Learning Long- and Short-term Representations for Temporal Knowledge Graph Reasoning

Mengqi Zhang
School of Artificial Intelligence, University of Chinese Academy of Sciences
CRIPAC, MAIS, Institute of Automation, Chinese Academy of Sciences
Beijing, China
mengqi.zhang@cripac.ia.ac.cn

Yuwei Xia
School of Cyber Security, University of Chinese Academy of Sciences
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
xiayuwei@iie.ac.cn

Qiang Liu*
School of Artificial Intelligence, University of Chinese Academy of Sciences
CRIPAC, MAIS, Institute of Automation, Chinese Academy of Sciences
Beijing, China
qiang.liu@nlpr.ia.ac.cn

Shu Wu
School of Artificial Intelligence, University of Chinese Academy of Sciences
CRIPAC, MAIS, Institute of Automation, Chinese Academy of Sciences
Beijing, China
shu.wu@nlpr.ia.ac.cn

Liang Wang
School of Artificial Intelligence, University of Chinese Academy of Sciences
CRIPAC, MAIS, Institute of Automation, Chinese Academy of Sciences
Beijing, China
wangliang@nlpr.ia.ac.cn

ABSTRACT

Temporal Knowledge graph (TKG) reasoning aims to predict missing facts based on historical TKG data. Most of the existing methods are incapable of explicitly modeling the long-term time dependencies from history and neglect the adaptive integration of the long- and short-term information. To tackle these problems, we propose a novel method that utilizes a designed Hierarchical Relational Graph Neural Network to learn the Long- and Short-term representations for TKG reasoning, namely HGLS. Specifically, to explicitly associate entities in different timestamps, we first transform the TKG into a global graph. Based on the built graph, we design a Hierarchical Relational Graph Neural Network that executes in two levels: The sub-graph level is to capture the semantic dependencies within concurrent facts of each KG. And the global-graph level aims to model the temporal dependencies between entities. Furthermore, we design a module to extract the long- and short-term information from the output of these two levels. Finally, the long- and short-term representations are fused into a unified one by Gating Integration for entity prediction. Extensive experiments on four datasets demonstrate the effectiveness of HGLS.

*To whom correspondence should be addressed.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583242

CCS CONCEPTS

• **Computing methodologies** → **Temporal reasoning**.

KEYWORDS

temporal knowledge graph, graph neural network, long- and short-term information

ACM Reference Format:
Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. 2023. Learning Long- and Short-term Representations for Temporal Knowledge Graph Reasoning. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583242

1 INTRODUCTION

Temporal Knowledge Graph (TKG) is a type of dynamic multi-relational graph data used to record evolutionary events and knowledge in the real world. Most of TKG data is automatically identified and extracted from a variety of international news articles, such as ICEWS [1] and GDELT [15] data. Each fact in a TKG is represented by a quadruple (s, r, o, t), such as (*Obama, run for, president, 2012*). Reasoning over TKG has gained much attention in recent years due to its great practical value in event prediction [4], question answering [20], and other areas.

Reasoning over TKG primarily has two settings: interpolation and extrapolation [13]. Given a TKG with timestamps from t_0 to t_n, interpolation [7, 8, 31, 33, 35] mainly aims at inferring missing facts that occur at time t, where $t_0 < t < t_n$. Oppositely, extrapolation [9, 11, 13, 17, 26, 27, 41] attempts to predict facts that occur at time t with $t > t_n$. In this paper, we mainly focus on predicting facts in future timestamps (*i.e.*, extrapolation setting).

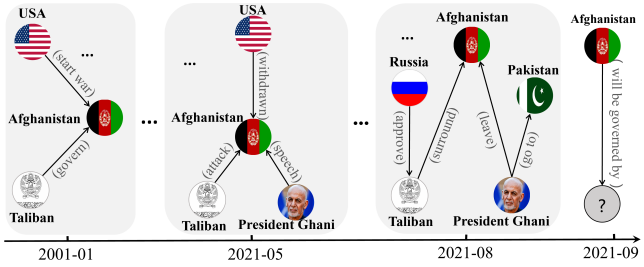


Figure 1: An example of reasoning over TKG. We display three KGs at different timestamps. Each edge indicates the interaction between two entities.

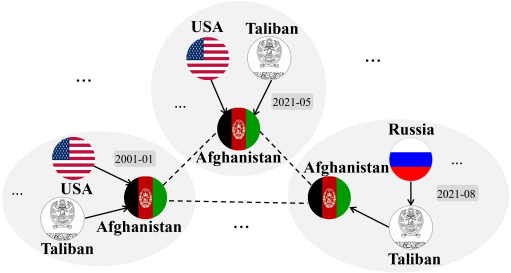


Figure 2: An example of converting a TKG to a global graph. We take *Afghanistan (Afgh)* as an example, which is the common entity among the three KGs in this figure. We link the *Afgh* entities of different timestamps by dotted lines to transform the KG sequence shown in Figure 1 into a global graph (Dotted lines between other common entities are omitted in this figure).

For accurately inferring a future fact, it is common to consider the long-ago history related to the fact and recent events because they carry important **long- and short-term time dependencies** for prediction. As shown in Figure 1, we illustrate an example of reasoning over TKG. The goal is to predict who will govern *Afghanistan (Afgh)* in September 2021. From the **long-term perspective**, the fact occurred in January 2001 demonstrates that *USA* started a war in *Afgh*. Though 2001 is long ago, what happened in 2001 significantly influences the situation of *Afgh* in 2021. From the **short-term perspective**, the recent facts that occurred in August 2021, such as (*Russia*, *Approve*, *Taliban*) and (*Taliban*, *Surround*, *Afgh*), obviously affect the situation of *Afgh* in September 2021.

Some preceding extrapolation TKG reasoning models such as Know-Evolve [26], model all the historical facts as a temporal point process. While some recent attempts, such as RE-NET [13] and RE-GCN [17], incorporate the Graph Neural Network (GNN) into sequence models to capture structural and temporal dependencies between entities. Moreover, xERTE [9] and TITer [24] design explainable models based on subgraph search and reinforcement learning, respectively. Although these models achieve promising results in TKG reasoning, there are still some drawbacks in modeling long- and short-term time dependencies.

Firstly, existing methods do not explicitly leverage long-term time dependencies. *On the one hand*, they are unable to explicitly model the long-term time dependencies of the same entities occurring at different timestamps. Most of them rely on recurrent sequence models to capture temporal dependencies in history, which encode long-term history sequence implicitly and thus easily result in the loss of crucial long-term information [29], such as the important implication of *Afgh* in 2001 for its state in 2021. *On the other hand*, they ignore the explicit associations of different entities across different timestamps. Some interactions between different entities in various timestamps are also necessary to take into account. For example, the impact of *USA* in May 2003 on *Afgh* in August 2021 is important but neglected by existing models. Such impact is more challenging to be captured especially when the time interval between entities is large.

Secondly, existing works neglect the adaptive integration of the long- and short-term time dependencies. Specifically, these two dependencies have different degrees of importance to various entities and relations at different times. Some entities or relations may be more affected by long-term dependencies, while others may be more sensitive to short-term dependencies.

