Learning Latent Relations for Temporal Knowledge Graph Reasoning

Mengqi Zhang[1,2] , **Yuwei Xia**[3,4] , **Qiang Liu**[1,2]*, **Shu Wu**[1,2] , **Liang Wang**[1,2]
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]Center for Research on Intelligent Perception and Computing (CRIPAC)
State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS)
Institute of Automation, Chinese Academy of Sciences
[3]Institute of Information Engineering, Chinese Academy of Sciences
[4]School of Cyber Security, University of Chinese Academy of Sciences
mengqi.zhang@cripac.ia.ac.cn,xiayuwei@iie.ac.cn,
{qiang.liu,shu.wu,wangliang}@nlpr.ia.ac.cn

Abstract

Temporal Knowledge Graph (TKG) reasoning aims to predict future facts based on historical data. Due to the limitations of construction tools and data sources, many important associations between entities may be omitted in TKG. We refer to these missing associations as *latent relations*. Most of the existing methods have some drawbacks in explicitly capturing *intra-time latent relations* between co-occurring entities and *inter-time latent relations* between entities that appear at different times. To tackle these problems, we propose a novel Latent relations Learning method for TKG reasoning, namely L^{2}TKG. Specifically, we first utilize a Structural Encoder (SE) to obtain representations of entities at each timestamp. Then we design a Latent Relations Learning (LRL) module to mine and exploit the intra- and inter-time latent relations. Finally, we extract the temporal representations from the output of SE and LRL for entity prediction. Extensive experiments on four datasets demonstrate the effectiveness of L^{2}TKG.

1 Introduction

Temporal Knowledge Graph (TKG) is a type of dynamic multi-relational graph data that is used to record evolutionary knowledge in the real world. 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 two primary settings: interpolation and extrapolation. Due to the high practical values in event prediction (Deng et al., 2020), question answer (Mavromatis et al., 2022), and so on, reasoning over TKG under the extrapolation setting has gained much attention in recent years, which mainly aims at predicting facts that occur at time t with $t > t_n$ for given a TKG with history from t_0 to t_n.

Most extrapolation models mainly utilize the temporal and structural information available in the TKG for reasoning. For example, RE-NET (Jin et al., 2020a) and RE-GCN (Li et al., 2021) incorporate recurrent neural networks and graph neural networks to capture the temporal and structural dependencies of historical TKG sequences. Moreover, xERTE (Han et al., 2021a) and TITer (Sun et al., 2021) develop sub-graph search and path search strategies to predict target entities based on existing TKG structures, respectively.

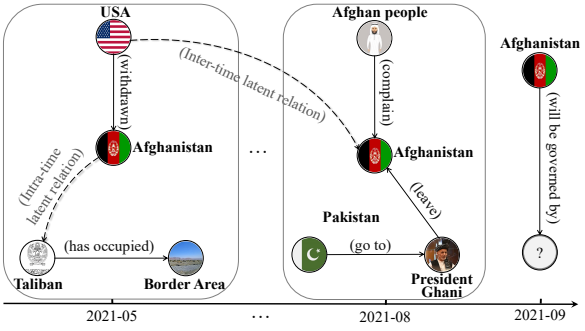


Figure 1: An example of reasoning over TKG. Each edge indicates the interaction between two entities. The gray dotted lines indicate two types of latent relations

Although these methods achieve promising results in TKG reasoning, they still suffer from the problem of missing associations in TKGs. Specifically, the majority of TKG data is identified and extracted automatically from a variety of news articles, such as ICEWS data (Boschee et al., 2015). Many crucial associations between entities may be omitted from TKGs due to the limitations of construction tools and data sources. We refer to such missing associations as *latent relations* between entities. Existing works are unable to explicitly mine and utilize these latent relations, which are mainly manifested in two aspects.

Firstly, existing methods fail to explicitly capture *intra-time latent relations* between co-occurring

entities. In the TKG reasoning process, some concurrent entities may not be connected, but they have a strong semantic correlation with each other. As shown in Figure 1, although *Afghanistan* and *Taliban* are not connected by any relations in the TKG in May 2021, in fact *Taliban* negotiated with *Afghanistan* at that time, which has a significant impact on the situation in *Afghanistan*. Therefore, it is essential to model the critical latent relations between concurrent entities. Most of the existing TKG reasoning models utilize Relational Graph Neural Networks (RGNNs) (Schlichtkrull et al., 2018; Li et al., 2021) to capture the semantic dependencies between entities at each timestamp. However, RGNNs highly rely on existing edges or associations, which makes it challenging to model critical semantic dependencies between some entities that are not directly connected.

Secondly, existing methods ignore the *inter-time latent relations* between entities that appear at various timestamps. Some entities at different timestamps may have strong semantic dependencies, which can provide crucial auxiliary information for TKG reasoning. Consequently, the associations between these entities must also be considered. Take Figure 1 as an example: the effect of the *USA* in May 2003 on *Afghanistan* in August 2021 is significant, but these two distinct entities cannot be directly related in TKG because they appear at different times. The existing TKG reasoning models focus on modeling the semantic dependencies of the same entities at different times, but are inadequate for the aforementioned entities at different times.

To deal with the aforementioned challenges, we propose a novel Latent relations Learning method for TKG reasoning, L^2TKG for brevity. The overall framework of L^2TKG is presented in Figure 2. Specifically, we first utilize a Structural Encoder (SE) to generate the representations of entities at each timestamp. Inspired by graph structural learning (Jin et al., 2020b; Zhu et al., 2021b; Liu et al., 2022), we design a Latent Relations Learning module (LRL) for learning the two types of missing associations in TKG reasoning. Based on the embedding of entities at each timestamp, LRL is able to establish new important associations between unconnected entities in a learnable manner, and then encode the learned latent relational graph to obtain more comprehensive representations of entities. Finally, we extract temporal representations from the output of SE and LRL for the entity prediction task.

In summary, our work makes the following main contributions:

- We highlight and explore the necessity of capturing critical missing associations in TKG reasoning.
- We introduce graph structure learning into TKG reasoning, and propose a novel and effective latent relations learning method to alleviate the problem of missing associations in TKG reasoning.
- We conduct extensive experiments on four typical TKG datasets, which demonstrate the effectiveness of our proposed model.

