{F}$ denote the sets of entities, relations, timestamps, and facts, respectively. A Temporal Knowledge Graph (TKG) can be described as $\mathcal{G}_{(1,T)} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T\}$, where $\mathcal{G}_t = (\mathcal{E}_t, \mathcal{R}_t, \mathcal{E}_t)$ is a multi-relationship directed TKG snapshot. In this context, a fact within a TKG can be represented as a quadruple (e_s, r, e_o, t), which equivalent to (e_s^t, 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 $t \in \mathcal{T}$, and $e^t = (e, t)$.

This paper focuses on the task of extrapolated temporal knowledge graph completion, which is geared toward predicting new facts at times $t > \mathcal{T}$. More precisely, when presented with known facts $\{(e_{s_i}, r_i, e_{o_i}, t_i) \mid t_i < t_q\}$ and a query $(e_q, r_q, ?, t_q)$ or $(?, r_q, e_o, t_q)$ that involves a timestamp not previ-

ously encountered, the aim is to generate a prioritized list of possible object/subject candidates that are most likely to complete the query.

3.2. Overall Architecture

We propose an enhancement to the temporal knowledge graph completion model by incorporating logical rules. The model, as depicted in Figure 2, comprises three main components: logical rule extraction, pre-training of rule embedding, and a rule-enhanced temporal knowledge graph completion model. Firstly, random walks are conducted in the temporal knowledge graph based on a specified transfer distribution. These random walks are then transformed into temporal logic rules, and the confidence score of each rule is computed. Subsequently, in the pretraining phase, the embeddings of relations, entities, and rules are initialized within the same vector space, and the embeddings for relations and rules are jointly trained. The objective of pretraining is to simultaneously maximize the likelihood of existing quadruples and logical rules. Finally, the rule embeddings are integrated into the existing temporal knowledge graph completion model, enabling it to provide answers to future queries.

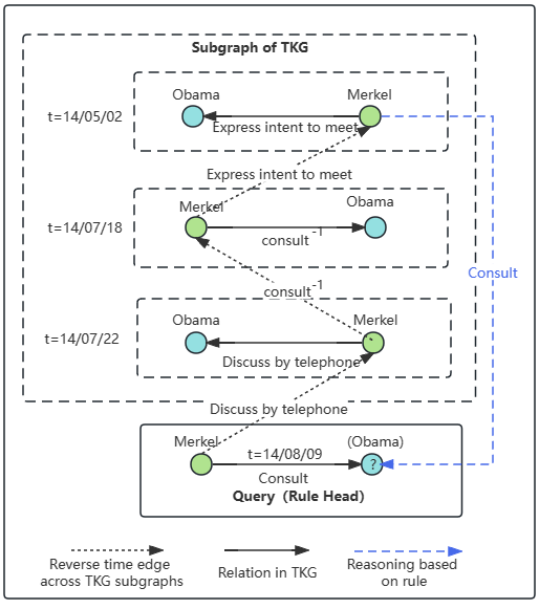

Figure 3: A series of subgraphs related to query $(Merkel,\ consult,\ ?,\ 14/08/09)$ in TKG, with entities $Merkel$ and $Obama$. The black dashed line represents the relationship between the rule body in rule, while the blue dashed line represents the relation between the rule body " $\rightarrow$ " the rule head. The correct result of the inference is $Obama$.

3.3. Temporal Logic Rule Extraction

3.3.1. Temporary Random Walk

For a query $(e_{l+1}, r_{l+1}, ?, t_{l+1})$, the process begins with a temporary random walk of length l originating from e_{l+1}, resulting in a non-increasing temporary random walk denoted as W.

For a rule of length l, a walk of length $l+1$ is sampled, where the additional step corresponds to the rule header r_h. In the first sampling step, uniform sampling selects an edge from all edges with the relation type as the head relation r_h to serve as the rule head $(e_1, r_h, e_{l+1}, t_{l+1})$. In subsequent sampling steps, define an exponential map $\hat{m} := (l+1)-(m-2)$, this transfer distribution is exponentially weighted based on the index mapping (Liu et al., 2022):

$$\mathbb{P}\left(u; m, e_{\hat{m}}, t_{\hat{m}}\right) = \frac{\exp\left(t_u - t_{\hat{m}}\right)}{\sum_{\hat{u}\in\mathcal{A}(m,e_{\hat{m}},t_{\hat{m}})} \exp\left(t_{\hat{u}} - t_{\hat{m}}\right)} \tag{1}$$

Where t_u represents the timestamp of edge u and $\mathcal{A}\left(m, e_{\hat{m}}, t_{\hat{m}}\right)$ represents the set of feasible edges for the next transition. $\mathbb{P}$ providing an incentive in the form of exponential-weighted probabilities for edges close to the previous edge.

