Artificial Intelligence 329 (2024) 104085

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

Temporal inductive path neural network for temporal knowledge graph reasoning

Hao Dong [a,b,1], Pengyang Wang [c,*], Meng Xiao [a,b], Zhiyuan Ning [a,b], Pengfei Wang [a,b,*], Yuanchun Zhou [a,b]

[a] Computer Network Information Center, Chinese Academy of Sciences, 2 Dongsheng South Rd, Haidian District, Beijing, China
[b] University of Chinese Academy of Sciences, 1 Yanqihu East Rd, Huairou District, Beijing, China
[c] Department of Computer and Information Science, The State Key Laboratory of Internet of Things for Smart City, University of Macau, Avenida da Universidade, Taipa, Macau, China

ARTICLE INFO

Keywords:
Temporal knowledge graph
Temporal reasoning
Graph neural networks
Knowledge graph reasoning

ABSTRACT

Temporal Knowledge Graph (TKG) is an extension of traditional Knowledge Graph (KG) that incorporates the dimension of time. Reasoning on TKGs is a crucial task that aims to predict future facts based on historical occurrences. The key challenge lies in uncovering structural dependencies within historical subgraphs and temporal patterns. Most existing approaches model TKGs relying on entity modeling, as nodes in the graph play a crucial role in knowledge representation. However, the real-world scenario often involves an extensive number of entities, with new entities emerging over time. This makes it challenging for entity-dependent methods to cope with extensive volumes of entities, and effectively handling newly emerging entities also becomes a significant challenge. Therefore, we propose **T**emporal **I**nductive **P**ath **N**eural **N**etwork (TiPNN), which models historical information in an entity-independent perspective. Specifically, TiPNN adopts a unified graph, namely history temporal graph, to comprehensively capture and encapsulate information from history. Subsequently, we utilize the defined query-aware temporal paths on a history temporal graph to model historical path information related to queries for reasoning. Extensive experiments illustrate that the proposed model not only attains significant performance enhancements but also handles inductive settings, while additionally facilitating the provision of reasoning evidence through history temporal graphs.

1. Introduction

Knowledge Graphs (KGs) are powerful representations of structured information that capture a vast array of real-world facts. They consist of nodes, which represent entities such as people, places, and events, and edges that define the relationships between these entities. These relationships are typically represented as triples in the form of (s, r, o), comprising a subject, a relation, and an object, such as *(Tesla, Founded by, Elon Musk)*. Temporal Knowledge Graphs (TKGs) is an extension of a traditional KGs that incorporates the dimension of time. In a TKG, these facts are annotated with timestamps or time intervals to indicate when they are or were

* Corresponding authors.
E-mail addresses: donghcn@gmail.com (H. Dong), pywang@um.edu.mo (P. Wang), shaow@cnic.cn (M. Xiao), ningzhiyuan@cnic.cn (Z. Ning), pfwang@cnic.cn (P. Wang), zyc@cnic.cn (Y. Zhou).
[1] This work was done when the first-author was the research assistant in IOTSC at University of Macau.

https://doi.org/10.1016/j.artint.2024.104085
Received 23 September 2023; Received in revised form 29 November 2023; Accepted 23 January 2024
Available online 1 February 2024

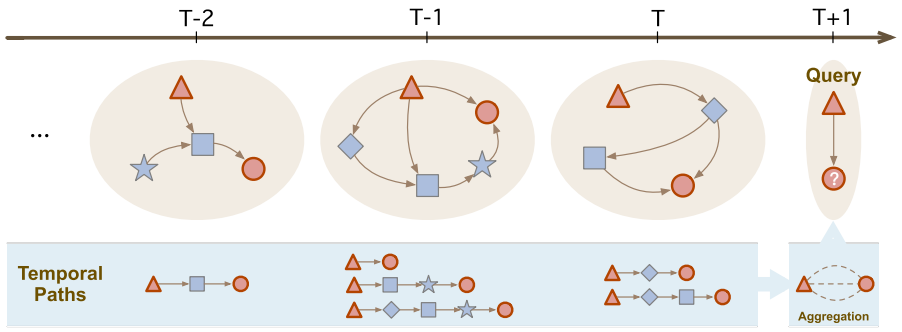

Fig. 1. An example of inference based on temporal paths modeling. The red triangular nodes and red circular nodes represent the subject entity and candidate object entity of the query at timestamp $T+1$, respectively. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

valid. Different from KGs, each fact in a TKG is represented as a quadruple, adding a timestamp feature, which can be denoted as (s, r, o, t), including a subject, a relation, an object, and a timestamp, such as *(Albert Einstein, Born in, Germany, 03/14/1879)*. The timestamp indicates the specific date or time when this event occurred, providing temporal information to the graph. Such unique representation empowers TKGs to capture the dynamics of multi-relational graphs over time, exhibiting temporal variations. As a result, TKGs have found widespread applications in diverse domains, like social network analysis [1,2], and event prediction [3–5] (e.g. disease outbreaks, natural disasters, and political shifts). These applications leverage the temporal information embedded in TKGs to make informed decisions and predictions based on historical knowledge.

Among various tasks, TKG reasoning focuses on inferring missing facts based on known ones and consists of two primary settings: interpolation and extrapolation [6]. Given the timestamp scope $[0, T]$ in a TKG, where T represents the maximum timestamp, in the interpolation setting, it focuses on the time range from 0 to T, i.e., completing missing facts within the range of past timestamps [7]. On the other hand, in the extrapolation setting, the reasoning timestamps extend beyond T, focusing on predicting the facts in the future [8,9,24]. This study particularly emphasizes the extrapolation setting.

Existing literature on TKG reasoning mainly focuses on exploring temporal features and structural information to capture complicated variations and logical connections between entities and relations. For example, RE-NET [6] and RE-GCN [8] are representative works that leverage RNN-based and GNN-based approaches to learn temporal and structural features of entities in TKGs. xERTE [10] utilizes the entity connection patterns from a constructed inference graph. These methods rely on entity modeling as nodes in the graph play a crucial role in knowledge representation. However, the real-world scenario often involves an extensive number of entities, with new entities emerging over time. This makes it challenging for entity-dependent methods to cope with extensive volumes of entities, and effectively handling newly emerging entities also becomes a significant challenge.

To address this, we previously introduced an innovative *entity-independent* modeling approach for TKGs with an evolutionary perspective in our previous work, referred to as DaeMon [9]. The entity-independent modeling approach brings several benefits: (i) modeling on graph representation is independent of entities, maintaining stable performance even for datasets with numerous entities. Besides, the memory occupation is independent of the number of entity categories. (ii) the entity-independent nature makes it insensitive to entity-specific features. Hence, when new entities emerge in future timestamps, the model can still handle these unseen nodes, showcasing robust generalization and reasoning capabilities. In essence, DaeMon focuses on learning query-aware latent path representations, incorporating a passing strategy between adjacent timestamps to control the path information flow toward later ones without explicitly modeling individual entities. Specifically, DaeMon introduces the concept of temporal paths to represent the logical paths between a query's subject entity and candidate object entities within the sequence of historical subgraphs. The model employs a path aggregation unit to capture local path information within each subgraph and utilizes path memory units to store aggregated representations of temporal paths along the timeline. Finally, DaeMon adopts a memory passing strategy to facilitate cross-temporal information propagation.

Here, we provide a generalized overview of inference based on temporal paths modeling, as illustrated in Fig. 1. For a given inference query, it starts from this query and collects the path between the subject entity and object entity in the historical context. Then, it utilizes a query-aware temporal method to represent the aggregation of temporal paths with temporal information for reasoning. For example, as shown in Fig. 1, it iteratively learns the path information between the red triangular and red circular nodes within the historical context. The aggregation of path information corresponding to the query is then utilized to infer the possibility of potential future interactions. Note that, during the learning of path information, the method does not focus on which intermediary nodes (i.e., the blue nodes in Fig. 1) are present in the path, but rather employs the relation representations present in the path to express it, showcasing an entity-independent characteristic.

While DaeMon benefits from this entity-independent modeling approach, it still exhibits certain limitations. First, during the learning of historical information, DaeMon independently performs graph learning on local subgraphs and then connects them through memory passing strategy to obtain the final query-aware path features for future fact reasoning. This process requires graph learning on each subgraph separately, leading to difficulties in modeling complex temporal characteristics and a significant increase in time complexity as the historical length grows. Second, the memory passing strategy is adopted to model temporal features in historical sequences, controlling the information flow from one timestamp to the next to achieve temporal evolution modeling. However, this temporal modeling approach is unidirectional, which not only gives rise to the long-term dependency problem but

Table 1
Notations and descriptions.

Notations	Descriptions
$\mathcal{G}$	a temporal knowledge graph
G_t	a subgraph corresponding to the snapshot at timestamp t
$\mathcal{V}, \mathcal{R}, \mathcal{T}$	the finite set of entities, relation types and timestamps in the temporal knowledge graph
$\mathcal{E}_t$	the set of fact edges at timestamp t
(s, r, o, t)	a quadruple (fact edge) at timestamp t
$(s, r, ?, t+1)$	a query with the missing object that is to be predicted at timestamp $t+1$
$\hat{G}$	a history temporal graph
r_τ	a temporal relation in history temporal graph
(s, r_τ, o)	a temporal edge with time attribute τ attached in history temporal graph
m	the length of the history used for reasoning
ω	the number of temporal path aggregation layers
$\mathbf{H}$, $\mathbf{h}$	the representation of temporal paths (and path)
$\mathbf{R}$	a learnable representation of relation types in $\mathcal{R}$
$\mathbf{\Psi}_r$	the query relation r-aware basic static embedding of temporal edge
$\mathbf{\Upsilon}$	the temporal embedding of temporal edge
$\mathbf{w}_r$	the query-aware temporal representation of a temporal edge
d	the dimension of embedding

also neglects relative temporal features between timestamps. Third, DaeMon defines temporal paths as virtually connected paths across the sequence of subgraphs. In practice, it independently learns several edges on each timestamp's subgraph and then obtains the path feature by fusing the edge information through memory passing strategy. The independent learning approach without real physical paths creates challenges in providing relevant reasoning evidence during the inference stage, which makes it difficult to present interpretable explanations to users.

To overcome the above challenges, here we propose a Temporal Inductive Path Neural Network (TiPNN). Specifically, we introduce the concept of ***history temporal graphs*** as a replacement for the original historical subgraph sequences to mitigate the complexity of learning historical information independently on multiple subgraphs. It allows us to model historical information more efficiently by combining it into a single unified graph. Furthermore, we redefine the notion of ***temporal paths*** as a logical semantics path between the query entity and candidate entities in history temporal graph and design ***query-aware temporal path processing*** to model the path feature on the single unified graph in the entity-independent manner for the final reasoning. It enables us to comprehensively learn both the temporal and structural features present in the history temporal graph. The adoption of the entity-independent approach allows the model to effectively handle the inductive setting on the history temporal graph. By jointly modeling the information within and between different timestamps in a single unified graph, it can capture a more intuitive and holistic view of the temporal reasoning logic, thus facilitating user understanding of the model's reasoning process and can even provide interpretable explanations for the reasoning results. Our main contributions are as follows:

- To holistically capture historical connectivity patterns, we introduce a unified graph, namely the history temporal graph, to retain complete features from the historical context. The history temporal graph integrates the relationship of entities, timestamps of historical facts, and temporal characteristics among historical facts.
- We define the concept of temporal paths and propose the Temporal Inductive Path Neural Network (TiPNN) for the task of extrapolated reasoning over TKGs. Our approach models the temporal path features corresponding to reasoning queries by considering temporal information within and between different timestamps, along with the compositional logical rules of historical subgraph sequences.
- Extensive experiments demonstrate that our model outperforms the existing state-of-the-art results. Additionally, we construct datasets and conduct validation experiments for inductive reasoning on TKGs. Furthermore, we present reasoning evidence demonstrations and analytical experiments based on history temporal graph inference.

2. Preliminaries

In this section, we provide the essential background knowledge and formally define the context. We also introduce the basics of temporal knowledge graph reasoning task. The details of essential mathematical symbols and the corresponding descriptions of TiPNN are shown in Table 1. It is important to clarify beforehand that we will indicate vector representations in the upcoming context using **bold** items.

2.1. Background of temporal knowledge graph

A temporal knowledge graph $\mathcal{G}$ is essentially a multi-relational, directed graph that incorporates timestamped edges connecting various entities. It can be formalized as a succession of static subgraphs arranged in chronological order, i.e., $\mathcal{G} = \{G_1, G_2, ..., G_t, ...\}$. A subgraph in $\mathcal{G}$ can be denoted as $G_t = (\mathcal{V}, \mathcal{R}, \mathcal{E}_t)$ corresponding to the snapshot at timestamp t, where $\mathcal{V}$ is the set of entities, $\mathcal{R}$ is the set of relation types, and $\mathcal{E}_t$ is the set of fact edges at timestamp t. Each element in $\mathcal{E}_t$ is a timestamped quadruple

$(subject, relation, object, timestamp)$, which can be denoted as (s, r, o, t) or (s_t, r_t, o_t), describing a relation type $r \in \mathcal{R}$ occurs between subject entity $s \in \mathcal{V}$ and object entity $o \in \mathcal{E}$ at timestamp $t \in \mathcal{T}$, where $\mathcal{T}$ denote the finite set of timestamps.

