RETIA: Relation-Entity Twin-Interact Aggregation for Temporal Knowledge Graph Extrapolation

Kangzheng Liu[†‡], Feng Zhao[†*], Guandong Xu[‡], Xianzhi Wang[‡], and Hai Jin[†]
[†]National Engineering Research Center for Big Data Technology and System
Services Computing Technology and System Lab, Cluster and Grid Computing Lab
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
[‡]Data Science and Machine Intelligence Lab, University of Technology Sydney, Sydney, Australia
[†]{frankluis, zhaof, hjin}@hust.edu.cn, [‡]{guandong.xu, xianzhi.wang}@uts.edu.au

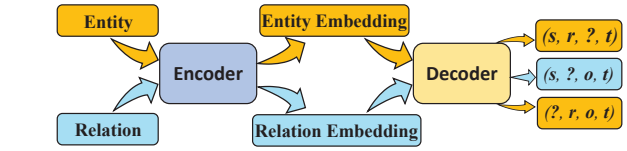
***Abstract*—Temporal knowledge graph (TKG) extrapolation aims to predict future unknown events (facts) based on historical information, and has attracted considerable attention due to its great practical significance. Accurate representations (embeddings) of entities and relations form the basis of TKG extrapolation. Recent work has been devoted to improving the rationality of entity representations. However, on the one hand, ignoring relation modeling results in incomplete relation representations; therefore, some approaches aggregate only immediately adjacent entities of relations, but this can lead to the "message islands" problem of relation modeling. On the other hand, ignoring the association constraints between entities and relations can make the embeddings of both entities and relations prone to overfitting. To address these challenges, we propose a novel method, namely, RETIA. For the former issue, we generate twin hyperrelation subgraphs for each historical subgraph and then aggregate both the adjacent entities and relations in the hyperrelation subgraphs through a graph convolutional network (GCN). For the latter concern, we propose a twin-interact module (TIM), which provides communication channels for relation aggregation and entity aggregation during the evolution of the historical sequence. Experiments conducted on five benchmark datasets demonstrate the substantial improvements achieved by RETIA in terms of multiple evaluation metrics. Our released code is available at https://github.com/CGCL-codes/RETIA.**

***Index Terms*—Temporal knowledge graph extrapolation, Twin-interact aggregation, Graph convolutional network**

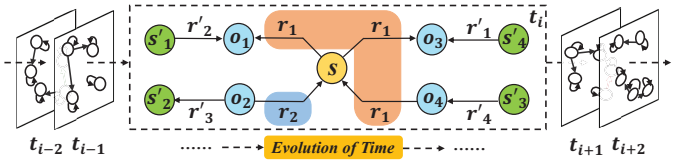
I. Introduction

Temporal knowledge graphs (TKGs) represent facts as quadruples $(subject, relation, object, time)$ and are actually sequences of temporal subgraphs divided by the time (timestamp) dimension. TKG extrapolation involves the prediction of incomplete facts in future subgraphs, including missing entities and missing relations, by modeling the subgraphs of historical timestamps. Due to its great practical significance, TKG extrapolation is widely used in scenarios such as stock forecasting [1] and crisis forewarning [2].

TKG extrapolation has recently become a research hotspot. A great deal of work [3]–[7] has been done regarding entity forecasting research, which has improved remarkably. For example, RE-NET [3] models the evolution of an entity representation over time through a recurrent neural network (RNN), and CEN [7] follows RE-GCN [8] to model the temporal evolution processes of entities via a recurrent relational GCN (R-GCN). However, few studies have modeled relation representations. As shown in Figure 1(a), the common approaches are codec-based architectures. Using encoders, we obtain the embeddings of entities and relations; then, through decoders, incomplete facts at a future timestamp t can be forecast. On the one hand, for future relation forecast $(s, ?, o, t)$, we need not only the embeddings of entities s and o at future timestamp t but also the representations of all the relations in a score function of the quadruples to calculate the forecasting scores; thus, inappropriate relation embeddings directly cause the model to confuse the relations in the candidate set and be unable to forecast the missing relations. On the other hand, for future entity forecast $(s, r, ?, t)$ or $(?, r, o, t)$, relation embeddings also act as the inputs of the decoders, in turn affecting the accuracy of the calculated entity scores. Therefore, it is critical to obtain appropriate representations of entities and relations through encoders.



(a) An illustration of the role of relation embeddings.



(b) An illustration of aggregating neighborhood information for entities and relations.

Fig. 1. An illustration of relation embedding and adjacent information aggregation for entities and relations.

A series of studies [7]–[11] has shown that aggregating adjacent structure information is essential for the process of learning embeddings. As shown in Figure 1(b), according to the research on GCNs [9], [10], [12] for relational data, to obtain an accurate representation of entity s, we need to aggregate the relation-connected adjacent entities $\{r_1$:

*Corresponding author

$\{o_1, o_3, o_4\}, r_2 : o_2\}$ of entity s. Despite the maturity of entity aggregation approaches, little work thus far has focused on aggregating adjacent information for relations in the scenarios of temporal dynamics. RE-GCN [8] only tries to model the embeddings of a relation r_1 via the immediately adjacent entity information $\{o_1, o_3, o_4\}$; it ignores the more important adjacent relation information $\{r_2^{'}, r_1^{'}, r_4^{'}\}$. On the one hand, from the expression of a specific relation, the adjacent relation information is located at the same representation level as the modeled relation r_1, so its importance in relation aggregation is no less than that of the adjacent entity information in entity aggregation. On the other hand, it is difficult to simultaneously consider the adjacent information integrity of relation aggregation with the traditional entity-centric modeling strategy. As shown in Figure 1(b), in the relation modeling of temporal dynamic scenarios, if the update of relation r_1 is taken as an example, when the adjacent relation $r_1^{'}$ of relation r_1 passes messages by entity o_3, o_3 should be updated. At this time, entity o_3 acts as not only the message receiver of relation $r_1^{'}$ but also the message sender of relation r_1. Therefore, the update of o_3 implies a bridge for message passing between the two relations r_1 and $r_1^{'}$. In contrast, some of the latest works, such as RE-GCN [8] and TiRGN [12], aggregate only the immediately adjacent entity information of relations. Under these circumstances, as Figure 1(b) shows, the relations r_1 and $r_1^{'}$ both require the aggregation of the adjacent entity o_3; then, as the aggregated object, o_3 is always the sender of the message-passing process and thus cannot receive a new message and update the embeddings. Therefore, messages from relations (e.g., r_1 or $r_1^{'}$) can never cross the immediately adjacent entities and propagate to outer relations (e.g., $r_1^{'}$ or r_1), which we call the "message islands" problem in relation modeling. The message islands in the subgraphs are centered on different relations and bounded by the entities immediately adjacent to them, thereby aggravating the incompleteness of the relation representations.

As Figure 1(b) shows, to obtain an accurate representation of the relation r_1 and the entity s, the entity-connected adjacent relations $\{o_1 : r_2^{'}, o_3 : r_1^{'}, o_4 : r_4^{'}\}$ of relation r_1 and the relation-connected adjacent entities $\{r_1 : \{o_1, o_3, o_4\}, r_2 : o_2\}$ should be aggregated. In contrast to focusing on only a particular static subgraph, the association constraints refer to the fact that the embeddings of entities o_1, o_3, and o_4 at the previous timestamp need to be involved in the update of the relation embedding r_1 at the next timestamp, and the relation embeddings r_1 and r_2 at the previous timestamp need to be involved in the update of the entity embedding s at the next timestamp. If the embeddings of entities and relations are considered only in parallel without consideration of the temporal interactions between them, the resulting representation becomes unreasonable due to an overfitting tendency. In TKGs, the modeling of the association constraints between entities and relations is closely related to the evolution of historical sequences. For example, following RE-GCN [8], TiRGN [12] updates the relation embeddings of the next historical subgraph with the mean-pooled one-hop entity embeddings of the previous historical subgraph and models the evolutionary constraints between entities and relations through a gated recurrent unit (GRU) [13]. However, both approaches ignore the relative positional associations between edges (relations) and nodes (entities). As shown in Figure 1(b), entity o_1 acts both as an in-degree node for the relations r_1 and $r_1^{'}$; nevertheless, entity o_4 acts as an in-degree node for relation $r_4^{'}$ but as an out-degree node for relation r_1. Therefore, the message interactions between entities and relations are constrained by the positional structure between them.

In this paper, we propose a novel representation learning method for TKG extrapolation, namely, **R**elation-**E**ntity **T**win-**I**nteract **A**ggregation (RETIA). To prevent message islands, we need to aggregate the complete adjacency information of the relations as entities; thus, we propose a relation aggregation module (RAM). As shown in Figure 2(a), we first map the relations in the original historical subgraph to those in a twin hyperrelation subgraph and then generate hyperrelations between the relations based on their entity-connected positional neighborhood. Then, we aggregate the complete adjacent information of the relations in the sequential hyperrelation subgraphs based on the relation-aggregating R-GCN [9] and a residual GRU [13] (R-GRU). On the other hand, to account for the association constraints between entities and relations, we propose a twin-interact module (TIM). As Figure 2(b) shows, we continuously transmit the entity embeddings of the previous historical subgraph from the entity aggregation module (EAM) to the RAM and participate in updating the relation and hyperrelation embeddings in the next historical hyperrelation subgraph. The relation embeddings in the previous historical hyperrelation subgraph are passed from the RAM to the EAM and participate in the aggregation of the entity embeddings in the next historical subgraph. In particular, we aggregate the relation embeddings to update the associated hyperrelation embeddings via hyper mean pooling and use hyper long short-term memory (LSTM) [14] to model the temporal evolution of the hyperrelations, which present the positional association constraints between the relations and entities.