To deal with the two aforementioned challenges, we propose a Hierarchical Relational Graph Neural Network to learn Long- and Short-term representations for TKG reasoning (HGLS). Specifically, we first transform the TKG into a global graph (§4.1), in which each KG with a different timestamp is a sub-graph. KGs at different timestamps are connected into a whole graph by linking common entities among them. As shown in Figure 2, the common entity *Afgh* at different times are connected with each other. In this way, the *USA* in January 2001 and the *Afgh* in August 2021 can be linked by a two-hop connection in the built global graph. And *Afgh* entities at different times are linked by a one-hop connection. These explicit associations are difficult to be established in the sequence data. To model complex semantic and temporal dependencies among entities in the built graph, we further design a Hierarchical Relational Graph Neural Network (HRGNN) (§4.2), which deals with the global graph from the sub-graph level and global-graph level. Afterward, we extract the long- and short-term representation from the two-level output of HRGNN (§4.3). Finally, we utilize a Gating Integration module (§4.4) to adaptively and dynamically fuse the long- and short-term representations for the entity prediction task.

In summary, our work makes the following main contributions:

- We design a new global graph construction method to explicitly associate historical entities in different times. Further, we develop a hierarchical relational graph neural network that explicitly captures the long-term dependencies by encoding the newly built global graph.
- We propose to utilize a Gating Integration module to dynamically and adaptively integrate long- and short-term representations of entities and relations.
- We conduct extensive experiments on four commonly used TKG benchmarks, which demonstrate the effectiveness of HGLS.

2 RELATED WORK

In this section, we review the existing approaches for TKG reasoning in the extrapolation setting [9, 11, 13, 17, 18, 26, 27, 34, 41], which

focuses on predicting new facts in the future based on historical events. We discuss the techniques used in the existing approaches and analyze their strengths and weaknesses.

Specifically, Know-Evolve [26] builds a temporal point process (TTP) to capture the continuous-time temporal dynamics, predicting future facts by estimating the conditional probability of TTP. CyGNet [41] proposes a copy-generation mechanism that utilizes repeat patterns in historical facts to predict future facts while ignoring the high-order semantic dependencies among concurrent entities. In recent years, as graph neural networks (GNNs) have been successfully applied in many dynamic scenarios, such as traffic prediction [2, 38] and recommender system [32, 39, 40], they have also been introduced to model structural semantic dependencies in TKG reasoning. RE-NET [13] and RE-GCN [17] are the most relevant to our work, and they focus on modeling long- and short-term information, respectively. RE-NET focuses on capturing long-term dependencies. It models the long-term historical interactions of the entities to be predicted as sequences, and incorporates RNNs and Relational GCNs [17, 22] to capture temporal and structural dependencies, respectively. However, due to the limitations of RNNs in modeling the dependency between the same entity at different times and the dependency between different entities at different times, RE-NET is unable to utilize long-term historical data effectively. Unlike RE-NET, RE-GCN primarily captures short-term information, which only considers adjacent structural dependencies of entities and introduces more static properties of entities to assist prediction. The underutilization of long-term historical information limits the performance of RE-GCN. To capture the fine-grained temporal information, TANGO [10] extends the neural ordinary differential equations to multi-relational graph convolutional networks for forecasting future links. However, Due to the problem of computational complexity, the insufficient utilization of long-term information still exists in TANGO.

There are also some studies [9, 24] solving the issues of TKG reasoning via path search. For example, xERTE [9] designs an explainable model, which finds an enclosing subgraph around the query by iterartive sampling and attention propagation. TITer [24] presents a TKG forecasting model based on Reinforcement Learning, which includes a times-shaped reward based on Dirichlet distribution to guide the model training. But the limited search length greatly limits the models' utilization of long-term information.

All the methods discussed above have limitations in modeling long- and short-term temporal dependencies, particularly in ignoring the explicit dependencies between different entities at different timestamps in long-term history. Furthermore, they overlook the adaptive integration of long- and short-term information.

3 PRELIMINARIES

In this section, we mainly introduce the temporal knowledge graph (TKG) and formulate the task of TKG reasoning.

Definition 1 (**Temporal Knowledge Graph**). Let $\mathcal{E}$ and $\mathcal{R}$ represent a set of entities and relations. A quadruple $q_t = (e_s, r, e_o, t)$ represents a relation $r \in \mathcal{R}$ that occurs between subject entity $e_s \in \mathcal{E}$ and object entity $e_o \in \mathcal{E}$ at time t. All quadruples occurring at time t constitute a knowledge graph $\mathcal{G}_t$. A temporal knowledge graph (TKG) $\mathcal{G}$ is defined as a sequence of knowledge graphs with different timestamps, i.e., $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_t\}$. $e_s^{t_i} \in \mathcal{G}_{t_i}$ indicates that entity e_s occurs at time t_i. In this paper, we define the long-term history of TKG as $\{\mathcal{G}_{t-M}, \mathcal{G}_{t-M+1}, \cdots, \mathcal{G}_t\}$ and the short-term history of TKG as $\{\mathcal{G}_{t-m}, \mathcal{G}_{t-m+1}, \cdots, \mathcal{G}_t\}$. In general, M is much larger than m.

Table 1: Important Symbols

Symbol	Explanation
$\mathcal{G}_t$	KG at time t in a TKG
e_s^t	Entity e_s appears at time t
$\mathcal{P}_t$	Global Graph at time t
$\mathbf{x}_s$, $\mathbf{x}_r$	Static embedding of entity e_s, relation r
$\mathbf{e}_{s,t}$, $\mathbf{e}_{r,t}$	Dynamic embedding of entity e_s, relation r at time t
$\mathbf{h}_{s,t}^l$, $\mathbf{z}_{s,t}^l$	Embedding of e_s^t at l-th layer of sub-graph level, global-graph level of HRGNN
$\mathbf{e}_{s,t}^L$, $\mathbf{e}_{r,t}^L$	Long-term representation of entity e_s, relation r at time t
$\mathbf{e}_{s,t}^S$, $\mathbf{e}_{r,t}^S$	Short-term representation of entity e_s, relation r at time t

Definition 2 (**Temporal Knowledge Graph Reasoning**). TKG Reasoning is generally categorized into *entity prediction* and *relation prediction*. The *entity prediction* task aims to predict the missing object entity of $(e_s, r, ?, t+1)$ or the missing subject entity of $(?, r, e_o, t+1)$ given historical KG sequence $\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_t\}$. And the *relation prediction* task aims to predict the missing relation of $(e_s, ?, e_o, t+1)$ given $\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_t\}$. This paper focuses on the entity prediction task, and the proposed model can be easily extended to the relation prediction task.

Let vector $\mathbf{x}_s \in \mathbb{R}^d$ and $\mathbf{x}_r \in \mathbb{R}^d$ denote static embedding of each entity e_s and relation r, where d is the dimension. In TKG scenarios, the semantics of entities and relations generally evolve over time. Under this assumption, each entity e_s and relation r at time t can be converted into low-dimensional embedding vector $\mathbf{e}_{s,t} \in \mathbb{R}^d$ and $\mathbf{e}_{r,t} \in \mathbb{R}^d$. The goal of our model is to utilize the historical KG sequences and static entity and relation embeddings to learn dynamic representations of each entity and relation for future time:

$$\mathbf{e}_{s,t+1}, \mathbf{e}_{r,t+1} \leftarrow \Theta\left(\{\mathcal{G}_i\}_{i=0}^t, \mathbf{x}_s, \mathbf{x}_r\right), s \in \mathcal{E}, r \in \mathcal{R}, \tag{1}$$

where Θ, $\mathbf{x}_s$, and $\mathbf{x}_r$ are learnable model function and parameters, respectively. Then, taking the learned embeddings $\mathbf{e}_{s,t+1}, \mathbf{e}_{r,t+1}$ as input to predict $\mathcal{G}_{t+1}$. Some important symbols used in this paper are listed in Table 1.