2.2. Formulation of reasoning task

Temporal knowledge graph reasoning task involves predicting the missing entity by answering a query like $(s, r, ?, t_q)$ using historical known facts $\{(s, r, o, t_i) | t_i < t_q\}$. Note that the facts in the query time period t_q are unknown in this task setting. For the sake of generalization, we assume that the fact prediction at a future time depends on a sequence of historical subgraphs from the closest m timestamps. That is, in predicting the missing fact at timestamp $t+1$, we consider the historical subgraph sequence $\{G_{t-m+1}, ..., G_t\}$ for the inference.

In addition, given an object entity query $(s, r, ?, t_q)$ at a future timestamp t_q, we consider all entities in the entity set $\mathcal{V}$ as candidates for the object entity reasoning. The final prediction is obtained after scoring and ranking all the candidates by a scoring function. When predicting the subject entity, i.e., $(?, r, o, t_q)$, we can transform the problem into object entity prediction form $(o, r^{-1}, ?, t_q)$. Therefore, we also insert the corresponding reverse edge (o, r^{-1}, s, t) when processing the history graph. Without loss of generality, the later section will be introduced in terms of object entity predictions.

3. Methodology

In this section, we introduce the proposed model, **T**emporal **I**nductive **P**ath **N**eural **N**etwork (TiPNN). We start with a model overview and then discuss each part of the model as well as its training and reasoning process in detail.

3.1. Model overview

The main idea of TiPNN is to model the compositional logical rules on multi-relational temporal knowledge graphs. By integrating historical subgraph information and capturing the correlated patterns of historical facts, we can achieve predictions for missing facts at future timestamps. Given a query $(s, r, ?, t+1)$ at future timestamp $t+1$, our focus is on utilizing historical knowledge derived from $\{G_{t-m+1}, ..., G_t\}$ to model the connected semantic information between the query subject s and all candidate objects $o \in \mathcal{V}$ in history for the query responding.

Each subgraph in a temporal knowledge graph with a different timestamp describes factual information that occurred at different moments. These subgraphs are structurally independent of each other, as there are no connections between the node identifiers within them. And the timestamps are also independently and discretely distributed, which makes it challenging to model the historical subgraphs in a connected manner. To achieve this, we construct a logically connected graph, named ***History Temporal Graph***, to replace the originally independent historical subgraphs, allowing a more direct approach to modeling the factual features of the previous timestamp. By utilizing connected relation features and relevant paths associated with the query, it can capture semantic connections between nodes and thus learn potential temporal linking logical rules.

Specifically, for a given query subject entity, most candidate object entities are (directly or indirectly) connected to it in a logically connected history temporal graph. Therefore, ***Temporal Path*** is proposed to comprehensively capture the connected edges between the query subject entity and other entities within the history temporal graph in a query-aware manner. By explicitly learning the path information between entities on the history temporal graph, the query-aware representation of the history temporal path between query subject and object candidates can be obtained. Based on the learned path feature, the inference of future missing facts can be made with a final score function. A high-level idea of this approach is to induce the connection patterns and semantic correlations of the historical context.

We introduce the construction of history temporal graph in Section 3.2 and discuss the formulation of temporal path in Section 3.3. Then we present the detail of query-aware temporal path processing in Section 3.4. Additionally, we describe the scoring and loss function in Section 3.5, and finally analyze the complexity in Section 3.6.

3.2. History temporal graph construction

To comprehensively capture the connectivity patterns between entities and complex temporal characteristics in the historical subgraphs, we construct a *history temporal graph*. One straightforward approach is to ignore the temporal information within the historical subgraphs and directly use all triplets from the subgraph sequence to form a complete historical graph. However, it is essential to retain the complete information of the historical subgraph sequence. The timestamps in the historical subgraph sequence are crucial for inferring missing facts for the future. The representations of entities and relationships can vary in semantic information across different timestamps in the historical subgraphs. Additionally, there is a temporal ordering between historical subgraphs at different time periods, which further contributes to the inference process.

Therefore, we consider incorporating the timestamps of historical subgraphs into the history temporal graph to maximize the integrity of the original historical subgraph sequence, which allows us to retain the essential temporal context needed for inferring missing facts effectively. Specifically, given a query $(s, r, ?, t+1)$ at future time $t+1$, we use its corresponding historical subgraph sequence $\{G_{t-m+1}, ..., G_t\}$ from the previous m timestamps to generate the history temporal graph $\hat{G}_{t-m+1:t}$ according to the form specified in Equation (1). Fig. 2 illustrates the construction approach of the history temporal graph.

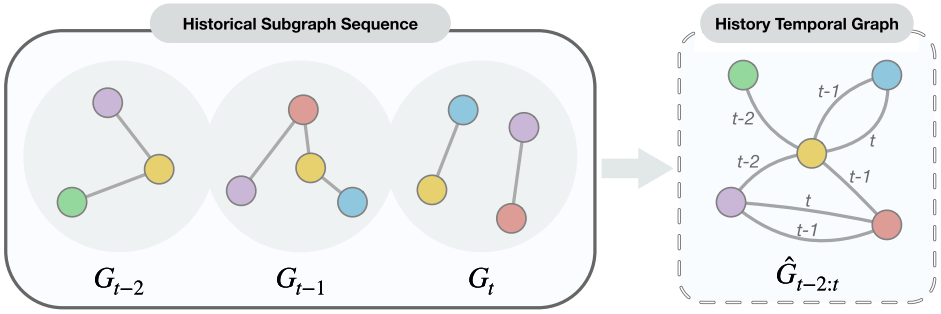

Fig. 2. Example of constructing the history temporal graph $\hat{G}_{t-2:t}$ with $m = 3$. For illustrative purposes, we use an undirected graph to demonstrate the construction method and omit the relationship types of edges in the historical subgraph sequence and the constructed history temporal graph. The time labels attached to the edges in the history temporal graph represent the timestamps of the corresponding edges in the historical subgraph sequence.

$$\hat{G}_{t-m+1:t} \leftarrow \left\{ (s, r_\tau, o) \middle| (s, r, o) \in G_\tau, \tau \in [t-m+1, t] \right\} \tag{1}$$

Note that r_τ denotes the *temporal relation* in history temporal graph $\hat{G}_{t-m+1:t}$, representing a relation type r with time attribute τ attached. That is, for a temporal edge $(s, r_\tau, o) \in \hat{G}_{t-m+1:t}$, it means (s, r, o) occurred in $G\tau$. For simplicity, we abbreviate the history temporal graph $\hat{G}_{t-m+1:t}$ as $\hat{G}_{<t+1}$ for the query at $t+1$ timestamp. Attaching the timestamp feature to the relation can be understood as expanding the set of relation types into a combined form of *relation-timestamp*, which allows us to capture the temporal aspect of relations among the entities and enables the modeling of the temporal path in history temporal graph.

3.3. Temporal path formulation

The prediction of future missing facts relies on the development trends of historical facts. To capture the linkages between the query subject entity and object candidate entities on history temporal graph, the concept of temporal path is put forward.

Definition 1. ***Temporal Path*** is a logical path that aggregates the semantics of all connected paths in history temporal graph between a subject entity and an object entity through arithmetic logical operations.

The aggregated temporal path can be used to simultaneously represent the temporal information and comprehensive path information from the subject entity to the object entity. This enables us to obtain a comprehensive query-aware paired representation for future facts reasoning by aggregating the information from various paths, which is from query subject entity to object candidates in history temporal graph.

Correspondingly, in the previous work [9], DaeMon individually addresses the logical connection patterns within each independent historical subgraph. Additionally, it utilizes a path memory unit to separately model temporal path patterns along the timeline. It's crucial to note that the temporal paths learned by DaeMon are virtual since they are not acquired on a unified graph, making it impossible for practical visualization on a unified graph. In other words, the temporal path defined in DaeMon represents a latent synthesis of paths across time by the neural network, unlike the explicitly existing path on the history temporal graph in TiPNN.

It is also essential to consider the relation type information from the query while learning paired representations. That is, for a query $(s, r, ?, t+1)$, we should also embed the relation type r into the paired representation (i.e., path representation) as well. This ensures that the relation type is incorporated into the learning process, enhancing the overall modeling of the query and enabling more accurate and context-aware inference of missing facts, and the detail will be discussed in Section 3.4. Here we denote the representation of temporal path in history temporal graph $\hat{G}_{<t+1}$ as $\mathbf{H}^{<t+1}_{(s,r)\to\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}|\times d}$ corresponding to the query $(s, r, ?)$ at future timestamp $t+1$, where $|\mathcal{V}|$ is the cardinality of object candidates set and d is the dimension of temporal path embedding. Specifically, $\mathbf{H}^{<t+1}_{(s,r)\to\mathcal{V}}$ describes the representations of temporal paths from query subject entity s to all object candidate entities in $\mathcal{V}$, considering the specific query $(s, r, ?)$. Each item in $\mathbf{H}^{<t+1}_{(s,r)\to\mathcal{V}}$, which is denoted as $\mathbf{h}^{<t+1}_{(s,r)\to o} \in \mathbb{R}^d$, represents a specific temporal path feature that learns the representation of temporal path from subject s to a particular candidate object o, where $o \in \mathcal{V}$.

Given query $(s, r, ?, t+1)$, we take an object candidate $o \in \mathcal{V}$ as an example. The temporal path between subject entity s and o we consider is the aggregation form of all paths that start with s and end with o in the topology of history temporal graph $\hat{G}_{<t+1}$. Formally, we describe $\mathbf{h}^{<t+1}_{(s,r)\to o}$ as follow:

$$\mathbf{h}^{<t+1}_{(s,r)\to o} = \bigoplus \mathcal{P}^{<t+1}_{s\to o} = \mathbf{p}_1 \oplus \mathbf{p}_2 \oplus \cdots \oplus \mathbf{p}_{|\mathcal{P}^{<t+1}_{s\to o}|} \Big|_{\mathbf{p}_k \in \mathcal{P}^{<t+1}_{s\to o}}, \tag{2}$$

where $\oplus$ denotes the paths aggregation operator that aggregates paths feature between query subject s and object candidate o, which will be introduced in the following section; $\mathcal{P}^{<t+1}_{s\to o}$ denotes the set of paths from s to o in history temporal graph $\hat{G}_{<t+1}$, and $|\mathcal{P}^{<t+1}_{s\to o}|$ denotes the cardinality of the path set. A path feature $\mathbf{p}_k \in \mathcal{P}^{<t+1}_{s\to o}$ is defined as follow when it contains edges as $(e_1 \to e_2 \to \cdots \to e_{|\mathbf{p}_k|})$:

$$\mathbf{p}_k = \mathbf{w}_r(e_1) \otimes \mathbf{w}_r(e_2) \otimes \cdots \otimes \mathbf{w}_r(e_{|\mathbf{p}_k|}), \tag{3}$$

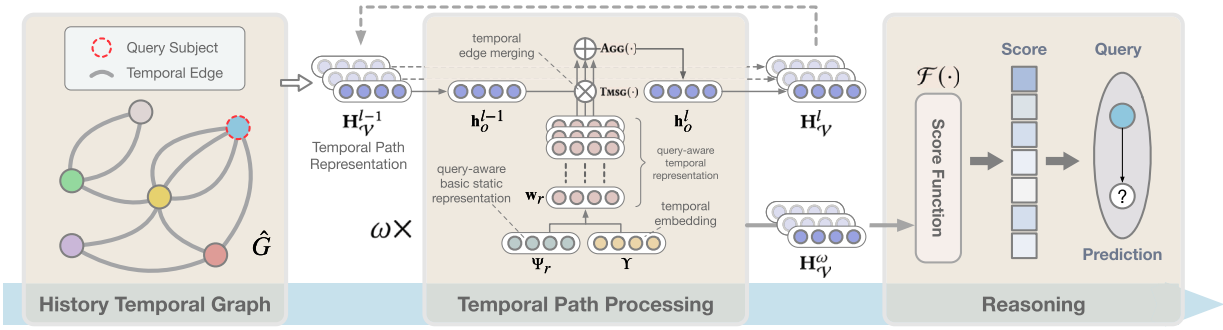

Fig. 3. An illustrative diagram of the proposed TiPNN model for query-aware temporal path processing. For a given query $(s, r, ?)$ at future timestamp, TiPNN engages in temporal path processing within the constructed history temporal graph $\hat{G}$ to perform prediction for the future timestamp. The temporal path feature $\mathbf{H}_{\mathcal{V}}$ is iteratively learned and updated by the query-aware ω-layers aggregation neural network. In each layer, the temporal edges in $\hat{G}$, enriched with temporal information, are individually modeled for basic static representation and temporal representation through $\mathbf{\Psi}_r$ and $\mathbf{\Upsilon}$, respectively. Subsequently, temporal edges are merged using $\textsc{Tmsg}(\cdot)$, followed by aggregation through $\textsc{Agg}(\cdot)$ to obtain the feature of the current layer's temporal path.

where $e_{(1,2,\ldots,|\mathrm{p}_k|)}$ denotes the temporal edges in path p_k, and $|\mathrm{p}_k|$ denotes the number of edges in path p_k; $\mathbf{w}_r(e_*)$ is the query-aware temporal representation of edge e_* and $\otimes$ denotes the operator of merging temporal edges information within the path.

