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
State Key Laboratory of Multimodal Artificial Intelligence Systems
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. However, due to the limitations in 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. We then 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 graphs (TKGs) play a vital role in capturing temporal facts in the real world. Each fact in a TKG is represented as a quadruple (s, r, o, t), such as (Obama, run for, president, 2012). Reasoning over TKGs involves two primary settings: interpolation and extrapolation. In recent years, there has been significant interest in the extrapolation setting due to its practical value in event prediction (Deng et al., 2020), question answering (Mavromatis et al., 2022), and other applications. In the extrapolation setting, the objective is to predict facts that occur at a time t with $t > t_n$, based on the historical information available in the TKG from t_0 to t_n.

*To whom correspondence should be addressed.

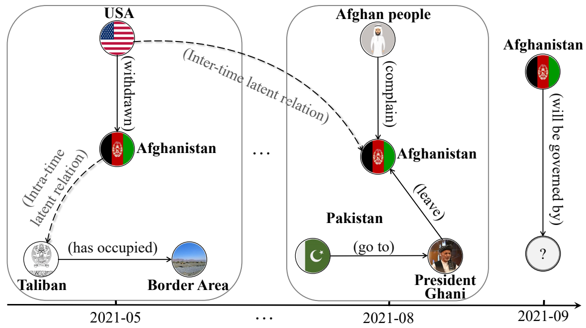


Figure 1: An example of reasoning over TKG. The gray dotted lines indicate two types of latent relations

Most extrapolation models 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. Additionally, 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.

While these methods demonstrate promising results in TKG reasoning, they still face the challenge of missing associations within TKGs. Specifically, the majority of TKG data is automatically identified and extracted from diverse 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 these missing associations as *latent relations* between entities. Existing approaches fail to explicitly discover and utilize these latent relations, which manifest primarily in two aspects.

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

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 12617–12631

tities. During TKG reasoning, certain concurrent entities may lack direct connections but exhibit strong semantic correlations. Figure 1 illustrates this phenomenon, where *Afghanistan* and *Taliban* are not connected in the TKG for May 2021. However, in reality, the *Taliban* was involved in negotiations with *Afghanistan* during that time, significantly impacting the situation in *Afghanistan*. Hence, it is crucial to model the critical latent relations among concurrent entities. Most existing TKG reasoning models rely on 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 heavily depend on existing edges or associations, making it challenging to model critical semantic dependencies among indirectly connected entities.

Secondly, existing methods ignore the *inter-time latent relations* between entities appearing at different timestamps. Some entities at various timestamps can exhibit strong semantic dependencies, providing essential auxiliary information for TKG reasoning. Therefore, it is necessary to consider the associations between these entities. Taking Figure 1 as an example, the impact of the *USA* in May 2021 on *Afghanistan* in August 2021 is significant. However, as these two entities appear at different times, they cannot be directly related in the TKG. Existing TKG reasoning models primarily focus on modeling the semantic dependencies of the same entities at different times but fall short when addressing entities at distinct timestamps.

To address the aforementioned challenges, we propose a novel Latent relations Learning method for TKG reasoning, L^{2}TKG for brevity. The overall framework of L^{2}TKG is presented in Figure 2. Specifically, we first employ 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. Utilizing the embeddings of entities at each timestamp, LRL enables the creation of new crucial associations between unconnected entities in a learnable manner and then encodes the learned latent relational graph to obtain more comprehensive representations of entities. Finally, we extract temporal representations from the output of SE and LRL components for the entity prediction task.

In summary, our work makes the following main contributions:

- We emphasize and investigate 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 predicts the future based on repeating patterns in historical facts.