4 METHODOLOGY

We now present the proposed HGLS, the framework of which is illustrated in Figure 3. There are four components in our model: (1) *Global Graph Construction*, which is to construct a global graph associating the historical KGs explicitly; (2) *Hierarchical Relational Graph Neural Network* that includes two-level operations, which is to capture semantic and temporal dependencies among entities in the global graph; (3) *Long- and Short-term Representation*, which is to obtain long- and short-term representations from the output of HRGNN for each entity and relation; (4) *Gating Integration*, which

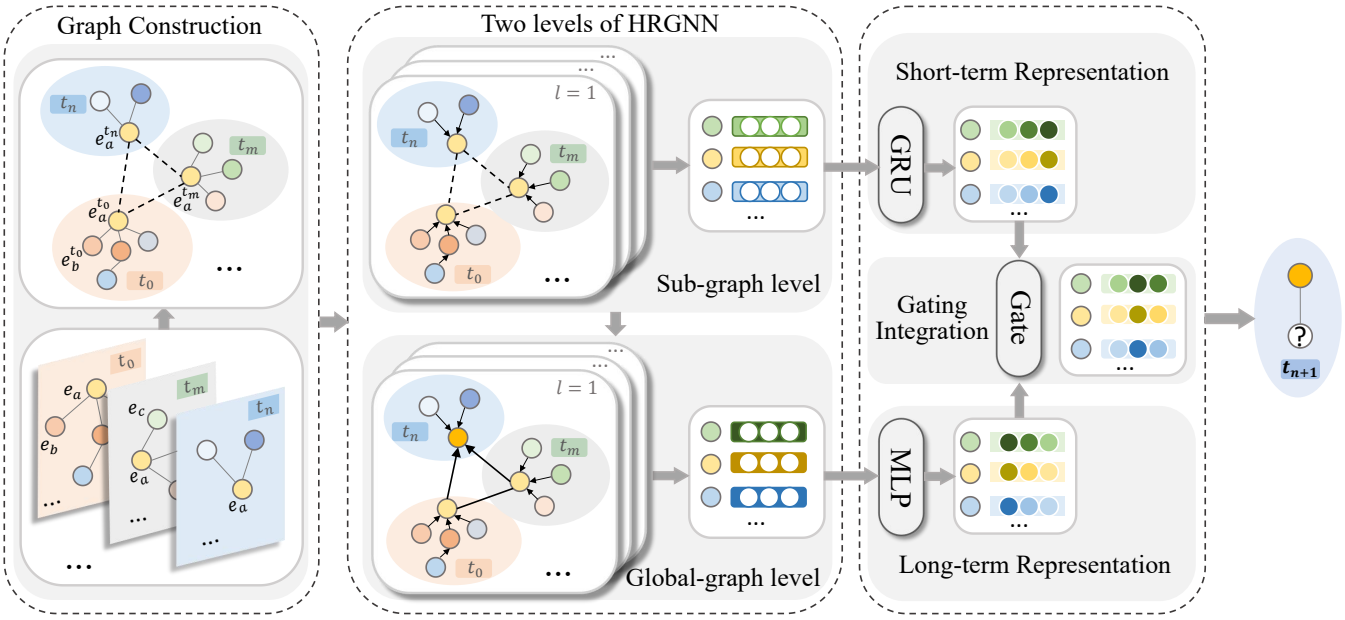


Figure 3: An illustration of HGLS model architecture. The temporal knowledge graph is firstly transformed into a *global graph* (§4.1), where each $\mathcal{G}_{t_i}$ is considered as a sub-graph. Then, the well-designed *Hierarchical Relational Graph Neural Network* (§4.2) is applied on the global graph, where *sub-graph level* operates on each sub-graph, and *global-graph level* performs operations on the whole graph. After that, we feed the outputs of the two levels into the MLP and GRU to obtain long- and short-term representations of each entity and relation, respectively (§4.3). Finally, the long- and short-term representations are fused into one unified representation by *Gating Integration* for the entity prediction task (§4.4).

is to adaptively fuse long- and short-term embeddings into one unified representation. Appendix A provides the pseudo-code of the overall framework.

4.1 Global Graph Construction

To capture the long-term time dependencies of entities, we first build a global graph to associate historical KGs explicitly.

Specifically, for any $\mathcal{G}_{t_i}$ and $\mathcal{G}_{t_j}$ with common entity e_s in the long-term sequence of TKG $\{\mathcal{G}_{t-M}, \mathcal{G}_{t-M+1}, \cdots, \mathcal{G}_t\}$, we add an edge between $e_s^{t_i}$ and $e_s^{t_j}$. In this way, we can connect the KGs at different times by the common entities. Consequently, the KG sequence can be transformed into one multiplex relation graph $\mathcal{P}_t$, namely *global graph*, where each $\mathcal{G}_{t_i}$ can be seen as a sub-graph of $\mathcal{P}_t$ and $e_s^{t_i}$ along with $e_s^{t_j}$ are set as two different nodes in the global graph. Taking the leftmost part of Figure 3 as an example, there are three KGs of various times, $\mathcal{G}_{t_0}$, $\mathcal{G}_{t_m}$, and $\mathcal{G}_{t_n}$, whose common entity is e_a. The three of them are connected to each other through e_a (by the dotted line in Figure 3). Even if the time interval among $\mathcal{G}_{t_0}$, $\mathcal{G}_{t_m}$, and $\mathcal{G}_{t_n}$ is large, their entities can still be associated explicitly in the global graph. For example, $e_a^{t_n}$ and $e_a^{t_0}$, $e_a^{t_n}$ and $e_b^{t_0}$ can be linked by one-hop and two-hop connections in the global graph, respectively. These explicit relationships in different timestamps fail to be captured in the sequence scenario.

Moreover, the constructed $\mathcal{P}_t$ is composed of two types of triplets. One is $(e_s^{t_i}, r_\tau, e_s^{t_j}), t_i, t_j < t$, where relation r_τ is utilized to associate entities that occur at different times and is referred to as a *time-related relation*. The other is $(e_s^{t_i}, r, e_o^{t_i},), t_i < t$, which indicates that entities e_s, e_o, and relation r occur at time t_i, where $r \in \mathcal{R}$. Each r has clear semantics and is called a *semantic relation*. Similar with each semantic relation r, we also transform r_τ into d-dimensional embedding vector $\mathbf{x}_\tau$. For brevity, in the later part we denote all types of relations as $r \in \mathcal{R} \cup \{r_\tau\}$.

By acting on the global graph with a multi-layer graph neural network, we can capture semantic and temporal dependencies between entities that appear concurrently or at different times. However, compared to typical homogeneous graphs and multi-relational graphs, the built graph is more complex, which brings more challenges in its encoding. Specifically, on the one hand, the global graph comprises more complicated relations, i.e., *time-related relation* and *semantic relation*, which are two distinct types of relationships. On the other hand, the majority of nodes in the global graph appear at different times, which contain temporal dependencies between them. To this end, we develop a more refined graph neural network to encode the global graph in the following section.

4.2 Hierarchical Relational Graph Neural Network

To efficiently encode the constructed global graph, in this section we design a Hierarchical Relational Graph Neural Network (HRGNN), which deals with the global graph at two levels. The first level, referred to as the sub-graph level, captures the semantic dependencies between entities among concurrent facts. The second level, dubbed

the global-graph level, models the temporal dependencies between entities at various times.