3.4. Query-aware temporal path processing

In this section, we discuss how to model temporal path representations for the queries at future timestamps based on history temporal graph, as illustrated in Fig. 3. It aids in predicting missing facts by providing connection patterns and temporal information into the temporal edges between entities and enabling practical inference for the future facts.

3.4.1. Temporal path aggregation layer

We propose a temporal path aggregation layer for comprehensively aggregating the connected temporal edges between the query subject entity s and all candidate object entities in $\mathcal{V}$ within the history temporal graph. Note that a temporal path representation is specific to a particular query. Different queries can lead to the same path, but they have different representations (e.g. when two queries share the same subject entity but have different query relations). This design ensures each path representations are context-dependent and take into account the query's unique characteristics, such as the query subject entity and query relation type, which can influence the meaning and relevance of the path in different contexts.

The entire aggregation process is carried out iteratively based on the sight outlined in Equations (2) and (3). Since the history temporal graph already contains the connection information and temporal feature of historical facts, we can directly capture the temporal path representations from history temporal graph relevant to a given query. We adopt the ω-layers message passing approach of graph neural network at history temporal graph to expand the iterative path length and learn query-aware temporal path features, which enables the continuous and simultaneous collection of multiple path information and their corresponding temporal edge features, for the temporal path representations.

Specifically, given a query $(s, r, ?)$ at future timestamp $t+1$, based on the structural and temporal feature of historical facts, the representation of temporal path $\mathbf{h}^{\prec t+1}_{(s,r)\to o}$ in history temporal graph $\hat{G}_{\prec t+1}$ (i.e., the aggregated feature of path set $\mathcal{P}^{\prec t+1}_{s\to o}$) is learned by the query-aware ω-layers aggregation neural network. That is, $\mathbf{H}^{\prec t+1}_{(s,r)\to\mathcal{V}}$ will be finally updated to represent the temporal paths from query subject s to all candidate objects $o \in \mathcal{V}$ within a limited number of hops, after finishing ω-th layer aggregation iteration. For the sake of simplicity, in the following text, we will use the query $(s, r, ?, t+1)$ as an example, where s is the query subject entity, r is the query relation type, and $t+1$ is the timestamp for the future fact to be predicted. We omit the superscript (and subscript) $\prec t+1$ and the indicator of query subject and relation pair (s, r) in notations. $\mathbf{H}^l_{\mathcal{V}}$ is used to denote the status of temporal path representation at the l-th iteration, and $\mathbf{h}^l_o \in \mathbf{H}^l_{\mathcal{V}}$ denotes the status of candidate $o \in \mathcal{V}$ at the l-th iteration, where $l \in [0, \omega]$. It should be stressed that $\mathbf{H}^l_{\mathcal{V}}$ (or $\mathbf{h}^l_o$) is still describing the representation of temporal path(s), rather than node(s) representation.

A temporal path is formed by logically aggregating multiple real existing paths from the history temporal graph. Each path consists of multiple temporal edges. Although the temporal edge exists in the form of *relation-timestamp*, it is still based on a specific relationship $r \in \mathcal{R}$. Therefore, we initialize a trainable representation for the relation set, which serves as a foundational feature when processing temporal edge features during temporal path aggregation with query awareness. Here, we introduce and discuss the learnable relation representation and initialization of temporal path representation.

- **Learnable Relation Representation**
  At the very beginning of the processing, we initialize the representation of all relation types, which are in the set of relation types $\mathcal{R}$, with a learnable parameter $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}|\times d}$, where $|\mathcal{R}|$ is the cardinality of relation types set. It is essential to note that the learnable relation types representation $\mathbf{R}$ is shared throughout the entire model and is not independent between each time step prediction, enabling the model to leverage the learned relationship representations consistently across different time points and queries.

- **Temporal Path Initialization**
  When $l = 0$, $\mathbf{H}_{\mathcal{V}}^0$ denotes the initial status of iteration that is used to prepare for the subsequent processing of temporal path iterations. Different from the common GNN-based approach and inspired by [11], we initialize the temporal path feature of $o \in \mathcal{V}$ as query relation representation $\mathbf{r} \in \mathbf{R}$ only when o is the same as the query subject entity s, and a zero embedding otherwise. Formally, for the query $(s, r, ?)$, any candidate $o \in \mathcal{V}$ and query relation representation $\mathbf{r}$, the temporal path representation $\mathbf{h}_o^0 \in \mathbf{H}_{\mathcal{V}}^0$ is initialized following

$$\mathbf{h}_o^0 \leftarrow \begin{cases} \mathbf{r} & if\ o \Leftrightarrow s, \\ \vec{0} & if\ o \nLeftrightarrow s. \end{cases} \tag{4}$$

  This initialization strategy ensures that the initial state is contextually sensitive to the given query, starting at the query subject with the query relation feature, providing a foundation for subsequent iterations to capture and model the temporal paths.

With the initialization of temporal path representations, now we introduce the iterative aggregation process of the temporal path. Taking $\mathbf{h}_o^l$ as an example,

$$\mathbf{h}_o^l = \textsc{Agg}\left(\left\{ \textsc{Tmsg}\left(\mathbf{h}_z^{l-1}, \mathbf{w}_r(z, p_\tau, o)\right) \middle| (z, p_\tau, o) \in \hat{G}_{\prec t+1} \right\} \right), \tag{5}$$

where $\textsc{Agg}(\cdot)$ and $\textsc{Tmsg}(\cdot)$ are the aggregation and temporal edge merging function, corresponding to the operator in Equation (2) and (3), respectively, which will be introduced in the following section; $\mathbf{w}_r \in \mathbb{R}^d$ denotes query-aware temporal representation of a temporal edge type; $(z, p_\tau, o) \in \hat{G}_{\prec t+1}$ is a temporal edge in history temporal graph $\hat{G}_{\prec t+1}$ that temporal relation p_τ occurs between an entity z and candidate entity o at the history time τ.

However, to emphasize the distinction from DaeMon [9], it is important to note that in DaeMon, the acquisition of temporal paths is learned independently on each isolated historical subgraph (i.e., G_i, where $i < t + 1$). Consequently, the final temporal path representation requires stacking m-times (i.e., history length) updates of $\mathbf{h}_o$ in DaeMon.

3.4.2. Aggregation operating

Equation (5) describes the iterative aggregation process of $\mathbf{h}_o$ in the l-th layer. Here we provide explanations for the temporal edge merging function $\textsc{Tmsg}(\cdot)$ and path aggregation function $\textsc{Agg}(\cdot)$, respectively. Intuitively, the $\textsc{Tmsg}(\cdot)$ function is responsible for iteratively searching for paths with a candidate object entity o as the endpoint, continuously propagating information along the temporal edges. On the other hand, the $\textsc{Agg}(\cdot)$ function is responsible for aggregating the information obtained from each iteration along the temporal edges. Due to the initial state of $\mathbf{H}_o^0$, where only $\mathbf{h}_s^0$ is given the initial information, after ω iterations of aggregation, $\mathbf{h}_o^\omega$ captures the comprehensive semantic connection and temporal dependency information of multiple paths from subject s to object candidate o in the history temporal graph $\hat{G}_{\prec t+1}$.

- **Temporal Edge Merging Function $\textsc{Tmsg}(\cdot)$**
  It takes the current temporal path representation $\mathbf{h}_z^{l-1}$, query relation representation $\mathbf{r}$, and the temporal edge information as input and calculates the updated potential path information for the candidate object o. Similar to message passing in knowledge graphs, we have adopted a vectorized multiplication method [12] to achieve feature propagation in history temporal graph, following

$$\textsc{Tmsg}\left(\mathbf{h}_z^{l-1}, \mathbf{w}_r(z, p_\tau, o)\right) = \mathbf{h}_z^{l-1} \otimes \mathbf{w}_r(z, p_\tau, o), \tag{6}$$

  where the operator $\otimes$ is defined as element-wise multiplication between $\mathbf{h}_z^{l-1}$ and $\mathbf{w}_r(z, p_\tau, o)$. The vectorized multiplication can be understood as scaling $\mathbf{h}_z^{l-1}$ by $\mathbf{w}_r(z, p_\tau, o)$ in our temporal edges merging [12]. $\mathbf{w}_r(z, p_\tau, o)$ represents the query-aware temporal representation of a temporal edge (z, p_τ, o), which needs to ensure its relevance to the query. Different from DaeMon [9], in the history temporal graph, each edge is appended with temporal information, preventing the consideration of features solely based on static relation types. Therefore, the approach of temporal relation encoding is proposed for the temporal edge representation.
  Temporal Relation Encoding. As mentioned earlier, temporal edge representation should align with the query's characteristics. We take into account the temporal edge features from the query relation representation, semantic and temporal characteristics of the temporal edges in the history temporal graph, generating comprehensive embedding with both static and temporal characteristics. Formally, $\mathbf{w}_r(z, p_\tau, o)$ can be derived as

$$\mathbf{w}_r(z, p_\tau, o) = g\left(\mathbf{\Psi}_r(p) \middle|\middle| \mathbf{\Upsilon}(\Delta\tau) \right), \tag{7}$$

  where $\mathbf{\Psi}_r(p) \in \mathbb{R}^d$ denotes the query relation r-aware basic static representation of edge type p, $\mathbf{\Upsilon}(\Delta\tau) \in \mathbb{R}^d$ denotes the temporal embedding of the temporal edge type p_τ, $||$ denotes the operator of concatenation, and $g(\cdot)$ is a feed-forward neural network. For the basic static representation of p, following [9], we obtain it through a linear transformation. Based on the query relation representation $\mathbf{r}$, we map a representation as the static semantic information of the temporal edge type p, following

$$\mathbf{\Psi}_r(p) = \mathbf{W}_p \mathbf{r} + \mathbf{b}_p, \tag{8}$$

where $\mathbf{W}_p$ and $\mathbf{b}_p$ are learnable parameters and serve as the weights and biases of the linear transformation, respectively, which makes the basic static information of the temporal edges derived from the given query relation embedding, ensuring the awareness of the query. And for the temporal embedding of the temporal edge, we use generic time encoding [13] to model the temporal information in temporal edges, following

$$\Delta\tau = |\tau - t_q|, \tag{9}$$

$$\boldsymbol{\Upsilon}(\Delta\tau) = \sqrt{\frac{1}{d}} \left[cos(\boldsymbol{w}_1\Delta\tau + \boldsymbol{\phi}_1), cos(\boldsymbol{w}_2\Delta\tau + \boldsymbol{\phi}_2), \cdots, cos(\boldsymbol{w}_d\Delta\tau + \boldsymbol{\phi}_d) \right], \tag{10}$$

where t_q denotes the query timestamp, $\Delta\tau$ denotes the time interval between the query timestamp t_q and the temporal edge timestamp τ, which measures how far apart τ is from t_q, namely *relative time distance*. Moreover, $\boldsymbol{w}_*$ and $\boldsymbol{\phi}_*$ are learnable parameters, and d is the dimension of the vector representation, which is the same as the dimension of the static representation. The $\textsc{Tmsg}(\cdot)$ function effectively bridges the temporal dependencies and semantic connections between the query subject and the candidate object through propagation along the temporal edges in paths. Besides, in the encoder of temporal relation, considering query relation feature allows to tailor the temporal edge representation to the specific relation type relevant to the query; temporal characteristics of edges help to model chronological order and time-based dependencies between different facts; and simultaneously modeling the basic static and temporal features of the temporal edge make more comprehensive semantics obtained.

- **Path Aggregation Function $\textsc{Agg}(\cdot)$**
  It considers the accumulated information from the previous iteration and the newly propagated information along the temporal edge to update the temporal path representation for a candidate object. We adopt principal neighborhood aggregation (PNA) proposed in [14], which leverages multiple aggregators (namely mean, maximum, minimum, and standard deviation) to learn joint feature, since previous work has verified its effectiveness [15,16]. We also consider the traditional aggregation function as a comparison, such as sum, mean, and max, which will be introduced in Section 4.3.

