Disentangled Multi-span Evolutionary Network against Temporal Knowledge Graph Reasoning

Hao Dong[1,2], Ziyue Qiao[3], Zhiyuan Ning[1,2], Qi Hao[4], Yi Du[1,2], Pengyang Wang[4*], Yuanchun Zhou[1,2]
[1]Computer Network Information Center, Chinese Academy of Sciences,
[2]University of Chinese Academy of Sciences,
[3]School of Computing and Information Technology, Great Bay University,
[4]State Key Laboratory of Internet of Things for Smart City, University of Macau
{donghcn, ziyuejoe, zyningcn}@gmail.com, {yc37440, pywang}@um.edu.mo,
{duyi, zyc}@cnic.cn

Abstract

Temporal Knowledge Graphs (TKGs) incorporate the temporal feature to express the transience of knowledge by describing when facts occur. TKG extrapolation aims to infer possible future facts based on known history, which has garnered significant attention in recent years. Some existing methods treat TKG as a sequence of independent subgraphs to model temporal evolution patterns, demonstrating impressive reasoning performance. However, they still have limitations: 1) In modeling subgraph semantic evolution, they usually neglect the internal structural interactions between subgraphs, which are actually crucial for encoding TKGs. 2) They overlook the potential smooth features that do not lead to semantic changes, which should be distinguished from the semantic evolution process. Therefore, we propose **Di**sentangled **M**ulti-span Evolutionary **Net**work (**DiMNet**) for TKG reasoning. Specifically, we design a multi-span evolution strategy that captures local neighbor features while perceiving historical neighbor semantic information, thus enabling internal interactions between subgraphs during the evolution process. To maximize the capture of semantic change patterns, we design a disentangle component that adaptively separates nodes' active and stable features, used to dynamically control the influence of historical semantics on future evolution. Extensive experiments demonstrate that DiMNet achieves substantial performance in TKG reasoning, outperforming the state-of-the-art up to 22.7% in MRR.

1 Introduction

Factual knowledge and entity semantics evolve over time, revealing intricate temporal properties. Recent studies have introduced Temporal Knowledge Graphs (TKGs) as an extension of static Knowledge Graphs (KGs) to capture time-sensitive facts (Dong et al., 2023, 2024). It introduces a time feature to record the timestamps or time intervals related to events and takes quadruples to represent a temporal fact, denoted as (s, r, o, t), such as $(Edison, BornIn, USA, 02/11/1847)$. TKGs have a broader range of applications, such as crisis warning (Liu and Fan, 2022) and real-time dialogue (Liu and Mazumder, 2021).

Similar to static KGs, TKGs also face the challenge of incompleteness. The task of reasoning on TKGs aims to complete missing links based on given known temporal facts. This includes two settings: interpolation and extrapolation (Jin et al., 2020). The former is used to predict missing facts within a known historical time range, while the latter predicts future facts based on known historical facts (Li et al., 2021). In this work, we focus on extrapolation reasoning that presents significant challenges and a wide scope for investigation.

A few early works start from the future query and extract relevant semantic information and occurrence patterns from history. RE-NET (Jin et al., 2020) is a representative work that encodes historical facts related to the query by RNN to obtain query-related representations. CyGNet (Zhu et al., 2021) directly captures repetitive patterns of facts that share the same entities and relations as the query. Although these methods can make the inference distribution at the prediction stage closer to the future, they overlook the structural information and relative temporal characteristics within the historical facts.

To make more comprehensive use of historical information and deeply capture the developmental patterns of history, several subgraph evolution methods have been proposed, such as RE-GCN (Li et al., 2021), RETIA (Liu et al., 2023), DaeMon (Dong et al., 2023), etc. They view the

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 12697–12707

TKG as a sequence of subgraphs and use GNN-based methods to model local structural dependencies in history, while employing sequence modeling to capture global semantic evolution patterns. However, this approach results in the structural modeling of the constructed subgraphs being independent of each other, hindering the mutual influence of internal structures between subgraphs. TITer (Sun et al., 2021a) connects the sequence of historical subgraphs into a whole by introducing auxiliary edges. TiPNN (Dong et al., 2024) and xERTE (Han et al., 2020a) sample the historical facts of the TKG into a single comprehensive graph, utilizing time encoding to capture the dynamic characteristics of facts. Although these methods break the isolation of subgraph internal structures, they neglect the features of the semantic evolution pattern of TKG.

However, the *internal structural interaction* among subgraphs and the *subgraphs' semantic evolution* over sequence are both crucial. Therefore, **it still remains challenges:**

(1) Existing models only focus on one of them. From the viewpoint of GNN's message passing, the semantic evolution over time essentially involves the influence of neighbors in different subgraphs on the central node, leading to changes in the central node's semantics. To enable beneficial interactions among subgraph structures, the most direct way is to allow the central node to perceive its historical neighbors. However, for a central node, it has neighbors at different distances, and the importance of these neighbors to the central node is distance-related. Therefore, when the central node perceives the historical neighbors' features, the distance between the central node and its neighbors in the historical subgraph should also be considered.

(2) When the neighbor changes along the timeline, the semantics of a node evolve. However, the changes in the neighbor may not be entirely thorough, and each central node also has its inherent attributes. Subgraph semantic evolution is a higher-order implicit modeling process. During the evolution, there are stable semantics of nodes that change relatively smoothly. Although (Li et al., 2021) introduced static entity types in evolutionary modeling, which is not a generalizable method since the duration of stable semantics is often unpredictable. For subgraph evolution, features with stable changes should be distinguished so that the model can maximize the capture of changes in node semantic information.

To this end, we propose the **Di**sentangled **M**ulti-span Evolutionary **Net**work (**DiMNet**) based on an encoder-decoder structure for the TKG reasoning task. The encoder models the evolutionary process of the historical subgraph sequence, and the decoder infers future facts from the evolutionary results. Specifically, in the encoding phase, we propose **Multi-span Evolution** approach based on graph neural network to enable interaction between the internal structures of subgraphs while preserving the modeling of semantic evolution across historical subgraphs. It is designed to perceive the semantic information of historical neighbors with equal spans during the local subgraph structural modeling stage, thus realizing the perception of the distance features of historical neighbors (First challenge). Additionally, during the subgraph evolution process, we design a **Disentangle Component** to dynamically and adaptively separate the mutually exclusive active and stable features of nodes as they evolve over time. The stable feature is used to guide the node evolution without deviating from the node's steady-state characteristics and intrinsic properties. The active feature is to guide the influence of multi-span historical neighbors on the current subgraph, thereby maximizing the modeling of change patterns (Second challenge). In general, this paper makes the following contributions:

- We propose a novel method DiMNet for TKG extrapolation, which models the evolution of historical subgraph sequences from the perspective of semantic change. Through a multi-span evolution strategy and a disentangle component, DiMNet is able to learn node semantic change patterns in a fine-grained manner.
- We design an inference strategy based on sampling virtual subgraphs specifically for the multi-span evolution encoding method to mitigate the issue of future topology uncertainty during the inference phase.
- Extensive experiments on four benchmark datasets demonstrate the effectiveness of DiMNet, showing superior performance compared to baselines for the TKG reasoning task and achieving new state-of-the-art results.