A non-increasing temporal random walk W of length l from entity e_{l+1} to entity e_1 in the TKG is defined as a sequence of edges

$$((e_{l+1}, r_l, e_l, t_l), (e_l, r_{l-1}, e_{l-1}, t_{l-1}), \ldots, (e_2, r_1, e_1, t_1))$$

where $t_l \geq t_{l-1} \geq \cdots \geq t_1$ and $(e_{i+1}, r_i, e_i, t_i) \in \mathcal{G}$ for $i \in [1, l]$.

The non-increasing temporal random walks adhere to time constraints, allowing only the edges to be traversed backwards in time, where it is also possible to walk along edges with the same timestamp (Liu et al., 2022). As shown in Figure 3, starting from the subject entity of the query quad $Merkel$ and following the non increasing reverse time edge of the cross time subgraph, a random walk W can be obtained as:

$$(Merkel,\ discuss\ by\ telephone, Obama, 14/07/22),$$
$$(Obama, consult^{-1}, Merkel, 14/07/18), (Merkel,$$
$$express\ intent\ to\ meet, Obama, 14/05/02)$$

Since there are no edges between TKG snapshots, a random walk cannot be transferred from one snapshot to another. Therefore, we add the three types of edges in turn. 1) Reverse edge, for each quadruple (e_s, r, e_o, t), add the reverse edge of (e_o, r^{-1}, e_s, t), where r^{-1} indicates the reciprocal relation of r. 2) Self-cyclic edge. Self-cyclic edges can allow the random walk to stay in one place except for the first step of the random walk. 3) Cross the edge of the time snapshot, if there is (e_s, r, e_o, t_i) and $t_i \leq t_j$, the random walk can walk

from node $e_s^{t_j}$ to node $e_o^{t_i}$ through the edge r. The non-increasing time edge along the cross-time subgraph indicates the effect of past facts on entities and helps to find answers in historical facts.

Before extracting rules from the TKG, the reverse edges for all relations in the TKG address the problem of being unable to infer certain bidirectional relations present in reality (e.g., spouses, siblings, handshakes, negotiations, etc.). For instance, the rule "$married(x, y) \Rightarrow married(y, x)$," even if it occurs only rarely in the TKG (1 to 2 data points), will have high confidence due to the presence of reverse edges.

3.3.2. Temporal Logic Rule

Under the results of random walk, entities and timestamps are replaced with variables to convert W into a temporal rule R. Let E_i and T_i be variables that represent entities and timestamps, respectively. Further, let $r_1, r_2, \ldots, r_l, r_{l+1} \in \mathcal{R}$ be fixed in the set of relations $\mathcal{R}$. The temporal logic rule R with temporal constraints of length l is defined as follows:

$$\left((E_1, r_{l+1}, E_{l+1}, T_{l+1}) \leftarrow \wedge_{i=1}^{l} (E_i, r_i, E_{i+1}, T_i)\right) \tag{2}$$

with the temporal constraints

$$T_1 \leq T_2 \leq \cdots \leq T_l < T_{l+1}$$

The left-hand side of R is known as the rule head, where r_{l+1} represents the head relation. The right-hand side is termed the rule body and is expressed as a conjunction of body atoms, specifically (E_i, r_i, E_{i+1}, T_i) (Liu et al., 2022). It's important to note that in the rule $\left((E_1, r_{l+1}, E_{l+1}, T_{l+1}) \leftarrow \wedge_{i=1}^{l} \left(E_i, r_i^{-1}, E_{i+1}, T_i\right)\right)$, the entities involved in it do not necessarily have to be distinct. This is because a pair of entities can have multiple interactions at different points in time. To maintain this information, instances where the same entity appears multiple times in W are replaced with the same random variable in R. As shown in Figure 3, entities $Merkel$ and $Obama$ have multiple interactions at different time points in random walk W. Therefore, in the extracted logic R, E_1 and E_2 are used to replace entities $Merkel$ and $Obama$, resulting in the following rule R:

$$\begin{aligned}
&(E_1, consult, E_2, T_4) \leftarrow \\
&(E_1, discuss\ by\ telephone, E_2, T_3) \\
&\wedge (E_2, consult^{-1}, E_1, T_3) \\
&\wedge (E_1, express\ intent\ to\ meet, E_2, T_1)
\end{aligned}$$

3.3.3. Rule Extraction

Given the possibility of having numerous logic rules derived from the random walks, we employ frequency-based statistical methods to assess the confidence score of closed loop path rules. This process helps identify temporal rules that are highly reliable for predicting future events.

In particular, the confidence score for each rule is determined as outlined in Eq. (3).

$$Conf(R) = \frac{\text{rule support}}{\text{body support}} \tag{3}$$

Denote "grounding" as the utilization of quadruples from the TKG to instantiate variables within the rules. The rule support and the body support is the number of sets in TKG that satisfy rule grounding and body grounding, where rule grounding refers to the replacement of all variables E_i in the entire rule R, and body grounding refers to the replacement of only variables in rule body, and all groundings must comply with the time constraints in Eq.(2).

Rules that achieve a confidence score higher than a pre-defined threshold are subsequently extracted. The rules extracted and filtered with high confidence from the TKG. The high confidence reflects the consistency between the rule and existing facts from the TKG. At this point, we believe that the rule with high confidence is valid.

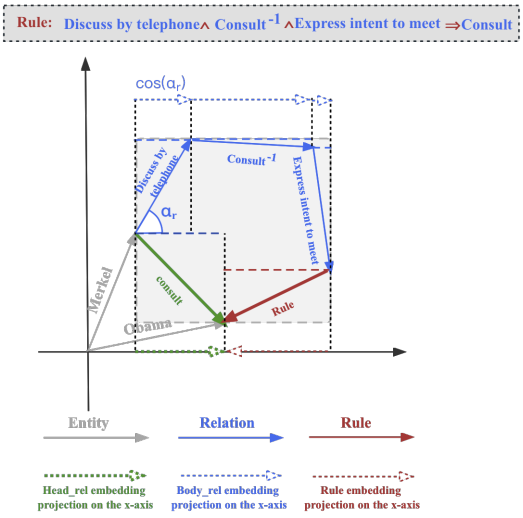

Figure 4: The figure represents the embedding of entities, relations, and rule in the embedding vector space of our model RENN. The gray arrows represent the embedding of entities $Merkel$ and $Obama$, the blue arrows represent the embedding of relations $Consult^{-1}$, $Express\ intent\ to\ meet$ and $Discuss\ by\ telephone$, and the red arrow represent the embedding of rule R. The green arrow represent the embedding of rule head relation $Consult$, and the dashed arrow represent the projection of a vector on the x-axis.

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

Table 1: Statistics on datasets.

3.4. Pretraining of Logic Embedding

The central aspect of RENN involves the integration of logical rules into neural networks. Our approach focuses on embedding these logical rules for subsequent incorporation into the neural network architecture.

To jointly pretraining the embedding of relations and logical rules in the same vector space, as shown in Figure 4, a natural idea is to model relations in rule body $r_l \wedge r_{l-1} \wedge \ldots \wedge r_1$ and logical rules R as the rotation of arrows from the subject entity to the object entity in query quad. The green arrow in Figure 4 represent the embedding of relation r^c_{l+1} between relations in rule body and rule theoretically similar to the relation r_q in the query quad, which is our target rule head relation r_{l+1}.