Some recent methods (Jin et al., 2020a; Li et al., 2021, 2022) combine graph neural networks (GNNs) and recurrent neural networks (RNNs) to model the semantic and temporal dependencies between entities. For instance, RE-NET (Jin et al., 2020a) incorporates RNNs and RGCNs to capture the temporal and structural dependencies from entity sequences. RE-GCN (Li et al., 2021) considers adjacent structural dependencies of entities while introducing static properties of entities. To incorporate global temporal information, TiRCN (Li et al., 2022) designs a global history encoder network that collects repeated historical facts. HGLS (Zhang et al., 2023b) designs a Hierarchical Graph Neural Network to explicitly encode long-term temporal information. Furthermore, TANGO (Han et al., 2021b) employs Neural Ordinary Differential Equations for fine-grained temporal information in TKG reasoning, specifically for forecasting future links. Additionally, some works (Han et al., 2021a; Sun et al., 2021) propose sub-graph or path search strategies for TKG reasoning. xERTE (Han et al., 2021a) designs an explainable model for entity prediction, utilizing a sub-graph search strategy to identify answer entities. TITer (Sun et al., 2021) performs a path search based on reinforcement learning to predict future entities, incorporating a time-shaped reward using the Dirichlet distribution for guiding model training. Recently, MetaTKG (Xia et al., 2022) proposes a temporal meta-learner to learn evolution patterns of facts. CENET (Xu et al., 2023) combines the contrastive learning strategy with TKG models to identify significant entities from historical and nonhistorical dependency. However, all of these above methods rely on existing associations between entities or structures in TKG and disregard the utilization of important latent associations between entities.

2.2 Graph Structure Learning

Graph Neural Networks (GNNs) have gained significant attention for their ability to handle graph-structured data and have shown promising performance in various tasks, including Recommender Systems (Wu et al., 2019; Chen and Wong, 2020; Zhang et al., 2021, 2020a, 2023a) and Natural Language Processing (Yao et al., 2019; Zhang et al., 2020b). However, it has been observed that graph data can contain noise, which can negatively impact the training of GNNs (Jin et al., 2020c). To address this issue, researchers have proposed graph structure learning (GSL) methods, which aim to jointly learn an optimized graph structure and node representations.

GSL models can be categorized into three main categories (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, PTDNet (Luo et al., 2021) proposes a parameterized topological denoising network to improve the robustness and generalization performance of GNNs by learning to drop task-irrelevant edges. LDS (Franceschi et al., 2019) introduces a method for simultaneous learning of graph structure and graph convolutional network parameters. This is achieved by solving an approximate bilevel program that determines a discrete probability distribution on the graph edges. 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.

Drawing inspiration from GSL approaches, our work centers on metric-learning-based methods. The goal is to discover new and important missing associations within TKG data while obtaining 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). This paper primarily emphasizes the *entity prediction* task within TKG reasoning. The objective of the *entity prediction* task is to forecast the missing object entity of $(e_s, r, ?, t+1)$ or the missing subject entity of $(?, r, e_o, t+1)$ based on the historical KG sequence $\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_t\}$.

Let $\mathbf{x}_s \in \mathbb{R}^d$ and $\mathbf{x}_r \in \mathbb{R}^d$ denote the static embedding of entity e_s and relation r, where d represents 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 $\{\mathcal{G}_i\}_{i=0}^t$, along with static entity and relation embeddings $\mathbf{x}_s$ and $\mathbf{x}_r$. The embeddings $\mathbf{x}_s$ and $\mathbf{x}_r$ serve as learnable parameters.

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 among concurrent entities at each timestamp using the existing TKG structure.

(2) *Latent Relations Learning* (LRL), which mines and exploits 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.

4.1 Structural Encoder

At each timestamp, there exist strong semantic dependencies among connected co-occurring entities. 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, the entity representation $\mathbf{h}_{s,t_i}^{\omega}$ at time t_i is obtained. We denote the embedding of e_s at time t_i as $\mathbf{h}_{s,t_i}$, omitting the superscript ω.

4.2 Latent Relations Learning

After capturing the semantic dependencies among concurrent entities at each timestamp, we introduce a latent relations learning module to identify and leverage significant missing associations: *intra-time latent relations* and *inter-time latent relations*, 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 $\mathcal{G} = \{\mathcal{G}_{t-L}, \cdots, \mathcal{G}_t\}$. In this context, the same entity appearing at different times is treated as distinct entities, such as $e_s^{t_i}$ and $e_s^{t_j}$. Consequently, the number of entities considered in this module is $N = \sum_{t_k=t-L}^{t} n_{t_k}$, where n_{t_k} represents the number of entities in $\mathcal{G}_{t_k}$, and L represents the length of historical sequence.

