



Temporal Knowledge Graph Embedding for Link Prediction

Yi Zhang¹, Zhi Deng², Dan Meng³, Liang Zhou¹, Mengfei Li¹, Qijie Liu¹,
and Chao Kong¹(✉)

¹ School of Computer and Information, Anhui Polytechnic University, Wuhu, China
zhangyi@ahpu.edu.cn, lzhou@ahpu.edu.cn, lmf@stu.ahpu.edu.cn,
lqj@stu.ahpu.edu.cn, kongchao@ahpu.edu.cn

² School of Computer Science, Northwestern Polytechnical University, Xi'an, China
dengcai@mail.nwpu.edu.cn

³ OPPO Research Institute, Shenzhen, China
mengdan@oppo.com

Abstract. Link prediction aims to infer the behavior of the network evolution process by predicting missed or future relationships based on currently observed connections. It has become an attractive area of research since it allows us to understand how networks will evolve. Early studies cast the link prediction task as an entity identifying problem on graphs and adopt vertex representation strategies to perform predictive analysis. Although these methods are effective to some extent, they overlook the special properties of network evolution.

In this paper, we propose a new method named TKGE, short for *Temporal Knowledge Graph Embedding*, to learn the evolutionary representations of temporal knowledge graph for link prediction task. Specifically, we employ the self-attention mechanism to incorporate the static structural information and dynamic temporal information by aggregating the context from related entities. By introducing the position embedding characterizing the dynamic information of temporal knowledge graph, TKGE can generate the evolutionary embedding of entities and relations for downstream applications, such as link prediction, recommender system, and so on. We conduct experiments on several real datasets. Both quantitative results and qualitative analysis verify the effectiveness and rationality of our TKGE method.

Keywords: Temporal knowledge graph · Representation learning · Self-attention · Link prediction

1 Introduction

The knowledge graph (KG), also known as the semantic network, represents the network of real-world entities and illustrates the relations between them. It

Y. Zhang, Z. Deng and D. Meng—These authors contribute equally to this work.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

X. Zhao et al. (Eds.): WISA 2022, LNCS 13579, pp. 3–14, 2022.

https://doi.org/10.1007/978-3-031-20309-1_1

has been widely used in various applications, such as language representation learning, question answering, recommender systems, and so on.

To date, existing KGs are usually constructed by machine learning algorithms automatically, but there are many hidden relations that have not been observed, so they are often incomplete. In view of this situation, the link prediction task in KG aims to predict the missing entities or relations, which is also called knowledge inference [1] and knowledge completion [2]. The real-world KGs are usually dynamic, in which more new entities and new types of relations are observed in the KGs over time. We call such KGs with dynamic changes over time as temporal knowledge graphs (TKGs). Figure 1 shows an example of a temporal knowledge graph that evolves over time from the year 1998 to 2013. As can be seen from the observed TKG facts in Fig. 1, Tencent was founded by Pony Ma in 1998 and had multiple relations with other entities from 2010 to 2013, resulting in the semantics of the entity of Tencent shifts over time. Hence, modeling temporal information in KGs is crucial to understanding how the knowledge evolves over time. In this paper, we study the problem of learning TKG embedding for link prediction.

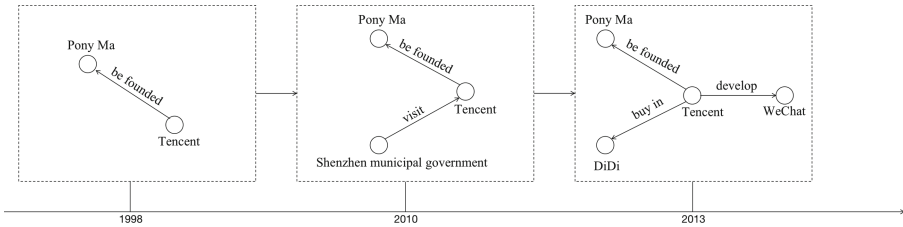


Fig. 1. An example of the evolution of temporal knowledge graph.