Our contributions are summarized as follows:

- To overcome the "message islands" problem and obtain accurate relation embeddings, we propose a RAM, that aggregates not only the immediately adjacent entities but also the adjacent relations of the relations in the hyperrelation subgraph.
- To capture the association constraints between the entities and relations in TKGs, especially the positional association constraints, we propose a TIM, which evolutionally models the interactions of the vector flows between the RAM and EAM.
- Extensive experiments are conducted on five public TKG datasets. The improvement achieved in terms of almost all the evaluation metrics demonstrates the effectiveness of our proposed method for TKG extrapolation.

The remainder of this paper is organized as follows. Related work, including that on existing static and dynamic

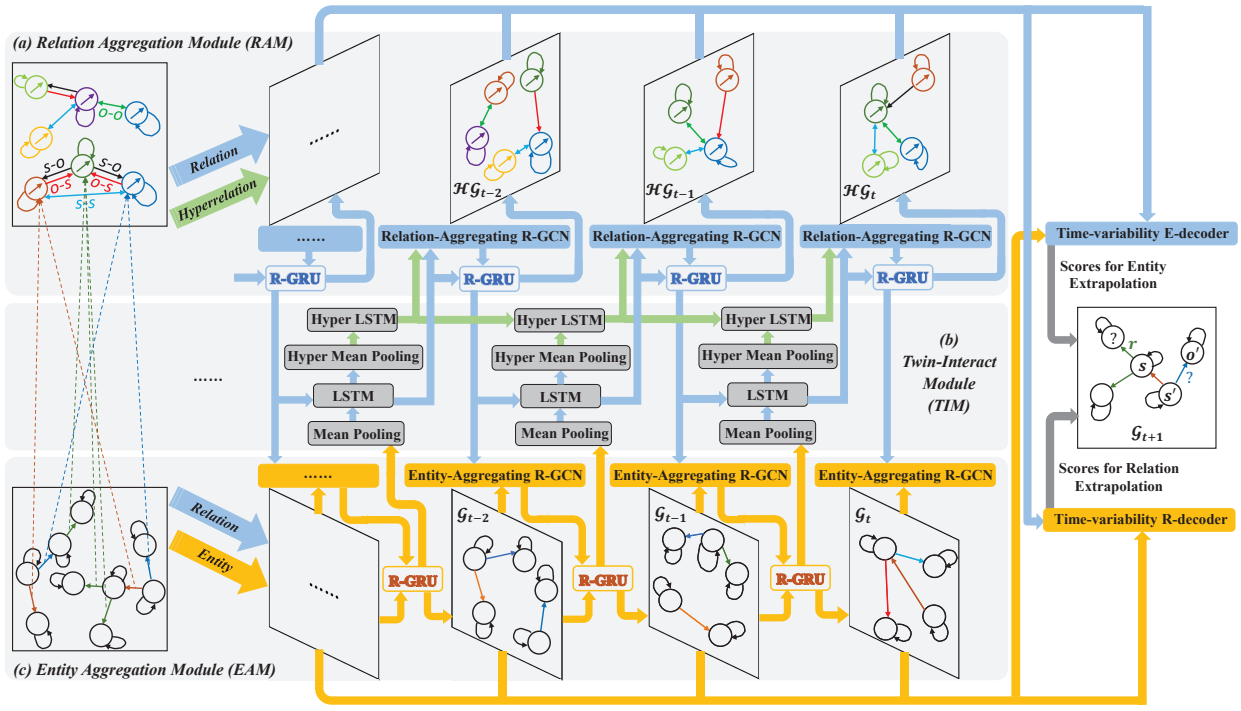


Fig. 2. The framework of our RETIA model. The blue, orange, and green bar lines indicate the transmission of the relation, entity, and hyperrelation embeddings, respectively. The leftmost dashed lines represent the mapping of the relations from the original subgraphs to the hyperrelation subgraphs.

extrapolation methods for TKGs, is introduced in Section II. The proposed model is detailed in Section III. In addition, the experiments and analyses are presented in Section IV, followed by the conclusion in Section V.

II. Related Work

Existing approaches to event forecasting over TKGs can be divided into two categories according to their data modeling techniques: static strategies and dynamic strategies.

1) Static Modeling: Static modeling methods do not take temporal dynamics into consideration. Translation-based methods include TransE [15] and TransH [16]; they map entities and relations onto a low-dimensional space. The matrix decomposition-based methods include DistMult [17] and ComplEx [18]. DistMult [17] models a relation as a matrix of linear transformations in a vector space. ComlpEx [18] extends the original representation learning paradigm to a complex space. ConvE [19] introduces 2D convolution for knowledge embeddings. Conv-TransE [20] preserves the translational property of embeddings based on ConvE. Rotation-based methods include RotatE [21], which models relations as rotations from subject entities to object entities. Traditional static methods learn the embeddings of entities and relations simultaneously in a unified low-dimensional space. However, some early GCN-based methods conduct information aggregation for only entity representations. R-GCN [9] adopts the concept of relation basis to assist in aggregating the neighborhood information for entities. Then, some advanced work modeled relation representations. Comp-GCN [10] leverages entity-relation composition operations to jointly embed both nodes and relations in a graph. StarE [22] (an improved version of Comp-GCN) introduces a representation learning method for hyperrelational graphs.

Compared with previous work on relation aggregation, our proposed RETIA features the following major differences. 1) Comp-GCN is an aggregation model for static graphs. However, RETIA is an extrapolation framework for a sequence of static graphs (i.e., TKG) considering temporal dynamics. 2) StarE associates additional information (tuples of key-value pairs, called statement qualifiers) with main triples to disambiguate or constrain the validity of a triple fact under different circumstances. The hyperrelational KGs it can model consist of facts with high-dimensional representations, such as $(s, r, o, (qr_1, qv_1), (qr_2, qv_2), ...)$, where the tuple (qr_i, qv_i) refers to qualifier pairs. It is generally used for representation learning of high-dimensional KGs but does not exactly fit temporal scenarios. For example, real-world facts (s, r, o) are generally valid for a certain timestamp range $[t_0, t_2]$, then the representation of $(s, r, o, (time, t_0), (time, t_1), (time, t_2))$ is still essentially a compression of temporal information into a static graph for message passing. Thus, neither the evolution of the temporal information $\{t_0, t_1, t_2\}$ nor the structural dependencies between timestamps can be taken into account. 3) RETIA depicts each timestamp as a separate static subgraph, in which case the facts (s, r, o, t_0), (s, r, o, t_1), (s, r, o, t_2) are on the subgraphs $\mathcal{G}_{t_0}$, $\mathcal{G}_{t_1}$, and $\mathcal{G}_{t_2}$, respectively. The hyperrelation subgraphs generated in the RAM consist of relation nodes and hyperrelations that express the relative positions between entities and relations. In summary, it aims to solve the problems of relation aggregation integrity and positional association constraint in temporal dynamic scenarios.

2) Dynamic Modeling: With dynamic methods, time information is modeled in two scenarios: interpolation and extrapolation. The methods that operate under the interpolation setting use global resources (past and future) to make predictions about current facts; examples include TTransE [23], HyTE [24], and TA-DistMult [25]. TTransE [23] and TA-DistMult [25] integrate time information into the relation embeddings of the corresponding occurring facts. HyTE [24] projects relations and entities to time-specific hyperplanes. In the extrapolation setting, the forecast facts correspond to a future timestamp, and only historical information is available; i.e., unknown future events are forecast. RE-NET [3] models historical information as conditional probabilities. CyGNet [4] uses a copy mechanism to extract one-hop repetitive entities from historical data. xERTE [5] constructs an inference graph for a query subgraph. TITer [6] extracts candidate paths through reinforcement learning. CluSTeR [26] also extracts query-related subgraphs through reinforcement learning and then models candidate entity embeddings with a GCN. TLogic [27] generates query paths based on temporal logic rules. CEN [7] addresses time-variability issues through online learning strategies. Nevertheless, these approaches fail to model relation representations for the following two reasons. First, different from a static modeling strategy, in dynamic modeling, the timestamps separate TKGs into independent subgraph spaces; thus, temporal models cannot learn the embeddings of entities and relations simultaneously in a unified space. Second, the connections between different subgraph spaces require explicit temporal evolution modeling. However, the abovementioned methods lack loss designs and thus cannot explicitly model relations.

RE-GCN [8] and TiRGN [12] both model relation representations and the association constraints between only relations and the entities immediately adjacent to them via mean pooling and a GRU [13], respectively. Therefore, they still fail to aggregate outer adjacent relation information, resulting in "message islands" and positional association constraints between the relations and entities. Our proposed RETIA is an extrapolation method that aggregates the complete adjacent information of relation nodes in hyperrelation subgraphs to overcome the problem of message islands in the context of the relation modeling of temporal dynamics and models positional association constraints on the basis of common association.

III. The RETIA Model

In this section, we introduce the proposed RETIA model.

A. Notations and Definitions

Table I provides the set of notations and the corresponding descriptions used in our RETIA model. Note that for the hyperrelation facts $(r_s, hyper\text{-}r, r_o, t)$ in $\mathcal{HG}_t$, the relations r_s and r_o are mapped from the original subgraph $\mathcal{G}_t$, and $hyper\text{-}r$ is generated according to the relative positions among the entities and relations. Formally, for a future subgraph $\mathcal{G}_{t+1}$ at a specific timestamp t+1 $\in [0, T\text{-}1]$, TKG interpolation involves predicting incomplete facts given the global information of all the temporal subgraphs $\{\mathcal{G}_\tau | 0 \leq \tau \leq T\text{-}1\}$. In contrast, TKG extrapolation involves forecasting a missing object entity $(s, r, ?, t{+}1)$, a missing subject entity $(?, r, o, t{+}1)$, or a missing relation $(s, ?, o, t{+}1)$ according to previous historical k-length temporal subgraphs $\{\mathcal{G}_\tau | t\text{-}k{+}1 \leq \tau \leq t\}$. For the original quadruples (s, r, o, t) and hyperrelation quadruples $(r_s, hyper\text{-}r, r_o, t)$ at any timestamp t, we add the inverse relation facts (o, r^{-1}, s, t) and the inverse hyperrelation facts $(r_o, hyper\text{-}r^{-1}, r_s, t)$ to the t^{th} subgraph and hyperrelation subgraph, respectively; thus, only the in-degree edges need to be considered, and the actual numbers of modeled relations and hyperrelations are $2M$ and $2H$.