4.2.1 Sub-graph Level: Modeling Semantic Dependencies among Concurrent Facts. For facts that occur concurrently, the entities generally have strong semantic relevance with their neighbors. Thus, we first consider capturing the semantic dependencies among concurrent facts to obtain the embedding of each node $\mathbf{h}_{s,t_i}$ in each sub-graph $\mathcal{G}_{t_i}$.

In particular, we utilize a relational graph convolution neural network [17, 22] as a semantic aggregator to obtain the embedding of each node in a sub-graph $\mathcal{G}_{t_i}$. Formally, the sub-graph level aggregator is defined as follows:

$$\mathbf{h}_{s,t_i}^{l+1} = \mathrm{f}\left(\frac{1}{|\mathcal{N}_{e_s,t_i}|}\sum_{e_o^{t_i}\in\mathcal{N}_{e_s,t_i}} \mathbf{W}_1^l\left(\mathbf{h}_{o,t_i}^l + \mathbf{x}_r\right) + \mathbf{W}_2^l\mathbf{h}_{s,t_i}^l\right), \quad (2)$$

where $\mathcal{N}_{e_s,t_i}$ is the set of neighbors of $e_s^{t_i}$ in sub-graph $\mathcal{G}_{t_i}$, $\mathrm{f}(\cdot)$ is the RReLU function, $\mathbf{W}_1^l$ and $\mathbf{W}_2^l \in \mathbb{R}^{d\times d}$ are trainable weight parameter matrices for aggregating and self-loop in the l-th layer, and the initial entity embedding $\mathbf{h}_{s,t_i}^0$ and $\mathbf{h}_{o,t_i}^0$ are set to static embedding $\mathbf{x}_s$ and $\mathbf{x}_o$. After ω-layer convolution, we can obtain entity representation $\mathbf{h}_{s,t_i}^\omega$ that only consider the semantic dependencies with its neighbors at time t_i. We omit the superscript ω and use $\mathbf{h}_{s,t_i}$ to denote the output embedding of the sub-graph level.

4.2.2 Global-graph Level: Modeling Temporal Dependencies between Entities. After modeling semantic dependencies, the embedding of each entity $\mathbf{h}_{s,t_i}$ in its sub-graph, i.e., the embedding containing the information of its at-appearance time, can be obtained. To further capture the temporal dependencies between entities at different times, we perform message propagation and aggregation operations on the global graph based on the output of the sub-graph level.

Specifically, due to the variability in the appearance time of adjacent nodes in the global graph, we first consider the influence of time interval on each relation between entity $e_s^{t_i}$ and $e_o^{t_j}$. In this level, the representation of each relation can be calculated by

$$\mathbf{z}_r^{i,j} = \mathbf{x}_r + \phi(|t_i - t_j|), \quad (3)$$

where $|t_i - t_j|$ represents the absolute value of time interval. Following [36], the time encoding function $\phi(\cdot)$ is defined as,

$$\phi(t) := \sqrt{\frac{1}{d}}\left[\cos(\mathbf{w}_1 t + \mathbf{p}_1), \cdots, \cos(\mathbf{w}_d t + \mathbf{p}_d)\right], \quad (4)$$

where $\mathbf{w}, \mathbf{p} \in \mathbb{R}^d$ are learnable parameter vectors.

Then, to capture the impact of temporal and semantic dependencies more precisely, we utilize an attention mechanism [19] to calculate the coefficient between two adjacent nodes. Formally, it can be formulated as:

$$\alpha_{s,o}^{i,j} = \frac{\exp\left(\mathrm{g}\left(\mathbf{a}^\mathrm{T}\mathbf{W}_3^l\left[\mathbf{z}_{s,t_i}^l \parallel \mathbf{z}_{o,t_j}^l \parallel \mathbf{z}_r^{i,j}\right]\right)\right)}{\sum_{e_o^{t_k}\in\widetilde{\mathcal{N}}_{e_s,t_i}}\exp\left(\mathrm{g}\left(\mathbf{a}^\mathrm{T}\mathbf{W}_3^l\left[\mathbf{z}_{s,t_i}^l \parallel \mathbf{z}_{o,t_k}^l \parallel \mathbf{z}_r^{i,k}\right]\right)\right)}, \quad (5)$$

where each initial input entity embedding $\mathbf{z}_{s,t_i}^0$ is the output of sub-graph level $\mathbf{h}_{s,t_i}$, $\widetilde{\mathcal{N}}_{e_s,t_i}$ is the set of neighbors of $e_s^{t_i}$ in $\mathcal{P}_t$, $\mathbf{a} \in \mathbb{R}^{3d}$ and $\mathbf{W}_3^l \in \mathbb{R}^{3d\times 3d}$ are learnable weight parameters in each layer, $\mathrm{g}(\cdot)$ is the LeakyReLU activation function, $\cdot^\mathrm{T}$ represents transposition, and $\parallel$ is the concatenation operation.

After that, we can obtain each entity embedding in the global graph by aggregating the embedding from all its neighbors adaptively,

$$\mathbf{z}_{s,t_i}^{l+1} = \mathrm{h}\left(\sum_{e_o^{t_k}\in\widetilde{\mathcal{N}}_{e_s,t_i}} \alpha_{s,o}^{i,k}\mathbf{W}_4^l\left(\mathbf{z}_{o,t_k}^l + \mathbf{z}_r^{i,k}\right) + \mathbf{W}_5^l\mathbf{z}_{s,t_i}^l\right), \quad (6)$$

where $\mathrm{h}(\cdot)$ is the ReLU activation function, $\mathbf{W}_4^l$ and $\mathbf{W}_5^l$ are weight parameter matrices for aggregating and self-loop in each layer. After β-layer operation in global-graph level, we can get the output $\mathbf{z}_{s,t_i}^\beta$. For simplicity, we use $\mathbf{z}_{s,t_i}$ to represent the output of the global-graph level.

4.3 Long- and Short-term Representations

In this section, we discuss how to obtain the long- and short-term dynamic representations from the output of HRGNN for each entity and relation.

4.3.1 Long-term Representation. Long-term representations reflect the semantics of entities and relations over a long period of time. Since the global-graph level of HRGNN captures the long-term temporal dependencies between entities, the output of this level can be used as the long-term representation of each entity. We feed the output into a nonlinear transformation to get the long-term representation for each entity:

$$\mathbf{e}_{s,t+1}^L = \sigma(\mathbf{W}_6\mathbf{z}_{s,t} + \mathbf{b}), \quad (7)$$

where $\sigma(\cdot)$ is the tanh function, $\mathbf{W}_6 \in \mathbb{R}^{d\times d}$ is the weight matrix, and $\mathbf{b} \in \mathbb{R}^d$ is a bias vector.

Compared with entities, the representations of relations are relatively stable in the long run [9]. So, we use their static embeddings as long-term representations:

$$\mathbf{e}_{r,t+1}^L = \mathbf{x}_r, r \in \mathcal{R}. \quad (8)$$

4.3.2 Short-term Representation. Short-term representations reflect semantic changes of entities and relations in recent times. To capture the short-term information of entities, we use Gated Recurrent Unit (GRU) [3] to encode the most recent m timestamps of each entity based on the output at the sub-graph level. The short-term representation of each entity can be obtained by

$$\mathbf{e}_{s,t+1}^S = \mathrm{GRU}_E\left(\mathbf{e}_{s,t}^S, \mathbf{h}_{s,t}\right), \quad (9)$$

where all entities share the same parameters for GRU_E, $\mathbf{h}_{s,t}$ is the output of node e_s^t in the sub-graph level.