However, the TKG embedding for link prediction is often a challenging task due to the following reasons: (1) it is arduous to simulate the strong time dependency in TKG; (2) there are some potential factors that affect the network evolution.

In this paper, we propose an approach, called TKGE, to learn the evolutionary embeddings of entities and relations in TKG. We formulate the link prediction task as a conditional probability problem covering entity prediction and relation prediction. For the provided approach, we would like to address the two challenges highlighted earlier. Specifically, we utilize self-attention mechanism to model TKG from both static structural information and dynamic evolutions of the graph to simulate the complex time dependency better. We employ the multi-head self-attention to model the TKG from many aspects. With multi-head self-attention, TKGE can perform very efficiently but still generate accurate evolutionary embeddings. Extensive results on the six datasets have shown the effectiveness and rationality of the proposed model.

2 Related Work

The link prediction task on TKG aims to predict missing quadruples. It is further divided into two subtasks: entity prediction and relation prediction. Traditional research on KG link prediction tasks mainly focuses on static KGs, that is, facts do not change with time, and less research on TKGs. However, the time sequence information in the TKG is very helpful to capture the dynamic trend of facts, and the interactive data in the real world is not invariable.

In general, TKG link predictions are categorized in two settings: interpolation and extrapolation. Given a TKG with the time interval $[t_0, t_n]$, interpolation aims to predict missing facts at timestamp t such that $t_0 \leq t \leq t_n$ and extrapolation, which aims to predict new facts at timestamp t such that $t \geq t_n$. For interpolation, DE-Simple [3] draws inspiration from diachronic word embeddings, developing a diachronic embedding function to generate a hidden representation for the entity at any given time. However, it is hard to help supplement the KG on future timestamps.

For extrapolation, existing methods are achieved through various techniques such as temporal point process framework [4, 5], CNN-based methods [6] and deep recurrent models [7, 8]. Mei et al., propose N-SM-MPP [4] model for link prediction using the Hawkes process to capture the dynamic temporal information. In addition, DyRep [5] considers representation learning as a latent mediation process. However, these two methods are more suitable for model TKG in continuous time. For the multi-relational, directed graph structure of each KG, REGCN [6] utilize GCN, which it's quite a powerful model for graph-structured data to characterize structural dependencies. Jin et al., develop an autoregressive framework [7], called RE-NET, utilizing recurrent event encoder and neighborhood aggregator to model the information of dynamic temporal and concurrent events in the same timestamp. Despite the effectiveness of the methods mentioned above, these methods have limitations to some degree, because they do not consider the special properties of TKG that is there exist some potential factors that affect the network evolution.

3 Problem Formulation

We first give notations used in this paper and then formalize the TKG representation learning problem to be addressed.

Notations. A TKG is defined as a sequence of static KGs with timestamps, $G = \{G_1, \dots, G_n\}$ where n represents the number of timestamps. Each KG G_t is a multi-relational, directed graph which contains all the facts that co-occur at timestamp t and we define it as $G_t = \{V, R, \mathcal{E}_t\}$ where V denotes the set of entities, R denotes the set of relations, and \mathcal{E}_t denotes the set of facts at timestamp t . Any fact in \mathcal{E}_t can be represented as a quadruple (subject entity, relation, object entity, timestamp) and is denoted by (s, r, o, t) , where $s, o \in V$ and $r \in R$.

Problem Definition. The TKG representation learning task aims to learn latent representations $\mathbf{h}_{o,t} \in \mathbb{R}^{F'}$ and $\mathbf{r}_t \in \mathbb{R}^{F'}$ for each node $o \in V$ and relation

$r \in R$ at time $t = \{1, 2, \dots, n\}$, such that $\mathbf{h}_{o,t}$ preserves both the graph structures and dynamic temporal information. Formally, the problem can be defined as:

Input: A sequence of static KGs with timestamps, $G = \{G_1, \dots, G_n\}$, entity embedding matrices \mathbf{H}_t and relation embedding matrices \mathbf{R}_t .

Output: A map function $f : V \cup R \rightarrow \mathbb{R}^{F'}$, which maps each entity and relation in G to a F' -dimensional embedding vector.