TABLE I
Set of notations used in the RETIA model.

Notations	Descriptions
$\mathcal{G}$	A TKG
$\mathcal{E}$	Set of entities in $\mathcal{G}$
$\mathcal{R}$	Set of relations in $\mathcal{G}$
$\mathcal{HR}$	Set of hyperrelations in $\mathcal{G}$
$\mathcal{T}$	Set of timestamps in $\mathcal{G}$
N	Number of entities (Size of $\mathcal{E}$)
M	Number of relations (Size of $\mathcal{R}$)
H	Number of hyperrelations (Size of $\mathcal{HR}$)
T	Number of timestamps ($t \in \{0, ..., T\text{-}1\}$)
$\mathcal{G}_t$	A temporal subgraph composed of facts (s, r, o, t)
$\mathcal{HG}_t$	Twin hyperrelation subgraph of $\mathcal{G}_t$ composed of hyperrelation facts $(r_s, hyper\text{-}r, r_o, t)$
d	Embedding dimensionality
k	Length of the historical subgraph sequence
$\mathbf{E}_t$	Embedding matrix of all the entities in $\mathcal{G}_t$
$\mathbf{R}_t$	Embedding matrix of all the relations in $\mathcal{G}_t$ and $\mathcal{HG}_t$
$\mathbf{HR}_t$	Embedding matrix of all the hyperrelations in $\mathcal{HG}_t$
$\mathbf{E}_0$	Input embedding matrix of all the entities for the first historical timestamp
$\mathbf{R}_0$	Input embedding matrix of all the relations for the first historical timestamp
$\mathbf{HR}_0$	Input embedding matrix of all the hyperrelations for the first historical timestamp

B. Architecture Overview

As shown in Figure 2, our proposed RETIA model is composed of the RAM, EAM, and TIM. For an entity or relation forecasting task at a future timestamp t+1, as Figure 2(a) shows, to solve the relation aggregation integrity and positional association constraint of temporal dynamics, we need to first determine the hyperrelations according to their relative positions between relations and entities. In the RAM, to address message islands in relation aggregation for temporal scenarios, we aggregate both the neighboring entity and relation information of the relations, rather than merely the immediately adjacent entities of the relations. Instead of limiting ourselves to the original entity-centric subgraphs, we use relation-aggregation R-GCN [9] in the k relation-centric hyperrelation subgraphs to solve this problem. The TIM is responsible for modeling the association constraints between the RAM and EAM in the historical sequence, especially the positional association constraints. Mean pooling is used to establish a common association between the entity embeddings of the previous timestamp and the relation embeddings of the next timestamp, and then LSTM [14] is used to model the

TABLE II
ILLUSTRATIONS OF THE HYPERRELATIONS.

Hyperrelations	Positional association constraints
o-s	The object of relation r_s is the subject of relation r_o
s-o	The subject of relation r_s is the object of relation r_o
o-o	The relations r_s and r_o have the common object
s-s	The relations r_s and r_o have the common subject

evolution of this common association. Hyper mean pooling is used to further establish the positional association between relation embeddings and hyperrelation embeddings, and then hyper LSTM [14] is used to model the evolution of the positional association.

C. Relation Aggregation Module (RAM)

As the left part of Figure 2 shows, for each subgraph $\mathcal{G}_t$ of the k-length historical sequence, we generate a twin hyperrelation subgraph $\mathcal{HG}_t$. According to previous work [28], we define $\mathcal{HR} = \{\text{o-s,s-o,o-o,s-s}\}$. Each hyperrelation demonstrates the entity-connected relative position between the two specific relations r_s and r_o. We present illustrations of the four types of hyperrelations in Table II. Specifically, as shown in Algorithm 1, for historical subgraph $\mathcal{G}_t$ at a specific timestamp t, we first traverse all the quadruples of the subgraph to obtain the relation-object adjacency matrix $\mathcal{RO}_t$ and the relation-subject adjacency matrix $\mathcal{RS}_t$. Then, taking the hyperrelation o-s as an example, in the original subgraph $\mathcal{G}_t$, if the object entity o of the relation r_s of the fact $(s^{'}, r_s, o)$ is the subject entity s of the relation r_o of the fact $(s, r_o, o^{'})$, the hyperrelation between the relations r_s and r_o is o-s in the corresponding twin hyperrelation subgraph. We obtain the adjacency matrix $\mathcal{OS}_t$ of the hyperrelation o-s through $\mathcal{RO}_t \times \mathcal{RS}_t$. By analogy, we obtain the adjacency matrices $\mathcal{SO}_t$, $\mathcal{OO}_t$, and $\mathcal{SS}_t$ of the hyperrelations s-o, o-o, and s-s through $\mathcal{RS}_t \times \mathcal{RO}_t$, $\mathcal{RO}_t \times \mathcal{RO}_t$, and $\mathcal{RS}_t \times \mathcal{RS}_t$, respectively. Specifically, we set the diagonal elements of $\mathcal{OO}_t$ and $\mathcal{SS}_t$ to zero to prevent the repeated generation of self-loop relation nodes. Finally, we generate $\mathcal{HG}_t$ through the hyperrelation adjacency matrices $\mathcal{OS}_t$, $\mathcal{SO}_t$, $\mathcal{OO}_t$, and $\mathcal{SS}_t$.

Next, we aggregate the neighborhood information of the relations in each temporal hyperrelation subgraph via the relation-aggregating R-GCN. We aggregate the adjacent information of each node (i.e., relation) in the graph via the message-passing operation:

$$\mathbf{r}_{r_o}^{(l)} = f\left(\sum_{hr \in \mathcal{HR}} \frac{1}{c_{r_o,hr}} \sum_{r_s \in \mathcal{R}_{r_o}^{hr}} \mathrm{W}_{hr}^{(l-1)}(\mathbf{r}_{r_s}^{(l-1)} + \mathbf{hr}^{(l-1)}) + \mathrm{W}_0^{(l-1)}\mathbf{r}_{r_o}^{(l-1)}\right) \tag{1}$$

where $\mathbf{r}_{r_o}^{(l-1)}, \mathbf{r}_{r_o}^{(l)} \in \mathbb{R}^{M \times d}$ indicates the embeddings of all the relations in the $(l-1)^{th}$ and l^{th} layers of the relation-aggregating R-GCN, respectively. $\mathcal{R}_{r_o}^{hr}$ indicates the set of relations that are adjacent to node r_o via the hyperrelation hr. $\mathbf{r}_{r_s}^{(l-1)}$ and $\mathbf{hr}^{(l-1)}$ indicate the embeddings of the adjacent relations and the corresponding hyperrelations in the $(l-1)^{th}$ layer of the relation-aggregating R-GCN. $c_{r_o,hr}$ represents the size of $\mathcal{R}_{r_o}^{hr}$. $f(\cdot)$ represents the adopted activation function (the reflected rectified linear unit ($RReLU$) function). $\mathrm{W}_{\mathrm{hr}}^{(l-1)}$ indicates the edge-specific parameters for aggregating the structural features according to different hyperrelations. $\mathrm{W}_0^{(l-1)}$ indicates the parameters for aggregating the self-loop features of all the relations. Thus, the output of the relation-aggregating R-GCN in the twin hyperrelation subgraph $\mathcal{HG}_t$ is the neighborhood-aggregated relation embeddings $\mathbf{R}_{\mathrm{Agg}}^{\mathrm{t}}$ at timestamp t. Generally, as Figure 2(a) shows, the relation-aggregating R-GCN in a certain hyperrelation subgraph $\mathcal{HG}_t$ at the t^{th} historical timestamp can be formally represented as:

$$\mathbf{R}_{\mathrm{Agg}}^{\mathrm{t}} = \mathrm{RAR\_GCN}(\mathbf{R}_{\mathrm{Lstm}}^{\mathrm{t}}, \mathbf{HR}_{\mathrm{t}}) \tag{2}$$

where $\mathbf{R}_{\mathrm{Agg}}^{\mathrm{t}} \in \mathbb{R}^{2\mathrm{M} \times \mathrm{d}}$ is the output of the relation-aggregating R-GCN at the t^{th} historical timestamp.

As shown in Figure 2(a), $\mathbf{R}_{\mathrm{Lstm}}^{\mathrm{t}} \in \mathbb{R}^{2\mathrm{M} \times \mathrm{d}}$ and $\mathbf{HR}_{\mathrm{t}} \in \mathbb{R}^{2\mathrm{H} \times \mathrm{d}}$ are the output relation embeddings and hyperrelation embeddings of the LSTM and the hyper LSTM in the TIM at the t^{th} timestamp, respectively, and they both contain structural information from the previous $(t-1)^{th}$ timestamp. Thus, we use the R-GRU to model the chronological dependencies between the relation embeddings in sequential subgraphs.