Similarly, the short-term representation of each relation is computed from the representation of the most recent m timestamps of relation r. We also adopt GRU to model the short-term pattern of relations,

$$\mathbf{e}_{r,t+1}^S = \mathrm{GRU}_R\left(\mathbf{e}_{r,t}^S, \mathbf{h}_{r,t}\right), \quad (10)$$

where $\mathbf{h}_{r,t}$ denotes the relation representation at time t, computed by aggregating the representations of entities interacting with the relation at time t:

$$\mathbf{h}_{r,t} = \mathrm{Meanpooling}(\mathbf{h}_{s,t}), \forall e_s \in \mathcal{N}_r^t, \quad (11)$$

the operator Meanpooling$(\cdot)$ acts on the entity set $\mathcal{N}_r^t$ associated with r at time t, all relations share the same parameters for GRU_R.

4.4 Gating Integration

To adaptively integrate long- and short-term representations into a unified representation, we adopt a learnable gating function to fuse entity embedding and relation embedding [12]. Formally, the entity representation can be obtained by

$$\mathbf{e}_{s,t+1} = \sigma(\mathrm{g}_e) \odot \mathbf{h}^L_{s,t+1} + (1 - \sigma(\mathrm{g}_e)) \odot \mathbf{h}^S_{s,t+1}, \tag{12}$$

$$\mathbf{e}_{r,t+1} = \sigma(\mathrm{g}_r) \odot \mathbf{h}^L_{r,t+1} + (1 - \sigma(\mathrm{g}_r)) \odot \mathbf{h}^S_{r,t+1}, \tag{13}$$

where $\mathrm{g}_e, \mathrm{g}_r \in \mathbb{R}^d$ are gate vector parameters to trade-off the long- and short- term information of each entity e and relation r, $\sigma(\cdot)$ is the Sigmoid function to constrain the value of each element in $[0, 1]$, and $\odot$ denotes element-wise multiplication.

4.5 Parameter Learning

In this section, we describe how to get the score for each quadruple $(e_s, r, e_o, t + 1)$ and the learning objective for training HGLS.

4.5.1 Score Function. We utilize ConvTransE [17, 23] as a decoder to calculate the probability of interaction between entity e_s and e_o under the relation r at time $t + 1$. Formally,

$$p_{t+1}(o|s, r, \mathcal{G}_{<t+1}) = \sigma\left(\mathbf{e}_{o,t+1}\, \text{ConvTransE}\left(\mathbf{e}_{s,t+1}, \mathbf{e}_{r,t+1}\right)\right). \tag{14}$$

Similarly, the probability that there is an interaction of r between e_s and e_o at time $t + 1$ can be obtained by

$$p_{t+1}(r|s, o, \mathcal{G}_{<t+1}) = \sigma\left(\mathbf{e}_{r,t+1}\, \text{ConvTransE}\left(\mathbf{e}_{s,t+1}, \mathbf{e}_{o,t+1}\right)\right), \tag{15}$$

where $\sigma(\cdot)$ is Sigmod function, $\mathbf{e}_{s,t+1}$, $\mathbf{e}_{o,t+1}$, and $\mathbf{e}_{r,t+1}$ are dynamic representations that contain both long- and short-term information.

4.5.2 Learning Objective. In addition to the entity prediction task, we also consider the relation prediction task to promote the learning of relation embeddings. Then, the two learning tasks can be defined as,

$$\mathcal{L}_e = -\sum_{t=0}^{T} \sum_{(e_s,r,e_o,t+1)\in G_{t+1}} \log p_{t+1}(o|s, r, \mathcal{G}_{<t+1}), \tag{16}$$

$$\mathcal{L}_r = -\sum_{t=0}^{T} \sum_{(e_s,r,e_o,t+1)\in G_{t+1}} \log p_{t+1}(r|s, o, \mathcal{G}_{<t+1}). \tag{17}$$

Thus, the objective function is as follows:

$$\mathcal{L} = \lambda_1 \mathcal{L}_e + (1 - \lambda_1)\mathcal{L}_r + \lambda_2 \|\Theta\|_2, \tag{18}$$

where λ_1 is a hyper-parameter to control the weight of different tasks, $\|\cdot\|_2$ is L_2 norm, and λ_2 is to control regularization strength.

5 EXPERIMENTS

In this section, we perform experiments on four temporal knowledge graph datasets to evaluate our model. We aim to answer the following questions through experiments.

- **Q1**: How does HGLS perform compared with state-of-the-art TKG forecasting methods on the entity prediction task?
- **Q2**: How do the long-term dependencies learned from the HRGNN module affect the performance of HGLS?
- **Q3**: How does the adaptive integration of long- and short-term dependencies affect the performance of HGLS?
- **Q4**: How sensitive is HGLS with different hyper-parameters?

5.1 Experimental Setup

◊ **Datasets.** We use four typical TKG datasets in our experiments: ICEWS14 [6], ICEWS18 [13], ICEWS05-15 [6], and GDELT [13]. The first three datasets are from the Integrated Crisis Early Warning System [1] and record the facts in 2014, 2018, and the facts from 2005 to 2015, respectively. GDELT is from the Global Database of Events, Language, and Tone [15]. We divide ICEWS14, ICEWS18, ICEWS05-15, and GDELT into training, validation, and test sets with a proportion of 80%, 10%, and 10% by timestamps following [17]. The details of data statistics are shown in Appendix B.

◊ **Baselines.** We compare our HGLS with static KG (SKG) and TKG reasoning models. The SKG models include DisMult [37],ComplEx [28], R-GCN [22], ConvE [5], and RotatE [25]. The TKG models include CyGNet [41], RE-NET [13], xERTE [9], RE-GCN [17], and TITer [24]. We provide implementation details of baselines and HGLS in Appendix C and D, respectively.

◊ **Evaluation Metrics.** In the experiments, we adopt widely-used metrics [13, 17], MRR and Hits@{1, 10} to evaluate the model performance. For a fair comparison with all baseline models, we follow the setup of [9, 17], utilizing the ground truth history during multi-step inference for all compared models. Without loss of generality [17], we report the experimental results under the raw setting.

5.2 Performance Comparison (RQ1)

The performances on entity prediction task of all models are shown in Table 2, from which we have some following observations:

- Most TKG models significantly outperform the static KG reasoning models (i.e., DisMult, ComplEx, R-GCN, ConvE, and RotatE) on all datasets, which confirmed the necessity of using temporal information for TKG predictions. HGLS also outperforms other TKG models in most of the evaluation metrics on four datasets, which verifies the effectiveness of our model and answers **Q1**. Specifically, HGLS outperforms CyGNet because CyGNet mainly considers the repetitive patterns and ignores the high-order semantic dependencies at each time. RE-Net utilizes only several historical interactions of the target entity in predictions, while RE-GCN only considers the facts that occurred in the most recent time and ignores the utilization of long-term information, making them generally perform worse than our model. HGLS also achieves better performance than xERTE and TITer on ICEWS data. This is likely due to the fact that xERTE and TITer predict the target entity with sub-graph-based search and path-based search, respectively, but the limited search length greatly limits the models' utilization of long-term information. Furthermore, the two models ignore the complex temporal dependencies among entities.
- Besides, we notice that all methods perform poorly on GDELT compared to their performances on other datasets. The reason may be that most of GDELT's entities are abstract concepts [17], which makes it difficult to learn their accurate representations in different quadruples and times. By learning sufficient historical information, our HGLS can obtain more general entity and

Table 2: Performance comparison on four datasets in terms of MRR (%), Hit@1 (%), and Hit@10 (%) (raw metrics). The best performance is highlighted in boldface, and the second best is underlined.