For a given temporal logic rule $R : r_l \wedge r_{l-1} \wedge \ldots \wedge r_1 \rightarrow r_{l+1}$, from Figure 4 we can easily find that the sum of the multiple rotation angles of rule body relation embedding and rule embedding in the vector space is equivalent to the corresponding angles $\alpha^c_{r_{l+1}}$ of rule head, i.e.,

$$\alpha_{r_{l+1}} \approx \alpha^c_{r_{l+1}} = \sum_{i=1}^{l} \alpha_{r_i} + \alpha_R \tag{4}$$

Relation embedding α_r represents the rotation angle α of the relation r in vector space. Specifically, we limit the modulus of each relation r and rule R to 1. Then as show in Figure 4, the projection of the embedding on the x-axis is $\cos(\alpha_r)$. Similarly, the sum of projections of the multiple rotation angles of rule body relation $\cos(\alpha_{r_i})$ and rule $\cos(\alpha_R)$ in the vector space is equivalent to the projections of corresponding angle $\cos(\alpha_{r_{l+1}})$ of rule head.

We could use the embedding of $[r_1, r_2, \ldots, r_l, R]$ as $[\alpha_{r_1}, \alpha_{r_2}, \ldots, \alpha_{r_l}, \alpha_R]$ to obtain the embedding of r^c_{l+1} to approximate the embedding of r_{l+1}. We employ an LSTM to generate the embedding of rule header $\alpha^c_{r_{l+1}}$. We construct a cascaded input LSTM network for establishing the relationship between the rule and its corresponding rule body. The input of the network is the sum of projection of the angle vector of rule body relations and rule, the output of this network $\alpha^c_{r_{l+1}}$ is theoretically expected to approximate the embedding of the rule head relation $\alpha_{r_{l+1}}$. Our training objective is to minimize the difference between the actual rule head relation r_{l+1} and the output approximate rule head relation r^c_{l+1}. To minimize the difference between the actual relation r_{l+1} and the approximate relation r^c_{l+1}, we define the euclidean distance $d(R, r_1, r_2, \ldots, r_{l+1})$ using a binomial norm as follows,

$$d(R, r_1, r_2, \ldots, r_{l+1}) = \left\| \mathrm{LSTM}\left[\sum_{i=1}^{\iota} \cos(\alpha_{r_i}) + \cos(\alpha_R)\right] - \cos(\alpha_{r_{l+1}}) \right\| \tag{5}$$

In Eq.(5) distance function, the objective is to align the sum of the angle vector of the rule body relations and the rule with the angle vector of the rule head relation.

In theory, when the rule R has a high confidence score, the embedding of the LSTM output and rule header relationship $\alpha_{r_{l+1}}$ should have high likelihood. To future pretrain the embeddings, the loss using a logsigmoid is defined as follows,

$$L(R, r_1, r_2, \ldots, r_{l+1}) = -\log \sigma(\gamma_r - d(R, r_1, r_2, \ldots, r_{l+1})) \tag{6}$$

3.5. Logic Enhanced TKG Completion Model

The temporal logic rule embedding can be used for any existing TKG completion models by taking a copy of the logic embedding as input to the models.

For example, TITer (Sun et al., 2021) is reinforcement learning besed model, which takes the dynamic embedding and historical path embedding of the generated entity as input to obtain the expected prediction. The injection of temporal logic rules could be the embedding matrix of all relations in TKG. In the pre-training of the RENN framework, we jointly trained the embedding of relations and rules in the same vector space, thereby jointly maximizing the likelihood of existing quads and logical rules. Using the pre-trained relation embedding matrix as the initialization of the relation embedding matrix in the TITer model. Compared to the embedding of the original TITer, the embedding of quads in RENN-TITer has higher similarity to logical rules.

The reinforcement learning strategy network $\Pi(\theta)$ of TITer utilizes the relation embedding r in the agent transfer path to generate historical path embedding h_n, n represent the number of steps in the history path in reinforcement learning.

$$\mathbf{h}_n = \mathrm{LSTM}\left(\mathbf{h}_{n-1}, \left[\mathbf{r}_{n-1}; \mathbf{e}^{t_{n-1}}_{n-1}\right]\right)$$

With input the dynamic entity embedding and historical path embedding into the hierarchy, the expected output entities and expected output relations are obtained. Finally, obtain the score of the candidate entities and select the entities with the highest score.

4. Experiments

4.1. Experimental Setup

4.1.1. Datasets

We experiment on ICEWS-14 (García-Durán et al., 2018), ICEWS18 (Jin et al., 2020), YAGO (Mahdisoltani et al., 2015) and WIKI for TKG completion. The detailed statistics of the datasets are listed in Table 1, including the number of quadruples in training set/validation set/test set, the number of entities and relations in the datasets.