2 Related Work

In this paper, we illustrate the related work about TKG reasoning under the extrapolation setting and graph structure learning.

2.1 TKG Reasoning under the Extrapolation Setting

TKG reasoning under the extrapolation setting aims to predict new facts in future timestamps based on historical TKG sequence.

Specifically, GHNN (Han et al., 2020) and Know-Evolve (Trivedi et al., 2017) use temporal point process (TTP) to model the TKG data for capturing the continuous-time temporal dynamics, and they predict the future facts by estimating the conditional probability of TTP. CyGNet (Zhu et al., 2021a) proposes a copy-generation mechanism that utilizes repeating patterns in historical facts to predict the future, but ignores the high-order relationships between entities and relations.

Some recent methods (Jin et al., 2020a; Li et al., 2021, 2022) combine graph neural networks and recurrent neural networks to model the semantic and time dependencies between entities. For example, RE-NET (Jin et al., 2020a) incorporates RNNs and RGCNs to capture the temporal and structural dependencies from sequences of entities to be predicted. Different from RE-NET, RE-GCN (Li et al., 2021) takes more adjacent structural dependencies of entities and relations into consideration and introduces some static properties of entities. To consider more global temporal information, TiRCN (Li et al., 2022) designs a global history encoder network collecting repeated historical facts. To

further capture fine-grained temporal information, TANGO (Han et al., 2021b) also adopts Neural Ordinary Differential Equations to the TKG reasoning for forecasting future links. Besides, some works (Han et al., 2021a; Sun et al., 2021) also propose sub-graph or path search strategies for TKG reasoning. xERTE (Han et al., 2021a) designs an explainable model for entity prediction, which provides a sub-graph search strategy to find answer entities. TITer (Sun et al., 2021) performs path search based on reinforcement learning to predict future entities, which includes a times-shaped reward based on Dirichlet distribution to guide the model training. Recently, CENET (Xu et al., 2023) combines the contrastive learning strategy with TKG models to identify significant entities from historical and non-historical dependency. However, all of these above methods rely on existing associations between entities or structures in TKG and ignore the utilization of important latent associations between entities.

2.2 Graph Structure Learning

Owing to the capability of dealing with graph structure data, Graph Neural Networks (GNNs) have been widely utilized and achieved promising performance in many tasks, such as Recommender System (Wu et al., 2019; Chen and Wong, 2020; Zhang et al., 2021) and Text Classification (Yao et al., 2019; Zhang et al., 2020). Recently, researchers have proposed that graph data may contain noises that may deteriorate the training of GNNs (Jin et al., 2020c). To deal with this problem, graph structure learning (GSL) is proposed, aiming to learn an optimized graph structure and node representations jointly.

There are three main categories of GSL models (Zhu et al., 2021b): *metric-learning-based* methods (Jiang et al., 2019; Chen et al., 2020; Cosmo et al., 2020; Li et al., 2018b), *probabilistic* methods (Franceschi et al., 2018, 2019; Zhang et al., 2019), and *direct-optimized* methods (Yang et al., 2019; Jin et al., 2020c). For example, IDGL (Chen et al., 2020) cast the graph learning problem as a similarity metric learning problem and leverages adaptive graph regularization for controlling the quality of the learned graph. DGM (Kazi et al., 2023) introduces a learnable Differentiable Graph Module that predicts edge probabilities in the graph. NeuralSparse (Chen et al., 2020) proposes a supervised graph sparsification technique that improves generalization power by learning to remove potentially task-irrelevant edges from input graphs.

In contrast to the above-mentioned work to optimize the existing graph structure, our work mainly uses a metric-learning-based approach to discover new and important missing associations in TKG and to obtain optimal entity representations for TKG reasoning.

3 Preliminaries

In this section, we introduce the definition of TKG, formulate the task of TKG reasoning, and explain some notations used in this paper.

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 quadruple occurring at time t constitute a knowledge graph $\mathcal{G}_t$. $e_s^t \in \mathcal{G}_t$ indicates that entity e_s occurs at time t. A temporal knowledge graph (TKG) $\mathcal{G}$ is defined as a sequence of knowledge graphs with different timestamps, i.e., $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_t\}$.

Definition 2 (**Temporal Knowledge Graph Reasoning**)**.** We mainly focus on the *entity prediction* task for TKG reasoning in the paper. 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 $\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_t\}$.

Let vector $\mathbf{x}_s \in \mathbb{R}^d$ and $\mathbf{x}_r \in \mathbb{R}^d$ represent static embedding of entity e_s and relation r, where d is the dimension. The general paradigm of TKG reasoning is to learn future representations of each entity for predicting $\mathcal{G}_{t+1}$ by using the historical KG sequences $\{\mathcal{G}_i\}_{i=0}^t$, static entity and relation embeddings $\mathbf{x}_s$ and $\mathbf{x}_r$.

4 Methodology

In this section, we present the proposed L^{2}TKG. The overall framework of L^{2}TKG is illustrated in Figure 2. There are three main components: (1) *Structural Encoder (SE)*, which captures the semantic dependencies between concurrent entities at each timestamp based on the existing TKG structure. (2) *Latent Relations Learning (LRL)*, which mines and exploits the critical intra-time and inter-time latent relations between entities. (3) *Temporal Representation Learning*, which extracts temporal representation for each entity from the output of SE and LRL.