4 Methodology

In this section, we first introduce our proposed method, TKGE, which consists of two components. The structural self-attention module is used to capture the static structural information of the KG at each timestamp t and the temporal self-attention module is used to capture the evolution of the dynamic KG over time. The overall framework of the model is shown in Fig. 2. First, the structural self-attention module uses self-attention [9, 10] to aggregate the local static structural information of each entity in each timestamp t generating structural embeddings for entities. Then, by introducing position embedding, temporal self-attention module can capture the evolution of KG in different timestamps. Finally, based on the representations of the entities and relations learned above, we can use various scoring functions to perform prediction tasks in future timestamps. It is worth mentioning that, intuitively, some latent facts in the real world can affect the evolution of graphics. This is evident in the case of citation networks, papers of different research fields may expand their citation papers at significantly varying rates. Since we introduce multi-head self-attention mechanism [11, 12] in the structural self-attention module and temporal self-attention module, which can not only characteristic the evolution of the KG from different aspects but also reduce the deviation of prediction and improve the stability of the model.

4.1 Structural Self-attention

To capture the static structural information in the KG at each timestamp t , this component is designed to realize the mapping from a series of TKGs $G = \{G_1, \dots, G_n\}$ to a series of entity embedding matrices $\{\mathbf{H}_1, \dots, \mathbf{H}_t\}$. The initial entity embedding matrix is obtained by random initialization.

Specifically, the structural self-attention mechanism calculates the structural representation of the entity by learning the importance of the relevant entity to the target entity and assigning different weights to each relevant entity: $\mathbf{h}_{o,t} = \sigma(\frac{1}{c_o} \sum \alpha_{s,o} W^s(\mathbf{h}_{s,t} + \mathbf{r}_t))$, where $\mathbf{h}_{s,t}$ and \mathbf{r}_t are the embeddings of entity s and relation r at timestamp t , respectively. c_o is the in-degree of the object entity o . $\sigma(\cdot)$ is a nonlinear activation function. In particular, $\mathbf{h}_{s,t} + \mathbf{r}_t$ represents the translation attribute between the entity and the relation in the triplet. The calculation formula of the specific weight $\alpha_{s,o}$ is as follows:

$$\mathbf{e}_{s,o} = \sigma(A_{s,o} \cdot \alpha^T [W^s \mathbf{h}_{ot} || W^s \mathbf{h}_{s,t}]), \forall (s, r, o) \in \mathcal{E}_t, \quad (1)$$

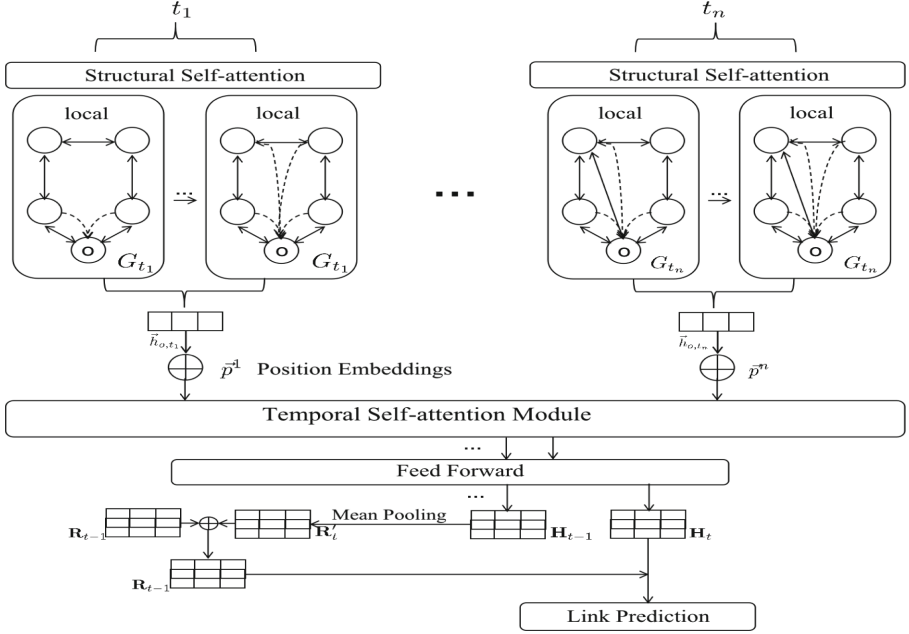


Fig. 2. The overall architecture of the TKGE model.