4.1.2. Evaluation Metrics

In the entity completion task, the model scores and ranks all candidate entities in the entity set for the knowledge to be inferred that lacks the object or subject entities in the test set, and selects the highest ranked entity as the answer. The evaluation indicators in the experiment include the Mean Reciprocal Ranking (MRR) (Lacroix et al., 2020) and the top 1/3/10 hit rate of the target entity Hit@1/3/10. Among them, the Mean Reciprocal Ranking indicator calculates the mean reciprocal ranking of all queries in the test set. The larger the indicator, the higher the ranking of correct answers and the better the sorting effect. The indicator Hit@N represents the proportion of the number of queries that hit answers in the top N of the sorting results to the test set. The larger the indicator, the better the sorting effect.

4.1.3. Baseline

We compare our model with three kinds of methods.

(1) Symbolic logic TKG completion method, TLogic (Liu et al., 2022) is currently the best TKG completion method based on symbolic logic, which extracts time-based logical rules through time random walks and obtains inference results through rule grounding.

(2) Interpolated TKG reasoning methods, including TTransE (Leblay and Chekol, 2018), TA-DistMult (García-Durán et al., 2018), DE-SimpleE (Goel et al., 2020), and TNTComplEx (Lacroix et al., 2020). We use these method to conduct the extrapolation task and report the result as in (Sun et al., 2021).

(3) Extrapolation TKG completion methods, including RE-NET (Jin et al., 2020), CyGNet (Zhu et al., 2021), xERTE (Han et al., 2021), TITer(Sun et al., 2021), L2TKG (Zhang et al., 2023a) and HGLS (Zhang et al., 2023b).

We use the RENN framework to pre-train TITer(Sun et al., 2021) and HGLS(Zhang et al., 2023b) methods, obtain our two experimental methods, RENN-TITer and RENN-HGLS.

4.1.4. Implementation Details

Our model is implemented using PyTorch. The parameters of the baselines are consistent with the settings in the original text.

The length of the logical rules extracted from TKGs is 1, 2, and 3, respectively. The confidence score and the rule body support threshold of the rule filter are 0.1 and 20, respectively. Using the Adam optimizer to optimize parameters, the rule learning rate is 0.0005. During the pre-training period, the batch size was set to 2048. The maximum number of steps for training is 50000.

4.2. Results and Analysis

4.2.1. Performance on TKG Completion.

Table 2 presents a comparison of the results obtained by RENN-TITer and RENN-HGLS on the test sets of four TKG datasets, relative to a baseline method. Our RENN pre-training approach is compared against TITer and HGLS models, using TITer and HGLS as baselines to evaluate the effectiveness of injecting logical rules. RENN pre-training has demonstrated improvements over TITer and HGLS methods on all the datasets, highlighting the efficacy of the pre-training method in incorporating logical rules. In comparison to all baseline methods, our two methods based on the RENN framework have consistently achieved the best results across nearly all evaluation metrics.

4.2.2. RENN's Effectiveness On Static Knowledge Graphs

The model RENN should exhibit logical and predictive effectiveness on static knowledge graphs without timestamps. We have conducted experiments on three widely used public static KG datasets - UMLS, Kinship, and Family. The baselines for comparison were two classical static knowledge graph embedding methods: RotatE and TransE. We applied our RENN framework, modified to exclude time handling, to both baseline methods and observed improvements over the baselines. The specific results are as Table 3.

This experiment successfully demonstrates the effectiveness of our RENN method for static knowledge graph reasoning tasks.