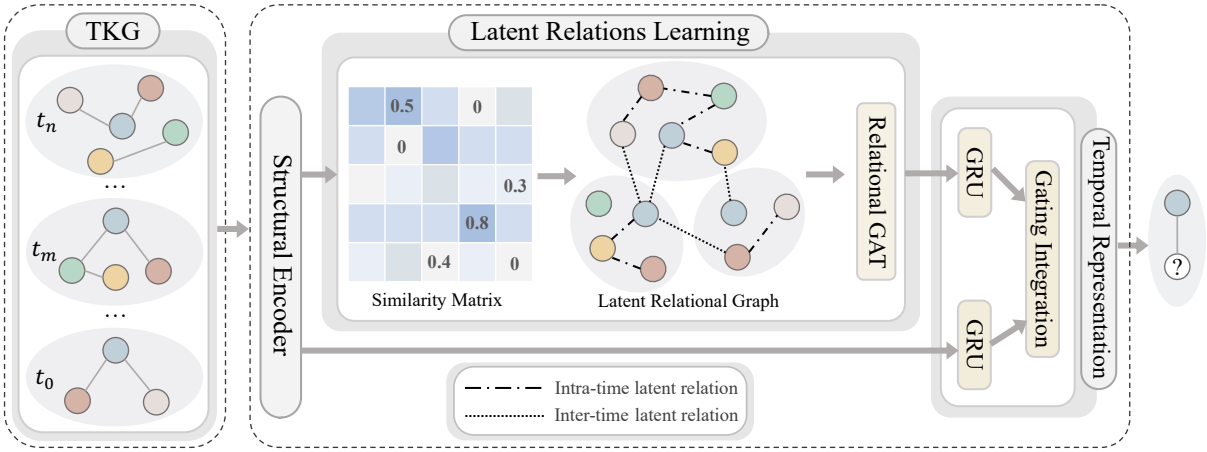


Figure 2: An illustration of L^{2}TKG model architecture. We first utilize a *Structural Encoder* (§4.1) to obtain representations of entities at each timestamp. Then, the well-designed *Latent Relations Learning* (§4.2) module sequentially calculates the similarity matrix, and constructs and encodes a latent relational graph to obtain a comprehensive representation of each entity. Finally, we extract the temporal representations from the output of SE and LRL for the entity prediction task (§4.3).

4.1 Structural Encoder

At each timestamp, connected co-occurring entities have strong semantic dependencies. To capture these semantic dependencies, we propose a structural encoder based on relational graph convolution neural network (Schlichtkrull et al., 2018; Li et al., 2021), which aims to obtain the embedding of each entity at the timestamp of its appearance.

Formally, the structural encoder can be defined as follows:

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

where $\mathcal{N}_{e_s}^{t_i}$ is the set of neighbors of e_s in $\mathcal{G}_{t_i}$, $f(\cdot)$ is the ReLU function, $\mathbf{W}_1$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$ are trainable weight parameter matrices in each 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 $\mathbf{h}_{s,t_i}^{\omega}$ at time t_i. We omit the superscript ω and use $\mathbf{h}_{s,t_i}$ to denote the embedding of e_s at time t_i.

4.2 Latent Relations Learning

After capturing the existing entity semantic dependencies among concurrent entities at each timestamp, we then design a latent relations learning module to discover and exploit important missing associations: *intra-time latent relation* and *inter-time latent relation*, between historical entities.

4.2.1 Learning latent relational graph

The purpose of this part is to mine latent relations between entities appearing in TKG sequence $\mathcal{G} = \{\mathcal{G}_{t-L}, \cdots, \mathcal{G}_t\}$. The same entity that appears at different times is treated as two separate entities, such as $e_s^{t_i}$ and $e_s^{t_j}$. Thus, the number of entities to be considered in this module is $N = \sum_{t_i=t-L}^{t} n_{t_k}$, where n_{t_k} is the number of entities in $\mathcal{G}_{t_k}$ and L is the history sequence length.

Without loss of generality, we assume that the highly associated entities are also similar in the embedding space. As a result, we first compute the similarity between entity embeddings. There are many similarity metrics that can be chosen. We use simple cosine metrics to compute the similarity:

$$d(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{W}_3\mathbf{x})^{\mathrm{T}}(\mathbf{W}_4\mathbf{y})}{\|\mathbf{W}_3\mathbf{x}\|\|\mathbf{W}_4\mathbf{y}\|}, \quad (1)$$

where $\cdot^{\mathrm{T}}$ represents transposition, $\mathbf{W}_3$ and $\mathbf{W}_4 \in \mathbb{R}^{d\times d}$ are learnable weight parameters.

To reduce the complexity of calculations, we only calculate the similarity between entity pairs that have not connected in the TKG sequence. Next, we will introduce in detail how to obtain the crucial intra-time and inter-time latent relations, respectively.

Intra-time latent relation learning. We calculate the similarity between any two entity representations appearing at the same timestep t_p but not becoming connected. The similarity matrix $\mathbf{S}^{t_p} \in \mathbb{R}^{n_{t_p}\times n_{t_p}}$ between unconnected entities at time

t_p is computed by

$$\mathbf{S}_{i,j}^{t_p} = d(\mathbf{h}_{e_i,t_p}, \mathbf{h}_{e_j,t_p}), \tag{2}$$

where $(e_i, e_j) \in \mathcal{G}_{t_p}$ and $\{(e_i, r, e_j, t_p)|\forall r\} \notin \mathcal{G}_{t_p}$. For other case, the value of $\mathbf{S}_{i,j}^{t_p}$ is set to 0.

To retain important latent relations and reduce noise interference, we use the sparse operation based on k-NN (Chen et al., 2009) for each matrix $\mathbf{S}^{t_p}$, that is: for each entity, we only keep latent relations with the top-k scores. In this way, the final similarity matrix at time t_p is calculated as:

$$\hat{\mathbf{S}}_{i,j}^{t_p} = \begin{cases} \mathbf{S}_{i,j}^{t_p}, & \mathbf{S}_{i,j}^{t_p} \in \text{top-k}(\mathbf{S}_{i,:}^{t_p}) \\ 0, & otherwise \end{cases}, \tag{3}$$

where $\mathbf{S}_{i,:}^{t_p}$ denotes the i-row of $\mathbf{S}^{t_p}$. Each $\hat{\mathbf{S}}^{t_p}$ records the important intra-time latent relations between entities at time t_k.