Algorithm 1 Hyperrelation subgraph construction algorithm

Input: A subgraph $\mathcal{G}_t$ with quadruples (s, r, o, t)
Output: A twin hyperrelation subgraph $\mathcal{HG}_t$ with quadruples $(r_s, hyper\text{-}r, r_o, t)$

for each quadruple in $\mathcal{G}_t$ **do**
  Determine the relation-object adjacency matrix $\mathcal{RO}_t$ and relation-subject adjacency matrix $\mathcal{RS}_t$.
end for
if $(s^{'}, r_s, o, t) \cap (s, r_o, o^{'}, t) \cap o = s$ **then**
  Obtain the adjacency matrix $\mathcal{OS}_t$ of the hyperrelation o-s through $\mathcal{RO}_t \times \mathcal{RS}_t$.
end if
if $(s, r_s, o^{'}, t) \cap (s^{'}, r_o, o, t) \cap s = o$ **then**
  Obtain the adjacency matrix $\mathcal{SO}_t$ of the hyperrelation s-o through $\mathcal{RS}_t \times \mathcal{RO}_t$.
end if
if $(s, r_s, o, t) \cap (s^{'}, r_o, o^{'}, t) \cap o = o^{'}$ **then**
  Obtain the adjacency matrix $\mathcal{OO}_t$ of the hyperrelation o-o through $\mathcal{RO}_t \times \mathcal{RO}_t$ and set the diagonal elements of $\mathcal{OO}_t$ to zero.
end if
if $(s, r_s, o, t) \cap (s^{'}, r_o, o^{'}, t) \cap s = s^{'}$ **then**
  Obtain the adjacency matrix $\mathcal{SS}_t$ of the hyperrelation s-s through $\mathcal{RS}_t \times \mathcal{RS}_t$ and set the diagonal elements of $\mathcal{SS}_t$ to zero.
end if
Generate $\mathcal{HG}_t$ through the hyperrelation adjacency matrices $\mathcal{OS}_t$, $\mathcal{SO}_t$, $\mathcal{OO}_t$, and $\mathcal{SS}_t$.
return $\mathcal{HG}_t$

Specifically, we normalize the aggregation operation of the relation-aggregating R-GCN by passing the input and output relation embeddings into a GRU cell to accommodate complex modeling:

$$\mathbf{R}_{\mathrm{t}} = \mathrm{R\_GRU}(\mathbf{R}^{\mathrm{t}}_{\mathrm{Agg}}, \mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}}) \tag{3}$$

where $\mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}} \in \mathbb{R}^{2\mathrm{M}\times\mathrm{d}}$ is the output of the LSTM in the TIM at the t^{th} historical timestamp; however, it contains hidden relation and entity information from the $(t-1)^{th}$ timestamp, as mentioned above. $\mathbf{R}_{\mathrm{t}} \in \mathbb{R}^{2\mathrm{M}\times\mathrm{d}}$ is the final relation embeddings of the t^{th} historical subgraph.

D. Entity Aggregation Module (EAM)

This module is designed to aggregate the neighboring entity and relation information of the entities in each historical subgraph of the TKGs.

As Figure 2(c) shows, we aggregate the neighborhood information of the entities of each historical subgraph through the entity-aggregating R-GCN. Similar to the RAM processing strategy, we adopt message-passing architecture here:

$$\mathbf{e}_o^{(l)} = f\left(\sum_{r\in\mathcal{R}}\frac{1}{c_{o,r}}\sum_{s\in\mathcal{E}_o^r}\mathrm{W}_r^{(l-1)}(\mathbf{e}_s^{(l-1)}+\mathbf{r}^{(l-1)})+\mathrm{W}_0^{(l-1)}\mathbf{e}_o^{(l-1)}\right) \tag{4}$$

where $\mathbf{e}_o^{(l-1)}, \mathbf{e}_o^{(l)} \in \mathbb{R}^{N\times d}$ indicate the embeddings of all the entities in the $(l-1)^{th}$ and l^{th} layers of the entity-aggregating R-GCN, respectively. $\mathcal{E}_o^r$ indicates the set of entities that are adjacent to node o and connected by relation r. $\mathbf{e}_s^{(l-1)}$ and $\mathbf{r}^{(l-1)}$ indicate the embeddings of the adjacent entities and the corresponding relations in the $(l-1)^{th}$ layer of the entity-aggregating R-GCN. $c_{o,r}$ represents the size of $\mathcal{E}_o^r$. Moreover, $f(\cdot)$ represents the adopted activation function ($RReLU$). $\mathrm{W}_{\mathrm{r}}^{(l-1)}$ indicates the edge-specific parameters used to aggregate the structural features according to different relations. $\mathrm{W}_0^{(l-1)}$ indicates the parameters used to aggregate the self-loop features of all the entities. Finally, the entity-aggregating R-GCN of the t^{th} historical subgraph can be formally represented as:

$$\mathbf{E}^{\mathrm{t}}_{\mathrm{Agg}} = \mathrm{EAR\_GCN}(\mathbf{E}_{\mathrm{t}-1}, \mathbf{R}_{\mathrm{t}}) \tag{5}$$

where $\mathbf{E}^{\mathrm{t}}_{\mathrm{Agg}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$ is the output of the entity-aggregating R-GCN at the t^{th} timestamp. $\mathbf{E}_{\mathrm{t}-1} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$ is the output of the R-GRU at the $(t-1)^{th}$ timestamp. $\mathbf{R}_{\mathrm{t}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$ is the relation embeddings obtained from the RAM at the t^{th} timestamp. Next, similar to the RAM, we again use an R-GRU to model the evolution of the entity embeddings over time in different historical subgraphs within the EAM. As shown in Figure 2(c), we pass the output of the R-GRU at the previous historical timestamp and the output of the entity-aggregating R-GCN at the next historical timestamp to the current R-GRU:

$$\mathbf{E}_{\mathrm{t}} = \mathrm{R\_GRU}(\mathbf{E}^{\mathrm{t}}_{\mathrm{Agg}}, \mathbf{E}_{\mathrm{t}-1}) \tag{6}$$

where $\mathbf{E}_{\mathrm{t}} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$ is the final entity embeddings of the original subgraph $\mathcal{G}_t$ at the t^{th} historical timestamp.

E. Twin-Interact Module (TIM)

As shown in the leftmost part of Figure 2, we generate hyperrelation subgraphs according to the relative positional associations between the relations and entities in the original subgraphs. Following MaKEr [28], we use four hyperrelations $\{\mathrm{o\text{-}s,s\text{-}o,o\text{-}o,s\text{-}s}\}$ to model the positional association constraints between the relations and entities.

In TKGs, when the interactions between relations and entities are modeled, the sequential evolution between historical subgraphs must be considered. As shown in Figure 2(b), the TIM actually builds evolutionary communication channels between the RAM and the EAM. In particular, the entity embeddings from the EAM of the $(t-1)^{th}$ timestamp participate in the mean pooling operation to update the relation embeddings; then, following previous work [8], [12], we preserve the distant features by concatenating the relation embeddings of the first historical timestamp:

$$\mathbf{R}^{\mathrm{t}}_{\mathrm{Mean}} = [\mathbf{R}_0; \mathit{MP}(\mathbf{E}_{\mathrm{t}-1}, E^{\mathrm{t}}_{\mathrm{r}})] \tag{7}$$

where $\mathbf{R}^{\mathrm{t}}_{\mathrm{Mean}} \in \mathbb{R}^{2\mathrm{M}\times2\mathrm{d}}$, $\mathbf{R}_0 \in \mathbb{R}^{2\mathrm{M}\times\mathrm{d}}$, and $\mathbf{E}_{\mathrm{t}-1} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$. *MP* indicates the mean pooling operation. E^t_r indicates the entities immediately connected to specific relations $\{r\}$ regardless of the in-degree or out-degree edges that are present at the t^{th} timestamp. This step models the general association constraints between the relations and entities. Then, we utilize LSTM to model the evolution of these interactions over time:

$$\mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}}, \mathbf{C}_{\mathrm{t}} = \mathrm{LSTM}(\mathbf{R}^{\mathrm{t}}_{\mathrm{Mean}}, (\mathbf{R}_{\mathrm{t}-1}, \mathbf{C}_{\mathrm{t}-1})) \tag{8}$$

where $\mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}} \in \mathbb{R}^{2\mathrm{M}\times\mathrm{d}}$ is regarded as the relation embedding input of the relation-aggregating R-GCN at the t^{th} timestamp. $\mathbf{R}_{\mathrm{t}-1} \in \mathbb{R}^{2\mathrm{M}\times\mathrm{d}}$ is the relation embedding output of the RAM for the previous timestamp. $\mathbf{C}_{\mathrm{t}} \in \mathbb{R}^{2\mathrm{M}\times2\mathrm{d}}$ and $\mathbf{C}_{\mathrm{t}-1} \in \mathbb{R}^{2\mathrm{M}\times2\mathrm{d}}$ are the temporary iterative vectors of the LSTM sequence modeling process, and we set $\mathbf{C}_0 = \mathbf{R}^0_{\mathrm{Mean}}$ at the first historical timestamp. Then, we further update the hyperrelation embeddings at the t^{th} historical timestamp according to the updated relation embeddings $\mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}}$ performing hyper mean pooling in the hyperrelation subgraphs:

$$\mathbf{HR}^{\mathrm{t}}_{\mathrm{Mean}} = [\mathbf{HR}_0; \mathrm{HMP}(\mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}}, R^{\mathrm{t}}_{\mathrm{hr}})] \tag{9}$$

where $\mathbf{HR}^{\mathrm{t}}_{\mathrm{Mean}} \in \mathbb{R}^{2\mathrm{H}\times2\mathrm{d}}$. $\mathbf{HR}_0 \in \mathbb{R}^{2\mathrm{H}\times\mathrm{d}}$ denotes the initialization embeddings of the hyperrelations. HMP indicates the hyper mean pooling operation. R^t_{hr} indicates the relations immediately connected to specific hyperrelations $\{hr\}$ regardless of the in-degree or out-degree edges that are present at the t^{th} timestamp. Thus, we embed the related relation and entity features into the hyperrelations that express the positional association constraints. Next, we model the evolutionary patterns of the positional association constraints over time between the original subgraph sequence and the hyperrelation subgraph sequence via hyper LSTM:

$$\mathbf{HR}_{\mathrm{t}}, \mathbf{HC}_{\mathrm{t}} = \mathrm{H\_LSTM}(\mathbf{HR}^{\mathrm{t}}_{\mathrm{Mean}}, (\mathbf{HR}_{\mathrm{t}-1}, \mathbf{HC}_{\mathrm{t}-1})) \tag{10}$$

where $\mathbf{HR}_{\mathrm{t}} \in \mathbb{R}^{2\mathrm{H}\times\mathrm{d}}$ and $\mathbf{HR}_{\mathrm{t}-1} \in \mathbb{R}^{2\mathrm{H}\times\mathrm{d}}$ are the hyperrelation embeddings at the t^{th} and $(t-1)^{th}$ historical timestamps,

respectively. $\mathbf{HC}_{\mathrm{t}} \in \mathbb{R}^{2\mathrm{H}\times 2\mathrm{d}}$ and $\mathbf{HC}_{\mathrm{t-1}} \in \mathbb{R}^{2\mathrm{H}\times 2\mathrm{d}}$ are the temporary iterative vectors of the hyper LSTM sequence modeling process, and we initialize $\mathbf{HC}_0 = \mathbf{HR}^0_{\mathrm{Mean}}$ at the first historical timestamp.