Finally, after the ω-th iteration of aggregation, $\mathbf{H}_{\mathcal{V}}^{\omega}$ will obtain the representation of the temporal paths from query subject s to all candidate objects in $\mathcal{V}$. Here we should note that, unlike the previous work [9], TiPNN integrates all historical subgraphs into a unified history temporal graph $\hat{G}$. When handling temporal path features, any temporal edge $(z, p_\tau, o) \in \hat{G}$ is associated with timestamp information. Additionally, the static representation of p_τ and its temporal representation are respectively processed by $\boldsymbol{\Psi}$ and $\boldsymbol{\Upsilon}$. Consequently, TiPNN eliminates the need, as in DaeMon, for a separate consideration of cross-temporal path fusion. The resulting $\mathbf{H}_{\mathcal{V}}^{\omega}$ already encompasses comprehensive semantic connections and temporal dependency information from multiple paths in history.

3.5. Learning and inference

TiPNN models the query-aware temporal path feature by aggregation process within the history temporal graph. By capturing comprehensive embedding with both static and temporal characteristics of temporal edges, and aggregating multiple paths that consist of temporal edges, TiPNN learns a comprehensive representation of temporal path. Different from most previous models, we utilize the temporal edge features in a path from query subject to candidate object, without considering any entity embedding during modeling processing, thus TiPNN can solve the inductive setting, which will be presented in Section 4.2 in detail.

3.5.1. Score function

Here we show how to apply the final learned temporal paths representation to the temporal knowledge graph reasoning. Given query subject s and query relation r at timestamp $t+1$, after obtaining temporal paths representation $\mathbf{H}_{(s,r)\to\mathcal{V}}^{\omega}$, we predict the conditional likelihood of the future object candidate $o \in \mathcal{V}$ using $\mathbf{h}_{(s,r)\to o}^{\omega} \in \mathbf{H}_{(s,r)\to\mathcal{V}}^{\omega}$, following:

$$p(o|s,r) = \sigma\left(\mathcal{F}\left(\mathbf{h}_{(s,r)\to o}^{\omega} \,\middle\|\, \boldsymbol{r} \right) \right), \tag{11}$$

where $\mathcal{F}(\cdot)$ is a feed-forward neural network, $\sigma(\cdot)$ is the sigmoid function and $\|$ denotes embedding concatenation. Note that we append the query relation embedding $\boldsymbol{r}$ to the temporal path feature $\mathbf{h}_{(s,r)\to o}^{\omega}$, and it helps to alleviate the insensitivity to unreachable distances for nodes within a limited number of hops, and also enhances the learning capacity of relation embeddings $\mathbf{R}$.

As we have added inverse quadruple (o, r^{-1}, s, t) corresponding to (s, r, o, t) into the dataset in advance, without loss of generality, we can also predict subject $s \in \mathcal{V}$ given query relation r^{-1} and query object o with the same model as:

$$p(s|o,r^{-1}) = \sigma\left(\mathcal{F}\left(\mathbf{h}_{(o,r^{-1})\to s}^{\omega} \,\middle\|\, \boldsymbol{r}^{-1} \right) \right). \tag{12}$$

3.5.2. Parameter learning

Reasoning on a given query can be seen as a binary classification problem. The objective is to minimize the negative log-likelihood of positive and negative triplets, as shown in Equation (13). In the process of generating negative samples, we follow the Partial

Completeness Assumption [17]. Accordingly, for each positive triplet in the reasoning future triplet set, we create a corresponding negative triplet by randomly replacing one of the entities with a different entity. It ensures that the negative samples are derived from the future triplet set at the same timestamp, allowing the model to effectively learn to discriminate between correct and incorrect predictions at the same future timestamp.

$$\mathcal{L}_{TKG} = -\log p(s,r,o) - \sum_{j=1}^{n} \frac{1}{n} log(1 - p(\overline{s_j}, r, \overline{o_j})), \tag{13}$$

where n is hyperparameter of negative samples number per positive sample; (s,r,o) and $(\overline{s_j}, r, \overline{o_j})$ are the positive sample and j-th negative sample, respectively.

Besides, to promote orthogonality in the learnable relation parameter $\mathbf{R}$ initialized at the beginning, a regularization term is introduced in the objective function, inspired by the work in [18]. The regularization term is represented following Equation (14), where $\mathbf{I}$ is the identity matrix, α is a hyperparameter and $\|\cdot\|$ denotes the L2-norm.

$$\mathcal{L}_{REG} = \left\| \mathbf{R}^T\mathbf{R} - \alpha\mathbf{I} \right\| \tag{14}$$

Therefore, the final loss of TiPNN is the sum of two losses and can be denoted as:

$$\mathcal{L} = \mathcal{L}_{TKG} + \mathcal{L}_{REG}. \tag{15}$$

3.6. Complexity analysis

The score of each temporal path from the query subject and candidate object can be calculated in parallel operation. Therefore, to see the complexity of the proposed TiPNN, we analyze the computational complexity of each query. The major operation in TiPNN is query-aware temporal path processing, and each aggregation iteration contains two steps: temporal edge merging function $\textsc{Tmsg}(\cdot)$ and path aggregation function $\textsc{Agg}(\cdot)$ (as shown in Equation (5)). We use $|\mathcal{E}|$ to denote the maximum number of concurrent facts in the historical subgraph sequence, and $|\mathcal{V}|$ to denote the cardinality of the entity set. $\textsc{Tmsg}(\cdot)$ has the time complexity of $O(m|\mathcal{E}|)$ since it performs Equation (6) for all edges in the history temporal graph, where m denotes the length of the history used for reasoning. $\textsc{Agg}(\cdot)$ has the time complexity of $O(|\mathcal{V}|)$ since the aggregation method we adopt will perform for every entity. And $\textsc{Agg}(\cdot)$ is executed after $\textsc{Tmsg}(\cdot)$ at each aggregation iteration, so the time complexity is finally $O(\omega(m|\mathcal{E}| + |\mathcal{V}|))$ when the number of aggregation layers is ω.

4. Experiments

In this section, we carry out a series of experiments to assess the effectiveness and performance of proposed TiPNN. Through these experiments, we aim to address and answer the following research questions:

- **RQ1:** How does the proposed model perform as compared with state-of-the-art knowledge graph reasoning and temporal knowledge graph reasoning methods?
- **RQ2:** How does each component of TiPNN (i.e., temporal relation encoding, temporal edge merging function, path aggregation function) affect the performance?
- **RQ3:** How do the core parameters in history temporal graph construction (i.e., sampling history length) and query-aware temporal path processing (i.e., number of temporal path aggregation layers) affect the reasoning performance?
- **RQ4:** How does the performance efficiency of TiPNN compared with state-of-the-art method for the reasoning?
- **RQ5:** How does the proposed model solve the inductive setting on temporal knowledge graph reasoning?
- **RQ6:** How does the proposed model provide reasoning evidence based on the history temporal path for the reasoning task?

4.1. Experimental settings

4.1.1. Datasets

We conduct extensive experiments on four representative temporal knowledge graph datasets, namely, ICEWS18 [6], GDELT [19], WIKI [20], and YAGO [21]. These datasets are widely used in the field of temporal knowledge reasoning due to their diverse temporal information. The ICEWS18 dataset is collected from the Integrated Crisis Early Warning System [22], which records various events and activities related to global conflicts and crises, offering valuable insights into temporal patterns of such events. The GDELT dataset [3] is derived from the Global Database of Events, Language, and Tone, encompassing a broad range of events across the world and enabling a comprehensive analysis of global events over time. WIKI [20] and YAGO [21] are two prominent knowledge bases containing factual information with explicit temporal details. WIKI covers a wide array of real-world knowledge with timestamps, while YAGO offers a structured knowledge base with rich semantic relationships and temporal annotations. And we focus on the subsets of WIKI and YAGO that have yearly granularity in our experiments. The details of the datasets are provided in Table 2. To ensure fair evaluation, we adopt the dataset splitting strategy employed in [6]. The dataset is partitioned into three sets: training, validation, and test sets, based on timestamps. Specifically, (timestamps of the train) < (timestamps of the valid) < (timestamps of the test).

Table 2
Statistics of Datasets ($\mathcal{E}_{train}$, $\mathcal{E}_{valid}$, $\mathcal{E}_{test}$ are the numbers of facts in training, validation, and test sets).

Datasets	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\mathcal{E}_{train}$	$\mathcal{E}_{valid}$	$\mathcal{E}_{test}$	$\lvert\mathcal{T}\rvert$
ICEWS18	23,033	256	373,018	45,995	49,545	304
GDELT	7,691	240	1,734,399	238,765	305,241	2,976
WIKI	12,554	24	539,286	67,538	63,110	232
YAGO	10,623	10	161,540	19,523	20,026	189

Table 3
Overall Performance Comparison of Different Methods. Evaluation metrics are time-aware filtered MRR and Hits@{1,3,10}. All results are multiplied by 100. The best results are highlighted in **bold**. And the second-best results are highlighted in underline. (Higher values indicate better performance.)

Model	ICEWS18				GDELT				WIKI				YAGO			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
DistMult	11.51	7.03	12.87	20.86	8.68	5.58	9.96	17.13	10.89	8.92	10.97	16.82	44.32	25.56	48.37	58.88
ComplEx	22.94	15.19	27.05	42.11	16.96	11.25	19.52	32.35	24.47	19.69	27.28	34.83	44.38	25.78	48.20	59.01
ConvE	24.51	16.23	29.25	44.51	16.55	11.02	18.88	31.60	14.52	11.44	16.36	22.36	42.16	23.27	46.15	60.76
RotatE	12.78	4.01	14.89	31.91	13.45	6.95	14.09	25.99	46.10	41.89	49.65	51.89	41.28	22.19	45.33	58.39
TTransE	8.31	1.92	8.56	21.89	5.50	0.47	4.94	15.25	29.27	21.67	34.43	42.39	31.19	18.12	40.91	51.21
TA-DistMult	16.75	8.61	18.41	33.59	12.00	5.76	12.94	23.54	44.53	39.92	48.73	51.71	54.92	48.15	59.61	66.71
DE-SimplE	19.30	11.53	21.86	34.80	19.70	12.22	21.39	33.70	45.43	42.60	47.71	49.55	54.91	51.64	57.30	60.17
TNTComplEx	21.23	13.28	24.02	36.91	19.53	12.41	20.75	33.42	45.03	40.04	49.31	52.03	57.98	52.92	61.33	66.69
TANGO-Tucker	28.68	19.35	32.17	47.04	19.42	12.34	20.70	33.16	50.43	48.52	51.47	53.58	57.83	53.05	60.78	65.85
TANGO-DistMult	26.65	17.92	30.08	44.09	19.20	12.17	20.40	32.78	51.15	49.66	52.16	53.35	62.70	59.18	60.31	67.90
CyGNet	24.93	15.90	28.28	42.61	18.48	11.52	19.57	31.98	33.89	29.06	36.10	41.86	52.07	45.36	56.12	63.77
RE-NET	28.81	19.05	32.44	47.51	19.62	12.42	21.00	34.01	49.66	46.88	51.19	53.48	58.02	53.06	61.08	66.29
RE-GCN	30.58	21.01	34.34	48.75	19.64	12.42	20.90	33.69	77.55	73.75	80.38	83.68	84.12	80.76	86.30	89.98
TITer	29.98	22.05	33.46	44.83	15.46	10.98	15.61	24.31	75.50	72.96	77.49	79.02	87.47	84.89	89.96	90.27
xERTE	29.31	21.03	33.40	45.60	18.09	12.30	20.06	30.34	71.14	68.05	76.11	79.01	84.19	80.09	88.02	89.78
CEN	30.84	21.23	34.58	49.67	20.18	12.84	21.51	34.10	78.35	74.69	81.47	84.45	83.49	79.77	85.85	89.92
GHT	29.16	18.99	33.16	48.37	20.13	12.87	21.30	34.19	48.50	45.08	50.87	53.69	57.22	51.64	60.68	67.17
DaeMon	31.85	22.67	35.92	49.80	20.73	13.65	22.53	34.23	82.38	78.26	86.03	88.01	91.59	90.03	93.00	93.34
TiPNN	**32.17**	**22.74**	**36.24**	**50.72**	**21.17**	**14.03**	**22.98**	**34.76**	**83.04**	**79.04**	**86.45**	**88.54**	**92.06**	**90.79**	**93.15**	**93.58**

4.1.2. Evaluation metrics

To evaluate the performance of the proposed model for TKG reasoning, we employ the commonly used task of link prediction on future timestamps. To measure the performance of the method, we report the Mean Reciprocal Rank (MRR) and Hits@{1, 3, 10} metrics. MRR is a measure that evaluates the average reciprocal rank of correctly predicted facts for each query. Hits@k is a set of metrics that measures the proportion of queries for which the correct answer appears in the top-k positions of the ranking list.