2 Related Work

We briefly review the classic traditional KG reasoning methods and then introduce some recent related TKG reasoning methods.

Traditional KG reasoning methods aim to model static knowledge and have seen significant advancements (Sun et al., 2019; Li et al., 2022a). There are some translation-based methods, such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014), and TransR (Lin et al., 2015), view relations as translations of subject entities to object entities in the vector space. For the semantic matching-based methods, such as RESCAL (Nickel et al., 2011), which measures the semantic matching between entities and relations through a tensor-based relational learning approach. Neural network-based methods, such as R-GCN (Schlichtkrull et al., 2017) and CompGCN (Vashishth et al., 2019), learn representations of entities and relations from the perspective of graph structural features.

To represent facts with temporal feature, recent literature has increasingly focused on learning TKGs. From the perspective of reasoning tasks, they are usually divided into interpolation and extrapolation reasoning. For interpolation, TTransE (Leblay and Chekol, 2018) extends TransE (Bordes et al., 2013) by binding timestamps to relations as translation features. HyTE (Dasgupta et al., 2018) links timestamps to their respective hyperplanes. TNTComplEx (Lacroix et al., 2020), building on ComplEx (Trouillon et al., 2016), treats TKGs as a 4th-order tensor and learns representations via canonical decomposition. However, these methods are not particularly effective for predicting future facts (Goel et al., 2020; Han et al., 2020b; Leblay and Chekol, 2018; Sadeghian et al., 2016), leading to the development of extrapolation methods. CyGNet (Zhu et al., 2021) proposes a copy-generation mechanism to collect repeated events for query head entities and relations. RE-NET (Jin et al., 2020) utilizes sequence modeling and GCN to capture temporal and structural dependencies in TKG sequences. RE-GCN (Li et al., 2021) considers both subgraph structures and static properties, modeling them in an evolving manner. TANGO (Han et al., 2021) employs neural ordinary differential equations for continuous-time reasoning. xERTE (Han et al., 2020a) constructs an enclosing subgraph around the query through iterative sampling and attention propagation for reasoning. DaeMon (Dong et al., 2023) proposes capturing query-aware temporal path features in sequential subgraphs to complete future missing facts. TiPNN (Dong et al., 2024) builds a history temporal graph from subgraph sequences and uses query-aware methods to learn relevant reasoning paths for queries.

As discussed earlier, there is still significant room for improvement in TKG reasoning methods. The proposed methods provide valuable solutions for learning semantic evolution patterns.

3 Preliminaries

A TKG can be formalized as a sequence of static subgraphs arranged in chronological order, i.e., $\mathcal{G} = \{G_1, G_2, ..., G_t, ...\}$. Each G_t in $\mathcal{G}$ can be represented as $G_t = (\mathcal{V}, \mathcal{R}, \mathcal{E}_t)$, where $\mathcal{V}$ is the set of nodes, $\mathcal{R}$ is the set of relation types, and $\mathcal{E}_t$ is the set of edges at timestamp t. Each element in $\mathcal{E}_t$ can be expressed as (s_t, r_t, o_t), describing a relation type $r \in \mathcal{R}$ occurring between the subject node $s \in \mathcal{V}$ and the object node $o \in \mathcal{V}$ at timestamp $t \in \mathcal{T}$, where $\mathcal{T}$ denotes the set of timestamps. Extrapolation reasoning of TKG aims to complete facts that occur in the future. Given an object node query $(s, r, ?, t_q)$ at a future timestamp t_q, we derive the reasoning results by considering all historical known facts $\{(s, r, o, t_i)|t_i < t_q\}$. Without loss of generality, inverse edges $(o, r^{-1}, ?, t_q)$ are also added to the TKG. This allows us to transform the query when predicting the subject, i.e., $(?, r, o, t_q)$, into the form of predicting the object $(o, r^{-1}, ?, t_q)$.

4 Methodology

4.1 Model Overview

The overall framework is illustrated in Figure 2. DiMNet equip a **Multi-span Evolution** strategy to enable the local structure to perceive the historical neighbor information of nodes during the evolution of the subgraph sequence. Through the multi-span approach, historical neighbor features can assist in the fine-grained updating of nodes at the current timestamp. We hope to involve historical neighbors with the same hop when the central node aggregates current different hop neighbors, as shown in Figure 1.

To maximize the modeling of dynamic change patterns in historical sequences, we integrate a cross-time **Disentangle Component** into the gaps between subgraph modeling, which can guide how evolved information influences the subsequent learning of historical subgraphs. It adaptively disentangles the semantic changes of nodes in adjacent subgraphs, separating active and stable features of nodes, and uses them for dynamic control of historical information in the subsequent multi-span evolution process.

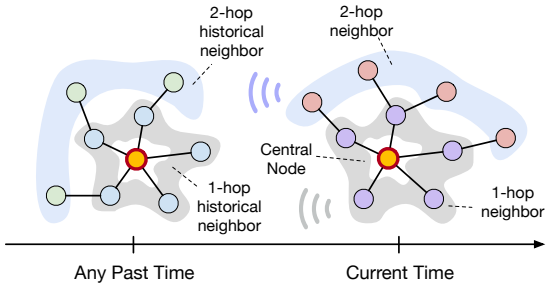


Figure 1: Illustration of the current central node perceiving historical neighbors.

As historical subgraphs evolve node semantics over time, the evolved node representations implicitly contain temporal and semantic features. Existing methods, such as (Li et al., 2021; Zhang et al., 2023), directly score the final representations to complete future facts. However, due to the unknown topology of subgraphs at future timestamps, there might be a distribution shift between the inferred future facts and the actual facts. We design a decoder specifically for multi-span evolution. It constructs an *Inference Virtual Subgraph* based on the query, then revisits the multi-span evolved history to make a query-aware final score.