Model	ICEWS14			ICEWS05-15			ICEWS18			GDELT		
	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
DisMult	25.31	17.93	42.22	17.43	10.08	30.12	16.59	10.01	31.69	15.64	9.37	29.01
ComplEx	32.33	23.21	52.37	23.14	14.56	41.63	18.84	11.41	25.78	12.23	8.30	20.36
RGCN	28.14	19.43	46.02	27.43	20.15	44.62	18.04	8.57	35.68	10.93	4.59	22.38
ConvE	30.93	21.74	50.18	25.25	16.07	44.34	24.28	15.61	44.59	17.28	10.34	30.63
RotatE	27.53	18.60	47.62	19.39	10.19	38.57	15.35	7.10	33.09	5.48	1.96	13.76
CyGNet	36.51	27.42	54.44	37.46	27.58	56.14	26.82	17.13	45.72	18.30	10.94	31.26
RE-NET	38.91	29.32	57.51	41.72	31.14	62.03	28.42	18.41	47.92	19.01	11.36	31.39
xERTE	39.72	31.18	57.23	44.46	34.23	63.82	27.69	18.99	45.82	18.74	11.21	32.14
RE-GCN*	39.48	29.52	58.64	44.49	33.58	65.84	29.11	19.11	48.64	18.93	11.54	32.35
TITer	41.18	32.13	57.94	43.99	33.67	64.11	28.09	**21.13**	44.01	18.63	11.04	32.34
HGLS	**47.00**	**35.06**	**70.41**	**46.21**	**35.32**	**67.12**	**29.32**	19.21	**49.83**	**19.04**	**11.79**	**33.23**

* indicates that we remove the static information from the model to ensure the fairness of comparisons between all baselines.

relation representations, and thus perform better than other state-of-the-art models.

5.3 Ablation Studies (RQ2 and RQ3)

To investigate the superiority of the HRGNN module in learning long-term dependencies (**Q2**) and the Gating Integration module in fusing long- and short-term information (**Q3**), we compare HGLS with different variants in terms of MRR: HGLS-L, which only considers the long-term dependencies for entities and relations; HGLS-S, which only considers the short-term dependencies for entities and relations; HGLS-Con, which uses the concatenation operation instead of Gating Integration module; HGLS-RGCN, which sets the global-graph level operation as Relational Graph Convolution Network; HGLS-RGAT, which neglects the time encoding in global-graph level. We show the variant models and their results in Table 3 and have the following findings:

- HGLS outperforms HGLS-L and HGLS-S on most evaluation metrics, which confirms that integrating long- and short-term information can effectively enhance the performance on the entity prediction task. Specifically, HGLS has a 20% improvement over the HGLS-S attributing to the introduction of long-term information on ICEWS14. In addition, by introducing short-term information, HGLS also has a 12% gain compared to HGLS-L on ICEWS15. Furthermore, simply concatenating long- and short-term representation may not achieve better results. HGLS utilizes a learnable gate module and achieves better performance than HGLS-Con, verifying that the Gating Integration module can effectively integrate long-term and short-term information.
- Compared to HGLS-RGCN, the performance of HGLS verifies that our HRGNN could effectively improve the information propagation in the global graph. HGLS generally achieves better performance than HGLS-RGAT in most cases. Such improvement might be attributed to the time encoding mechanism, which enhances the ability to distinguish important neighbor nodes under the complex relations in the global graph. So, it is essential to design more refined information propagation mechanisms for the built global graph.

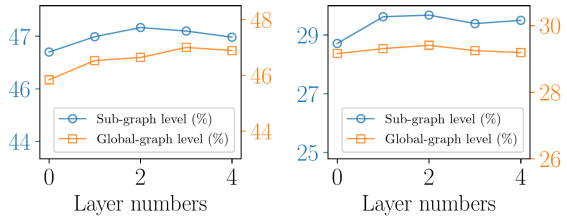
Table 3: Ablation studies on entity prediction task in terms of MRR (%) (raw metrics).

Model	ICEWS14	ICEWS05-15	ICEWS18	GDELT
HGLS-L	46.07	41.20	28.31	18.50
HGLS-S	39.16	44.70	29.16	18.53
HGLS-Con	45.24	45.04	28.92	18.65
HGLS-RGCN	46.64	45.65	29.28	18.85
HGLS-RGAT	46.97	45.15	29.15	18.91
HGLS	**47.00**	**46.21**	**29.32**	**19.04**

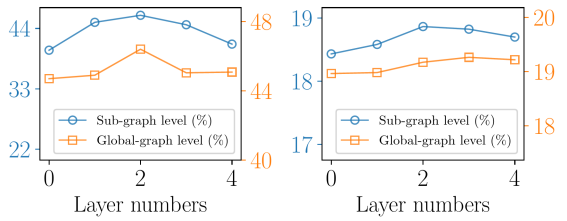
5.4 Sensitivity Analysis (RQ4)

To further explore the sensitivity of HGLS to important hyper-parameters (**Q4**), we study how two hyper-parameters, the layer number of each level in HRGNN and the length of long- and short-term history affect the performance of HGLS.

5.4.1 Effect of HRGNN Layer numbers. HRGNN is a vital module of HGLS. The number of layers at each level decides the degree of modeling semantic and temporal dependencies in the global graph. In this part, we conduct our method with different layer numbers at each level of HRGNN module on four datasets. We set sub-graph level layer number ω and global-graph level layer number β in the range of $\{0, 1, 2, 3, 4\}$. When adjusting one of the levels, the other level uses the optimal number of layers. For simplicity, HGLS_{S_k} indicates the model with k layers for sub-graph level operation, and

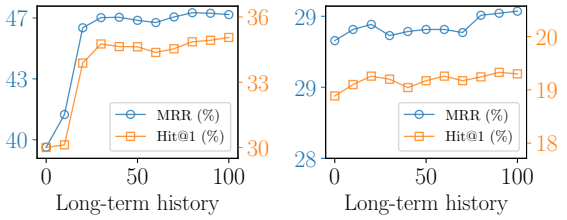


(a) ICEWS14 (b) ICEWS18



(c) ICEWS05-15 (d) GDELT

Figure 4: Performance (MRR %) of HGLS with different layer numbers at each level (the left y-axis shows sub-graph level value, and the right y-axis shows global-graph level value).

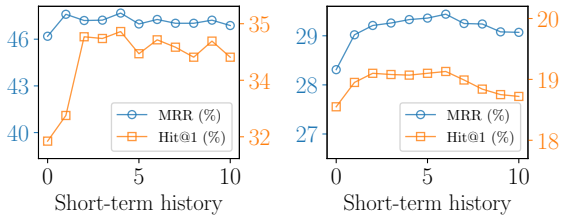


(a) ICEWS14 (b) ICEWS18

Figure 5: Performance of HGLS with different lengths of long-term history (the left y-axis shows MRR value, and the right y-axis shows Hit@1 value).