$$\alpha_{s,o} = \frac{\exp(e_{s,o})}{\sum_{(s,r), \exists(s,r,o) \in \mathcal{E}_t} \exp(e_{s,r})}, \quad (2)$$

where $A_{s,o}$ represents the weight matrix of the link between the subject entity and the object entity at the current timestamp t . In this paper, the number of edge occurrences is used as the weight. $\alpha \in \mathbb{R}^{2D}$ and $W^s \in \mathbb{R}^{F \times D}$ denote the weight vector and the parameter matrices, respectively. \parallel is the concatenation operation and $\sigma(\cdot)$ denotes the non-linear activation function. Significantly, this module is composed of multiple stacked structural self-attention layers. By stacking multiple layers, our method can further consider the high-order relation between entities to generate the final structural embedding of the entity at each timestamp.

4.2 Temporal Self-attention

To capture the dynamic temporal information in TKG, we first utilize position embedding $\{\mathbf{p}^1, \dots, \mathbf{p}^T\}$, $\mathbf{p}^t \in \mathbb{R}^F$ to describe the dynamic temporal sequence information of each static KG. Then, temporal self-attention module takes a series of embeddings for a particular entity o at different timestamps calculated by $\mathbf{h}_{o,t} + \mathbf{p}^t$ as input and returns a sequence of temporal entity embeddings at different timestamps. Meanwhile, this component utilize GRU to realize the mapping from a series of TKGs $G = \{G_1, \dots, G_n\}$ to a

series of relation embedding matrices $\{\mathbf{R}_1, \dots, \mathbf{R}_n\}$. The initial relation embedding matrix is obtained by random initialization. Specifically, the input and the output for each entity o is denoted by $\{\mathbf{x}_{o,t_1}, \dots, \mathbf{x}_{o,t_n}\}, \mathbf{x}_{o,t} \in \mathbb{R}^{D'}$ and $\{\mathbf{h}_{o,t_1}, \dots, \mathbf{h}_{o,t_n}\}, \mathbf{h}_{o,t} \in \mathbb{R}^{F'}$. Where n, D' and F' denote the number of timestamps and the dimension of the input and the output vector. In practice, we compute the input and the output embeddings for entity o at different timestamps, packed together into a matrix $\mathbf{X}_o \in \mathbb{R}^{T \times D'}$ and $\mathbf{H}_o \in \mathbb{R}^{T \times F'}$. Specifically, the output embedding of the entity is mainly calculated according to $\mathbf{H}_o = \beta_o(\mathbf{X}_o \mathbf{W}_v)$, where $\beta_o \in \mathbb{R}^{T \times T}$ is the weight matrix of attention. This calculation first converts the queries, keys, and values to different spaces through the learned matrix $\mathbf{W}_q \in \mathbb{R}^{D' \times F'}$, $\mathbf{W}_k \in \mathbb{R}^{D' \times F'}$, $\mathbf{W}_v \in \mathbb{R}^{D' \times F'}$, and then uses the scaled dot-product attention mechanism to calculate the attention score and weight:

$$e_o^{ij} = \left(\frac{((\mathbf{X}_o \mathbf{W}_q)(\mathbf{X}_o \mathbf{W}_k)^\top)_{ij}}{\sqrt{F'}} + M_{ij} \right). \quad (3)$$

$$B_o^{ij} = \frac{\exp(e_o^{ij})}{\sum_{k=1}^n \exp(e_o^{ik})}, \quad (4)$$

where $\mathbf{M} \in \mathbb{R}^{T \times T}$, is the mask matrix, which is used to encode autoregressive attributes, that is, to use information from a few previous timestamps to describe the information at a later timestamp. If $i \leq j$, $M_{ij} = 0$, otherwise $M_{ij} = -\infty$.