Inter-time latent relation learning. We calculate the similarity between any two entity representations appearing at different timesteps t_p and t_q.

$$\mathbf{Q}_{i,j}^{t_p,t_q} = d(\mathbf{h}_{e_i,t_p}, \mathbf{h}_{e_j,t_q}), \tag{4}$$

where $e_i \in \mathcal{G}_{t_p}, e_j \in \mathcal{G}_{t_q}, t_p \neq t_q$. For other cases, the value of $\mathbf{Q}^{t_p,t_q}$ is 0. Similar to intra-time latent relation learning, we also perform sparsification on the similarity matrix:

$$\hat{\mathbf{Q}}_{i,j}^{t_p,t_q} = \begin{cases} \mathbf{Q}_{i,j}^{t_p,t_q}, & \mathbf{Q}_{i,j} \in \text{top-k}(\mathbf{Q}_{i,:}^{t_p,t_q}) \\ 0, & otherwise \end{cases}. \tag{5}$$

Each $\hat{\mathbf{Q}}^{t_p,t_q}$ records the important inter-time latent relation between entities at different times.

We choose k values independently for the sparse operations of the two latent relations learning, denoted as k_1 and k_2, respectively. Based on the learned similarity matrices, we then build a latent relational graph $\mathcal{P}$. In specific, if $\hat{\mathbf{S}}_{i,j}^{t_p} > 0$, we construct intra-time latent relation between $e_i^{t_p}$ and $e_j^{t_p}$ in $\mathcal{P}$. Similarly, if $\hat{\mathbf{Q}}_{i,j}^{t_p,t_q} > 0$, we construct inter-time latent relation between $e_i^{t_p}$ and $e_j^{t_q}$. We only consider latent relations and omit original relations of the TKG sequence in the graph $\mathcal{P}$. Similar to existing relations, we also transform the two types of latent relations into low-dimensional embedding vectors, which are learnable parameters. To facilitate presentation, we directly use numerical numbers $\{1, ..., N\}$ to denote the nodes in $\mathcal{P}$ in the next section.

4.2.2 Encoding latent relational graph

After obtaining the latent relational graph $\mathcal{P}$, we perform message propagation and aggregation operations on it to capture the semantic dependencies of entities under the newly learned associations.

In specific, we first utilize a graph attention mechanism (Lv et al., 2021) to calculate the coefficient between two adjacent nodes i and j under the learned latent relation r in $\mathcal{P}$:

$$\alpha_{ij} = \frac{\exp\left(f\left(\mathbf{a}^{\mathrm{T}}\mathbf{W}_3\left[\mathbf{z}_i^l \parallel \mathbf{z}_j^l \parallel \mathbf{z}_r^{ij}\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(f\left(\mathbf{a}^{\mathrm{T}}\mathbf{W}_3\left[\mathbf{z}_i^l \parallel \mathbf{z}_k^l \parallel \mathbf{z}_r^{ik}\right]\right)\right)},$$

where initial embedding $\mathbf{z}_i^{(0)}$ is the corresponding entity embedding obtained by Structural Encoder (§4.1), $\mathbf{z}_r^{ij}$ is the embedding of latent relation between node i and node j, $\mathcal{N}_i$ is the set of neighbors of i in $\mathcal{P}$, $\mathbf{a} \in \mathbb{R}^{3d}$ and $\mathbf{W}_5 \in \mathbb{R}^{3d\times 3d}$ are learnable weight parameters in each layer, $f(\cdot)$ is the LeakyReLU activation function, and $\parallel$ is the concatenation operation.

After that, we obtain a more comprehensive representation for each entity by aggregating the embeddings from its neighbors in the latent relational graph,

$$\mathbf{z}_i^{l+1} = g\left(\sum_{k\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}_6\left(\mathbf{z}_k^l + \mathbf{z}_r^{ik}\right) + \mathbf{W}_7\mathbf{z}_i^l\right),$$

where $g(\cdot)$ is the ReLU activation function, $\mathbf{W}_6$ and $\mathbf{W}_7$ are weight parameter matrices in each layer. For simplicity, we use $\mathbf{z}_i$ to represent $\mathbf{z}_i^\beta$ after β-layer operation.

4.3 Temporal Representations Learning

In addition to the semantic dependencies under different relations, the temporal patterns of entities are also crucial for TKG reasoning. This section discusses how to obtain the temporal representations of entities based on the output of SE and LRL.

4.3.1 Global temporal representation

Since the LRL module captures the semantic dependencies of the entity under the new associations, its output contains more global information. We further input them into GRU to get the global temporal representation of each entity:

$$\mathbf{e}_{s,t+1}^G = \text{GRU}_G\left(\mathbf{e}_{s,t}^G, \mathbf{z}_{s,t}\right), \tag{6}$$

where $\mathbf{z}_{s,t}$ corresponds to the output representation of LRL (§4.2) at entity e_s^t.

4.3.2 Local temporal representation

Local temporal representation reflects the semantic changes of entities in recent times. Following (Li et al., 2021, 2022), we adopt GRU to encode the most recent m timestamps of each entity based on the output of the structural encoder:

$$\mathbf{e}^L_{s,t+1} = \mathrm{GRU}_L\left(\mathbf{e}^L_{s,t}, \mathbf{h}_{s,t}\right), \tag{7}$$

where $\mathbf{h}_{s,t}$ is the corresponding entity embedding obtained by Structural Encoder (§4.1).

4.3.3 Gating Integration

To facilitate model reasoning, we adopt a learnable gating function (Hu et al., 2021) to adaptively integrate the global and local temporal representations into a unified temporal representation. Formally,

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

where $\mathrm{g}_e \in \mathbb{R}^d$ is a gate vector parameter to trade-off two types of temporal information of each entity e, $\sigma(\cdot)$ is to constrain the value of each element in $[0, 1]$, and $\odot$ denotes element-wise multiplication.

4.4 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 our model.

We first calculate the probability of interaction between entity e_s and e_o under the relation r at time $t+1$. Formally,

$$p_{t+1}(o|s,r) = \sigma\left(\mathbf{e}_{o,t+1}\,\mathrm{f}\left(\mathbf{e}_{s,t+1}, \mathbf{x}_r\right)\right),$$

where $f(\cdot)$ is decoder function ConvTransE (Li et al., 2021), $\mathbf{e}_{s,t+1}$ and $\mathbf{e}_{o,t+1}$ are temporal representations that contain both global- and local temporal information.

The learning tasks can be defined as,

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

Thus, the objective function is as follows:

$$\mathcal{L} = \mathcal{L}_e + \lambda_1 \|\Theta\|_2,$$

where $\|\cdot\|_2$ is L_2 norm and λ_1 is to control regularization strength.