4.2 Multi-span Evolution Backbone

Given a historical subgraph sequence with length m, i.e., $\{G_{t-m+1}, ..., G_t\}$, the goal of the evolution backbone is to model the subgraphs along the historical timeline into a sequence of node embeddings $\{\hat{\boldsymbol{H}}_{t-m+1}, ..., \hat{\boldsymbol{H}}_t\}$.

In DiMNet, the evolutionary component adopts a GNN-based approach to model the structure of subgraphs at each timestamp. To smoothly incorporate historical neighbor information from different hops, we employ an ω-layers multi-span iteration to integrate intermediate features from previous subgraphs. Each layer is designed in two parts: **message aggregation** and **feature update**.

4.2.1 Message Aggregation Process

For a subgraph at timestamp t, for an object node o at the l-th layer ($l \in [1, \omega]$) is as follows:

$$\boldsymbol{h}^l_{o,t} = \texttt{Agg}\Big(\Big\{\underbrace{\boldsymbol{W}^l_{nbr}(\ddot{\boldsymbol{h}}^{l-1}_{s,t} \oplus \boldsymbol{r}^l)}_{NeighborMessage}\Big\} \uplus \Big\{\underbrace{\boldsymbol{W}^l_{sf}\ddot{\boldsymbol{h}}^{l-1}_{o,t}}_{Self-loop}\Big\}\Big). \quad (1)$$

The central node o considers neighbor features and self-loop features during the aggregation. The term $\boldsymbol{r}^l \in \mathbb{R}^d$ denotes the relation type representation between s and o. It is important to note that, considering the different contributions of edges to semantic evolution at different layers, we process the relation embedding in a layer-specific manner using a linear function, i.e., $\boldsymbol{r}^l = \boldsymbol{W}^l_{\texttt{REL}}\boldsymbol{r} + \boldsymbol{b}^l_{\texttt{REL}}$, where $\boldsymbol{r} \in \boldsymbol{\mathcal{R}}$ is a learnable original relation representation. $\boldsymbol{W}^l_{nbr}, \boldsymbol{W}^l_{sf} \in \mathbb{R}^{d\times d}$ are layer-specific transformation parameters for neighbor and self-loop features, respectively. $\oplus$ is defined as the element-wise summation operator. $\uplus$ denotes the operation of merging the two types of feature messages. $\texttt{Agg}(\cdot)$ represents the message aggregation method, for which we use the widely adopted Principal Neighborhood Aggregation (PNA) proposed in (Corso et al., 2020), which leverages multiple aggregators (namely mean, maximum, minimum, and standard deviation) to learn joint features. For $\ddot{\boldsymbol{h}}^{l-1}_{s,t}, \ddot{\boldsymbol{h}}^{l-1}_{o,t} \in \mathbb{R}^d$, they are derived from the previously obtained final updated features, as follows:

$$\ddot{\boldsymbol{H}}^{l-1}_t = \hat{\boldsymbol{H}}^{l-1}_t \oplus \boldsymbol{W}^l_{\texttt{CE}}\hat{\boldsymbol{H}}^l_{t-1}. \quad (2)$$

Here, we use $\ddot{\boldsymbol{H}}^*_* \in \mathbb{R}^{|\mathcal{V}|\times d}$ with uppercase to denote the representations of the entire set of nodes $\mathcal{V}$. For $s, o \in \mathcal{V}$, there is $\ddot{\boldsymbol{h}}^{l-1}_{s,t}, \ddot{\boldsymbol{h}}^{l-1}_{o,t} \in \ddot{\boldsymbol{H}}^{l-1}_t$. $\hat{\boldsymbol{H}}^*_*$ represents the final updated features, which will be introduced in Sec. 4.2.2. $\boldsymbol{W}^l_{\texttt{CE}} \in \mathbb{R}^{d\times d}$ is a trainable transformation matrix used for the weighted summation of updated features from two different timestamps to achieve Cross-time Evolution (CE). Through the above observations, we can conclude that the input of each layer is generated by: the output from the same layer at the previous timestamp, and the output from the previous layer at the current timestamp.

4.2.2 Feature Update after Aggregation

After completing the feature aggregation described in Eq. 1, $\boldsymbol{h}^l_{o,t}$ is expected to have captured the subgraph and historical l-hop neighbor information for node o, and the same applies to other nodes $o' \in \mathcal{V}$. Similarly, we use $\boldsymbol{H}^l_t \in \mathbb{R}^{|\mathcal{V}|\times d}$ to represent the features obtained for the entire set of nodes at timestamp t after l-th layer aggregation. To better capture the temporal pattern of the subgraph sequence and non-trivial influence of historical neighbors on the semantic evolution process, we employ a multi-span gating mechanism to assist in updating the node semantics based on the aggregation process, as follows:

$$\hat{\boldsymbol{H}}^l_t = \boldsymbol{U}^l_t \otimes \boldsymbol{H}^l_t + (1 - \boldsymbol{U}^l_t) \otimes \hat{\boldsymbol{H}}^l_{t-1}. \quad (3)$$

Eq. 3 describes how to obtain the final updated representation $\hat{\boldsymbol{H}}^l_t \in \mathbb{R}^{|\mathcal{V}|\times d}$ of nodes using the aggregated features $\boldsymbol{H}^l_t$ after l-th layer aggregation.

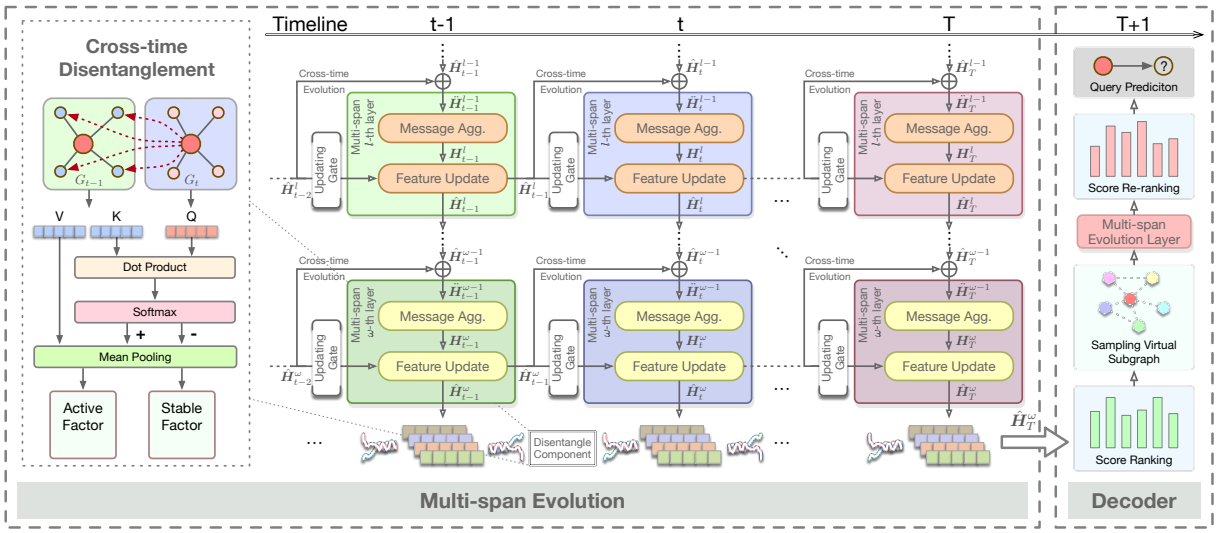


Figure 2: The Overall Architecture of DiMNet.