Methods	ICEWS14				ICEWS18				YAGO				WIKI			
	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
TLogic	43.0	33.6	48.3	61.3	29.8	20.5	33.9	48.5	-	-	-	-	-	-	-	-
TTransE	13.4	3.1	17.3	34.6	8.3	1.9	8.6	21.9	31.2	18.1	40.9	51.2	29.3	21.7	34.4	42.4
TA-DistMult	26.5	17.1	30.2	45.5	16.8	8.6	18.4	33.6	54.9	48.2	59.6	66.7	44.5	39.9	48.7	51.7
DE-SimplE	32.7	24.4	35.7	49.1	19.3	11.5	21.9	34.8	54.9	51.6	57.3	60.2	45.4	42.6	47.7	49.6
TNTComplEx	32.1	23.4	36.0	49.1	27.5	19.5	30.8	42.9	57.9	52.9	61.3	66.7	45.0	40.0	49.3	52.0
CyCNET	32.7	23.7	36.3	50.7	24.9	15.9	28.3	42.6	52.1	45.4	56.1	63.8	33.9	29.1	36.1	41.9
RE-NET	38.3	28.7	41.3	54.5	28.8	19.1	32.4	47.5	58.0	53.1	61.1	66.3	49.7	46.9	51.2	53.5
xERTE	40.8	32.7	45.7	57.3	29.3	21.0	33.5	46.5	84.2	80.1	88.0	89.8	71.1	68.1	76.1	79.0
L2TKG	47.4	35.4	-	**71.1**	33.4	22.2	-	55.0	-	-	-	-	-	-	-	-
TITer	41.7	32.7	46.5	58.4	29.9	22.1	33.5	44.8	87.5	84.9	89.9	90.3	73.3	70.9	74.8	76.7
HGLS	47.0	35.1	52.7	70.4	29.3	19.2	-	49.8	79.9	75.7	82.6	87.0	75.6	71.6	77.9	82.4
RENN-TITer	41.9	32.9	46.8	58.6	30.2	**22.3**	33.8	44.9	**88.5**	**86.1**	**90.9**	**91.3**	75.3	**73.2**	76.8	78.4
RENN-HGLS	**47.8**	**36.3**	**53.9**	70.3	**33.6**	22.1	**38.1**	**57.1**	80.9	75.9	82.9	87.3	**77.6**	72.9	**79.3**	**83.7**

Table 2: Results on TKG completion.The best performance is highlighted in boldface. The confidence interval for the experiment is $confidence(0.1, 1]$, $number\_of\_body\_support(20, +\infty)$.

Methods	UMLS				Kinship				Family			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	70.4	55.4	82.6	92.9	30.0	14.3	35.2	63.7	81.3	67.5	94.6	98.5
RENN-TransE	74.8	61.8	85.1	93.4	34.7	20.7	39.8	62.3	82.0	68.9	94.6	98.6
RotatE	80.2	69.6	89.0	96.3	67.2	52.8	76.4	93.5	94.1	85.3	97.4	99.0
RENN-RotatE	82.7	74.9	88.9	95.5	67.3	53.8	77.5	95.0	97.5	96.7	98.5	98.6

Table 3: Two classical static knowledge graph inference methods TransE and RotatE compare the RENN enhancement results on different static knowledge graph datasets with the original results.

4.2.3. Unseen Entity Inference

To provide more detailed statistics, $Unseen_{more}$ and $Unseen_{less}$ in Table 4 summarizes two cases of unseen entities in the test set of ICEWS14. The unseen entities are entities that do not exist in the training and validation sets. The table includes the following details: the number of unseen entities in the test set N^{ent}_{unseen}, the number of quadruples where the object entity is unseen N^{ob}_{unseen}, the number of quadruples where the subject entity is unseen N^{su}_{unseen}, the number of quadruples where both the subject entity and object entity are unseen N^{obs}_{unseen}.

Dataset	N^{ent}_{unseen}	N^{ob}_{unseen}	N^{su}_{unseen}	N^{obs}_{unseen}
$Unseen_{more}$	235	165	169	24
$Unseen_{less}$	76	80	80	10

Table 4: Number of unseen entities in the test set of dataset ICEWS14.

Table 5 shows the prediction results of our method RENN-HGLS and the original HGLS method without injecting logical rules on the $Unseen_{less}$ and $Unseen_{more}$ test sets of ICEWS14, respectively. The difference in the results obtained by the RENN-HGLS method on two different unseen entity test sets is 0.69%, 0.19%, 0.73% and 2.21% respectively, which is significantly lower than the difference in the HGLS method on the two test sets as 1.93%, 1.92%, 2.29%, 1.94%. The above data indicates that the pre-training method using the RENN framework has stronger ability in inferring unseen entities.