5 Experiments

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

- **Q1**: How does L^{2}TKG perform compared with state-of-the-art TKG reasoning methods on the entity prediction task?
- **Q2**: How does L^{2}TKG perform in learning missing associations?
- **Q3**: How do different components affect the L^{2}TKG performance?
- **Q4**: How sensitive is L^{2}TKG with different hyper-parameter settings?

5.1 Experimental Setup

5.1.1 Datasets

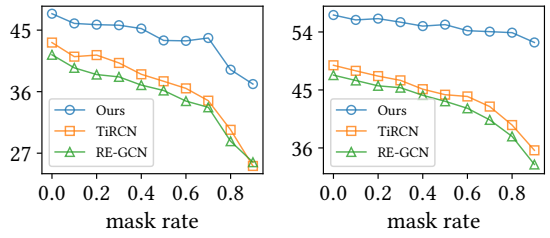
We evaluate our L^{2}TKG on four representative TKG datasets in our experiments: ICEWS14 (García-Durán et al., 2018), ICEWS18 (Jin et al., 2020a), ICEWS05-15 (García-Durán et al., 2018), and GDELT (Jin et al., 2020a). The first three datasets are from the Integrated Crisis Early Warning System (Boschee et al., 2015) and record the facts in 2014, 2018, and the facts from 2005 to 2015, respectively. The last one is from the Global Database of Events, Language, and Tone (Leetaru and Schrodt, 2013). The details of data split strategy and data statistics are shown in Appendix A.

5.1.2 Baselines

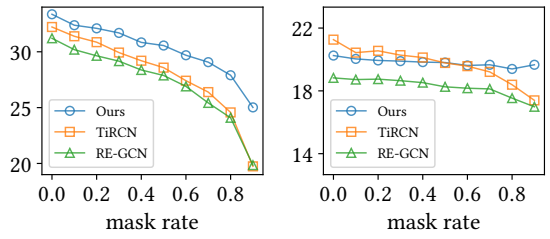
We compare L^{2}TKG with static KG (SKG) reasoning models: DisMult (Yang et al., 2015),ComplEx (Trouillon et al., 2016), R-GCN (Schlichtkrull et al., 2018), ConvE (Dettmers et al., 2018), and RotatE (Sun et al., 2019), and TKG models: CyGNet (Zhu et al., 2021a), RE-NET (Jin et al., 2020a), xERTE (Han et al., 2021a), TIEer (Sun et al., 2021), RE-GCN (Li et al., 2021), TiRCN (Li et al., 2022), and CENET (Xu et al., 2023). We provide implementation details of baselines and L^{2}TKG in Appendix B and C, respectively.

5.1.3 Evaluation Metrics

We adopt widely-used metrics (Jin et al., 2020a; Li et al., 2021), MRR and Hits@{1, 10} to evaluate the model performance in the experiments. For a fair comparison, we follow the setup of Li et al. (2022), using the ground truth history during multi-step inference, and report the experimental

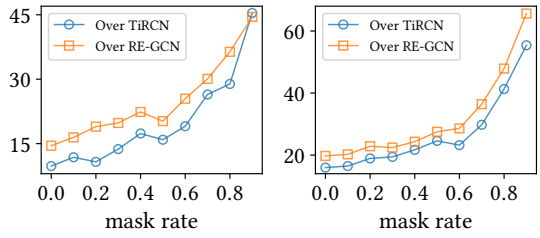


(a) ICEWS14 (b) ICEWS05-15

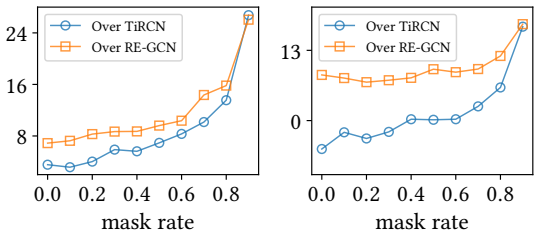


(c) ICEWS18 (d) GDELT

Figure 3: Performance of L^{2}TKG, TiRCN, and RE-GCN under different mask rates in terms of MRR (%).



(a) ICEWS14 (b) ICEWS05-15



(c) ICEWS18 (d) GDELT

Figure 4: The relative improvements (%) of L^{2}TKG over TiRCN and RE-GCN under different mask rates.

results under the time-aware filtered setting for all compared models.

5.2 Performance Comparison (RQ1)

The performances on entity prediction task of all models are shown in Table 1. From the results we have some following observations:

L^{2}TKG achieves the best performance on all ICEWS datasets with most evaluation metrics, which verifies the effectiveness of our model. Specifically, L^{2}TKG significantly outperforms all compared static models, demonstrating the importance of modeling temporal information in TKG reasoning. Our model is better than RE-GCN and TiRCN. The reason might be that RE-GCN only utilizes the most recent historical sequence of TKG and neglects the global historical information of the entities. Although TiRCN considers more historical dependencies than RE-GCN, it only utilizes the first-order repetitive patterns of global history. Our L^{2}TKG not only encodes some recent information but also exploits more learned latent relations between historical entities, allowing it to make better use of global historical data than TiRCN. Compared with L^{2}TKG and TiRCN, both the RE-NET and CyGNET ignore the use of local temporal information about entities and thus perform less well than most TKG models. Different from our model, xERTE and TITer predict the target entity with sub-graph-based search and path-based search, respectively, but their search methods rely on existing paths, which limits their search scope and impairs their performance.

Compared with the ICEWS data, the GDELT data has a higher number of facts appearing at each time, and the problem of missing association is less severe, so our model has limited improvement compared to SOTA models.