Assuming no loss of generality, we posit that highly associated entities also exhibit similarity within 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

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

where $(e_i, e_j) \in \mathcal{G}_{t_p}$ and $(e_i, r, e_j, t_p) \notin \mathcal{G}_{t_p}$, for all $r \in \mathcal{R}$. 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:

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

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_p.

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

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

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:

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

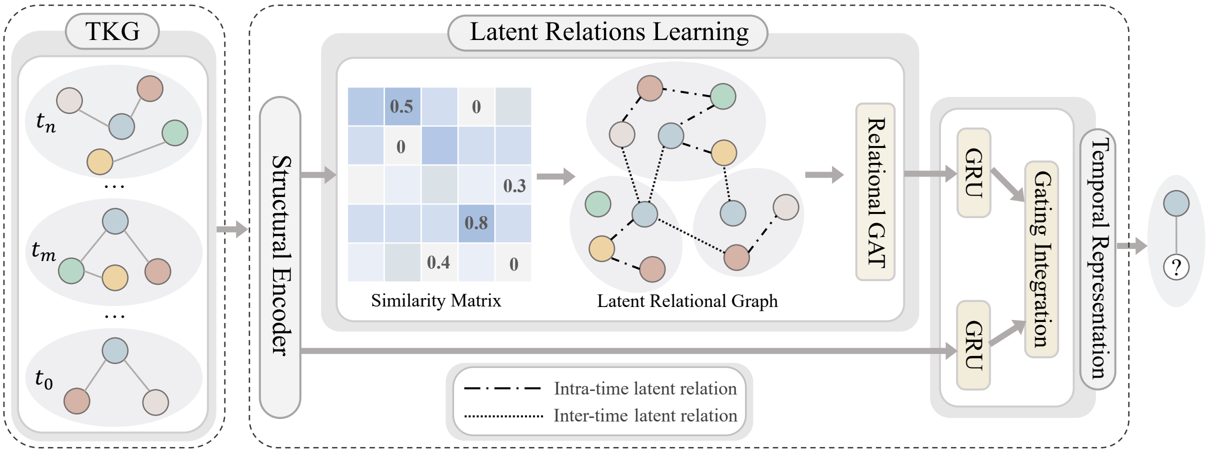


Figure 2: An illustration of L^{2}TKG model architecture. We first utilize a *Structural Encoder* (§4.1) to obtain representations of entities at each timestamp. Then, the well-designed *Latent Relations Learning* (§4.2) module sequentially calculates the similarity matrix, and constructs and encodes 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).

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

We independently choose Top-k values for the sparse operations in learning the two types of latent relations, denoted as k_1 and k_2, respectively. Based on the acquired similarity matrices, we proceed to construct a latent relational graph denoted as $\mathcal{P}$. Specifically, if $\hat{\mathbf{S}}^{t_p}_{i,j} > 0$, we construct an intra-time latent relation between $e_i^{t_p}$ and $e_j^{t_p}$ within $\mathcal{P}$. Similarly, if $\hat{\mathbf{Q}}^{t_p,t_q}_{i,j} > 0$, we construct an inter-time latent relation between $e_i^{t_p}$ and $e_j^{t_q}$ within $\mathcal{P}$. In this graph $\mathcal{P}$, we solely consider latent relations and omit original relations of the TKG sequence. Furthermore, similar to existing relations, we transform the two types of latent relations into low-dimensional embedding vectors, which serve as learnable parameters. To facilitate presentation, we directly employ numerical numbers $\{1, ..., N\}$ to represent the nodes in $\mathcal{P}$ in the subsequent 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:

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

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

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), as well as TKG models such as 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 results under the time-aware filtered setting for all compared models.

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*	<u>43.18</u>	<u>33.12</u>	<u>62.24</u>	<u>48.83</u>	<u>38.62</u>	<u>69.20</u>	<u>32.22</u>	**22.24**	<u>51.88</u>	**21.67**	**13.63**	**37.60**
$\mathrm{L}^2\mathrm{TKG}$	**47.40**	**35.36**	**71.05**	**57.43**	**41.86**	**80.69**	**33.36**	<u>22.15</u>	**55.04**	<u>20.53</u>	<u>12.89</u>	<u>35.83</u>
$\Delta Improve.$	9.77%	6.73%	14.15%	17.61%	8.39%	16.60%	3.54%	–	6.09%	–	–	–

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.