In contrast to the traditional filtered setting used in previous works [23,6,24], where all valid quadruples appearing in the training, validation, or test sets are removed from the ranking list of corrupted facts, we consider that this setting is not suitable for TKG reasoning tasks. Instead, we opt for a more reasonable evaluation, namely time-aware filtered setting, where only the facts occurring at the same time as the query are filtered from the ranking list of corrupted facts, which is aligned with recent works [31,25,9].

4.1.3. Compared methods

We conduct a comprehensive comparison of our proposed model with three categories of baselines: (1) *KG Reasoning Models*. It consists of traditional KG reasoning models that do not consider timestamps, including DistMult [12], ComplEx [26], ConvE [27], and RotatE [28]. (2) *Interpolated TKG Reasoning models*. We also compare our proposed model with four interpolated TKG reasoning methods, namely TTransE [20], TA-DistMult [29], DE-SimplE [7], and TNTComplEx [30]. (3) *Extrapolated TKG Reasoning models*. We include state-of-the-art extrapolated TKG reasoning methods that infers future missing facts, including TANGO-Tucker [25], TANGO-DistMult [25], CyGNet [24], RE-NET [6], RE-GCN [8], TITer [31], xERTE [10], CEN [32], GHT [33], and DaeMon [9].

4.1.4. Implementation details

For history temporal graph construction, we perform a grid search on the history length m and present overview results with the lengths 25, 15, 10, 8, corresponding to the datasets ICEWS18, GDELT, WIKI, and YAGO in Table 3, which is described in detail presented in Fig. 7. The embedding dimension d is set to 64 for temporal path representation. For query-aware temporal path processing, we set the number of temporal path aggregation layers ω to 6 for ICEWS18 and GDELT datasets, and 4 for WIKI and YAGO datasets. We conduct layer normalization and shortcut on the aggregation layers. The activation function $relu$ is adopted for the aggregation of the temporal path. Similar to [11], we use different temporal edge representations in different aggregation layers, that is the learnable parameter in $\mathbf{\Psi}$ and $\mathbf{\Upsilon}$ is independent of each aggregation iteration [34] [35]. To facilitate parameter learning,

we have set the number of negative sampling to 64 and the hyperparameter α in the regularization term to 1. We use Adam [36] for parameter learning, with a learning rate of $1e-4$ for YAGO and $5e-4$ for others. The maximum epoch of training has been set to 20. We also design an experiment in an inductive setting, which predicts facts with unseen entities in the training set, to demonstrate the inductive ability of our proposed model. All experiments were conducted with EPYC 7742 CPU and 8 TESLA A100 GPUs.

4.2. Experimental results (RQ1)

The experiment results on the TKG reasoning task are shown in Table 3 in terms of time-aware filtered MRR and Hits@{1,3,10}. The results demonstrate the effectiveness of our proposed model with a convincing performance and validate the superiority of TiPNN in handling temporal knowledge graph reasoning tasks. It consistently outperforms all baseline methods and achieves state-of-the-art (SOTA) performance on the four TKG datasets.

In particular, TiPNN exhibits better performance compared to all static models (listed in the first block of Table 3) and the temporal models in the interpolation setting (presented in the second block of Table 3). This is attributed to TiPNN's incorporation of temporal features of facts and its ability to learn temporal inductive paths, which enables it to effectively infer future missing facts. By leveraging comprehensive historical semantic information, TiPNN demonstrates remarkable proficiency in handling temporal knowledge graph reasoning tasks. Compared with the temporal models under the extrapolation setting (those presented in the third block of Table 3), the proposed model also achieves better results. TiPNN demonstrates its capability of achieving superior results by effectively modeling temporal edges. In contrast to the previous model, it starts from the query and obtains query-specific temporal path representations, allowing for more accurate predictions about the future. Thanks to the message propagation mechanism between temporal edges, it leverages both temporal and structural information to learn customized representations tailored to each query. This enables TiPNN to make more precise predictions for future missing facts. The impressive performance of DaeMon demonstrated its ability to model relation features effectively. However, we further enhance the integration of relation and temporal features by weakening the barriers between different time-stamped subgraphs using the constructed history temporal graph. It enables us to capture the characteristics of historical facts over a larger span of information and reduces the parameter loss caused by subgraph evolution patterns.

4.3. Ablation study (RQ2)

Note that TiPNN involves three main operational modules when processing query-aware temporal paths: the temporal relation encoder, the temporal edge merging operation, and the path aggregation operation. The temporal relation encoder is responsible for learning the features of temporal edges. The temporal edge merging operation is used to generate path features for each path from the query subject entity to the candidate object entity. The path aggregation module is responsible for consolidating all path features for pair-wise representations of temporal paths.

To show the impact of these components, we conducted an ablation analysis on the temporal encoder Υ of temporal relation encoding and discussed the influence of temporal encoding independence on the results. We also compared three variants of the temporal edge merging operation in $\textsc{Tmsg}(\cdot)$, as well as replacements for the aggregator in the path aggregation module $\textsc{Agg}(\cdot)$. These comprehensive evaluations allowed us to assess the effectiveness of each component and gain insights into their respective contributions to the overall performance of TiPNN.

4.3.1. Effectiveness of temporal relation encoding

The temporal encoder Υ plays a crucial role in modeling the temporal order of historical facts and the time factor of temporal edges in the temporal paths. Therefore, in this study, we aim to evaluate the ability of the temporal encoder in predicting future facts. To do so, we conducted an ablation analysis by removing the temporal encoder component from TiPNN and comparing its performance. The results are illustrated in Fig. 4, where we present a performance comparison between TiPNN with and without the temporal encoder.

The experimental results demonstrate that when the temporal encoder is removed, the performance of TiPNN decreases across all datasets. This outcome aligns with our expectations since relying solely on the query-aware static representation from the temporal relation modeling cannot capture the temporal information of historical facts. As a consequence, the absence of the temporal encoder hinders the accurate learning of temporal features among past facts.

Furthermore, we also investigated the impact of the independence of parameters in the temporal encoder on all datasets. Specifically, we considered the parameters $\boldsymbol{w}*$ and $\boldsymbol{\phi}*$ in Υ of Equation (10), which are responsible for modeling the temporal features of the temporal edge (z, p_τ, o), to be independently learned for each static type p. In other words, we assumed that the same relative time distance may have different temporal representations for different relation types. Thus, for each distinct edge type p, we utilized a relation-aware temporal encoder to represent the temporal features for relative time distance $\Delta\tau$. We conducted experiments on all datasets, as shown in Table 4. We use 'Shared Param.' to indicate that the parameters in Υ are shared and not learned independently for each edge type p, and 'Specific Param.' to indicate that the parameters are not shared and are learned independently based on the edge types.

Based on Table 4(a), we observe that using shared parameters yields better performance on the ICEWS18 and GDELT datasets. Conversely, from Table 4(b), we find that independently learning the parameters of the temporal encoder is more beneficial on the WIKI and YAGO datasets. We analyze this phenomenon and suggest that WIKI and YAGO have fewer edge types, and independently learning specific parameters for each edge type allows for easier convergence, making this approach more advantageous compared to learning shared parameters.

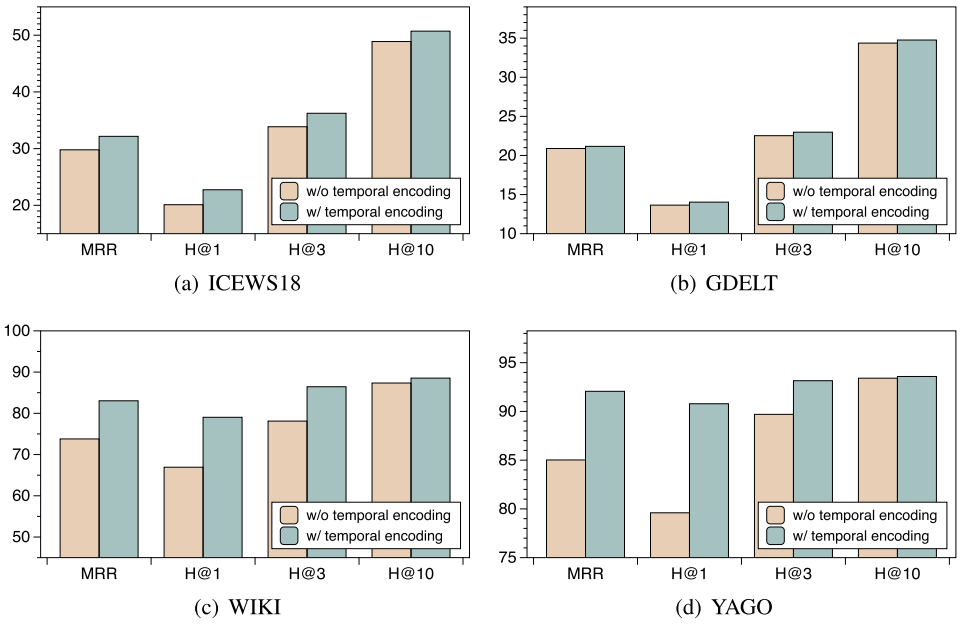

Fig. 4. Ablation results on temporal encoding.

Table 4
Impact of the independence setting of parameters in temporal encoder.

(a) ICEWS18 and GDELT

Settings	ICEWS18				GDELT			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Shared Param.	**32.17**	**22.74**	**36.24**	**50.72**	**21.17**	**14.03**	**22.98**	**34.76**
Specific Param.	30.36	20.80	34.38	49.29	21.00	13.88	22.80	34.60

(b) WIKI and YAGO

Settings	WIKI				YAGO			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Shared Param.	82.97	78.89	86.39	88.40	91.32	89.65	92.83	93.24
Specific Param.	**83.04**	**79.04**	**86.45**	**88.54**	**92.06**	**90.79**	**93.15**	**93.58**

4.3.2. *Variants of temporal edge merging method*

The temporal edge merging operation $\textsc{Tmsg}(\cdot)$ utilizes the message passing mechanism to iteratively extend the path length while merging the query-relevant path information. As mentioned in Equation (6), we employ the scaling operator from DistMult [12] for computing the current temporal path representation with the features of temporal edges through element-wise multiplication. Additionally, we have also conducted experiments using the translation operator from TransE [23] (i.e., element-wise summation) and the rotation operator from RotatE [28] to provide further evidence supporting the effectiveness of the temporal edge merging operation.

We conducted experimental validation and comparison on two types of knowledge graphs: the event-based graph ICEWS18 and the knowledge-based graph YAGO. Fig. 5 displays the results obtained with different merging operators. The findings demonstrate that TiPNN benefits from these excellent embedding methods, performing on par with DistMult and RotatE, and outperforming TransE, particularly evident on MRR and H@1 in the knowledge-based graph YAGO.

4.3.3. *Variants of path aggregation method*

The path aggregation operation $\textsc{Agg}(\cdot)$ is performed after merging the temporal edges at each step, and the aggregated feature serves as the temporal path representation from the query subject entity to the object entity candidates for future fact reasoning. As mentioned in Section 3.4.2, we employ the principal neighborhood aggregation (PNA) [14] as the aggregator to aggregate the propagated messages. PNA aggregator considers mean, maximum, minimum, and standard deviation as aggregation features to obtain comprehensive neighbor information. To validate its effectiveness, we compare it with several individual aggregation operations: SUM, MEAN, and MAX.

Similarly, we conduct experiments on both the event-based graph ICEWS18 and the knowledge-based graph YAGO. Fig. 6 illustrates the results with different aggregation methods. The findings demonstrate that TiPNN performs optimally with the PNA

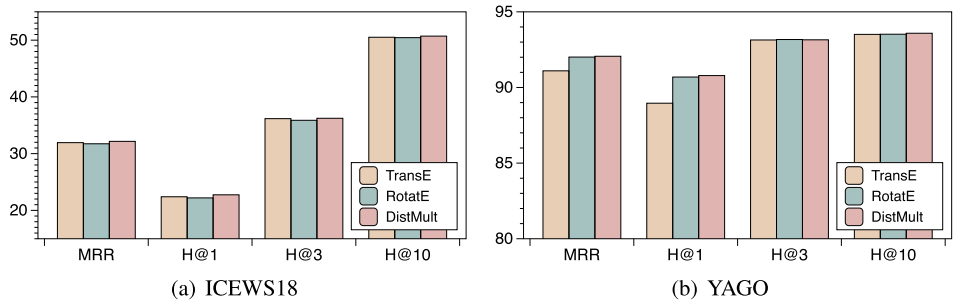

Fig. 5. Ablation results on temporal edge merging method.