5.3 Performance Comparison in Learning Missing Associations (RQ2)

To further verify the ability of L^{2}TKG to mine and exploit the latent relations, we execute L^{2}TKG on datasets with varying degrees of missing associations. On the ICEWS and GDELT datasets, we mask off $\{0.1, ..., 0.9\}$ of the existing relations in the KG of each timestamp, respectively. We show the performance of RE-GCN, TiRCN, and L^{2}TKG under different mask ratios in Figure 3, and the relative improvements of L^{2}TKG over RE-GCN and TiRCN in Figure 4. From the results, we have the following observations:

From Figure 3 we find that the performance of all models decreases to different degrees as the mask rate increases, which is due to the gradual decrease of historical association information in the dataset. Nevertheless, our model performance degrades relatively flat and maintains good performance in the case of severe missing association information (mask rate > 0.6). In Figure 4, the rel-

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	37.65	27.43	57.90	40.42	29.44	61.60	27.12	17.21	46.85	20.22	12.35	35.82
RE-NET	39.86	30.11	58.21	43.67	33.55	62.72	29.78	19.73	48.46	19.55	12.38	34.00
xERTE	40.79	32.70	57.30	46.62	37.84	63.92	29.31	21.03	46.48	19.45	11.92	34.18
TITer	41.73	32.74	58.44	47.60	38.29	64.86	29.98	22.05	44.83	18.19	11.52	31.00
RE-GCN*	41.99	32.93	61.92	47.39	37.65	68.56	30.13	19.11	48.86	19.13	11.54	32.35
CENET	41.30	32.58	58.22	47.13	37.25	67.61	29.65	19.98	48.23	19.73	12.04	34.98
TiRGN*	43.18	33.12	62.24	48.83	38.62	69.20	32.22	**22.24**	51.88	**21.67**	13.63	37.60
L^2TKG	**47.40**	**35.36**	**71.05**	**57.43**	**41.86**	**80.69**	**33.36**	22.15	**55.04**	20.53	**13.67**	**37.79**
$\Delta Improve.$	9.77%	6.73%	14.15%	17.61%	8.39%	16.60%	3.54%	–	6.09%	–	0.29%	0.51%

Table 1: Performance comparison on four datasets in terms of MRR (%), Hit@1 (%), and Hit@10 (%) (time-aware metrics). The best performance is highlighted in boldface, and the second-best is underlined. * indicates that we remove the static information from the model to ensure the fairness of comparisons between all baselines.

ative performance improvement of our model compared to RE-GCN and TiRCN gradually increases. In particular, the model performance improves substantially when the mask rate exceeds 0.6. These findings all indicate that our latent relations learning method can effectively mine and exploit the missing associations between entities and alleviate the problem of missing associations in history.

5.4 Ablation Studies (RQ3)

To investigate the superiority of each component in our model, we compare L^2TKG with different variants in terms of MRR. Specifically, we modify L^2TKG by removing the latent relation learning module (**w/o LRL**), intra-time relation learning of LRL (**w/o LRL-Intra**), inter-time relation learning of LRL (**w/o LRL-Inter**), local temporal representation module (**w/o Ltr**), global temporal representation module (**w/o Gtr**), and structural encoder (**w/o SE**), respectively. We show their results in Table 2 and have the following findings:

L^2TKG significantly outperforms L^2TKG (**w/o LRL**) on all datasets, which confirms that our latent relations learning module effectively discovers and utilizes missing important associations in TKG sequence to assist prediction tasks. L^2TKG (**w/o LRL-Intra**) and L^2TKG (**w/o LRL-Inter**) also achieves better performance than L^2TKG (**w/o LRL**). The improvements verify that both learned inter-time and intra-time latent relations contribute to model performance. Compared with L^2TKG (**w/o LRL-Intra**) and (**w/o LRL-Inter**), the performance of L^2TKG is further improved, which means that two latent relations play different roles in promoting the prediction of the model, and it is necessary to use both latent relations together.

Model	ICEWS14	ICEWS05-15	ICEWS18	GDELT
w/o LRL	38.32	44.49	28.74	19.46
w/o LRL-Intra	47.08	55.84	33.05	20.36
w/o LRL-Inter	47.00	56.30	33.30	20.41
w/o Ltr	36.40	43.00	32.15	19.03
w/o Gtr	40.64	49.27	29.61	20.24
w/o SE	44.34	47.01	31.18	19.78
L^2TKG	**47.40**	**57.43**	**33.36**	**20.53**

Table 2: Ablation studies on datasets in terms of MRR (%) with time-aware metrics.

L^2TKG also obtains significant improvements over L^2TKG (**w/o Ltr**) and L^2TKG (**w/o Gtr**), indicating that both global- and local-temporal information can effectively enhance the performance on the prediction task. The improvement between L^2TKG and L^2TKG (**w/o SE**) verifies the importance of capturing the semantic dependencies among co-occurring at each timestamp.

5.5 Sensitivity Analysis (RQ4)

The structural encoder (SE) and latent relation learning (LRL) are two vital modules in our model. To further investigate their effects, we study how the k values of sparse operations (Intra-time and Inter-time learning) and the layer numbers of SE and LRL affect the performance of L^2TKG. The results and analyses are shown in Appendix D.

6 Conclusion

In this paper, we have proposed a novel method $\mathrm{L}^2\mathrm{TKG}$ for reasoning over TKG. We first obtain the embedding of each historical entity based on the structural encoder. Then, a well-designed latent relations learning module is proposed to mine and encode the two types of latent relations, obtaining comprehensive entity embeddings. Finally, we extract temporal representations of entities from the outputs of LRL and SE for final prediction. Experimental results on four benchmarks and extensive analysis demonstrate the effectiveness and superiority of $\mathrm{L}^2\mathrm{TKG}$ in TKG reasoning.

Limitations

In this section, we talk about the limitations of our model. Specifically, selecting k values in the LRL module requires human involvement. Different types of data or entities may depend on different k values. Although most k values within a reasonable range result in model performance gains, finding the optimal value through human involvement in the selection alone is difficult. In the future, we will study the automatic optimization of the k values to further improve the model's ability to learn latent relations.

References

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Jie Chen, Haw-ren Fang, and Yousef Saad. 2009. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10(9).

Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling information loss of graph neural networks for session-based recommendation. *KDD*.

Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *NIPS*, pages 19314–19326.

Luca Cosmo, Anees Kazi, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. 2020. Latent patient network learning for automatic diagnosis.

Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *KDD*, pages 1585–1595.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, pages 1568–1577.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *ICML*, pages 1972–1982.

A García-Durán, Sebastijan Dumani, and M. Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*, pages 4816–4821.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021a. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *ICLR*.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021b. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *EMNLP*, pages 8352–8364.

Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. 2020. Graph hawkes neural network for forecasting on temporal knowledge graphs. In *AKBC*.

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*, pages 754–763.

Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *CVPR*, pages 11305–11312.

W. Jin, M. Qu, X. Jin, and X. Ren. 2020a. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *EMNLP*, pages 6669–6683.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020b. Graph structure learning for robust graph neural networks. In *KDD*, pages 66–74.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020c. Graph structure learning for robust graph neural networks. *KDD*.

Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. 2023. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pages 1–49. Citeseer.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018a. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018b. Adaptive graph convolutional neural networks. In *AAAI*.

Yujia Li, Shiliang Sun, and Jing Zhao. 2022. Tirgn: Time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In *IJCAI*, pages 2152–2158.

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*, pages 408–417.

Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. 2022. Towards unsupervised deep graph structure learning. In *WWW*, pages 1392–1403.

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*, page 1150–1160.

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*, volume 36, pages 5825–5833.

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*, pages 8024–8035.

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*, pages 593–607.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. TimeTraveler: Reinforcement learning for temporal knowledge graph forecasting. In *EMNLP*, pages 8306–8319.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *ICML*, pages 3462–3471.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080.

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*.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*.

Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. 2023. Temporal knowledge graph reasoning with historical contrastive learning. In *AAAI*.

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*.

Liang Yang, Zesheng Kang, Xiaochun Cao, Di Jin, Bo Yang, and Yuanfang Guo. 2019. Topology optimization based graph convolutional network. In *IJCAI*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. *AAAI*, 33(01):7370–7377.

Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. 2021. Mining latent structures for multimedia recommendation. *ACM MM*.

Yingxue Zhang, Soumyasundar Pal, Mark J. Coates, and Deniz Üstebay. 2019. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI*.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. *ACL*.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021a. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *AAAI*, pages 4732–4740.

Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2021b. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*.

A Dataset

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 (Li et al., 2021). The statistics of four TKG datasets are summarized in Table 3.

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

Table 3: The statistics of the datasets. Time gap represents time granularity between temporally adjacent facts.

B Baselines

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

DisMult (Yang et al., 2015), a model that proposes a simplified bilinear formulation to capture relational semantics.

ComplEx (Trouillon et al., 2016), a model that converts the embedding into complex vector space to handle symmetric and antisymmetric relations.

R-GCN (Schlichtkrull et al., 2018), a graph neural network that handles the highly multi-relational graph data.

ConvE (Dettmers et al., 2018), a model that adopts a 2D convolutional neural network to model the interactions between entities and relations.

RotatE (Sun et al., 2019), a model that defines each relation as a rotation from the subject entity to object entity in the complex vector space.

TKG reasoning models compared to L^2TKG include:

CyGNet[1] (Zhu et al., 2021a), a model that utilizes recurrence patterns in historical facts to predict future facts.

RE-NET[2] (Jin et al., 2020a), a model that adopts RNN to capture the historical dependencies of each query and RGCN to model the structural dependencies of each entity.

RE-GCN[3] (Li et al., 2021), a recurrent evolution network model that learns the evolution of entities and relations based on the results of a relational graph neural network. Moreover, the static properties of entities are also incorporated via a static graph module (Since other models do not utilize additional information, we remove the static properties in RE-GCN to ensure the fairness of comparisons among models).

xERTE[4] (Han et al., 2021a), an explainable model that designs a temporal relational attention mechanism to extract sub-graphs around the query.

TITer[5] (Sun et al., 2021), a reinforcement learning-based model, which includes a time-shaped reward based on Dirichlet distribution to guide the model training.

TiRCN[6] (Li et al., 2022), a model that utilizes a local recurrent graph encoder network to capture the historical dependency of events at adjacent timestamps and uses the global history encoder network to collect repeated historical facts (We also remove the static properties to ensure the fairness of comparisons among models).

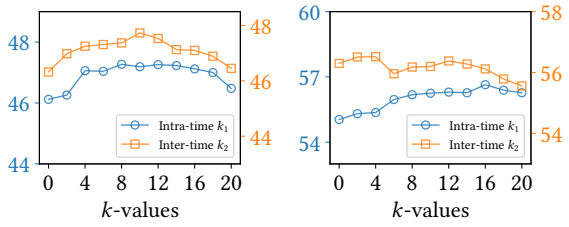
CENET[7] (Xu et al., 2023), a model based on contrastive learning that learns both the historical and non-historical dependencies to distinguish the most potential entities.

C Implementation Deatils

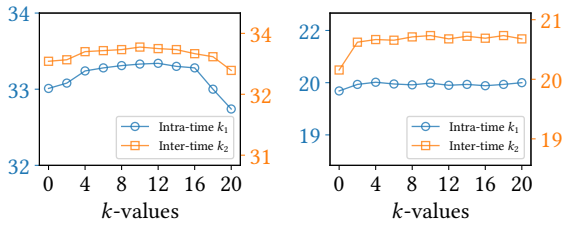
We implement our L^2TKG in **Pytorch** (Paszke et al., 2019) and **DGL** Library (Wang et al., 2019). We use Adam optimizer (Kingma and Ba, 2015) with learning rate set to 0.001 and l_2 regularization λ_2 set to 10^{-5}. The embedding size is fixed to 200 for all methods. For the L^2TKG hyper-parameters, we apply a grid search on the validation set: the k_1 and k_2 values are searched in $\{2, 4, ..., 20\}$, the SE layer number ω and LRL layer number β in $\{1, 2, 3, 4\}$, and the length of local temporal representation m in $\{1, 2, \cdots, 10\}$.