5.2 Performance Comparison (RQ1)

The performance of all models on the entity prediction task is presented in Table 1. Based on the results, we made the following observations:

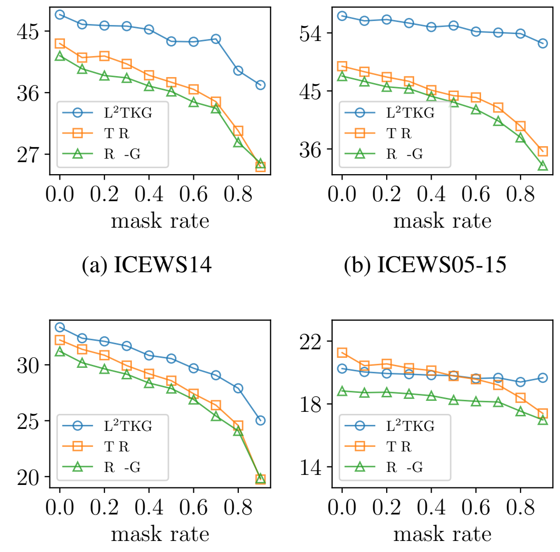
$\mathrm{L}^2\mathrm{TKG}$ achieves the best performance on all ICEWS datasets with most evaluation metrics, which verifies the effectiveness of our model. Specifically, $\mathrm{L}^2\mathrm{TKG}$ significantly outperforms all compared static models, demonstrating the importance of modeling temporal 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 $\mathrm{L}^2\mathrm{TKG}$ 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 $\mathrm{L}^2\mathrm{TKG}$ 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. In contrast to our model, xERTE and TITer employ sub-graph-based search and path-based search, respectively, for target entity prediction. However, their search methods are constrained by existing paths, limiting their search scope and compromising their performance.

In the case of the GDELT data, it contains a higher number of facts at each time, and the issue of missing associations is less severe. Consequently, our model exhibits limited improvement compared to state-of-the-art models.

5.3 Performance Comparison in Learning Missing Associations (RQ2)

To further validate the capacity of $\mathrm{L}^2\mathrm{TKG}$ in discovering and leveraging latent relations, we evaluate its performance on datasets with different levels of missing associations. On the ICEWS and GDELT datasets, we mask a range of $\{0.1, ..., 0.9\}$ of the existing relations in the knowledge graph for each timestamp. Figure 3 presents the performance comparison of RE-GCN, TiRCN, and $\mathrm{L}^2\mathrm{TKG}$ using various mask ratios, while Figure 4 illustrates the relative improvements of $\mathrm{L}^2\mathrm{TKG}$ over RE-GCN and TiRCN. From the results, we have the following observations:

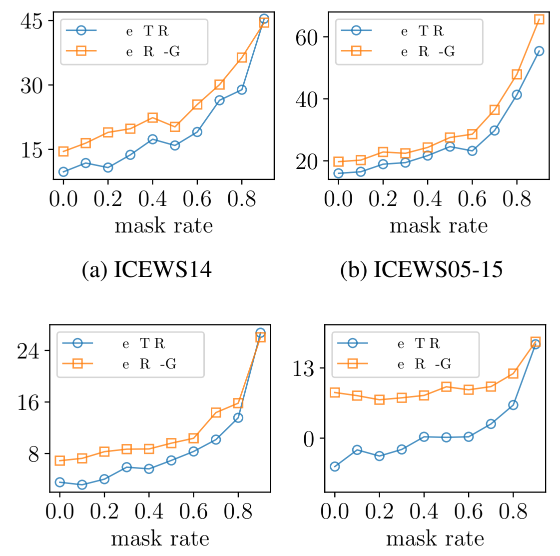
From Figure 3 we find that the performance of all models decreases to different degrees as the mask rate increases, which is due to the gradual decrease of historical association information in the dataset. Nonetheless, our model suffers a relatively small drop in performance and maintains satisfactory results even when faced with significant missing association information (mask rate > 0.6). In Figure 4, the relative 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 associa-



(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.