Model	Dataset	MRR	Hit@1	Hit@3	Hit@10
RENN-HGLS	$Unseen_{less}$	48.50	36.44	54.62	72.51
	$Unseen_{more}$	47.81	36.25	53.89	70.30
HGLS	$Unseen_{less}$	48.93	36.98	55.01	72.35
	$Unseen_{more}$	47.00	35.06	52.72	70.41

Table 5: The prediction results on unseen entities, with the confidence interval for the experiment as $confidence(0.1, 1]$, $number\_of\_body\_support(20, + \infty)$.

Dataset	Filter parameter	I_R	R_H	R_{L_1}	R_{L_2}	R_{L_3}
ICEWS14	$filter^{0}_{0}$	28661	428	8338	7991	12335
	$filter^{0.1}_{20}$	4803	428	1906	402	2495
WIKI	$filter^{0}_{0}$	570	48	83	7	480
	$filter^{0.1}_{20}$	334	48	56	1	277

Table 6: The I_R indicates the number of Initial rules, R_H indicates the number of different rule heads, R_{L_1}, R_{L_2} and R_{L_3} represents the number of rules with lengths of 1, 2 and 3 respectively.

4.3. Ablation and Parameter Studies

4.3.1. Necessity of Rule Confidence Filtering

Taking ICEWS14 and WIKI data sets as examples, the number of rules get filtered out in the experiments shown in Table 6, where the filter confidence and number of support sets thresholds are 0.1 and 20. Results show that the rules from ICEWS14 and WIKI were respectively filtered out 23,858 and 236.

Figure 5 shows the experimental results of different parameters in the RENN-HGLS method's rule filter on ICEWS14. Two experiments filtered out rules are conducted with confidence scores less than 0.01 and support sets less than 2, and rules with confidence scores less than 0.1 and support sets less than 20. In addition, we set both filtering parameters to 0 to obtain experimental results on the removal rule filtering mechanism. The experi-

Filter parameter	Dataset	MRR	Hit@1	Hit@3	Hit@10
$filter_2^{0.01}$	ICEWS14	40.15	30.95	44.86	57.46
	WIKI	73.35	71.03	74.84	76.67
	YAGO	88.52	86.01	90.89	91.31
$filter_{20}^{0.1}$	ICEWS14	41.86	32.88	46.81	58.57
	WIKI	75.30	73.17	76.83	78.53
	YAGO	88.74	86.16	91.22	91.55

Table 7: Results of RENN-TITer with various rule filter parameters.The superscript represents the confidence score of the filtered rule, while the subscript represents the number of supported sets of the filtered rule. Respectively filtered out rules are conducted with confidence scores less than 0.01 and support sets less than 2, and rules with confidence scores less than 0.1 and support sets less than 20.

mental results on all evaluation indicators indicate that the method have good performance when the filtered rules have high confidence and a large number of support sets. There is a necessity for the existence of a rule filtering mechanism.

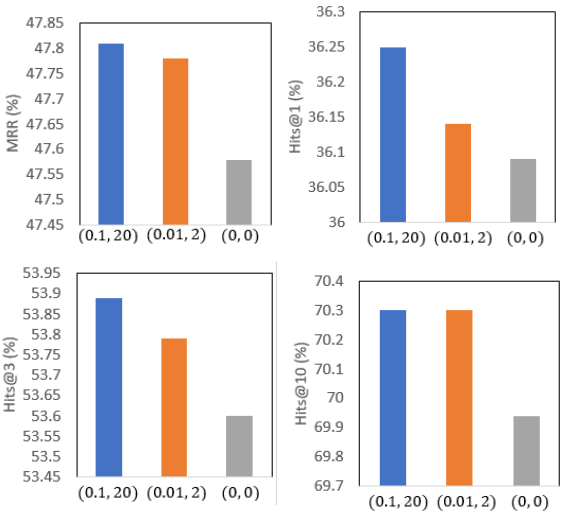

Figure 5: Results of RENN-HGLS with various rule filter parameters.

Table 7 shows the experimental results of the RENN-TITer on datasets ICEWS14 and YAGO, with rule filter parameters of 0.01, 2, and 0.1, 20, respectively. The experimental results show that rules with higher confidence and more support sets after filtering have better performance. The rule filtering mechanism increases the weight of more important rules, which helps to improve the accuracy of experiments.