Here, $\otimes$ is defined as the element-wise multiplication operator. $\boldsymbol{U}_t^l \in \mathbb{R}^{|\mathcal{V}|\times d}$ represents the gating weights at timestamp t, which are obtained by

$$\boldsymbol{U}_t^l = \sigma\big(\boldsymbol{W}_{\texttt{UG}}^l \boldsymbol{\mathcal{A}}_{t-1} + \boldsymbol{b}_{\texttt{UG}}^l\big), \tag{4}$$

where $\boldsymbol{W}_{\texttt{UG}}^l \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_{\texttt{UG}}^l \in \mathbb{R}^d$ are the trainable weights and bias parameters in the Updating Gate (UG) computation process, respectively. As for $\boldsymbol{\mathcal{A}}_{t-1}$, it represents an active factor used to guide the influence of historical neighbors on feature update, that can be disentangled during the evolution process, which will be introduced in Sec. 4.3.

We can see that $\boldsymbol{U}_t^l$, $\boldsymbol{W}_{\texttt{UG}}^l$, and $\boldsymbol{b}_{\texttt{UG}}^l$ are all layer-specific, ensuring that the influence of updated features from different historical layers on the current timestamp's update is unique. This multi-span strategy aligns with our goal by modeling the cross-time same-layer feature transmission, allowing the central node to perceive the semantic changes of historical neighbors.

4.2.3 Feature Initialization

Here we introduce the feature initialization at the starting layer and timestamp. In the training phase, we initialize the trainable embeddings for the set of nodes and relations, denoted as $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}|\times d}$ and $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|\times d}$, respectively. For $\forall t > 0$, which means for timestamps other than the starting timestamp of the subgraph sequence, the input for the 1-st layer is derived as follows:

$$\hat{\boldsymbol{h}}_{o,t}^0 = g\Big(\boldsymbol{p}_{o,t} \,\big|\big|\, \texttt{MP}\{\boldsymbol{\mathcal{R}}_{\mathcal{K}(o)}\}\Big), \tag{5}$$

where $\mathcal{K}(o)$ represents the edges that o serve as objects, $\texttt{MP}\{\cdot\}$ denotes the mean pooling operation performed on relation embeddings corresponding to $\mathcal{K}(o)$, $||$ represents the vector concatenation operation, and g is a 2-layer fully connected network used to reduce the dimensionality to $\mathbb{R}^d$. And $\boldsymbol{p}_{o,t} \in \boldsymbol{P}_t$ is determined as follows:

$$\boldsymbol{P}_t = \boldsymbol{I}_t \otimes \boldsymbol{\mathcal{V}} + (1 - \boldsymbol{I}_t) \otimes \hat{\boldsymbol{H}}_{t-1}^{\omega}, \tag{6}$$

$$\boldsymbol{I}_t = \sigma\big(\boldsymbol{W}_{\texttt{IG}} \boldsymbol{\mathcal{B}}_{t-1} + \boldsymbol{b}_{\texttt{IG}}\big). \tag{7}$$

Similar to Eq. 3 & Eq. 4, a gating mechanism parameterized by Initialization Gate weight $\boldsymbol{W}_{\texttt{IG}}$ is adopted here. $\hat{\boldsymbol{H}}_{t-1}^{\omega}$ represents the features obtained after the final ω-th layer update at $t-1$, ensuring that while evolving across same-layer, the sequential temporal characteristics of the structure are also preserved. And $\boldsymbol{\mathcal{B}}_{t-1}$, in contrast to $\boldsymbol{\mathcal{A}}_{t-1}$ in Eq. 4, represents a stable factor used to guide the initialization of the layer, which will also be introduced in Sec. 4.3. For $t = 0$, $\boldsymbol{P}_t \leftarrow \boldsymbol{\mathcal{V}}$, and $\boldsymbol{\mathcal{A}}_{t-1}, \boldsymbol{\mathcal{B}}_{t-1} \leftarrow \vec{0}$.

4.3 Cross-time Disentanglement

Since node semantic changes occur along with the changes in subgraph structures over time, here we design a cross-time disentangle component to learn how node semantics change after each subgraph structure, utilizing the updated features at two adjacent timestamps, $\hat{\boldsymbol{H}}_t^{\omega}$ and $\hat{\boldsymbol{H}}_{t-1}^{\omega}$, where ω denotes the final layer of aggregation in the evolution. We aim to disentangle a pair of mutually exclusive factors, namely the **active factor** and the **stable factor**, to represent the activity and smoothness of the semantic changes of nodes between timestamp t and $t-1$.

Considering that the semantic changes of $\hat{\boldsymbol{h}}^{\omega}_{o,t}$ relative to $\hat{\boldsymbol{h}}^{\omega}_{o,t-1}$ are due to the differences in the neighbor at timestamp t compared to $t-1$. Therefore, we start from the updated feature of a central node o at timestamp t and the related 1-hop neighbors $\mathcal{N}_{o,t-1}$ at timestamp $t-1$, since 1-hop neighbors have already collected sufficient neighbor features in $\hat{\boldsymbol{H}}^{\omega}_{t-1}$, where $\mathcal{N}_{o,t-1} = \{s|(s,r,o) \in G_{t-1}\}$. Specifically, we design a mutually exclusive attention mechanism to disentangle the adjacent updated states. For $\forall s \in \mathcal{N}_{o,t-1}$ and $(s,r,o) \in G_{t-1}$, we calculate the attention vectors:

$$\boldsymbol{Q}^t_o = \boldsymbol{W}_{\mathtt{Q}}(\hat{\boldsymbol{h}}^{\omega}_{o,t} \,||\, \boldsymbol{r}), \tag{8}$$