For ICEWS14, ICEWS05-15, ICEWS18, and GDELT, the optimal k_1 values are 8, 10, 6, and 6. The optimal k_2 values are 10, 10, 6, and 8. The optimal LRL layer number β are 2, 2, 1, and 2. The optimal length of local temporal representation m for are 3, 5, 6, and 1, respectively. The optimal SE layer number ω is 2 for all datasets. For the SE, we set the block dimension to 2×2 and the dropout rate for each layer to 0.2. For the ConvTransE of the score function, the number of kernels, kernel

[1] https://github.com/CunchaoZ/CyGNet
[2] https://github.com/INK-USC/RE-Net
[3] https://github.com/Lee-zix/RE-GCN
[4] https://github.com/TemporalKGTeam/xERTE
[5] https://github.com/JHL-HUST/TITer
[6] https://github.com/Liyyy2122/TiRGN
[7] https://github.com/xyjigsaw/CENET

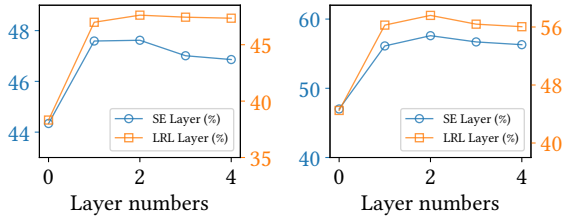


(a) ICEWS14 (b) ICEWS05-15

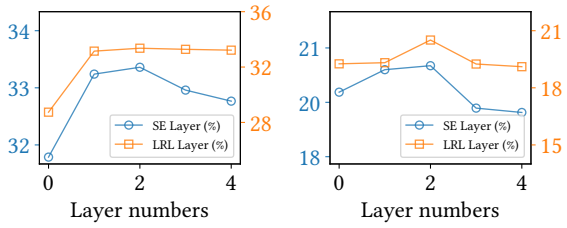


(c) ICEWS18 (d) GDELT

Figure 5: Performance of L^{2}TKG under different k-values in terms of MRR (%).



(a) ICEWS14 (b) ICEWS05-15



(c) ICEWS18 (d) GDELT

Figure 6: Performance of L^{2}TKG under different layer numbers of SE and LRL in terms of MRR (%).

size, and the dropout rate are set to 50, 2×3, and 0.2, respectively.

To improve the efficiency of L^{2}TKG while ensuring the performance, we properly process historical TKG data when predicting query $(e_s, r, ?, t+1)$. Specifically, we only use the historical KG sequence in which e_s has appeared for the learning of latent relations. For example, entity e_s has appeared at time t_1, t_2, and t_3, where $t_3 < t+1$. Then we input the representations of entities in $\{\mathcal{G}_{t_1}, \mathcal{G}_{t_2}, \mathcal{G}_{t_3}\}$ into the LRL module to mine and exploit important latent relations.

For the compared methods, we use the default hyper-parameters except for dimensions. We run the evaluation five times with different random seeds and report the mean value of each method. All experiments are conducted on NVIDIA Tesla V100 (32G) and Intel Xeon E5-2660.

D Sensitivity Analysis (RQ4)

The structural encoder (SE) and latent relation learning (LRL) are two vital modules in our model. This section studies how hyper-parameters, the k value of the sparse operations (Intra-time and Inter-time learning), and the layer numbers of LRL and SE affect the performance of L^{2}TKG.

D.1 Effect of k Values in LRL

The values of k_1 and k_2 determine the number of newly learned intra-time and inter-time latent relations. Figure 5 shows the model performance

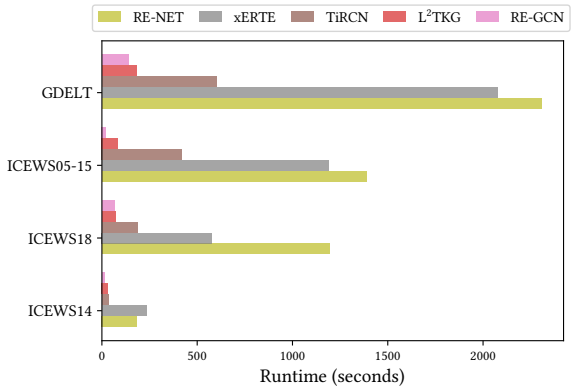


Figure 7: Runtime (seconds) comparison to some baselines.

under varied k_1 and k_2 values, respectively. When adjusting one k_i value, the other k_i uses the optimal value. $k_i = 0$ means that our model does not consider the corresponding types of latent relations learning.

From the results, we can find that the model performance improves at the initial stage where the two k values increase, which verifies that the two latent relations can provide more effective information for TKG reasoning. However, when k continues to increase, the trend will decrease. The reason might be that many unimportant latent relations are introduced as noise to interfere with the model. This demonstrates the necessity of k-NN sparsification in the LRL module.

D.2 Effect of LRL Layer Nubmer β

The number of layers in LRL decides the degree of utilizing the latent relations. In this part, we conduct our model when the LRL layer number β is in the range of $\{0, 1, 2, 3, 4\}$. The results are shown in Figure 6 (yellow line). We can find our method achieves significant improvement between $\beta = 0$ and $\beta > 0$, which validates the rationality of mining the latent associations in TKG reasoning. When further stacking the LRL layer, the performance of $\mathrm{L^2TKG}$ begins to deteriorate, which is probably because the LRL suffers from the over-smoothing problem (Li et al., 2018a).

D.3 Effect of SE Layer Number ω

The number of layers in SE determines the degree of modeling semantic dependencies among concurrent facts. We also set the SE layer number ω in the range of $\{0, 1, 2, 3, 4\}$ and conduct our model. From the results in Figure 6 (blue line), we can find that our model achieves the best performance when $\omega = 2$ and significantly outperforms the value at $\omega = 0$, which demonstrates that utilizing the high-order neighbor information in concurrent entities can enhance the semantic representations of entities at each timestamp. As the number of layers further increases ($\omega > 2$), the model's performance begins to decline, which may be because the use of higher-order information makes it easy to introduce noise and lead to over-smoothing.

E Efficiency

To investigate the efficiency of our model, we compare $\mathrm{L^2TKG}$ with RE-GCN, TiRCN, xERTE, and RE-NET in terms of inference time on the test set. From Figure 7, we can find that although our $\mathrm{L^2TKG}$ mines and exploits many important latent relations from historical entities, the inference speed is still higher than TiRCN, xERTE, and RE-NET. We attribute this efficiency to the sparsification operations of LRL (§4.2) and proper data processing (Appendix C). Besides, $\mathrm{L^2TKG}$ is mainly based on the GNN model that can perform parallel computation, thus ensuring a better balance of performance and efficiency.