Intuitively, the relation embedding contains the information of the entity in the corresponding fact, so the relation’s embedding at timestamp t will be affected by the entity information related to the relation r at timestamp t and its own information at timestamp $t - 1$, where the related entities is denoted by $E_{r,t} = \{i|(i, r, o, t) \text{ or } (s, r, i, t) \in \mathcal{E}_t\}$. Inspired by [6], by applying the average pooling operation in the embedding matrix of the entities related to the relation r at timestamp $t - 1$, $H_{t-1, E_{r,t}}$, the calculation formula of relation embedding is: $\mathbf{r}'_t = [\text{pooling}(\mathbf{H}_{t-1, E_{r,t}})] \parallel \mathbf{r}$, where \mathbf{r} is the embedding vector of the relation r . Especially, when there is no relation at timestamp t , $\mathbf{r}'_t = 0$. Then, according to the relation embedding matrix \mathbf{R}' and \mathbf{R}_{t-1} at timestamp $t - 1$, this paper utilizes GRU to obtain the updated relationship matrix.

After obtaining the evolutionary embeddings of entities and relations, we perform link prediction problems in a probabilistic way. Considering the good results shown by GCN in KG link prediction, we choose ConvTransE [13] as the decoder. Therefore, the specific calculation formula of the entity conditional probability vector is as follows:

$$\mathbf{p}(o|s, r, \mathbf{H}_t, \mathbf{R}_t) = \sigma(\mathbf{H}_t \text{ConvTransE}(\mathbf{s}_t, \mathbf{r}_t)), \quad (5)$$

where $\sigma(\cdot)$ denotes the sigmoid function. Similarly, we can get the relational conditional probability vector according to the following formula:

$$\mathbf{p}(r|s, o, \mathbf{H}_t, \mathbf{R}_t) = \sigma(\mathbf{R}_t \text{ConvTransE}(\mathbf{s}_t, \mathbf{o}_t)), \quad (6)$$

where $\mathbf{s}_t, \mathbf{r}_t$ and \mathbf{o}_t are the corresponding elements in the entity and relation embedding matrices \mathbf{H}_t and \mathbf{R}_t , respectively.

4.3 Parameter Learning

This paper takes the entity and relation prediction task as a multi-classification problem, where each category corresponds to an entity or relation, and $\mathbf{y}_{t+1}^o \in \mathbb{R}^{|V|}$ or $\mathbf{y}_{t+1}^r \in \mathbb{R}^{|R|}$ is used to represent the label vectors of the entity or relation prediction task under the $t + 1$ timestamp. If a value is 1 indicates the corresponding fact happens. The corresponding loss functions are as follows:

$$L^o = \sum_{t=0}^{n-1} \sum_{(s,r,o,t+1) \in \mathcal{E}_{t+1}} \sum_{i=0}^{|V|-1} y_{t+1,i}^o \log p_i(o|s, r, \mathbf{H}_t, \mathbf{R}_t), \quad (7)$$

$$L^r = \sum_{t=0}^{n-1} \sum_{(s,r,o,t+1) \in \mathcal{E}_{t+1}} \sum_{i=0}^{|R|-1} y_{t+1,i}^r \log p_i(r|s, o, \mathbf{H}_t, \mathbf{R}_t), \quad (8)$$

where n represents the number of timestamps in the training set, $y_{t+1,i}^o$ and $y_{t+1,i}^r$ is the value of the i -th element of \mathbf{y}_{t+1}^o and \mathbf{y}_{t+1}^r respectively.

4.4 Discussion

To verify the efficiency of our algorithm, this section analyzes the time complexity of the model. First, the self-attention mechanism mainly includes three steps: similarity calculation, softmax, and weighted summation. After the analysis, the time complexity of this part is $O(n^2d)$, where n is the maximum length of the sequence, and d is the dimension of embedding. In addition, we use GRU component to calculate relation embedding, where the time complexity of pooling operation is $O(|R|D)$, where D is the maximum number of related entities with relation r at timestamp t , and R is the number of elements in the relation set. Therefore, the total time complexity is $O(n^2d + |R|D)$.