tions 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^{2}TKG with different variants in terms of MRR. Specifically, we modify L^{2}TKG by removing the latent relation learning module (w/o LRL), intra-time relation learning of LRL (w/o LRL-Intra), inter-time relation learning

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^{2}TKG	**47.40**	**57.43**	**33.36**	**20.53**

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

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^{2}TKG significantly outperforms L^{2}TKG (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^{2}TKG (w/o LRL-Intra) and L^{2}TKG (w/o LRL-Inter) also achieves better performance than L^{2}TKG (w/o LRL). The improvements verify that both learned inter-time and intra-time latent relations contribute to model performance. Compared with L^{2}TKG (w/o LRL-Intra) and (w/o LRL-Inter), the performance of L^{2}TKG 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.

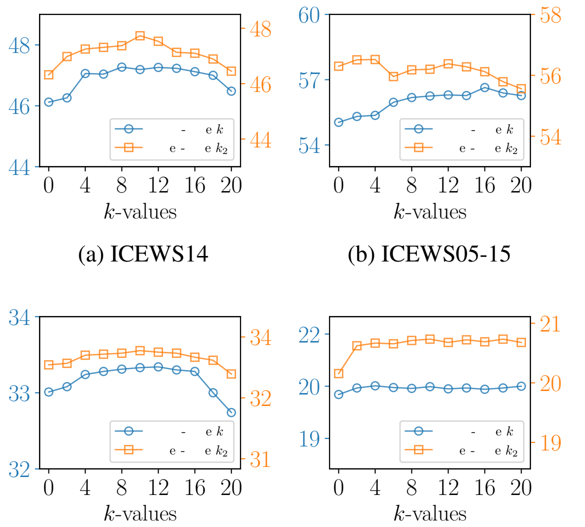
L^{2}TKG also obtains significant improvements over L^{2}TKG (w/o Ltr) and L^{2}TKG (w/o Gtr), indicating that both global- and local-temporal information can effectively enhance the performance on the prediction task. The improvement between L^{2}TKG and L^{2}TKG (w/o SE) verifies the importance of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

6 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.

6.1 Effect of k Values in LRL

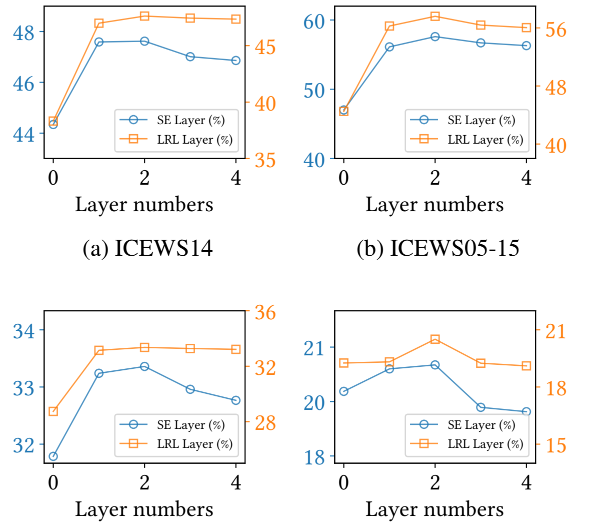
The values of k_1 and k_2 determine the number of newly learned intra-time and inter-time latent rela-



(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 (%).

tions, respectively. Figure 5 illustrates the performance of model for different values of k_1 and k_2. When adjusting one k_i value, the other k_i utilizes the optimal value. A k_i value of 0 indicates that our model does not consider the corresponding types of latent relation learning.

From the results, we can find that the performance of L^{2}TKG improves initially as the two k values increase. This finding confirms that the two latent relations can provide more effective information for TKG reasoning. However, as k continues to increase, the trend begins to decline. This decline could be attributed to the introduction of numerous unimportant latent relations that act as noise, thereby interfering with the model. This demonstrates the necessity of employing k-NN sparsification in the LRL module.

6.2 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 L^{2}TKG begins to deteriorate, which is probably because the LRL suffers from the over-smoothing problem (Li et al., 2018a).

6.3 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.

7 Conclusion

In this paper, we have proposed a novel method L^{2}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 L^{2}TKG in TKG reasoning.