4.3.2. Maximum Rule Length Research

Using the RENN framework, rules with maximum lengths of $[1, \ldots, 5]$ were extracted from the ICEWS14 dataset. Under filter parameters of 0.1 and 20, RENN-HGLS injected rules with different maximum lengths into the embedding method

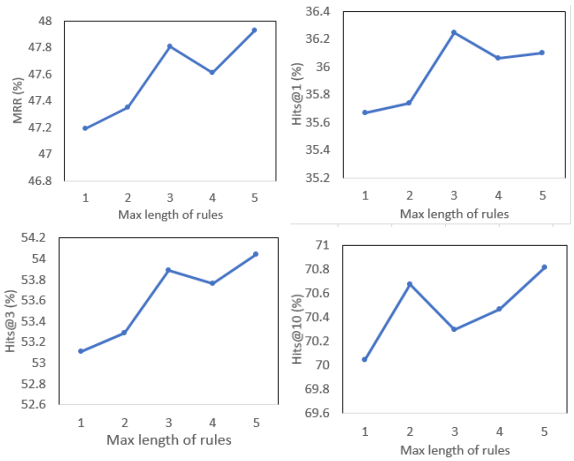

Figure 6: Rule length research.

through pre-training. As shown in Figure 6, the experimental effect was optimal at a maximum length of 5, but it greatly increased the running time and computational complexity of the model. Therefore, we used the relatively good maximum length of 3 to extract the rule with a length of $[1, 2, 5]$.

5. Conclusion

In this paper, we propose a novel model to represent and model logical rules and quadruples for temporal knowledge graphs completion. The model pre-trains by learning the embedding of each logical rule and its corresponding relationship to improve the likelihood of existing quadruples and logical rules, injecting prior logical rules into the embeddings of the TKG. In addition, the filtering mechanism based on rule confidence scores limits the contribution of different weight logical rules Our experiments on multiple temporal knowledge graph datasets verify the effectiveness of the model.

The knowledge graph typically records only accurate facts explicitly and does not label incorrect information. Consequently, when missing facts are present in the knowledge graph, the model may struggle to distinguish whether a quad not appearing in the record is a false fact to be excluded or a missing fact that has been overlooked. Future work could involve implementing mechanisms to address this challenge, such as refining algorithms for fact validation or developing techniques for automated error detection and correction within the knowledge graph.

6. Acknowledgements

This work is supported by the Social Science Planning Foundation of Liaoning Province under Grant L22CTQ002.

7. References

Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. 2023. Temporal knowledge graph completion: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, Macao, SAR, China*, pages 6545–6553.

Melisachew Wudage Chekol, Giuseppe Pirrò, Joerg Schoenfisch, and Heiner Stuckenschmidt. 2017. Marrying uncertainty and time in knowledge graphs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA*, pages 88–94.

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, Brussels, Belgium*, pages 4816–4821.

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

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *9th International Conference on Learning Representations, Virtual Event, Austria*.

Yongquan He, Peng Zhang, Luchen Liu, Qi Liang, Wenyuan Zhang, and Chuang Zhang. 2021. HIP network: Historical information passing network for extrapolation reasoning on temporal knowledge graph. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Virtual Event / Montreal, Canada*, pages 1915–1921.

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

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. In *8th International Conference on Learning Representations, Addis Ababa, Ethiopia*.

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

Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. 2022. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, Virtual Event*, pages 4120–4127.

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, CIDR 2015, Asilomar, CA, USA*.

Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2021. An embedding-based approach to rule learning in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1348–1359.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Virtual Event / Punta Cana, Dominican Republic*, pages 8306–8319.

Youri Xu, Haihong E, Meina Song, Wenyu Song, Xiaodong Lv, Haotian Wang, and Jinrui Yang. 2021. RTFE: A recursive temporal fact embedding framework for temporal knowledge graph completion. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online*, pages 5671–5681.

Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. 2023a. Learning latent relations for temporal knowledge graph reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics,Toronto, Canada*, pages 12617–12631.

Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. 2023b. Learning long- and short-term representations for temporal knowledge graph reasoning. In *Proceedings of the ACM Web Conference 2023,Austin, TX, USA*, pages 2412–2422.

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, Virtual Event*, pages 4732–4740.