In general, at a particular historical timestamp t, as shown in Figure 2(b), the TIM generates relation embeddings $\mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}}$ and hyperrelation embeddings $\mathbf{HR}_{\mathrm{t}}$ that both contain the positional association constraints between the relations and entities at the t^{th} timestamp according to the output entity embeddings $\mathbf{E}_{\mathrm{t-1}}$ obtained from the EAM at the $(t-1)^{th}$ timestamp. Then, the RAM generates relation embeddings $\mathbf{R}_{\mathrm{t}}$ that aggregate the complete adjacent information at the t^{th} timestamp according to the obtained $\mathbf{R}^{\mathrm{t}}_{\mathrm{Lstm}}$ and $\mathbf{HR}_{\mathrm{t}}$. Finally, the EAM obtains the updated entity embeddings $\mathbf{E}_{\mathrm{t}}$ at the t^{th} timestamp according to the relation embeddings $\mathbf{R}_{\mathrm{t}}$ from the RAM and the entity embeddings $\mathbf{E}_{\mathrm{t-1}}$ of the $(t-1)^{th}$ previous timestamp.

F. Time-Variability Training Strategy

The time-variability problem arises because different historical timestamps play different roles in future event forecasting (i.e., TKG extrapolation). To overcome the time-variability challenge, following CEN [7], we comprehensively consider the entity and relation representations over different historical timestamps.

For the entity forecasting task $(s, r, ?, t+1)$ and the relation forecasting task $(s, ?, o, t+1)$ at the $(t+1)^{th}$ future timestamp, we obtain the entity embedding sequence $\{\mathbf{E}_{\mathrm{t-k+1}}, \cdots, \mathbf{E}_{\mathrm{t-1}}, \mathbf{E}_{\mathrm{t}}\}$ and the relation embedding sequence $\{\mathbf{R}_{\mathrm{t-k+1}}, \cdots, \mathbf{R}_{\mathrm{t-1}}, \mathbf{R}_{\mathrm{t}}\}$ for all the subgraphs in the time-variability k-length history. We adopt Conv-TransE [20] as a component unit of the time-variability E-decoder and the time-variability R-decoder. Specifically, the entity and relation decoding processes based on the embeddings $\{\mathbf{E}_{\mathrm{t}}, \mathbf{R}_{\mathrm{t}}\}$ at a specific historical timestamp t can be represented as:

$$\mathbf{p}^{\mathrm{e}}_{\mathrm{t}}(\mathrm{o}|\mathrm{s}, \mathrm{r}, \mathbf{E}_{\mathrm{t}}, \mathbf{R}_{\mathrm{t}}) = f(\mathrm{Conv\_TransE}(\mathrm{s_t}, \mathrm{r_t}) \cdot \mathbf{E}_{\mathrm{t}}) \tag{11}$$

$$\mathbf{p}^{\mathrm{r}}_{\mathrm{t}}(\mathrm{r}|\mathrm{s}, \mathrm{o}, \mathbf{E}_{\mathrm{t}}, \mathbf{R}_{\mathrm{t}}) = f(\mathrm{Conv\_TransE}(\mathrm{s_t}, \mathrm{o_t}) \cdot \mathbf{R}_{\mathrm{t}}) \tag{12}$$

where $\mathrm{s_t} \in \mathbb{R}^{\mathrm{d}}$, $\mathrm{o_t} \in \mathbb{R}^{\mathrm{d}}$, and $\mathrm{r_t} \in \mathbb{R}^{\mathrm{d}}$ are the embeddings of the entities s, o and the relation r of a certain query at the t^{th} historical timestamp. $f(\cdot)$ denotes the softmax function. $\mathbf{p}^{\mathrm{e}}_{\mathrm{t}}$ (i.e., $\mathbf{p}^{\mathrm{e}}_{\mathrm{t}}(\mathrm{o}|\mathrm{s}, \mathrm{r}, \mathbf{E}_{\mathrm{t}}, \mathbf{R}_{\mathrm{t}})$) and $\mathbf{p}^{\mathrm{r}}_{\mathrm{t}}$ (i.e., $\mathbf{p}^{\mathrm{r}}_{\mathrm{t}}(\mathrm{r}|\mathrm{s}, \mathrm{o}, \mathbf{E}_{\mathrm{t}}, \mathbf{R}_{\mathrm{t}})$) are an N-dimensional vector and an M-dimensional vector, respectively; the dimensionality of each denotes the probability score of forecasting the corresponding entity or relation as a missing object or relation.

We train the model with each timestamp as a batch. For an event extrapolated to a certain future timestamp $(t+1)$, the training process is regarded as an N-label and an M-label classification problem for the entity forecasting task and relation forecasting task, respectively. We use the cross-entropy loss function; thus, the loss functions of these two tasks can be expressed as:

$$\mathcal{L}^{t+1}_e = -\sum_{(s,r,o)\in\mathcal{V}_{t+1}} \sum_{i\in\mathcal{E}} o^{t+1}_i \ln\left(\mathbf{p}^{\mathrm{e}}_{\mathrm{t-k+1}} + \cdots + \mathbf{p}^{\mathrm{e}}_{\mathrm{t-1}} + \mathbf{p}^{\mathrm{e}}_{\mathrm{t}}\right) \tag{13}$$

$$\mathcal{L}^{t+1}_r = -\sum_{(s,r,o)\in\mathcal{V}_{t+1}} \sum_{j\in\mathcal{R}} r^{t+1}_j \ln\left(\mathbf{p}^{\mathrm{r}}_{\mathrm{t-k+1}} + \cdots + \mathbf{p}^{\mathrm{r}}_{\mathrm{t-1}} + \mathbf{p}^{\mathrm{r}}_{\mathrm{t}}\right) \tag{14}$$

where $\mathcal{V}_{t+1}$ represents all the facts at the $(t+1)^{th}$ future timestamp. o^{t+1}_i and r^{t+1}_j denote the i^{th} ground-truth object entity and the j^{th} ground-truth relation, respectively, in the $(t+1)^{th}$ temporal subgraph $\mathcal{G}_{t+1}$. Note that the time-variability strategy requires us to consider the newly emerging historical facts at the $(t+1)^{th}$ timestamp when training the facts at the $(t+2)^{th}$ future timestamp to address the time-variability problem.

To train these two temporal tasks simultaneously, we set the learning weights for the two losses. Therefore, for the training process at the $(t+1)^{th}$ future timestamp, the final loss can be represented as $\mathcal{L}_{t+1} = \lambda\mathcal{L}^{t+1}_e + (1-\lambda)\mathcal{L}^{t+1}_r$. λ is the learning weight of entity forecasting.

G. Computational Complexity Analysis

In this section, we analyze the computational complexity of our proposed RETIA model. The time complexity of generating the hyperrelation subgraphs is $O(V)$, where V is the maximum number of facts contained in a separated historical timestamp. For the RAM and the EAM, the time complexities of relation aggregation and entity aggregation are $O(kM)$ and $O(kN)$, respectively. For the TIM, the computational complexities of mean pooling and LSTM are $O(kMP)$ and $O(kd^2)$, respectively, where P is the maximum number of entities adjacent to a relation at a specific historical timestamp. Similarly, the computational complexities of hyper mean pooling and the hyper LSTM are $O(kHP^{'})$ and $O(kd^2)$, respectively, where $P^{'}$ is the maximum number of relations associated with a hyperrelation in the corresponding hyperrelation subgraph. Thus, the computational complexity of the RETIA model is $O(k(M+N+MP+HP^{'}+d^2)+V)$.

IV. Experiments

In this section, we evaluate the performance of our proposed RETIA model on five popular TKG datasets.

A. Experimental Setup

1) Datasets: We use five public TKG datasets to demonstrate the effectiveness of our proposed method. They are YAGO [29], WIKI [30], ICEWS14 [25], ICEWS05-15 [25], and ICEWS18 [3]. The ICEWS series, which includes the ICEWS14, ICEWS05-15, and ICEWS18 datasets, is from the Integrated Crisis Early Warning System [31]. The YAGO and WIKI datasets are supplemented with time information based on the traditional static KGs YAGO3 and Wikipedia. Following extensive previous work [3], [7], [8], [12], we split the datasets into training, validation, and test sets using proportions of 80%/10%/10%. We detail information about the adopted datasets in Table V.

2) Baseline Methods: We compare the performance of our proposed RETIA model with that of multiple static and dynamic modeling methods (including interpolation and extrapolation). Note that the static methods are trained without

the time dimension and the interpolation methods are trained with both historical and future data; thus, they are not good at future event forecasting when provided with only historical information. The static modeling methods include DistMult [17], ConvE [19], ComplEx [18], Conv-TransE [20], RotatE [21], and R-GCN [9]. Some modeling methods that operate under the interpolation setting include TTransE [23], HyTE [24], and TA-DistMult [25]. We focus on comparisons with the modeling methods under the extrapolation setting, as these approaches are designed to address the task of forecasting future events (i.e., TKG extrapolation). These methods include RE-NET [3], CyGNet [4], xERTE [5], CluSTeR [26], RE-GCN [8], TITer [6], TLogic [27], CEN [7], and TiRGN [12]. Detailed descriptions of the abovementioned modeling methods are presented in Section II.