HGLS_{G_k} indicates the model with k layers for global-graph level operation. The results are shown in Figure 4. The main observations are as follows:

- Increasing the sub-graph level layer numbers substantially enhances the entity prediction. Clearly, the performance reaches the highest value when ω is 2 for all datasets. HGLS_{S_k} $(k > 0)$ achieves consistent improvement over HGLS_{S_0}, which does not consider the semantic dependencies among concurrent facts explicitly. We attribute the improvement to the utilization of high-order neighbor information in concurrent facts, which enhances the semantic representation of entities in each timestamp.
- Similarly, the model performance can be improved by increasing the number of layers in the global-graph level. Specifically, HGLS achieves the best performance on ICEWS14, ICEWS18, ICEWS05, and GDELT when β is 3, 2, 2, and 3, respectively. HGLS_{G_1} outperforms HGLS_{G_0} in all cases, which verifies the necessity of explicitly modeling the long-term dependencies of entities. Moreover, HGLS_{G_2} generally achieves better performance than HGLS_{G_1}. Such improvement might be attributed to the explicit utilization of information across time between different entities. To sum up, the above results illustrate that the long-term temporal dependencies of entities can be captured by the global graph.
- When further stacking the propagation layer at each level, the performance of HGLS begins to deteriorate. This is likely due to over smoothing, which is consistent with the findings in [16].



(a) ICEWS14 (b) ICEWS18

Figure 6: Performance of HGLS with different lengths of short-term history (the left y-axis shows MRR value, and the right y-axis shows Hit@1 value).

5.4.2 Effect of long- and short-term history. To investigate how long-and short-term information affects the performance, we conduct HGLS with different lengths of long- and short-term historical KGs on ICEWS14 and ICEWS18. The results are shown in Figure 5 and 6, respectively. More specifically, we find that the performance steadily grows when the length of long-term history increases from 0 to 50, implying that introducing long-term information can effectively enhance the performance of entity prediction tasks. As the length continued to grow, the performance of the model gradually remained stable. One possible reason is that longer historical information may be less useful for prediction tasks and may introduce additional noise. For the short-term history, increasing its length can obtain the improvement of performance. However, continuing to increase the length does not improve the performance and even starts to degrade in Hit@1. The reason might be the limitation of the GRU net in modeling long sequences.

6 CONCLUSION

In this paper, we have proposed a novel method HGLS for TKG reasoning. The method transforms the TKG sequence into a global graph to explicitly associate historical entities in different times. A Hierarchical Relational Graph Neural Network (HRGNN) module is designed to capture long-term dependencies information among entities by hierarchically encoding the built global graph. Additionally, a Gating Integration module is developed to adaptively integrate long- and short-term information for each entity and relation. The experimental results on four benchmarks and extensive analysis demonstrate the effectiveness and superiority of HGLS in the entity prediction task over TKG.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (U19B2038, 62141608, 62206291).

REFERENCES

[1] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.
[2] Cen Chen, Kenli Li, Sin G Teo, Xiaofeng Zou, Kang Wang, Jie Wang, and Zeng Zeng. 2019. Gated residual recurrent graph neural networks for traffic prediction. In *AAAI*, Vol. 33. 485–492.
[3] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.
[4] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1585–1595.
[5] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *AAAI*. 1811–1818.
[6] A García-Durán, Sebastijan Dumani, and M. Niepert. 2018. Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *EMNLP*. 4816–4821.
[7] R. Goel, SM Kazemi, M. Brubaker, and P. Poupart. 2020. Diachronic Embedding for Temporal Knowledge Graph Completion. *AAAI* (2020), 3988–3995.
[8] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020. DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion. In *EMNLP*. 7301–7316.
[9] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021. Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs. In *ICLR*.
[10] Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021. Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs. In *EMNLP*. 8352–8364.
[11] Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. 2020. Graph Hawkes Neural Network for Forecasting on Temporal Knowledge Graphs. In *AKBC*.
[12] Linmei Hu, Tianchi Yang, Luhao Zhang, Wanjun Zhong, Duyu Tang, Chuan Shi, Nan Duan, and Ming Zhou. 2021. Compare to The Knowledge: Graph Neural Fake News Detection with External Knowledge. In *ACL*. 754–763.
[13] W. Jin, M. Qu, X. Jin, and X. Ren. 2020. Recurrent Event Network: Autoregressive Structure Inferenceover Temporal Knowledge Graphs. In *EMNLP*. 6669–6683.
[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[15] Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, Vol. 2. Citeseer, 1–49.
[16] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.
[17] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal Knowledge Graph Reasoning Based on Evolutional Representation Learning. In *SIGIR*. 408–417.
[18] Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, and Fuchun Sun. 2022. Reasoning over Different Types of Knowledge Graphs: Static, Temporal and Multi-Modal. https://doi.org/10.48550/ARXIV.2212.05767
[19] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are We Really Making Much Progress? Revisiting, Benchmarking and Refining Heterogeneous Graph Neural Networks. In *KDD*. 1150–1160.
[20] Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N Ioannidis, Adesoji Adeshina, Phillip R Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. 2022. Tempoqr: temporal question reasoning over knowledge graphs. In *AAAI*, Vol. 36. 5825–5833.
[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8024–8035.
[22] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. 593–607.
[23] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion. In *AAAI*. 3060–3067.
[24] Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. TimeTraveler: Reinforcement Learning for Temporal Knowledge Graph Forecasting. In *EMNLP*. 8306–8319.
[25] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.
[26] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In *ICML*. 3462–3471.
[27] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*.
[28] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*. 2071–2080.
[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[30] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. 2019. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019).
[31] J. Wu, M. Cao, Jck Cheung, and W. L. Hamilton. 2020. TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion. In *EMNLP*. 5730–5746.
[32] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*, Vol. 33. 346–353.
[33] Tianxing Wu, Arijit Khan, Melvin Yong, Guilin Qi, and Meng Wang. 2022. Efficiently embedding dynamic knowledge graphs. *Knowledge-Based Systems* 250 (2022), 109124.
[34] Yuwei Xia, Mengqi Zhang, Qiang Liu, Shu Wu, and Xiao-Yu Zhang. 2022. MetaTKG: Learning Evolutionary Meta-Knowledge for Temporal Knowledge Graph Reasoning. In *EMNLP*. 7230–7240.
[35] Chenjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020. Temporal Knowledge Graph Completion Based on Time Series Gaussian Embedding. In *ISWC*. 654–671.
[36] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *ICLR*.
[37] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.).
[38] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *IJCAI*. 3634–3640.
[39] Mengqi Zhang, Shu Wu, Meng Gao, Xin Jiang, Ke Xu, and Liang Wang. 2020. Personalized graph neural networks with attention mechanism for session-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3946–3957.
[40] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. 2022. Dynamic Graph Neural Networks for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–1.
[41] 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 *AAAI*. 4732–4740.

A PSEUDOCODE

Algorithm 1 provides the pseudo-code of the overall framework.

B DATASET STATISTICS

The statistics of four TKG datasets are summarized in Table 4.

C BASELINES

The static KG reasoning models compared with our work are shown as follows:

- *DisMult* [37], a model that proposes a simplified bilinear formulation to capture relational semantics.
- *ComplEx* [28], a model that converts the embedding into complex vector space to handle symmetric and antisymmetric relations.
- *R-GCN* [22], a graph neural network that handles the highly multi-relational graph data.
- *ConvE* [5], a model that adopts a 2D convolutional neural network to model the interactions between entities and relations..
- *RotatE* [25], a model that defines each relation as a rotation from the subject entity to object entity in the complex vector space.