Limitations

In this section, we discuss the limitations of our model. Specifically, the selection of k values in the LRL module necessitates human involvement. Various types of data or entities may rely on distinct k values. While the majority of k values within a reasonable range lead to improvements in model performance, identifying the optimal value solely through human involvement poses challenges. Moving forward, we will investigate the automatic optimization of k values to enhance the model's capacity for acquiring latent relations.

Acknowledgement

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

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*.

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.

Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust graph neural network via topological denoising. In *WSDM*, pages 779–787.

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*.

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*, pages 7230–7240.

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*.

Mengqi Zhang, Shu Wu, Meng Gao, Xin Jiang, Ke Xu, and Liang Wang. 2020a. Personalized graph neural networks with attention mechanism for session-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3946–3957.

Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. 2023a. Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4741–4753.

Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. 2023b. Learning long-and short-term representations for temporal knowledge graph reasoning. In *WWW*, pages 2412–2422.

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. 2020b. 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.

B Baselines

The comparison of static KG reasoning models with our work is presented as follows:

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

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

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

ConvE (Dettmers et al., 2018) is 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.

The temporal KG reasoning models compared to our model are:

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

RE-NET[2] (Jin et al., 2020a) is a model that adopts RNN and RGCNs to capture the temporal and structural dependencies from entity sequences.

RE-GCN[3] (Li et al., 2021) is a Recurrent Evolution network based on Graph Convolution Network (GCN), which learns the evolutional representations of entities and relations at each timestamp by modeling the KG sequence recurrently. It also incorporates the static properties of entities through a static graph module. However, to ensure fairness in comparisons among models, we remove the static properties in RE-GCN, as other models do not utilize additional information.

xERTE[4] (Han et al., 2021a) is an explainable model that designs a sub-graph search strategy to identify answer entities.

TITer[5] (Sun et al., 2021) is a reinforcement learning-based model, which performs a path search to predict future entities.

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

CENET[7] (Xu et al., 2023) is 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^{2}TKG 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^{2}TKG 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.

[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

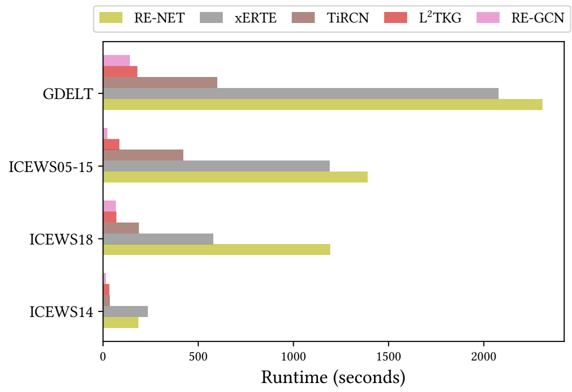


Figure 7: Runtime (seconds) comparison to some baselines.

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

To enhance the efficiency of L^{2}TKG while maintaining performance, we appropriately preprocess historical TKG data when predicting queries of the form $(e_s, r, ?, t+1)$. 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 Efficiency

To examine the efficiency of our model, we compared L^{2}TKG with RE-GCN, TiRCN, xERTE, and RE-NET in terms of inference time on the test set. As shown in Figure 7, despite the fact that our L^{2}TKG uncovers and leverages numerous significant latent relations from historical entities, its inference speed surpasses that of TiRCN, xERTE, and RE-NET. We attribute this efficiency to the sparsification operations of LRL (§4.2) and the appropriate processing of data (Appendix C). Moreover, the fundamental components of L^{2}TKG primarily rely on the GNN model, which enables parallel computation, thus ensuring a more optimal balance between performance and efficiency.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitation

☑ A2. Did you discuss any potential risks of your work?
There are no potential risks in our paper.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1(Introduction)

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 2,3,4

☐ B1. Did you cite the creators of artifacts you used?
No response.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 5

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
The use of existing artifacts in our paper is only for research.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The use of existing artifacts in our work is only for research.

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
The documentation of the artifacts is not the focus of our research.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5.1.1 and Appendix A

C ☑ Did you run computational experiments?

Section 5, 6, Appendix C and D.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Yes, Appendix C reports the computing infrastructure used.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 6 and Appendix C.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Appendix C

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix C

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.