3) Evaluation Protocol: We evaluate the effectiveness of our proposed RETIA model through a link prediction task. Four evaluation metrics are widely adopted for such tasks. They are the mean reciprocal rank (MRR), hits at 1 (Hits@1), hits at 3 (Hits@3), and hits at 10 (Hits@10), which all reflect the rankings of ground-truth missing entities or relations in the obtained extrapolation results. Following RE-GCN [8], for entity extrapolation, we report the mean results of the subject forecasting and object forecasting tasks; for relation extrapolation, we report only the results of the MRR metric due to the small number of relations; and for the YAGO and WIKI datasets, we report only the results of the MRR, Hits@3, and Hits@10 metrics.

Many previous studies [5]–[8], [12], [26], [27], [32] have proven that the traditional static evaluation metrics under the filtered setting are not suitable for scenarios with temporal dynamics; thus, some such works have adopted the dynamic time-aware filtered setting. However, regardless of whether a static filtered setting or a time-aware filtered setting is used, the approaches for handling one-to-many or many-to-many fact forecasting are crude. Taking an object forecasting task $(s, r, ?, t+1)$ with the ground-truth entity o_4 as an example, when there are multiple valid facts with the same subject s and relation r at the future timestamp $t+1$, then the objects $\{o_0, o_1, o_2, \cdots\}$ of these facts are conflicting entities for o_4. The time-aware filtered setting simply filters all the conflicting candidate entities except for the ground-truth missing entity o_4 of the specific task and thus tends to obtain better results. Without loss of generality, we report the results obtained under the raw setting instead.

4) Implementation Details: We implemented our RETIA model using PyTorch. Then, we trained the model on a Tesla V100 GPU. Considering time variability, we conducted model learning through two processes: general training and online continuous training. During the general training process, we utilized the training set and configured the parameters according to the model performance achieved on the entire validation set. During online continuous training, we utilized the facts acquired at the newly emerging historical timestamps and updated the obtained parameters according to the model performance achieved for the next (i.e., future) validation or test timestamp. We set the training batch size as the size of each timestamp to adapt to the abovementioned training process. We chose the historical length k from $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ according to the model performance on the validation set in general training. Finally, we set k to 3 for the YAGO and WIKI datasets, 4 for the ICEWS18 dataset, and 9 for the ICEWS14 and ICEWS05-15 datasets. For the relation-aggregating R-GCN and the entity-aggregating R-GCN, we set the number of layers to 2 and the dropout rate of each layer to 0.2. For each Conv-TransE unit in the time-variability E-decoder and R-decoder, we set the kernel size to 3×2, the number of kernels to 50, and the dropout rate to 0.2. We set the learning weight of the entity forecasting task λ to 0.7 for all the datasets. Following RE-GCN [8] and TiRGN [12], we also added static graph constraints when dealing with the ICEWS14, ICEWS18, and ICEWS05-15 datasets. The Adam optimizer was used for parameter learning, and the learning rates for general training and online continuous training were both set to 0.001. For the static modeling baseline methods, we removed the time dimension from all the TKG datasets. The embedding dimensionality d was set to 200, which was the same experimental setting as that in RE-GCN [8]. Some of the baseline results were also adopted from [8].

For some extrapolation baseline methods, including xERTE [5], TITer [6], TLogic [27], CEN [7], and TiRGN [12], their open source codes and default parameters were used to obtain results under the raw setting. For CluSTeR [26], which does not possess open source codes, we reported the corresponding results obtained in the original paper. TLogic [27] can process only the datasets of the ICEWS series because other datasets do not have content references for entities and relations in the real world. We reorganized the input format of the TLogic model for the ICEWS14 and ICEWS05-15 datasets because we kept the dataset-splitting strategy of TLogic consistent with that of the other baseline models. For CEN [7], we reported the results obtained under the online setting, addressing the time-variability problem.

B. Results of TKG Extrapolation

1) Entity Forecasting: The results of the entity forecasting task are shown in Table III and Table IV. The best results are in bold, and the second-best results are underlined. The proposed RETIA model performs much better than the static modeling methods because it ignores the time dimension of the facts in the TKGs, and conflicting facts at different historical timestamps are compressed into a static KG, which makes it almost impossible to obtain accurate representations of the entities and relations. The performance achieved by the dynamic modeling methods under the interpolation setting is generally better than that of the static methods because the dynamic modeling methods take time information into consideration. Nevertheless, some of the interpolation methods, specifically TTransE [23] and HyTE [24], perform poorly because these two models focus on only embedding the timestamps into the low-dimensional space while ignoring the evolutionary patterns passed between historical subgraphs. We focus on

TABLE III
PERFORMANCE ACHIEVED IN TERMS OF ENTITY FORECASTING ON THE ICEWS14, ICEWS18, AND ICEWS05-15 DATASETS WITH RAW METRICS.

Method	ICEWS14				ICEWS05-15				ICEWS18			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
DistMult	20.32	6.13	27.59	46.61	19.91	5.63	27.22	47.33	13.86	5.61	15.22	31.26
ConvE	30.30	21.30	34.42	47.89	31.40	21.56	35.70	50.96	22.81	13.63	25.83	41.43
ComplEx	22.61	9.88	28.93	47.57	20.26	6.66	26.43	47.31	15.45	8.04	17.19	30.73
Conv-TransE	31.50	22.46	34.98	50.03	30.28	20.79	33.80	49.95	23.22	14.26	26.13	41.34
RotatE	25.71	16.41	29.01	45.16	19.01	10.42	21.35	36.92	14.53	6.47	15.78	31.86
R-GCN	28.03	19.42	31.95	44.83	27.13	18.83	30.41	43.16	15.05	8.13	16.49	29.00
TTransE	12.86	3.14	15.72	33.65	16.53	5.51	20.77	39.26	8.44	1.85	8.95	22.38
HyTE	16.78	2.13	24.84	43.94	16.05	6.53	20.20	34.72	7.41	3.10	7.33	16.01
TA-DistMult	26.22	16.83	29.72	45.23	27.51	17.57	31.46	47.32	16.42	8.60	18.13	32.51
RE-NET	35.77	25.99	40.10	54.87	36.86	26.24	41.85	57.60	26.17	16.43	29.89	44.37
CyGNet	34.68	25.35	38.88	53.16	35.46	25.44	40.20	54.47	24.98	15.54	28.58	43.54
xERTE	32.23	24.29	36.41	48.76	38.07	28.45	43.92	57.62	27.98	19.26	32.43	46.00
CluSTeR	**46.00**	33.80	-	**71.20**	44.60	34.90	-	63.00	32.30	20.60	-	55.90
RE-GCN	41.50	30.86	46.60	62.47	46.41	35.17	52.76	67.64	30.55	20.00	34.73	51.46
TITer	40.90	31.77	45.84	57.67	46.62	36.46	52.29	65.23	28.44	20.06	32.07	44.33
TLogic	41.80	31.93	47.23	60.53	45.99	34.49	52.89	67.39	28.41	18.74	32.71	47.97
CEN	41.64	31.22	46.55	61.59	49.57	37.86	56.42	71.32	29.70	19.38	33.91	49.90
TiRGN	43.88	33.12	49.48	64.98	48.72	37.17	55.48	70.53	32.06	21.08	36.75	53.62
RETIA	45.29	**34.60**	**50.88**	66.06	**52.17**	**40.21**	**59.42**	**73.98**	**34.16**	**22.97**	**39.27**	**55.96**

TABLE IV
PERFORMANCE ACHIEVED IN TERMS OF ENTITY FORECASTING ON THE YAGO AND WIKI DATASETS WITH RAW METRICS.

Method	YAGO			WIKI		
	MRR	Hits@3	Hits@10	MRR	Hits@3	Hits@10
DistMult	44.05	49.70	59.94	27.96	32.45	39.51
ConvE	41.22	47.03	59.90	26.03	30.51	39.18
ComplEx	44.09	49.57	59.64	27.69	31.99	38.61
Conv-TransE	46.67	52.22	62.52	30.89	34.30	41.45
RotatE	42.08	46.77	59.39	26.08	31.63	38.51
R-GCN	20.25	24.01	37.30	13.96	15.75	22.05
TTransE	26.10	36.28	47.73	20.66	23.88	33.04
HyTE	14.42	39.73	46.98	25.40	29.16	37.54
TA-DistMult	44.98	50.64	61.11	26.44	31.36	38.97
RE-NET	46.81	52.71	61.93	30.87	33.55	41.27
CyGNet	46.72	52.48	61.52	30.77	33.83	41.19
xERTE	64.29	74.50	87.38	52.85	60.96	71.89
RE-GCN	63.07	71.17	82.07	51.53	58.29	69.53
TITer	64.97	74.80	87.44	57.36	63.80	72.52
CEN	63.39	71.68	83.16	51.98	58.96	70.61
TiRGN	64.71	74.17	87.01	53.20	60.78	72.07
RETIA	**67.58**	**78.42**	**88.06**	**70.11**	**78.30**	**84.77**

TABLE V
DETAILS OF THE TKG DATASETS.

#Datasets	ICEWS14	ICEWS05-15	ICEWS18	YAGO	WIKI
#Entities	6,869	10,094	23,033	10,623	12,554
#Relations	230	251	256	10	24
#Training	74,845	368,868	373,018	161,540	539,286
#Validation	8,514	46,302	45,995	19,523	67,538
#Test	7,371	46,159	49,545	20,026	63,110
#Granularity	24 hours	24 hours	24 hours	1 year	1 year

comparisons with the dynamic modeling methods under the extrapolation setting.