$$\boldsymbol{K}^t_s = \boldsymbol{W}_{\mathtt{K}}(\hat{\boldsymbol{h}}^{\omega}_{s,t-1} \,||\, \boldsymbol{r}), \tag{9}$$

$$\boldsymbol{V}^t_s = \boldsymbol{W}_{\mathtt{V}}(\hat{\boldsymbol{h}}^{\omega}_{s,t-1}), \tag{10}$$

where $\hat{\boldsymbol{h}}^{\omega}_{s,t-1}$ represents the updated feature of node o's neighbor s at timestamp $t-1$. $\boldsymbol{W}_{\mathtt{Q}}, \boldsymbol{W}_{\mathtt{K}} \in \mathbb{R}^{2d\times d}$, $\boldsymbol{W}_{\mathtt{V}} \in \mathbb{R}^{d\times d}$ denotes the weight matrices for query, key and value vector, respectively. Note that in the computation of $\boldsymbol{Q}^t_o$ and $\boldsymbol{K}^t_s$, we concatenate the relation embedding $\boldsymbol{r}$ to distinguish the impact of different edge types. For simplicity, we omit the bias terms here. Then, we can calculate the attention score between the neighbor $s \in \mathcal{N}_{o,t-1}$ and the central node o through the dot product operation, as follows:

$$e_{<s,o>} = \frac{\boldsymbol{Q}^t_o \cdot \boldsymbol{K}^t_s}{\sqrt{d}}. \tag{11}$$

The positive and negative forms of $e_{<s,o>}$ are passed through the *softmax* function to thus obtain two inversely proportional normalized scores:

$$\eta^t_{s,o} = \frac{\exp(e_{<s,o>})}{\sum_{u\in\mathcal{N}_{o,t-1}} \exp(e_{<u,o>})}, \tag{12}$$

$$\overline{\eta}^t_{s,o} = \frac{\exp(-e_{<s,o>})}{\sum_{u\in\mathcal{N}_{o,t-1}} \exp(-e_{<u,o>})}. \tag{13}$$

Here, we denote η as the active score and $\overline{\eta}$ as the stable score. Intuitively, a neighbor with higher η will have lower $\overline{\eta}$. Finally, we perform mean pooling on all the neighbor features $\boldsymbol{V}^t_s$ based on the two scores to obtain a pair of mutually exclusive factors, namely active factor $\boldsymbol{\alpha}$ and stable factor $\boldsymbol{\beta}$:

$$\forall u \in \mathcal{N}_{o,t-1},\ \ \boldsymbol{\alpha}^o_t = \mathtt{GRU}(\boldsymbol{\alpha}^o_{t-1}, \mathtt{MP}\{\eta^t_{u,o}\boldsymbol{V}^t_u\}), \tag{14}$$

$$\forall u \in \mathcal{N}_{o,t-1},\ \ \boldsymbol{\beta}^o_t = \mathtt{MP}\{\overline{\eta}^t_{u,o}\boldsymbol{V}^t_u\}. \tag{15}$$

Note that $\boldsymbol{\alpha}^o_t \in \boldsymbol{\mathcal{A}}_t$ and $\boldsymbol{\beta}^o_t \in \boldsymbol{\mathcal{B}}_t$. As mentioned in Sec. 4.2, they are used to guide the dynamic gating parameters $\boldsymbol{U}_*$ and $\boldsymbol{I}_*$ (corresponding to Eq. 4 and Eq. 7, respectively). Here we see that the calculation of $\boldsymbol{\alpha}^o_t$ incorporates a GRU iterative process. This endows the active factor with temporal characteristics, allowing it to learn the temporal patterns of activity along the subgraph sequence and thus providing better temporal features to guide semantic evolution.

4.4 Decoder and Inference

After the subgraph sequence of a TKG undergoes multi-span evolution, the complex evolutionary features of node semantics are captured. Based on this, we design an inference scheme specifically for the multi-span evolution encoding.

4.4.1 Score Function

Given a historical subgraph sequence ending with G_T, where T is the final timestamp in the given historical sequence, the final evolution result $\hat{\boldsymbol{H}}^{\omega}_T$ retains all the semantic and temporal information from the subgraph sequence. Therefore, for the query $(s,r,?)$ at timestamp $T+1$, we consider using $\hat{\boldsymbol{H}}^{\omega}_T$ to decode and calculate the probability of interaction between subject node s and object candidates $\forall o \in \mathcal{V}$ under the relation r.

Specifically, we use the widely adopted scoring function ConvTransE (Shang et al., 2019) to score the missing object node following Eq. 16, where $\hat{\boldsymbol{h}}^{\omega}_{s,T}, \hat{\boldsymbol{h}}^{\omega}_{o,T} \in \hat{\boldsymbol{H}}^{\omega}_T$, $\mathcal{C}$ denotes ConvTransE layer, σ is sigmoid function.

$$\phi_{T+1}(o|s,r) = \sigma\Big(\hat{\boldsymbol{h}}^{\omega}_{o,T}\mathcal{C}(\hat{\boldsymbol{h}}^{\omega}_{s,T}, \boldsymbol{r})\Big) \tag{16}$$

4.4.2 Inference on Virtual Subgraph

We note that the topologies between subgraphs are mutually independent. This means that when predicting the subgraph at a future timestamp, solely relying on the updated features $\hat{\boldsymbol{H}}^{\omega}_T$ obtained at timestamp T is insufficient, as DiMNet does not incorporate query-related features during the encoding phase. Therefore, inspired by the query-aware method (Dong et al., 2023, 2024), we sample a few edges using the scoring results derived by Eq. 16 to form a virtual graph, which is used for the final scoring inference, as follows:

$$G^{T+1}_{\mathtt{INF}} = \Big\{(s,r,o_i)\Big|o_i \in \text{Top-}k\big[\Phi_{T+1}(s,r)\big]\Big\}, \tag{17}$$

where $\Phi_{T+1}(s,r)$ represents the scores of all candidate tail entities corresponding to the query $(s,r,?)$,

i.e., $\phi_{T+1}(o_i|s,r) \in \Phi_{T+1}(s,r)$. s and r are the subject and relation of the query at timestamp $T+1$. Top-$k[\cdot]$ denotes sampling top k candidates with the highest scores.