Algorithm 1: Training procedure

Input: Train set: TKG sequence $\{\mathcal{G}_0, \cdots, \mathcal{G}_T\}$; Initial model parameters.
Output: A trained HGLS model.

while not converged **do**
  for t *in* $[1 : T-1]$ **do**
    /* Global Graph Construction */
    Obtain global graph $\mathcal{P}_t$ from $\{\mathcal{G}_{t-M}, \cdots, \mathcal{G}_t\}$;
    /* Hierarchical Relational GNN */
    $\mathbf{h}_{s,t} \leftarrow$ Sub-graph level, Eq (2);
    $\mathbf{z}_{s,t} \leftarrow$ Global-graph level, Eq (3), (4), (5), (6) ;
    /* Long- and short-term Representations */
    $\mathbf{h}^L_{s,t+1}, \mathbf{h}^L_{r,t+1} \leftarrow$ Eq (7), (8);
    $\mathbf{h}^S_{s,t+1}, \mathbf{h}^S_{r,t+1} \leftarrow$ Eq (9), (10);
    /* Gating Integration */
    $\mathbf{e}_{s,t+1}, \mathbf{e}_{r,t+1} \leftarrow$ Gating integration, Eq (12), (13) ;
    /* Learning Objective */
    Predicting $\widetilde{\mathcal{G}}_{t+1} \leftarrow$ Eq (14), (15);
    $\mathcal{L} \leftarrow$ Compute the loss between $\widetilde{\mathcal{G}}_{t+1}$ and $\mathcal{G}_{t+1}$, Eq (16), (17), (18) ;
    Update model parameters.

Due to ignoring the time information of KG in the static model, we also choose some state-of-the-art TKG reasoning models to compare with HGLS, which include:

- *CyGNet*[1] [41], a model that utilizes recurrence patterns in historical facts to predict future facts.
- *RE-NET*[2] [13], a model that adopts RNN to capture the historical dependencies of each query and RGCN to model the structural dependencies of each entity.
- *xERTE*[3] [9], an explainable model that designs a temporal relational attention mechanism to extract sub-graph around the query.
- *RE-GCN*[4] [17], a model that uses a recurrent evolution network based on relational graph neural networks to learn the evolution of entities and relations over time. Moreover, the static properties of entities are also incorporated via a static graph module. Since other compared models do not utilize additional information, we remove the static properties in RE-GCN to ensure the fairness of comparisons among models.
- *TITer*[5] [24], a reinforcement learning-based model, which includes a time-shaped reward based on Dirichlet distribution to guide the model training.

D IMPLEMENTATION DETAILS

We implement our HGLS in **Pytorch** [21] and **DGL** Library [30]. We use Adam optimizer [14] with learning rate set to 0.001 and l_2 regularization λ_2 set to 10^{-5}. The embedding size is fixed to 200 for all methods. For the HGLS hyper-parameters, we apply a grid search on the validation set: the length of short-term history m in $\{1, 2, \cdots, 10\}$, the layer number of sug-graph level ω and global-graph level β in $\{1, 2, 3, 4\}$, and the task coefficient λ_1 in $\{0.1, 0.2, \cdots, 1\}$.

For ICEWS14 and ICEWS18, we set the length of long-term history M to 365, and for ICEWS05-15 and GDELT, we set it to 500. The optimal length of short-term history m for ICEWS14, ICEWS05-15, ICEWS18, and GDELT are 3, 5, 6, and 1, respectively. For HRGNN, the optimal sub-graph level layer number is 2 for all datasets, and the optimal global-graph level layer number are 3, 2, 2, and 3 for ICEWS14, ICEWS105-15, ICEWS18, and GDELT, respectively. For the R-GCN used in the sub-graph level of HRGNN, we set the block dimension to 2×2 and the dropout rate for each layer to 0.2. For ConvTransE of score function, the number of kernels, kernel size, and the dropout rate are set to 50, 2×3, and 0.2, respectively. The optimal task coefficient λ_1 is set to 0.7 for all datasets. For the compared methods, we use the default hyper-parameters except for dimensions. All experiments are conducted on NVIDIA Tesla V100 (32G) and Intel Xeon E5-2660.

[1] https://github.com/CunchaoZ/CyGNet
[2] https://github.com/INK-USC/RE-Net
[3] https://github.com/TemporalKGTeam/xERTE
[4] https://github.com/Lee-zix/RE-GCN
[5] https://github.com/JHL-HUST/TITer

Table 4: The statistics of the datasets. Time gap represents time granularity between temporally adjacent facts.

Datasets	ICEWS14	ICEWS05-15	ICEWS18	GDELT
# $\mathcal{E}$	6,869	10,094	23,033	7,691
# $\mathcal{R}$	230	251	256	240
# Train	74,845	368,868	373,018	1,734,399
# Valid	8,514	46,302	45,995	238,765
# Test	7,371	46,159	49,545	305,241
Time gap	24 hours	24 hours	24 hours	15 mins

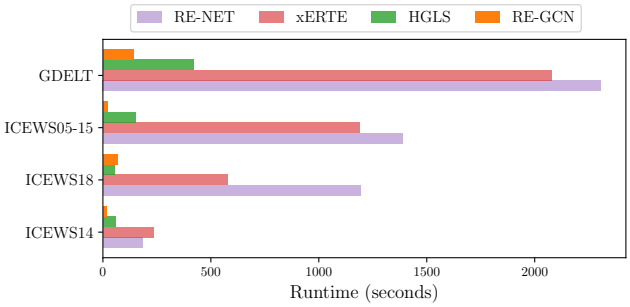


Figure 7: Runtime (seconds) comparison to some baselines.

E EFFICIENCY

To investigate the efficiency of our proposed model, we compare HGLS with RE-GCN, xERTE, and RENET in terms of inference time on the test set. Figure 7 shows that HGLS is faster than xERTE and RE-NET, even though it models both long- and short-term dependencies from history. RE-GCN is much faster than other models because it only considers recent historical facts. RE-NET uses a recurrent neural network to process historical queries recursively,

and xERTE searches the target entity by expanding sub-graphs iteratively. Since many computations in these two models cannot be parallelized, they are slower than the other models. HGLS is mainly based on the GNN model, which can perform parallel computation, thus ensuring a better balance of performance and efficiency.

F CASE STUDY

To understand how our model facilitates the utilization of long-term historical information, we visualize some quadruples associated with the test entities from ICEWS14 based on the attention coefficients of the global-graph level of HRGNN, which are shown in Table 5. The two cases show that our HGLS not only can take advantage of the historical interactions of the test entity, such as the interactions (*Nabih Berri*, *Consult*, *Lawmaker*, 2014/10/21) and (*Nabih Berri*, *Make an appeal or request*, *Legislature*, 2014/04/16) when inferring (*Nabih Berri*, *Make statement*, ?, 2014/11/02), but also can explicitly utilize the historical interactions that are not directly related to the test entity, such as the fact (*Legislature*, *Consult*, *Hezbollah*, 2014/04/16).

Table 5: Case study. Bolded entities indicate the answers of test quadruples. There are two types of associated quadruples: those that are directly associated with the test entity and those that are not. * denotes the latter.

Test quadruple	Associated quadruples
(Nabih Berri, Make statement , **Hezbollah** (?), 2014/11/02)	(Nabih Berri, Consult, Lawmaker, 2014/10/21) (Nabih Berri, Make an appeal or request, Legislature, 2014/04/16) (Legislature, Consult, Hezbollah, 2014/04/16)*
(John Kerry, Express intent to meet, **Federica Mogherini** (?), 2014/11/25)	(John Kerry, Praise, Iran, 2014/11/24) (John Kerry, Consult, Mohammad Javad Zarif, 2014/10/15) (Mohammad Javad Zarif, Discuss by telephone, Federica Mogherini, 2014/10/15)*