Regarding the extrapolation-based modeling methods, RETIA outperforms the RE-NET [3] and CyGNet [4] models because these two methods do not aggregate the neighborhood information of entities and thus fail to model the internal structures of the historical subgraphs. The xERTE [5], CluSTeR [26], and CEN [7] approaches model the temporal evolution of the entity embeddings between subgraphs while aggregating the adjacent information of entities, but they do not model the relation embeddings. The reinforcement learning-based methods, including CluSTeR [26] and TITer [6], extract a limited number of entities from the entity set to make up the candidate set; thus, the ranking cardinality is much lower than that of other methods, and better results can be easily obtained. This is why RETIA is slightly weaker than CluSTeR on the ICEWS14 dataset. Rule-based methods, including TLogic [27], conduct more targeted modeling. However, TLogic also ignores structural aggregation information during the evolution of historical subgraphs; therefore, RETIA performs better than TLogic on the ICEWS series datasets, which is the dataset that TLogic performs well on. Moreover, RETIA performs better than RE-GCN [8] and TiRGN [12] because these methods aggregate only the adjacent entity information of the relation embeddings, as mentioned above, which leads to incomplete relation representations due to the "message islands" problem and further affects the score decoding process of entity forecasting. On the other hand, all of these extrapolation methods ignore the positional association constraints between the embeddings of the entities and relations, thus resulting in the fuzzy aggregation of the structural information in each subgraph. Therefore, our proposed RETIA model outperforms the existing entity forecasting baselines on almost all the datasets and evaluation metrics.

2) Relation Forecasting: The results of relation forecasting are shown in Table VII. Similarly, the best results are bolded, and the second-best results are underlined. Since little work has been performed on relation forecasting, we compare RETIA with only the representative models. As mentioned above, methods that do not model relation embeddings cannot forecast future relations. In particular, the compared static modeling methods include ConvE [19] and Conv-TransE [20]. The

TABLE VI
ABLATION STUDY RESULTS OBTAINED ON ALL THE DATASETS.

Module	YAGO		WIKI		ICEWS14		ICEWS05-15		ICEWS18	
	Entity	Relation	Entity	Relation	Entity	Relation	Entity	Relation	Entity	Relation
wo. Entity Aggregation Module (EAM)	2.34	57.34	0.61	36.21	0.13	13.72	11.31	19.94	0.08	14.66
wo. Relation Aggregation Module (RAM)	61.30	15.94	45.78	12.39	29.95	3.63	30.54	3.90	15.66	2.49
RETIA	**67.58**	**98.91**	**70.11**	**98.21**	**45.29**	**42.05**	**52.17**	**43.19**	**34.16**	**41.78**

TABLE VII
PERFORMANCE ACHIEVED IN TERMS OF RELATION FORECASTING ON ALL THE DATASETS WITH RAW METRICS.

Method	YAGO	WIKI	ICEWS14	ICEWS05-15	ICEWS18
ConvE	91.33	78.23	38.80	37.89	37.73
Conv-TransE	90.98	86.64	38.40	38.26	38.00
RGCRN	90.18	88.88	38.04	38.37	37.14
RE-GCN	97.74	97.92	41.06	40.63	40.53
TiRGN	93.58	98.12	**42.57**	42.12	**41.78**
RETIA	**98.91**	**98.21**	42.05	**43.19**	**41.78**

TABLE VIII
RUN-TIME COMPARISONS OF THE EXTRAPOLATION METHODS ON ALL THE DATASETS (D: DAYS; H: HOURS; MIN: MINUTES; S: SECONDS).

Method	ICEWS14	ICEWS05-15	ICEWS18	YAGO	WIKI
RE-NET	3.07 min	19.88 min	23.15 min	8.23 min	26.07 min
CyGNet	58.62 s	20.34 min	4.38 min	21.40 s	1.06 min
xERTE	14.81 min	3.67 h	2.62 h	29.22 min	2.58 h
RE-GCN	3.33 s	46.51 s	6.86 s	0.29 s	0.53 s
TITer	2.93 min	22.66 min	2.26 d	1.62 h	22.35 min
TLogic	37.91 min	20.63 h	1.37 d	-	-
CEN	5.42 s	1.73 min	12.08 s	1.24 s	4.38 s
TiRGN	17.36 min	9.46 h	2.11 h	18.90 min	39.23 min
RETIA	**8.46 min**	**3.93 h**	**28.71 min**	**6.40 s**	**18.06 s**

compared dynamic modeling methods include RGCRN [33], RE-GCN [8], and TiRGN [12]. Among them, RE-NET [3] extends a heterogeneous graph model (GCRN) [33] to an RGCRN by replacing the GCN with an R-GCN.

The RETIA model performs significantly better than the static approaches because this kind of method does not model time information. In the comparison with the dynamic modeling methods, the superior performance of RETIA on almost all the datasets demonstrates that aggregating adjacent entity and relation features simultaneously for relation embeddings helps obtain accurate representations. We note that RETIA does not work as well on the ICEWS14 dataset as it does on TiRGN because TiRGN uses historical one-hop repetitive relations to limit the scope of the candidate set. Specifically, it simply kicks relations that do not exist historically out of the candidate set, which occasionally yields better performance on certain datasets. On the other hand, in the comparison with the baseline models, the relation forecasting improvement achieved by RETIA is much smaller than its entity forecasting improvement because we model the forecasting process as a multilabel classification task, and the number of relations in the dataset is much smaller than that of entities, making relation forecasting significantly easier than entity forecasting.

3) Comparison on Prediction Time: We compare the run time of RETIA with that of the important extrapolation baseline methods on all the datasets. Note that the time consumption results for RE-NET were taken from [8], since we were unable to run their open-source codes due to program crashes. As shown in Table VIII, we underline the results for which the time efficiency is better than that of RETIA. RETIA spends more run time than RE-NET and CyGNet on the ICEWS series datasets, but consumes far less time on the YAGO and WIKI datasets. Compared to RE-GCN and CEN, RETIA consumes more time on all the datasets due to its higher model complexity. xERTE and TLogic are more time-consuming because they both use sampling mechanisms to generate a large number of query-related subgraphs. The time consumed by TITer depends on the specific reinforcement learning strategy. TiRGN is less time-efficient because it takes considerable time to extract historical repetitive entities and relations from the global historical space. In summary, RETIA guarantees a limited increase in time efficiency while providing excellent extrapolation performance.

C. Ablation Study

In this section, we conduct an ablation test to study the roles of the RAM and the EAM in the model. The study of the TIM is separately provided in Section IV-D.

As Table VI shows, the best results are in bold, and the second-best results are underlined. We conduct an ablation study on all the experimental datasets and report the results in terms of the most representative MRR metric. The expressions "wo. Entity Aggregation Module (EAM)" and "wo. Relation Aggregation Module (RAM)" represent the removal of the EAM or RAM from the complete RETIA model, and "**RETIA**" indicates the complete proposed model. We record the entity forecasting and relation forecasting results under the "Entity" and "Relation" columns, respectively.

In the experiment, we randomly initialize the entities and relations and then remove the EAM by keeping the initialized entity embeddings unchanged and updating the relation embeddings normally. On the other hand, we randomly initialize the entities, relations, and hyperrelations and then remove the RAM by keeping the initialized relation embeddings unchanged and iteratively updating the entity embeddings. It is observed that the absence of the EAM or the RAM can significantly degrade the performance of the model, proving that accurate entity and relation representations are essential for both entity and relation forecasting. Without the RAM, a significant degradation can be observed in the model's relation forecasting performance; in the absence of the EAM,

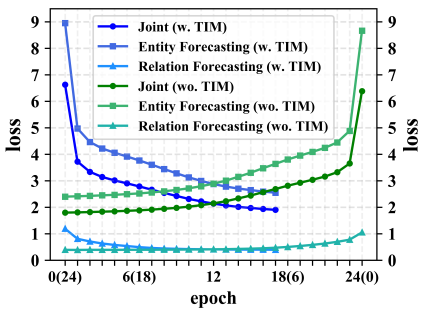


Fig. 3. Study on the role of the TIM in the general training process with the YAGO dataset.

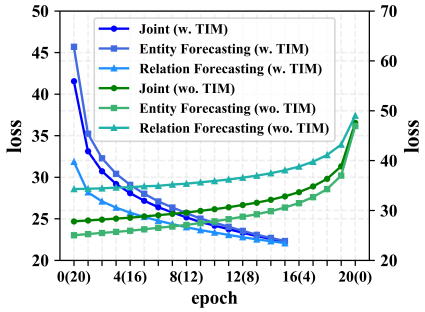


Fig. 4. Study on the role of the TIM in the general training process with the ICEWS14 dataset.

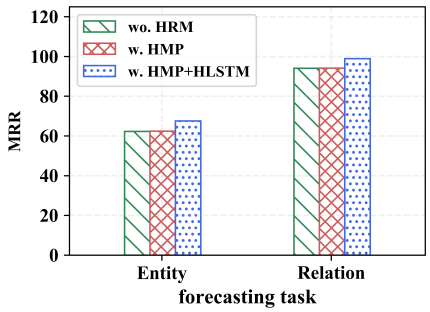


Fig. 5. Study on the positional association constraints with the YAGO dataset.

a significant decrease can be observed in the entity prediction performance of the model. This proves that the expressions of entity embeddings and relation embeddings play major roles in entity and relation forecasting tasks, respectively. In particular, the loss of entity aggregation modeling is catastrophic for the entity forecasting task, with the MRR metric reaching only 0.08 even on the most challenging ICEWS18 dataset (which contains the largest number of entities). On the other hand, relation aggregation modeling can significantly affect the relation forecasting performance of the model and reduce the accuracy of entity forecasting to a certain extent. When the complete neighborhood structure information is aggregated for both the entities and the relations and the evolutionary association constraints are simultaneously modeled via the TIM, RETIA realizes a greater level of improvement.

D. On the Twin-Interact Module

In this section, we investigate the overall contribution of the TIM to model performance and further study the role of evolutionary modeling of the positional association constraints on the YAGO and ICEWS14 datasets.

1) On the Association Constraints: On the one hand, we randomly initialize the entity and relation embeddings and send them to the EAM to iteratively update the entity embeddings while keeping the relation embeddings unchanged. On the other hand, we iteratively update the relation embeddings by sending the randomly initialized relation and hyperrelation embeddings to the RAM while keeping the hyperrelation embeddings unchanged. Note that the relation embeddings in the EAM and RAM are two different and inconsistent individuals. As Figure 3 and Figure 4 show, we report the results of the general training process of the model conducted with and without the TIM based on the YAGO and ICEWS14 datasets. "w. TIM" indicates the model with the TIM, and the losses of different epochs are plotted from left to right in blue. However, "wo. TIM" indicates that the TIM is removed from the model, and the losses are plotted from right to left in green. "Joint" indicates the joint losses with the task weight λ that are actually backpropagated during training.