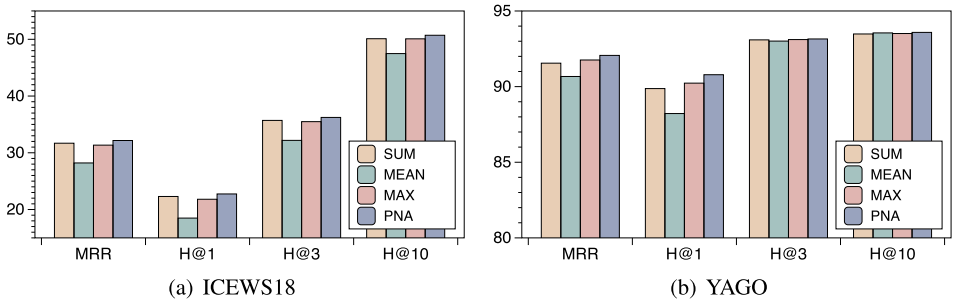

Fig. 6. Ablation results on path aggregation method.

Table 5
Density and Interval of Datasets ($|\mathcal{V}_{avg}|$ is the average number of entities involved in the subgraph of each timestamp).

Datasets	ICEWS18	GDELT	WIKI	YAGO
Interval	24 hours	15 mins	1 year	1 year
$\lvert\mathcal{V}\rvert$	23,033	7,691	12,554	10,623
$\lvert\mathcal{V}_{avg}\rvert$	986.44	393.18	2,817.47	1,190.72
$\frac{\lvert\mathcal{V}\rvert}{\lvert\mathcal{V}_{avg}\rvert}$	23.35	19.56	4.46	8.92

aggregator and performs the worst with the MEAN aggregator, consistently in both YAGO and ICEWS18, except for the H@3 and H@10 metrics on YAGO, where the differences are less pronounced.

4.4. Parameter study (RQ3)

To provide more insights on query-aware temporal path processing, we test the performance of TiPNN with different sampling history lengths m, and the number of temporal path aggregation layers ω.

4.4.1. Sampling history length

In order to comprehensively capture the interconnection patterns between entities in the historical subgraphs, we constructed a history temporal graph to learn query-aware temporal paths between entities, enabling the prediction of future missing facts. Unlike previous methods based on subgraph sequences [8,32], we fuse multiple historical subgraphs for reasoning. However, it may lead to the inclusion of redundant edge information from different timestamps when the historical length is excessively long, resulting in unnecessary space consumption in the learning process. Moreover, facts that are far from the prediction target timestamp have minimal contribution to the prediction, even with the addition of relative distance-aware temporal encoding in the temporal relation encoder. Conversely, if the history length is too short, it can hinder the model's ability to capture cross-time features between entities, thereby reducing the modeling performance for temporal patterns in history temporal graph. Hence, we conducted a discussion on the history length to find a balance. The experimental results of the history length study are shown in Fig. 7.

Since TiPNN initializes the learning of temporal paths based on the query, it requires the history temporal graph's topological structure to include as many entities as possible to ensure that the learned temporal paths from the query subject entity to each candidate object entity capture more temporal connection patterns. We present the density and interval for each dataset, as shown in Table 5, where $|\mathcal{V}|$ denotes the number of entities in the dataset.

For WIKI and YAGO, which have longer time intervals (i.e., 1 year), each subgraph contains more factual information, and the average number of entities in each subgraph is naturally higher. As a result, these datasets demonstrate relatively stable performance

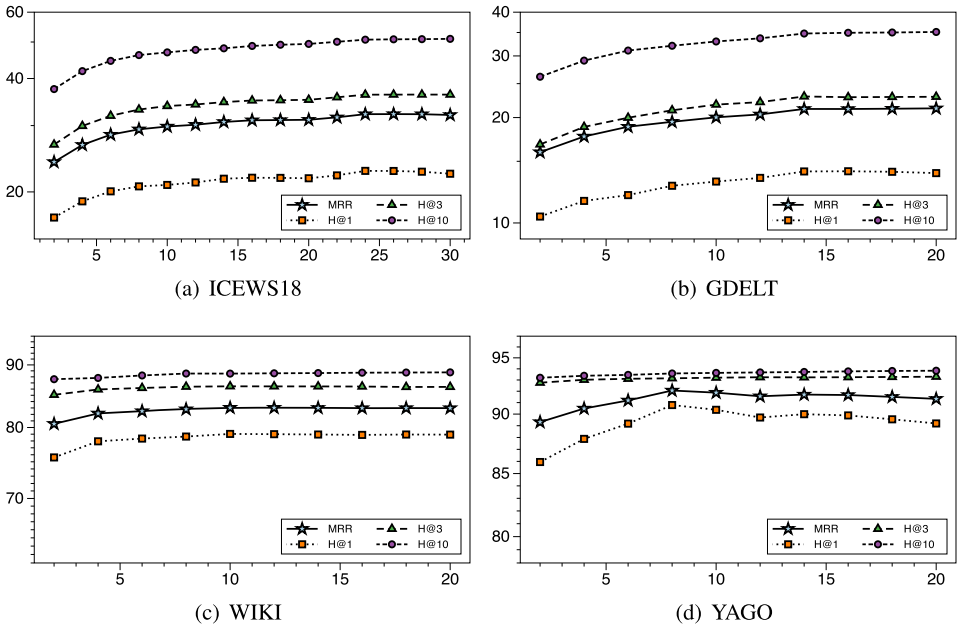

Fig. 7. The performance of different history length settings.

across various history length settings, as shown in Fig. 7(c&d). On the other hand, ICEWS18 and GDELT datasets exhibit different behavior, as depicted in Fig. 7(a&b). Their time intervals are relatively shorter compared to WIKI and YAGO. Consequently, they require longer history lengths to compensate for data sparsity and incorporate more comprehensive historical information in the history temporal graph.

To assess the impact of different history lengths on the model's performance, we calculate the ratio of $|\mathcal{V}|$ to $|\mathcal{V}_{avg}|$ and present it in the last row of Table 5. We observe that the model's performance tends to stabilize around this ratio, providing a useful reference for parameter tuning. Finally, we choose the balanced parameters for ICEWS18, GDELT, WIKI, and YAGO as 25, 15, 10, and 8, respectively.

4.4.2. Number of temporal path aggregation layers

The temporal path aggregation layer is a fundamental computational unit in TiPNN. It operates on the constructed history temporal graph using the temporal edge merging and path aggregation operation to learn query-aware temporal path representations for future fact inference. The number of layers in the temporal path aggregation layer directly impacts the number of hops in the temporal message passing on the history temporal graph, which corresponds to the logical maximum distance of temporal paths. Therefore, setting the number of layers faces a similar challenge as determining the history length in Section 4.4.1.

When the number of layers is too high, TiPNN captures excessively long path information in temporal paths, leading to increased space consumption. However, these additional paths do not significantly improve TiPNN's performance because distant nodes connected to the target node via multiple hops are less relevant for inference of future facts. Therefore, considering distant nodes in the inference process has limited impact on the accuracy of predictions for future facts. On the other hand, if the number of layers is too low, TiPNN may not effectively capture long-distance logical connections in the history temporal graph. This may result in insufficient learning of comprehensive topological connection information and precise combination rules of temporal paths.

To find an optimal setting for the number of layers in temporal path aggregation, we conducted experiments on ICEWS18 and YAGO as shown in Fig. 8, which illustrates the results for varying the number of temporal path aggregation layers. The results indicate that ICEWS18 performs best with 6 layers, and the number of layers has a minor impact on the results. As the number of layers increases, there is no significant improvement in the model's accuracy. On the other hand, YAGO performs best with 4 layers, and beyond 4 layers, the model's performance declines noticeably. We consider that this difference in performance could be attributed to the fact that ICEWS18, as an event-based graph, involves longer-distance logical connections for inferring future facts. On the contrary, YAGO, being a knowledge-based graph, has sparser relation types, resulting in more concise reasoning paths during inference. We finally set the optimal number of layers for ICEWS18 and GDELT to 6, and 4 for WIKI and YAGO.

4.5. Comparison on prediction time (RQ4)

To further analyze the efficiency of TiPNN, we compared the inference time of TiPNN with DaeMon [9] on the TKG reasoning task. For a fair comparison, we use the test sets of four datasets and align the model parameters under the same setting and environment. The runtime comparison between the two models is illustrated in Fig. 9, where we present the ratio of their inference times in terms of multiples of the unit time. The comparative experiments demonstrate that TiPNN achieves significant reductions in runtime

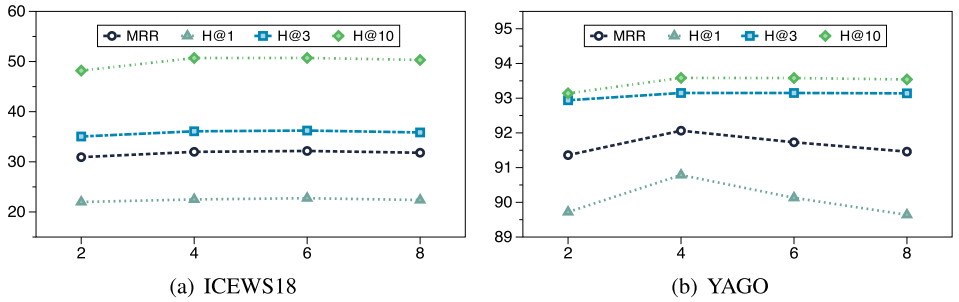

Fig. 8. The performance of different numbers of aggregation layers.

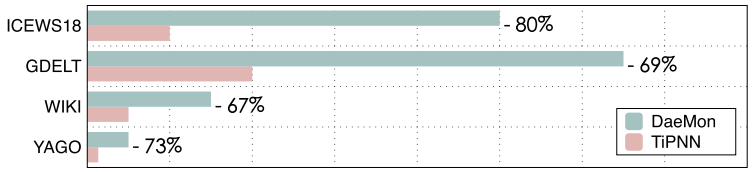

Fig. 9. Runtime comparison (the runtime is proportionally represented as multiples of the unit time).

compared to DaeMon, with time savings of approximately 80%, 69%, 67%, and 73% on the ICEWS18, GDELT, WIKI, and YAGO datasets, respectively.

The efficiency of TiPNN is mainly attributed to its construction of history temporal graphs, which enables the modeling of logical features between temporal facts and the capture of semantic and temporal patterns from historical moments. In contrast, DaeMon relies on graph evolution methods to handle the potential path representations, while TiPNN employs a reduced number of message-passing layers to learn real existing path features in the history temporal graph. Additionally, TiPNN leverages temporal edges to simultaneously capture relational and temporal features, eliminating the need for separate sequence modeling units to learn temporal representations. As a result, TiPNN exhibits lower complexity and higher learning efficiency compared to DaeMon.

4.6. Inductive setting (RQ5)

To validate the model's inductive reasoning ability, we also design experiments for TiPNN in an inductive setting. The inductive setting is a common scenario in knowledge graphs, where during the training process, the model can only access a subset of entities, and during testing, it needs to reason about unseen entities [37]. This setting aims to simulate real-world situations where predictions are required for new facts based on existing knowledge. In the context of the temporal knowledge graph, the inductive setting requires the model to perform reasoning across time, meaning that the learned representations from the historical subgraphs should generalize to future timestamps for predicting missing facts. This demands the model to possess the ability to generalize to unseen entities, leveraging the learned temporal connectivity patterns from the historical subgraphs.

In the inductive setting experiment, we train the model using only a portion of entities and then test its inference performance on the reasoning of the unseen entities. This type of validation allows for a comprehensive evaluation of the model's generalization and inductive reasoning capabilities, validating the effectiveness of TiPNN in predicting future missing facts.

In the context of TKGs, there are few existing datasets suitable for inductive validation. Therefore, we follow the rules commonly used in KG for inductive settings and construct an inductive dataset specifically tailored for TKGs [37]. Finally, we conduct experiments on our proposed model using the inductive dataset to evaluate its inference performance under the inductive setting.

4.6.1. Inductive datasets

In order to facilitate the inductive setting, we create a fully-inductive benchmark dataset by sampling disjoint entities from the YAGO dataset [21]. Specifically, the inductive dataset consists of a pair of TKGs: YAGO^1 and YAGO^2, satisfying the following conditions: (i) they have non-overlapping sets of entities and (ii) they share the same set of relations. To provide a comprehensive evaluation, we have sampled three different versions of the pair inductive dataset based on varying proportions of entity set cardinality. Table 6 provides the statistical data for these inductive datasets, where $|\mathcal{V}|$ denotes the number of entities in the dataset, $|\mathcal{R}|$ denotes the number of relations in the dataset. The labels v1(5:5), v2(6:4), and v3(7:3) in the table correspond to the three different datasets created with different entity set partition ratios.

4.6.2. Results of inductive experiment

In the Inductive setting, we conduct inference experiments and validations by cross-utilizing the training and test sets of a pair of TKGs: YAGO^1 and YAGO^2. Specifically, we train on the training set of YAGO^1 and test on the test set of YAGO^2, and vice versa, to achieve cross-validation in each version of the dataset. Additionally, we include experimental results in the transductive setting as

Table 6
Statistics of Inductive Datasets ($\mathcal{E}_{train}$, $\mathcal{E}_{valid}$, $\mathcal{E}_{test}$ are the numbers of facts in training, validation, and test sets).