Based on the virtual graph $G^{T+1}_{\texttt{INF}}$, we perform multi-span evolution (Eq. 1-7) once again to obtain $\hat{\boldsymbol{H}}^{\omega}_{T+1}$, which is used to fed into the scoring function again to obtain the final scoring results:

$$\phi'_{T+1}(o|s,r) = \sigma\Big(\hat{\boldsymbol{h}}^{\omega}_{o,T+1}\mathcal{C}(\hat{\boldsymbol{h}}^{\omega}_{s,T+1}, \boldsymbol{r})\Big). \quad (18)$$

4.5 Learning and Optimization

Predicting missing facts can be viewed as a multi-label learning task. Therefore, based on the final scoring results, the prediction loss is formalized as:

$$\mathcal{L}_{pred} = \sum_{t=1}^{|\mathcal{T}|} \sum_{(s,r,o)\in G_t} \sum_{u\in\mathcal{V}} y^u_t \log \phi'_t(u|s,r), \quad (19)$$

where $|\mathcal{T}|$ represents the number of timestamps in the training set. And y^u_t is the ground truth label, where the value is 1 indicates that the fact occurs and 0 otherwise.

Additionally, during the evolution process, we disentangle the updated features of adjacent timestamps to obtain the active factor $\boldsymbol{\alpha}$ and the stable factor $\boldsymbol{\beta}$. To ensure that the stable factor captures smooth features during training, we introduce an additional distance constraint loss:

$$\mathcal{L}_{dis} = \sum_{t=1}^{m} \sum_{u\in\mathcal{V}} \Big(1 - \text{CosSim}[\boldsymbol{\beta}^u_{t-1}, \boldsymbol{\beta}^u_t]\Big). \quad (20)$$

Note that, for simplicity, we omit the outer summation symbol over the absolute timestamps in the training set, and use t to denote the relative timestamps in the input historical sequence, with m representing the length of the historical sequence. CosSim$[\cdot]$ denotes the computation of the cosine similarity between vectors, used to ensure that the stable factors obtained at adjacent timestamps are similar. Based on the above discussion, the final loss is $\mathcal{L} = \mathcal{L}_{pred} + \mathcal{L}_{dis}$.

5 Experiments

5.1 Experimental Setup

Datasets. We adopt four widely used datasets: ICEWS14 (Han et al., 2020a), ICEWS05-15 (Li et al., 2021), ICEWS18 (Jin et al., 2020), and GDELT (Leetaru and Schrodt, 2013). Specifically, all of ICEWS datasets are the subsets generated from the Integrated Crisis Early Warning System (Boschee et al., 2015), which contains political events with specific timestamps. GDELT is derived from the Global Database of Events, Language, and Tone. We follow the data processing strategies mentioned in (Jin et al., 2020; Dong et al., 2024), which split the dataset into train/valid/test by timestamps that (timestamps of the train) $<$ (timestamps of the valid) $<$ (timestamps of the test). Detailed dataset statistics are provided in Table 2.

Evaluation Metrics. We use Mean Reciprocal Rank (MRR) and Hits@{1, 3, 10} as performance evaluation metrics. Some existing methods use filtered metrics that exclude all valid quadruples in the dataset, but it has been shown to be unsuitable for TKG reasoning tasks (Dong et al., 2024; Han et al., 2021), thus we report the more reasonable time-aware filtered metrics, which only filter out valid facts at the query timestamp from rank list.

Baseline Methods. We conduct a comparison with interpolated and extrapolated TKG methods. (1) Interpolated TKG: TTransE (Leblay and Chekol, 2018), TA-DistMult (Garcia-Duran et al., 2018), DE-SimplE (Goel et al., 2020), and TNT-ComplEx (Lacroix et al., 2020). (2) Extrapolated TKG: TANGO-Tucker (Han et al., 2021), TANGO-DistMult (Han et al., 2021), CyGNet (Zhu et al., 2021), RE-NET (Jin et al., 2020), RE-GCN (Li et al., 2021), TITer (Sun et al., 2021b), xERTE (Han et al., 2020a), CEN (Li et al., 2022b), GHT (Sun et al., 2022), DaeMon (Dong et al., 2023), and TiPNN (Dong et al., 2024).

5.2 Implementation Details

For all datasets, the embedding dimension d is set to 128, and we use Adam (Kingma and Ba, 2014) for parameter learning with a learning rate of $1e-3$. For the multi-span evolution, we perform a grid search on the history length m and present overview results with the lengths 10, 2, 10, 5, corresponding to the datasets ICEWS14, ICEWS05-15, ICEWS18, and GDELT in Table 1, of which parameter sensitivity analysis is shown in Figure 3. The number of evolution layers ω in each timestamp is set to 3 for the ICEWS14, ICEWS18, GDELT datasets, and 1 for the ICEWS05-15 dataset. Layer normalization and shortcuts are applied for the message aggregation process of each layer. The adopted activation function is $RReLU$. For the cross-time disentanglement, we incorporate a multi-head design into the attention mechanism to learn more complex features. The number of heads is set to 4 for ICEWS14 and ICEWS18, and to 1 for ICEWS05-15 and GDELT. The sampling number

Model	ICEWS14				ICEWS05-15				ICEWS18				GDELT			
	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
TTransE	13.43	3.11	17.32	34.55	15.57	4.80	19.24	38.29	8.31	1.92	8.56	21.89	5.50	0.47	4.94	15.25
TA-DistMult	26.47	17.09	30.22	45.41	24.31	14.58	27.92	44.21	16.75	8.61	18.41	33.59	12.00	5.76	12.94	23.54
DE-SimplE	32.67	24.43	35.69	49.11	35.02	25.91	38.99	52.75	19.30	11.53	21.86	34.80	19.70	12.22	21.39	33.70
TNTComplEx	32.12	23.35	36.03	49.13	27.54	9.52	30.80	42.86	21.23	13.28	24.02	36.91	19.53	12.41	20.75	33.42
TANGO-Tucker	26.25	17.30	29.07	44.18	42.86	32.72	48.14	62.34	28.68	19.35	32.17	47.04	19.42	12.34	20.70	33.16
TANGO-DistMult	24.70	16.36	27.26	41.35	40.71	31.23	45.33	58.95	26.65	17.92	30.08	44.09	19.20	12.17	20.40	32.78
CyGNet	32.73	23.69	36.31	50.67	36.81	26.61	41.63	56.22	24.93	15.90	28.28	42.61	18.48	11.52	19.57	31.98
RE-NET	38.28	28.68	41.43	54.52	43.32	33.43	47.77	63.06	28.81	19.05	32.44	47.51	19.62	12.42	21.00	34.01
RE-GCN	41.78	31.58	46.65	61.51	48.03	37.33	53.85	68.27	30.58	21.01	34.34	48.75	19.64	12.42	20.90	33.69
TITer	41.73	32.74	46.46	58.44	47.69	37.95	52.92	65.81	29.98	22.05	33.46	44.83	15.46	10.98	15.61	24.31
xERTE	40.79	32.70	45.67	57.30	46.62	37.84	52.31	63.92	29.31	21.03	33.40	45.60	18.09	12.30	20.06	30.34
CEN	42.17	32.10	47.59	61.43	46.84	36.38	52.45	67.01	30.84	21.23	34.58	49.67	20.18	12.84	21.51	34.10
GHT	37.40	27.77	41.66	56.19	40.31	29.99	45.04	60.51	29.16	18.99	33.16	48.37	20.13	12.87	21.30	34.19
DaeMon	40.68	31.53	45.58	56.73	44.50	35.55	49.64	60.75	31.85	22.67	35.92	49.80	20.73	13.65	22.53	34.23
TiPNN	41.79	32.76	46.92	58.80	45.35	36.27	50.58	62.16	32.17	22.74	36.24	50.72	21.17	14.03	22.98	34.76
DiMNet	**45.72**	**34.41**	**51.37**	**67.99**	**58.93**	**48.55**	**65.16**	**78.33**	**34.13**	**23.29**	**38.42**	**55.80**	**21.93**	**14.03**	**23.57**	**37.49**