During the general training process, when the model performance in the current epoch is lower than that in the historical best epoch for five consecutive iterations, we stop the model training procedure. When the association constraints between entities and relations are considered, the training process of the model converges more easily. During the training process, on both datasets, the losses of the modeling process including the association constraint are reduced to a low level in a relatively short period of time. For the YAGO dataset, the "wo. TIM" loss eventually drops to a level similar to that of "w. TIM"; however, for the ICEWS14 dataset, the "wo. TIM" model has difficulty converging due to the more complex data structure and smaller time granularity. On the other hand, as Table IX shows, we compare the forecasting performance of the model under two conditions on the two datasets. Note that the comparisons are the final results obtained after conducting online continuous training. If the association constraints between entities and relations (wo. TIM) are not modeled, the model not only does not converge easily in the general training process but also performs poorly on the test set. In summary, the learned embeddings tend to overfit the data because they ignore the association constraints.

TABLE IX
STUDY ON THE ROLE OF THE TIM IN THE FORECASTING PROCESS WITH THE YAGO AND ICEWS14 TEST SETS.

Module	YAGO				ICEWS14			
	Entity		Relation		Entity		Relation	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
wo. TIM	66.27	85.68	69.23	86.49	42.61	63.09	36.44	57.77
w. TIM	**67.58**	**88.06**	**98.91**	**99.93**	**45.29**	**66.06**	**42.05**	**73.65**

2) Capturing the Positional Association Constraints via Hyperrelations: The hyperrelations are used to model the positional association constraints between entities and relations. We further explore the ability of hyper mean pooling and the hyper LSTM to capture the positional association constraints by changing the hyperrelation embeddings that the TIM delivers to the RAM. As Figure 5 shows, "wo. HRM" indicates that the hyperrelations are not modeled, and we take the initialized embeddings as RAM inputs instead. "w. HMP" refers to using hyper mean pooling to aggregate the adjacent relation information of the hyperrelations. "w. HMP+HLSTM" indicates that the evolutionary dependencies between the hyperrelations of the subgraphs are modeled using the hyper LSTM based on hyper mean pooling.

For both the entity and relation forecasting tasks, the

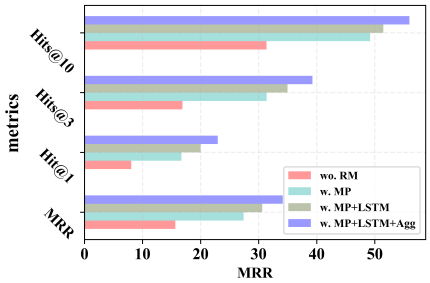


Fig. 6. Study on the role of relation modeling in entity forecasting with the ICEWS18 dataset.

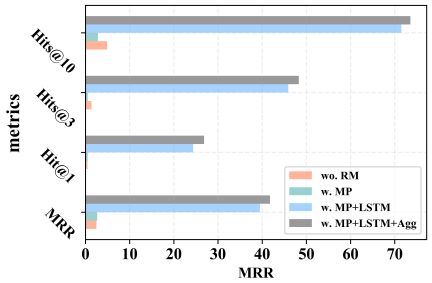


Fig. 7. Study on the role of relation modeling in relation forecasting with the ICEWS18 dataset.

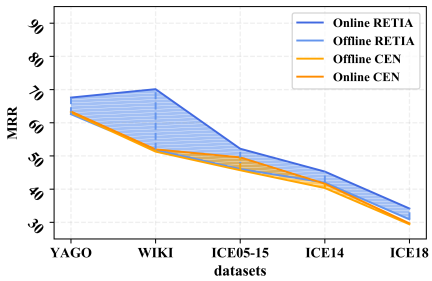


Fig. 8. Study on the time-variability training strategy in entity forecasting with all the datasets.

performance achieved by inputting the initial hyperrelation embeddings into the RAM for aggregation almost reaches that of the hyper mean pooling operation. On the one hand, this is because the hyperrelation generation algorithm already includes the positional association constraints between entities and relations, and the evolutionary modeling of relations captures the common association constraints. On the other hand, hyper mean pooling directly replaces the hyperrelation representations with the mean of the immediately adjacent relation embeddings, which affects the layer normalization process [34] of complex networks. When using the hyper LSTM to model the chronological evolution of the positional association constraints, the model makes further improvements in both the entity forecasting and the relation forecasting tasks. This proves that in the process of capturing the association constraints, the temporal dependencies between subgraphs play a more important role than the structural information inside the subgraphs.

E. On Relation Embeddings

In this section, we study the ability of the relation representation modeling approach of our proposed RETIA to overcome the "message islands" problem on the most representative ICEWS18 dataset.

As Figure 6 and Figure 7 show, we compare the effects of different degrees of relation modeling on entity and relation forecasting tasks, respectively. "wo. RM" indicates that the relations are not modeled, and the scores are calculated directly by inputting the initialized embeddings into the decoder. "w. MP" refers to utilizing mean pooling to aggregate the immediately adjacent entity features for relation embeddings. "w. MP+LSTM" refers to modeling the chronological evolution of the relations via LSTM based on the mean pooling operation. "w. MP+LSTM+Agg" indicates that based on the mean pooling and LSTM operations, we further aggregate the adjacent relation information for the modeled relations through the RAM instead of stopping the message-passing process for the relations at the level of the immediately adjacent entities. This step is the key to overcoming the "message islands" problem.

The initial relation embeddings obtained without any modeling can still achieve a certain level of entity forecasting performance, but it is fatal to relation forecasting, as it can make the model almost lose its forecasting ability. There is still a performance gap between the mean pooling operation and the simultaneous mean pooling and LSTM operations. As mentioned above, this is because using only the mean pooling operation can lead to an impairment in the layer normalization process conducted for the relation embeddings, thus having a greater impact on the relation forecasting task.The advanced baseline methods (RE-GCN and TiRGN) model the relations at the 3rd level (w. MP+LSTM), which leads to the "message islands" problem. Our proposed hyperrelation subgraph aggregation method can effectively further improve the achieved entity and relation forecasting performance. Because RETIA no longer rigidly adheres to the characteristics of the immediately adjacent entities, through message-passing architecture, it makes each relation cross the one-hop gap and passes the feature information out.

F. On the Time-variability Training Strategy

In this section, we investigate the influence of the time-variability training strategy on all the datasets.

Since the CEN model, which also considers the time-variability problem, does not model the relation representations, as shown in Figure 8, we comparatively report only the impact of the online continuous training process on the more representative entity forecasting task. Across all the datasets, RETIA achieves greater improvements than the advanced baseline method under the time-variability training strategy. This proves that reasonably modeling relations and considering the association constraints between entities and relations are more conducive to solving domain obstacles such as the time-variability problem.

V. Conclusion

In this paper, we propose RETIA to address the challenge of "message islands" in relation modeling and to capture the positional association constraints between relations and entities. RETIA evolutionally aggregates adjacent entity and relation features to produce relation embeddings on a twin hyperrelation subgraph sequence, thus spanning the message-passing gap. In addition, RETIA captures positional association constraints by modeling the structural information and chronological dependencies of hyperrelations. Extensive experiments demonstrate its significant improvements over the baseline models.

REFERENCES

[1] Salvatore Carta, Alessandro Sebastian Podda, Diego Reforgiato Recupero, and Maria Madalina Stanciu. Explainable ai for financial forecasting. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 51–69. Springer, 2021.

[2] Yuying Li. Corporate financial fraud identification and crisis forewarning based on the partial least squares method. *International Journal of Engineering Intelligent Systems*, 30(3), 2022.

[3] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 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, 2020.

[4] Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4732–4740, 2021.

[5] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*, 2020.

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

[7] Zixuan Li, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. Complex evolutional pattern learning for temporal knowledge graph reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 290–296, 2022.

[8] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 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.

[9] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[10] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2019.

[11] Ren Li, Yanan Cao, Qiannan Zhu, Guanqun Bi, Fang Fang, Yi Liu, and Qian Li. How does knowledge graph embedding extrapolate to unseen data: a semantic evidence view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5781–5791, 2022.

[12] Yujia Li, Shiliang Sun, and Jing Zhao. Tirgn: Time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2152–2158. ijcai.org, 2022.

[13] Kyunghyun Cho, B van Merrienboer, Caglar Gulcehre, F Bougares, H Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[16] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[17] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, 2015.

[18] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[19] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[20] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067, 2019.

[21] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2018.

[22] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7346–7359, 2020.

[23] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. Towards time-aware knowledge graph completion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1715–1724, 2016.

[24] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2001–2011, 2018.

[25] Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias 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, 2018.

[26] Zixuan Li, Xiaolong Jin, Saiping Guan, Wei Li, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 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, 2021.

[27] Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4120–4127, 2022.

[28] Mingyang Chen, Wen Zhang, Zhen Yao, Xiangnan Chen, Mengxiao Ding, Fei Huang, and Huajun Chen. Meta-learning based knowledge extrapolation for knowledge graphs in the federated setting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 1966–1972. ijcai.org, 2022.

[29] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

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

[31] Elizabeth Boschee, Jennifer Lautenschlager, Sean OBrien, Steve Shellman, James Starz, and Michael Ward. Icews coded event data. *Harvard Dataverse*, 12, 2015.

[32] Yifu Gao, Linhui Feng, Zhigang Kan, Yi Han, Linbo Qiao, and Dongsheng Li. Modeling precursors for temporal knowledge graph reasoning via auto-encoder structure. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2044–2051. ijcai.org, 2022.

[33] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pages 362–373. Springer, 2018.

[34] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.