Datasets		$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\mathcal{E}_{train}$	$\mathcal{E}_{valid}$	$\mathcal{E}_{test}$	Interval
v1 (5:5)	YAGO^1	3,980	10	39,588	4,681	6,000	1year
	YAGO^2	3,963	10	37,847	4,544	5,530	1year
v2 (6:4)	YAGO^1	4,590	10	46,070	5,685	5,904	1year
	YAGO^2	3,293	10	33,091	4,836	4,244	1year
v3 (7:3)	YAGO^1	5,705	10	64,036	7,866	9,663	1year
	YAGO^2	2,407	10	20,884	2,545	3,180	1year

Table 7
Inductive Performance of TiPNN (→ points from training set to test set corresponding to the inductive setting, and the row results without arrow marked is the corresponding transductive setting as the baseline. 'Bias' denotes represents the difference in comparison).

Dataset	v1				v2				v3			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
YAGO^1	90.85	89.31	92.10	92.58	91.19	89.79	92.35	92.73	91.67	90.46	92.72	93.09
YAGO^2 →YAGO^1	90.75	89.09	92.11	92.64	91.07	89.60	92.27	92.73	91.19	89.58	92.62	92.94
Bias (±)	≤ 0.22				≤ 0.19				≤ 0.88			
YAGO^2	91.33	89.83	92.61	93.02	92.88	91.61	93.93	94.27	90.66	89.22	91.70	92.43
YAGO^1 →YAGO^2	91.15	89.47	92.65	93.07	92.88	91.63	93.90	94.36	90.96	89.66	91.95	92.63
Bias (±)	≤ 0.36				≤ 0.09				≤ 0.44			

a baseline for comparison with the inductive setting. We used 'bias' to display the difference in results between the two settings. As shown in Table 7, the experimental results demonstrate that the bias is within a very small range.

The success of TiPNN in handling the Inductive setting is not surprising. This is because TiPNN models the structural semantics and temporal features of historical facts without relying on any learnable entity-related parameters. Instead, it leverages temporal edge features to model temporal path features relevant to the query. This inherent capability of TiPNN allows it to naturally perform inference on datasets that include unseen entities, making it well-suited for the inductive setting.

4.7. Reasoning evidence (RQ6)

Since we have integrated historical information into the constructed history temporal graph and perform inference on future facts by modeling temporal paths, we can use the path in the history temporal graph to provide evidence for the inference process and visualize the reasoning basis. An intuitive idea is that the model should provide important reasoning paths from the history temporal graph that contribute significantly to the inference for the corresponding response. Although the temporal path representation we obtain is a comprehensive representation logically aggregated from multiple paths in the history temporal graph, we can still estimate the importance of each individual path following the local interpretation method [38,39]. As described in Section 3.5.1, we convert the learned temporal path representation into corresponding scores, and thus, we can estimate the importance of paths through backtracking.

Drawing inspiration from path interpretation [15], we define the importance of a path as the weights assigned to it during the iteration process. We achieve this by computing the partial derivative of the temporal path scores with respect to the path's weights. Specifically, for a reasoning response $(s, r, o, t+1)$ of a missing object query $(s, r, ?, t+1)$, we consider the top-k paths as the basis for inference, which is defined as shown in Equation (16).

$$\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_k = \underset{\mathrm{P} \in \mathcal{P}_{s \to o}^{<t+1}}{\text{Top-}k} \frac{\partial\, p(s, r, o)}{\partial\, \mathrm{P}} \tag{16}$$

In practice, since directly computing the importance of an entire path is facing a challenge, we calculate the importance of each individual temporal edge in the history temporal graph, which can be obtained using automatic differentiation. And then, we aggregate the importance of temporal edges with a summation operation within each path to determine the corresponding path's importance. By applying beam search to traverse and compute the importance of each path, we can finally obtain the top-k most important paths as our reasoning evidence.

We selected several queries from the test set of event-based graph ICEWS18 to conduct an analysis and discussion of the reasoning evidence. As shown in Table 8, it presents the responses corresponding to these queries and their top-2 related reasoning evidence. Here, we provide our analysis and interpretation of the reasoning evidence for the first three queries. (I) For the first query *(Shinzo Abe, Make a visit, ?, 10/23/2018)*, TiPNN's response is China. The most significant reasoning clue is that Shinzo Abe expressed intent to meet or negotiate on 10/22/2018. Additionally, on 10/21/2018, there was also an expression of intent to meet or negotiate, followed by the action of *Make a visit* on 10/22/2018. These two clues are logically consistent and form a reasoning pathway: *Express intent to meet or negotiate → Make a visit*. (II) For the query *(European Union, Engage in diplomatic cooperation, ?, 10/23/2018)*, the

Table 8
Reasoning Evidence of Responses to Selected Queries from ICEWS18. The queries, responses, and the corresponding top-2 reasoning evidence with importance (higher values indicate higher importance) are provided. The attached superscript $^{-1}$ denotes the inverse relations. The timestamp in the quadruple is converted into mm/dd/yyyy format for representation.

Query:	**(Shinzo Abe, Make a visit, ?, 10/23/2018)**
Response:	China
0.784	<Shinzo Abe, Express intent to meet or negotiate, China, 10/22/2018>.
0.469	<Shinzo Abe, Express intent to meet or negotiate, Li Keqiang, 10/21/2018>→ <Li Keqiang, Express intent to meet or negotiate^{-1}, Shinzo Abe, 10/10/2018>→ <Shinzo Abe, Make a visit, China, 10/22/2018>.
Query:	**(European Union, Engage in diplomatic cooperation, ?, 10/23/2018)**
Response:	United Kingdom
1.613	<European Union, Engage in diplomatic cooperation, United Kingdom, 10/14/2018>.
1.044	<European Union, Express intent to engage in diplomatic cooperation (such as policy support), United Kingdom, 10/22/2018>.
Query:	**(Russia, Host a visit, ?, 10/24/2018)**
Response:	Head of Government (Italy)
2.350	<Russia, Make a visit^{-1}, Head of Government (Italy), 10/23/2018>.
1.338	<Russia, Express intent to meet or negotiate^{-1}, Head of Government (Italy), 10/22/2018>.
Query:	**(Citizen (Thailand), Threaten with military force, ?, 10/25/2018)**
Response:	Police (Thailand)
0.951	<Citizen (Thailand), fight with small arms and light weapons^{-1}, Police (Thailand), 10/13/2018>.
0.376	<Citizen (Thailand), Accuse, Thailand, 10/03/2018>→ <Thailand, Return, release person(s), Citizen (Thailand), 10/10/2018>→ <Citizen (Thailand), fight with small arms and light weapons^{-1}, Police (Thailand), 10/13/2018>.
Query:	**(Health Ministry (India), Criticize or denounce^{-1}, ?, 10/27/2018)**
Response:	Citizen (India)
0.757	<Health Ministry (India), Make an appeal or request, Citizen (India), 10/26/2018>.
0.367	<Health Ministry (India), Make statement, Government (India), 10/19/2018>→ <Government (India), Criticize or denounce^{-1}, Citizen (India), 10/26/2018>.

response is United Kingdom. One of the reasoning clues is that on 10/14/2018, European Union engaged in diplomatic cooperation with the United Kingdom, and on 10/22/2018 (the day before the query time), European Union expressed intent to engage in diplomatic cooperation with the United Kingdom. This is easily understandable as it follows a logical reasoning pathway: *Express intent to engage in diplomatic cooperation* → *Engage in diplomatic cooperation*. (III) For the query *(Russia, Host a visit, ?, 10/24/2018)*, the response is Head of Government (Italy). One reliable reasoning clue is that on 10/23/2018, Russia was visited by the Head of Government (Italy). This is also not surprising, as it follows a logical reasoning pathway: *Make a visit*$^{-1}$ → *Host a visit*, which is in line with common sense rules.

Through these examples of reasoning evidence, we can observe that TiPNN is capable of learning temporal reasoning logic from the constructed history temporal graphs. These reasoning clues provide users with more comprehensible inference results and intuitive evaluation criteria. One can utilize these clues to understand the reasoning process for future temporal facts, thereby increasing confidence in the inference results and enabling potential refinements or improvements when needed.

5. Related work

We divide the related work into two categories: (i) knowledge graph reasoning methods, and (ii) temporal knowledge graph reasoning methods.

5.1. Knowledge graph reasoning

In recent years, knowledge graph representation learning has witnessed significant developments, aiming to embed entities and relations into continuous vector spaces while capturing their semantics [23,28,40]. These methods can be broadly categorized into three categories: translation-based methods, semantic matching methods, and neural network-based methods. Translation-based methods, such as TransE [23]. It considers relations as translations from subject entities to object entities in the vector space. Building upon TransE, several improved methods have been proposed, including TransH [41], TransR [42], TransD [43] and TransG [44], which introduce various strategies to enhance the modeling of entity-relation interactions. For semantic matching methods, RESCAL [45] proposes a tensor-based relational learning approach capable of collective learning. DistMult [12] simplifies RESCAL using diagonal

matrices for efficiency, while HoIE [46] and ComplEx [47] extend the representation capacity by incorporating higher-order interactions and complex-valued embeddings [48], respectively. Neural network-based methods have also gained attention, with approaches like GCN [49], R-GCN [50], WGCN [51], VR-GCN [52], and CompGCN [53], which integrate content and structural features within the graph, allowing for joint embedding of entities and relations in a relational graph [54,55]. These methods effectively capture complex patterns and structural dependencies in KGs, pushing the boundary of KG representation learning. Despite the successes of existing methods in reasoning with static KGs, they fall short when it comes to predicting temporal facts due to the lack of temporal modeling.

5.2. Temporal knowledge graph reasoning

An increasing number of studies have started focusing on the representation learning of temporal knowledge graphs, aiming to consider the temporal ordering of facts and capture the temporal knowledge of events. Typically, TKG reasoning methods can be categorized into two main classes based on the range of query timestamps: interpolation reasoning and extrapolation reasoning.

For interpolation reasoning, the objective is to infer missing facts from the past within observed data [56,57,29]. TTransE [20] extends the TransE [23] by incorporating relations and timestamps as translation parameters between entities. TA-TransE [58] integrates the timestamps corresponding to fact occurrences into the relation representations, while HyTE [56] associates timestamps with corresponding hyperplanes. Additionally, based on ComplEx [47], TNTComplEx [30] adopts a unique approach where the TKGs are treated as a 4th-order tensor, and the representations are learned through canonical decomposition. However, these methods are not effectively applicable for predicting future facts [7,59,20,60].

In contrast, the extrapolation setting, which this work focuses on, aims to predict facts in future timestamps based on historical TKG sequences. Know-Evolve [61] uses the temporal point process to represent facts in the continuous time domain. However, it falls short of capturing long-term dependencies. Similarly, DyREP [62] posits representation learning as a latent mediation process and captures the changes of the observed processes. CyGNet [24] utilizes a copy-generation mechanism that collects frequently repeated events with the same subject entities and relations to the query for inferring. RE-NET [6] conducts GRU and GCN to capture the temporal and structural dependencies in a sequential manner. RE-GCN [8] considers both structural dependencies and static properties of entities at the same time, modeling in evolving manner. TANGO [25] designs neural ordinary differential equations to represent and model TKGs for continuous-time reasoning. Most existing methods are primarily transductive in nature, only focusing on modeling the graph features based on entities while overlooking the intrinsic temporal logical rules within TKG reasoning. As an illustration, xERTE [10] assumes that relations do not evolve and constructs an inference graph by sampling nodes from historical subgraphs, learning time-aware entity embeddings on it, yet neglecting complex relationship patterns. From another perspective, some path-based reasoning methods were proposed, which not only provide a new perspective for TKG reasoning but also serve as a reference for this work. For example, TITer [31] augments the historical subgraph sequence by adding auxiliary edges and employs reinforcement learning methods to guide path searching for inference. As a result, there is significant room for development in TKG reasoning, and a need for more efficient methods that can address the robustness and scalability requirements in real-world situations.

6. Conclusion

In this work, we have introduced TiPNN, an innovative query-aware temporal path reasoning model for TKGs reasoning tasks, addressing the challenge of predicting future missing temporal facts under the temporal extrapolated setting. First, we presented a unified graph, namely the history temporal graph, which represents the comprehensive features of the historical context. This constructed graph allows for a more comprehensive utilization of temporal features between historical facts during graph representation learning. Second, we introduced a novel concept of temporal paths, designed to capture query-relevant logical semantic paths on the history temporal graph, which can provide rich structural and temporal context for reasoning tasks. Third, a query-aware temporal path processing framework was also designed, integrated with the introduced temporal edge merging and path aggregation functions. It enables the modeling of temporal path features over the history temporal graph for future temporal fact reasoning.