Table 1: Overall Performance Comparison of Different Methods. The best results are highlighted in **bold**.

Datasets	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\mathcal{E}_{train}$	$\mathcal{E}_{valid}$	$\mathcal{E}_{test}$	$\lvert\mathcal{T}\rvert$
ICEWS14	7,128	230	63,685	13,823	13,222	365
ICEWS05-15	10,488	251	368,868	46,302	46,159	4,017
ICEWS18	23,033	256	373,018	45,995	49,545	304
GDELT	7,691	240	1,734,399	238,765	305,241	2,976

Table 2: Statistics of Datasets ($\mathcal{E}_{train}$, $\mathcal{E}_{valid}$, $\mathcal{E}_{test}$ denote the numbers of facts in train/valid/test sets.).

k is set to 50 for all datasets, during the construction of the virtual subgraph for scoring. The maximum epoch in the training stage has been set to 60. All experiments were conducted with TESLA A100 GPUs. Code and datasets are available at https://github.com/hhdo/DiMNet.

5.3 Experimental Results

Table 1 presents the overall performance of DiMNet in TKG reasoning task. From the results, we find DiMNet outperforms all baseline methods on the four TKG datasets. Specifically, it surpasses the existing best methods in terms of MRR by 8.4%, 22.7%, 6.1%, and 3.6% on ICEWS14, ICEWS05-15, ICEWS18, and GDELT datasets, respectively.

Compared to interpolated TKG reasoning methods, DiMNet considers the temporal characteristics of facts and employs an evolutionary modeling strategy to maximize the capture of semantic evolution features. This gives it a natural advantage in TKG reasoning, showcasing excellent performance in completing future missing facts. In comparison with existing extrapolated TKG reasoning methods, DiMNet still demonstrates outstanding performance. Unlike previous methods, in modeling the semantic evolution of historical subgraph sequences, DiMNet incorporates a multi-span strategy that allows the model to perceive intermediate features of historical neighbor semantic updates while capturing local structural semantics, facilitating the learning of multi-span semantic evolution, which is overlooked by existing evolution-based methods (Li et al., 2021; Liu et al., 2023; Dong et al., 2023). The disentangle component in DiMNet adaptively separates the active and stable features of nodes during semantic changes, guiding the influence of historical neighbor features on subsequent semantic updates in the multi-span evolution process, thereby maximizing the capture of semantic change patterns. Furthermore, the specifically designed decoder synergizes with historical evolutionary modeling to enhance the performance in completing future missing facts.

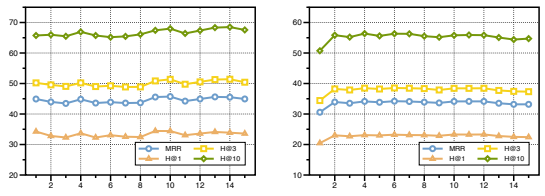
5.4 Ablation Study

To verify the impact of each component, we conducted ablation study on all datasets. The results of variants are shown in Table 3.

Multi-span evolution strategy (denoted as w/o Multi-span): Note that we don't remove the entire evolution strategy but rather disable the perception of historical neighbor features during the local structural semantics capturing. This way, the basic temporal features can still be retained, thereby maximizing the impact of the multi-span strategy. From the results, it is evident that without the perception of historical neighbors, the performance of DiMNet significantly drops as it can no longer capture the fine-grained multi-span updates and evolution.
Disentangle component (w/o Disentangle): We replaced the active factor $\boldsymbol{\mathcal{A}}$ and stable factor $\boldsymbol{\mathcal{B}}$ calculated by disentangle component with zero vectors, removed `GRU` iterative process (in Eq. 14), and deleted $\mathcal{L}_{dis}$ loss term. Results show that without the dynamic disentangled features, DiMNet also suffers significant performance loss because the dis-

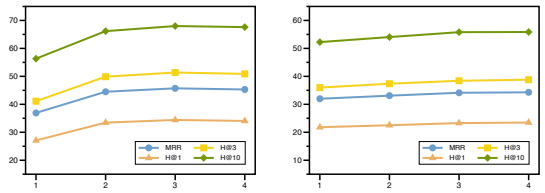
Model	ICEWS14				ICEWS05-15				ICEWS18				GDELT			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
DiMNet	**45.72**	**34.41**	**51.37**	**67.99**	**58.93**	**48.55**	**65.16**	**78.33**	**34.13**	**23.29**	**38.42**	**55.80**	**21.93**	**14.03**	**23.57**	**37.49**
w/o Multi-span	40.75	30.16	45.64	61.62	51.17	40.80	56.97	70.88	30.74	20.67	34.57	50.94	20.99	13.06	22.55	37.03
w/o Disentangle	34.34	24.90	38.16	52.99	53.22	43.14	59.05	72.01	33.60	22.88	37.97	54.91	20.51	12.64	22.56	36.23
w/o Multi-span & Disentangle	36.09	25.35	40.72	57.45	50.36	39.99	55.96	70.25	30.88	20.61	34.74	51.57	20.71	12.69	22.20	36.55
w/o $G_{\texttt{INF}}$	36.10	26.42	40.33	54.96	45.45	35.15	50.93	65.22	30.02	20.25	33.61	49.37	19.81	12.54	21.10	33.92

Table 3: Experimental Results of Ablation Study.



(a) ICEWS14 (b) ICEWS18

Figure 3: Performance on Different Sequence Length.



(a) ICEWS14 (b) ICEWS18

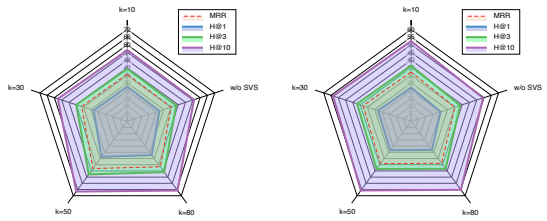
Figure 4: Performance on Different Layer Number.

entangle component adaptively separates complex changing features while providing updated guidance for multi-span evolution. Additionally, when both the multi-span mechanism and disentangle component are removed (denoted as w/o Multi-span & Disentangle), DiMNet almost degrades to the stacking of $m * \omega$ GNN layers, of which the results are still not satisfactory. **Sampling virtual subgraph** (w/o $G_{\texttt{INF}}$): We also attempted to remove the inference strategy using the virtual subgraph for re-scoring, and the results were similarly poor. This aligns with our intuition and further proves the effectiveness of DiMNet's design.

5.5 Parameter Study

Here we test the performance on different historical sequence lengths m, GNN layer numbers ω, and virtual subgraph sampling numbers k on ICEWS14 and ICEWS18 datasets.

Analysis of Sequence Length m: As shown in Figure 3, historical sequence length m does not significantly affect performance on both datasets. Especially for ICEWS18, the performance change tends to stabilize, demonstrating the robustness of DiMNet. In practice, to balance performance and complexity, we set the optimal number of m to 10 for both ICEWS14 and ICEWS18.



(a) ICEWS14 (b) ICEWS18

Figure 5: Performance on Different Virtual Subgraph Sampling Number.

Analysis of Layer Number ω: As shown in Figure 4, it indicates that ICEWS14 and ICEWS18 perform best with 3 layers. Additionally, the layer number has a relatively sensitive impact on performance for the ICEWS14 dataset, and the model performance tends to stabilize as the number of layers increases in both of the datasets.

Analysis of Sampling Number k: The results are shown in Figure 5, where w/o SVS indicates the removal of Sampling Virtual Subgraph. We can see that it significantly improves model's performance. However, for ICEWS14, when k increases from 50 to 80, the model performance shows a declining trend. This is because the sampled subgraph is based on scoring, and a larger k value introduces more erroneous noise, leading to poorer results.

6 Conclusion

In this paper, we propose DiMNet for TKG reasoning, which models historical subgraph evolution from a semantic change perspective. Based on message passing, the multi-span evolution strategy enables interactions among subgraphs while capturing semantic evolution. To enhance modeling of semantic changes, we introduce a disentangle component that separates active and stable features, guiding updates in subsequent subgraphs. We further design an inference strategy using virtual subgraph sampling to address uncertainty in future topology. Extensive experiments show that DiMNet yields substantial improvements and achieves new state-of-the-art results.

Limitations

The limitations of this work can be summarized in two points. First, while the multi-span evolution strategy effectively encodes historical subgraph sequences in temporal knowledge graphs, it heavily relies on local neighborhood information within the graph structure. Therefore, DiMNet may face performance bottlenecks when dealing with sparse graph structures. Second, the disentangle component in DiMNet can adaptively separate the mutually exclusive active and stable node features. However, due to its black-box modeling manner, it is challenging to generate interpretable visualizations of these disentangled features. Designing a robust and interpretable framework for semantic evolution modeling could be a promising direction for future work.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (No.92470204), the Science and Technology Development Fund (FDCT), Macau SAR (file no. 0123/2023/RIA2, 001/2024/SKL), the National Natural Science Foundation of China (No. 62406056), and the Youth Innovation Promotion Association CAS.

References

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

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

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271.

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

Hao Dong, Zhiyuan Ning, Pengyang Wang, Ziyue Qiao, Pengfei Wang, Yuanchun Zhou, and Yanjie Fu. 2023. Adaptive path-memory network for temporal knowledge graph reasoning. *arXiv preprint arXiv:2304.12604*.

Hao Dong, Pengyang Wang, Meng Xiao, Zhiyuan Ning, Pengfei Wang, and Yuanchun Zhou. 2024. Temporal inductive path neural network for temporal knowledge graph reasoning. *Artificial Intelligence*, page 104085.

Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821.

R. Goel, S. M. Kazemi, M. Brubaker, and P. Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3988–3995.

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

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8352–8364.

Zhen Han, Yunpu Ma, Peng Chen, and Volker Tresp. 2020b. Dyernie: Dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion. *arXiv preprint arXiv:2011.03984*.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

T. Lacroix, G. Obozinski, and N. Usunier. 2020. Tensor decompositions for temporal knowledge base completion. *arXiv preprint arXiv:2004.04926*.

J. Leblay and M. W. Chekol. 2018. Deriving validity time in knowledge graph. In *Companion of the the Web Conference*, pages 1771–1776.

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

Yizhi Li, Wei Fan, Chao Liu, Chenghua Lin, and Jiang Qian. 2022a. TranSHER: Translating knowledge graph embedding with hyper-ellipsoidal restriction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8517–8528, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutional representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 408–417.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Bing Liu and Sahisnu Mazumder. 2021. Lifelong and continual learning dialogue systems: learning during conversation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15058–15063.

Kangzheng Liu, Feng Zhao, Guandong Xu, Xianzhi Wang, and Hai Jin. 2023. Retia: relation-entity twin-interact aggregation for temporal knowledge graph extrapolation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1761–1774. IEEE.

Xiaoli Liu and Min Fan. 2022. Identification and early warning of financial fraud risk based on bidirectional long-short term memory model. *Mathematical Problems in Engineering*, 2022.

Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. 2011. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584.

Ali Sadeghian, Miguel Rodriguez, Daisy Zhe Wang, and Anthony Colas. 2016. Temporal reasoning over event knowledge graphs. In *Workshop on Knowledge Base Construction, Reasoning and Mining*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

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

H. Sun, J. Zhong, Y. Ma, Z. Han, and K. He. 2021a. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. *arXiv preprint arXiv:2109.04101*.

Haohai Sun, Shangyi Geng, Jialun Zhong, Han Hu, and Kun He. 2022. Graph hawkes transformer for extrapolated reasoning on temporal knowledge graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7481–7493.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021b. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. *arXiv preprint arXiv:2109.04101*.

Z. Sun, Z. H. Deng, J. Y. Nie, and J. Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

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

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*.

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

Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. 2023. Learning long-and short-term representations for temporal knowledge graph reasoning. In *Proceedings of the ACM Web Conference 2023*, pages 2412–2422.

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