Overall, the proposed model avoids the need for separate graph learning on each temporal subgraph, making use of the unified graph to represent the information of historical feature to enhance the efficiency of the reasoning process. By starting from the query, capturing and learning over query-aware temporal paths, TiPNN accounts for both structural information and temporal dependencies between entities of separate subgraphs in historical context, and achieves the prediction of future missing facts while offering interpretable reasoning evidence that facilitates users' analysis of results and model fine-tuning. Besides, in learning historical patterns, the modeling process adopts an entity-independent manner, that means TiPNN doesn't rely on specific entity representations, enabling it to naturally handle TKG reasoning tasks under the inductive setting. We have conducted extensive experiments on TKG reasoning benchmark datasets to evaluate the performance of TiPNN. The results demonstrate that the proposed model exhibits superior effectiveness and attains new state-of-the-art achievements.

The path-based, entity-independent approach employed in TiPNN is well-suited for TKG reasoning tasks as it explores interaction patterns among historical entities, deducing logical semantic interactions that occurred in the past. Although this approach has been applied to traditional KG reasoning [11], the explicit presence of time information and implicit temporal patterns in TKGs renders many excellent KG methods unsuitable for practical applications with a temporal dimension. Moreover, TiPNN consolidates historical subgraph sequences into a unified graph, allowing for more comprehensive utilization of interactions among entities at different times, which helps overcome the limitations and complexities associated with traditional history-evolution methods [8,9]

when modeling temporal features. While TiPNN provides a novel reference for the TKG reasoning domain, several challenges remain for further research. First, current TKG research is constrained to validation on small-scale datasets, and the performance potential for large-scale datasets is yet to be fully explored. Second, TKGs can be applied to a variety of practical scenarios such as event analysis and trajectory prediction, offering meaningful extensions beyond their current scope. Third, despite the outstanding performance and maturity of existing KG reasoning methods, efficiently integrating advanced KG techniques into the TKG domain poses a significant challenge.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This research is funded by the Natural Science Foundation of China under Grant No. 61836013, the Science and Technology Development Fund (FDCT), Macau SAR (file no. 0014/2022/AFJ, 0123/2023/RIA2, 001/2024/SKL), the Start-up Research Grant of University of Macau (File no. SRG2021-00017-IOTSC), the Postdoctoral Fellowship Program of CPSF (No. GZC20232736), the China Postdoctoral Science Foundation Funded Project (No. 2023M743565).

References

[1] X. Wang, G. Sun, X. Fang, J. Yang, S. Wang, Modeling spatio-temporal neighbourhood for personalized point-of-interest recommendation, in: L. De Raedt (Ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence, 2022, pp. 3530–3536, 31st International Joint Conference on Artificial Intelligence, IJCAI 2022; Conference date: 23-07-2022 Through 29-07-2022.
[2] M.M. Gervais, Rich economic games for networked relationships and communities: development and preliminary validation in Yasawa, Fiji, Field Methods 29 (2) (2017) 113–129, https://doi.org/10.1177/1525822X16643709.
[3] W. Jin, M. Qu, X. Jin, X. Ren, Recurrent event network: autoregressive structure inference over temporal knowledge graphs, arXiv preprint, arXiv:1904.05530, 2019.
[4] L. Bai, X. Ma, M. Zhang, W. Yu, TPmod: a tendency-guided prediction model for temporal knowledge graph completion, ACM Trans. Knowl. Discov. Data 15 (3) (Apr 2021), https://doi.org/10.1145/3443687.
[5] P. Wang, K. Liu, L. Jiang, X. Li, Y. Fu, Incremental mobile user profiling: reinforcement learning with spatial knowledge graph for modeling event streams, in: KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23–27, 2020, ACM, 2020, pp. 853–861.
[6] W. Jin, M. Qu, X. Jin, X. Ren, Recurrent event network: autoregressive structure inference over temporal knowledge graphs, in: EMNLP, 2020.
[7] R. Goel, S.M. Kazemi, M. Brubaker, P. Poupart, Diachronic embedding for temporal knowledge graph completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 3988–3995.
[8] Z. Li, X. Jin, W. Li, S. Guan, J. Guo, H. Shen, Y. Wang, X. Cheng, Temporal knowledge graph reasoning based on evolutional representation learning, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 408–417.
[9] H. Dong, Z. Ning, P. Wang, Z. Qiao, P. Wang, Y. Zhou, Y. Fu, Adaptive path-memory network for temporal knowledge graph reasoning, arXiv preprint, arXiv:2304.12604, 2023.
[10] Z. Han, P. Chen, Y. Ma, V. Tresp, Explainable subgraph reasoning for forecasting on temporal knowledge graphs, in: International Conference on Learning Representations, 2020.
[11] Z. Zhu, Z. Zhang, L.-P. Xhonneux, J. Tang, Neural Bellman-Ford networks: a general graph neural network framework for link prediction, Adv. Neural Inf. Process. Syst. 34 (2021).
[12] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, arXiv preprint, arXiv:1412.6575, 2014.
[13] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, arXiv preprint, arXiv:2002.07962, 2020.
[14] G. Corso, L. Cavalleri, D. Beaini, P. Liò, P. Veličković, Principal neighbourhood aggregation for graph nets, Adv. Neural Inf. Process. Syst. 33 (2020) 13260–13271.
[15] Z. Zhu, M. Galkin, Z. Zhang, J. Tang, Neural-symbolic models for logical queries on knowledge graphs, arXiv preprint, arXiv:2205.10128, 2022.
[16] S. Miao, M. Liu, P. Li, Interpretable and generalizable graph learning via stochastic attention mechanism, in: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato (Eds.), Proceedings of the 39th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 162, 2022, pp. 15524–15543, https://proceedings.mlr.press/v162/miao22a.html.
[17] L.A. Galárraga, C. Teflioudi, K. Hose, F. Suchanek, AMIE: association rule mining under incomplete evidence in ontological knowledge bases, in: Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 413–422.
[18] P. Xu, J.C.K. Cheung, Y. Cao, On variational learning of controllable representations for text without supervision, in: International Conference on Machine Learning, in: PMLR, 2020, pp. 10534–10543.
[19] K. Leetaru, P.A. Schrodt, GDELT: global data on events, location and tone, 1979–2012, in: ISA Annual Convention, vol. 2, Citeseer, 2013, pp. 1–49.
[20] J. Leblay, M.W. Chekol, Deriving validity time in knowledge graph, in: Companion of the Web Conference, 2018, pp. 1771–1776.
[21] F. Mahdisoltani, J. Biega, F. Suchanek, YAGO3: a knowledge base from multilingual Wikipedias, in: 7th Biennial Conference on Innovative Data Systems Research, CIDR Conference, 2014.
[22] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, M. Ward, ICEWS coded event data, https://doi.org/10.7910/DVN/28075, 2015.
[23] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Adv. Neural Inf. Process. Syst. 26 (2013).
[24] C. Zhu, M. Chen, C. Fan, G. Cheng, Y. Zhang, Learning from history: modeling temporal knowledge graphs with sequential copy-generation networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 4732–4740.

[25] Z. Han, Z. Ding, Y. Ma, Y. Gu, V. Tresp, Learning neural ordinary equations for forecasting future links on temporal knowledge graphs, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 8352–8364.
[26] T. Trouillon, J. Welbl, S. Riedel, R. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, JMLR.org, 2016.
[27] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: 32nd AAAI Conference on Artificial Intelligence, AAAI-18, New Orleans, LA, USA, 2–7 February 2018, 2017.
[28] Z. Sun, Z.H. Deng, J.Y. Nie, J. Tang, Rotate: knowledge graph embedding by relational rotation in complex space, arXiv preprint, arXiv:1902.10197, 2019.
[29] A. Garcia-Duran, S. Dumančić, M. Niepert, Learning sequence encoders for temporal knowledge graph completion, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 4816–4821.
[30] T. Lacroix, G. Obozinski, N. Usunier, Tensor decompositions for temporal knowledge base completion, arXiv preprint, arXiv:2004.04926, 2020.
[31] H. Sun, J. Zhong, Y. Ma, Z. Han, K. He, Timetraveler: reinforcement learning for temporal knowledge graph forecasting, arXiv preprint, arXiv:2109.04101, 2021.
[32] Z. Li, S. Guan, X. Jin, W. Peng, Y. Lyu, Y. Zhu, L. Bai, W. Li, J. Guo, X. Cheng, Complex evolutional pattern learning for temporal knowledge graph reasoning, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 290–296.
[33] H. Sun, S. Geng, J. Zhong, H. Hu, K. He, Graph Hawkes transformer for extrapolated reasoning on temporal knowledge graphs, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, 2022, pp. 7481–7493.
[34] A. Sadeghian, M. Armandpour, P. Ding, D.Z. Wang, DRUM: End-to-End Differentiable Rule Mining on Knowledge Graphs, Curran Associates Inc., Red Hook, NY, USA, 2019.
[35] F. Yang, Z. Yang, W.W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, Adv. Neural Inf. Process. Syst. 30 (2017).
[36] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[37] K.K. Teru, E. Denis, W.L. Hamilton, Inductive relation prediction by subgraph reasoning, arXiv:1911.06962, 2020.
[38] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K. Müller, How to explain individual classification decisions, J. Mach. Learn. Res. 11 (2010) 1803–1831.
[39] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 818–833.
[40] Y. Li, W. Fan, C. Liu, C. Lin, J. Qian, TranSHER: translating knowledge graph embedding with hyper-ellipsoidal restriction, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 8517–8528, https://aclanthology.org/2022.emnlp-main.583.
[41] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28, 2014.
[42] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, 2015.
[43] G. Ji, S. He, L. Xu, K. Liu, J. Zhao, Knowledge graph embedding via dynamic mapping matrix, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015, pp. 687–696.
[44] H. Xiao, M. Huang, Y. Hao, X. Zhu, TransG: a generative mixture model for knowledge graph embedding, arXiv preprint, arXiv:1509.05488, 2015.
[45] M. Nickel, V. Tresp, H.-P. Kriegel, et al., A three-way model for collective learning on multi-relational data, in: ICML, vol. 11, 2011, pp. 3104482–3104584.
[46] M. Nickel, L. Rosasco, T. Poggio, Holographic embeddings of knowledge graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, 2016.
[47] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International Conference on Machine Learning, in: PMLR, 2016, pp. 2071–2080.
[48] Z. Liu, Y. Lin, M. Sun, Representation Learning for Natural Language Processing, Springer Nature, 2020.
[49] M. Welling, T.N. Kipf, Semi-supervised classification with graph convolutional networks, in: J. International Conference on Learning Representations, ICLR 2017, 2016.
[50] M. Schlichtkrull, T.N. Kipf, P. Bloem, R.v.d. Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, arXiv preprint, arXiv:1703.06103, 2017.
[51] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, B. Zhou, End-to-end structure-aware convolutional networks for knowledge base completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3060–3067.
[52] R. Ye, X. Li, Y. Fang, H. Zang, M. Wang, A vectorized relational graph convolutional network for multi-relational network alignment, in: IJCAI, 2019, pp. 4135–4141.
[53] S. Vashishth, S. Sanyal, V. Nitin, P. Talukdar, Composition-based multi-relational graph convolutional networks, arXiv preprint, arXiv:1911.03082, 2019.
[54] Z. Ning, Z. Qiao, H. Dong, Y. Du, Y. Zhou, Lightcake: a lightweight framework for context-aware knowledge graph embedding, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2021.
[55] Z. Qiao, Z. Ning, Y. Du, Y. Zhou, Context-enhanced entity and relation embedding for knowledge graph completion, arXiv preprint, arXiv:2012.07011, 2020.
[56] S.S. Dasgupta, S.N. Ray, P. Talukdar, HyTE: hyperplane-based temporally aware knowledge graph embedding, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2001–2011.
[57] C. Esteban, V. Tresp, Y. Yang, S. Baier, D. Krompaß, Predicting the co-evolution of event and knowledge graphs, in: 2016 19th International Conference on Information Fusion, FUSION, IEEE, 2016, pp. 98–105.
[58] A. García-Durán, S. Dumančić, M. Niepert, Learning sequence encoders for temporal knowledge graph completion, arXiv preprint, arXiv:1809.03202, 2018.
[59] Z. Han, Y. Ma, P. Chen, V. Tresp, DyERNIE: dynamic evolution of Riemannian manifold embeddings for temporal knowledge graph completion, arXiv preprint, arXiv:2011.03984, 2020.
[60] A. Sadeghian, M. Rodriguez, D.Z. Wang, A. Colas, Temporal reasoning over event knowledge graphs, in: Workshop on Knowledge Base Construction, Reasoning and Mining, 2016.
[61] R. Trivedi, H. Dai, Y. Wang, L. Song, Know-evolve: deep temporal reasoning for dynamic knowledge graphs, in: International Conference on Machine Learning, in: PMLR, 2017, pp. 3462–3471.
[62] R. Trivedi, M. Farajtabar, P. Biswal, H. Zha, DyRep: learning representations over dynamic graphs, in: International Conference on Learning Representations, 2019.