5 Experiments

To evaluate the entity and relation embeddings learned by TKGE, we employ them to address the link prediction problem in TKG. Link prediction in TKG can be regarded as a classification task, which is mainly divided into entity prediction and relation prediction. Through experiments, we aim to answer the following research questions:

RQ1: How does TKGE perform compared with state-of-the-art temporal knowledge graph embedding methods?

RQ2: Is the construction of structural and temporal self-attention component helpful to learn more desirable representations for temporal knowledge graph?

RQ3: How do the key hyper-parameters affect the performance of TKGE?

In what follows, we first introduce the experimental settings, and then answer the above research question in turn.

Table 1. Descriptive statistics of datasets.

Name	$ V $	$ R $	$ \varepsilon_{train} $	$ \varepsilon_{valid} $	$ \varepsilon_{test} $	Time interval
ICEWS18	23,033	256	373,018	45,995	49545	24 h
ICEWS14	6,869	230	74,845	8,514	7,371	24 h
ICEWS05-15	10,094	251	368,868	46,302	46,159	24 h
WIKI	12,554	24	539,286	67,538	67,538	1 year
YAGO	10,623	10	161,540	19,523	20,026	1 year
GDELT	7691	240	1,734,399	238,765	305,241	15 mins

5.1 Experimental Settings

Dataset. We utilize six commonly used TKGs datasets: ICEWS18, ICEWS14, ICEWS05-15, WIKI, YAGO, and GDELT. The first three datasets come from the data captured and processed by the Integrated Crisis Early Warning System (ICEW). The data in the GDELT comes from various international news reports, such as the New York Times, Washington Post, etc. The statistics of the datasets are summarized in Table 1.

Evaluation Protocols. To evaluate the performance of link prediction task, we performed the same processing on the dataset as RE-GCN. For the ICEWS14 and ICEWS05-15 datasets, we randomly select 80% of the instances as the training set, 10% of the instances as the validation set, and the remaining 10% as the test set. The details of the dataset division are shown in Table 1. Based on previous research, this paper selects two evaluation indicators commonly used in link prediction, namely Mean Reciprocal Ranks (MRR) and Hits@K.

Baselines. We compare TKGE with two types of baselines: TKG link prediction models under the interpolation setting and TKG link prediction models under the extrapolation setting. For interpolation, we select HyTE [14], TTransE [15] and TA-DistMult [16] as comparison method. Similarly, for extrapolation, we select RGCRN [17], CyGNet [18], RE-NET [7] and RE-GCN [6] as comparison method. Especially, RGCRN is an extended model of GCRN, which originally for the homogeneous graphs. Specifically, RE-NET replace GCN with R-GCN.

Parameter Settings. We implement our method TKGE in TensorFlow and carry it out on Tesla V100. The training epoch is limited to 200. We adopt the Adam optimizer for parameter learning with a learning rate of 10^{-3} . For all models, we set the dimension of embedding as 200 and the batch size as 256 for a fair comparison.

5.2 Performance Comparison (RQ1)

Entity Prediction. In this task, the results of all the comparison methods are presented in Table 2. Firstly, the TKGE is superior to TKG link prediction models under the interpolation setting, such as HyTE, TTransE, and TA-DistMult, because TKGE additionally captures the static structural information of the

Table 2. Entity prediction performance on different datasets.

Model	ICEWS18			ICEWS14			ICEWS05-15		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
HyTE	7.48%	7.34%	16.04%	16.79%	24.85%	43.96%	16.05%	20.16%	34.73%
TTransE	8.46%	8.98%	22.35%	12.84%	15.75%	33.62%	16.52%	20.76%	39.27%
TA-DistMult	16.45%	18.12%	32.52%	26.23%	29.75%	45.25%	27.55%	31.46%	47.33%
RGCRN	23.49%	26.67%	41.97%	33.35%	36.65%	51.52%	35.94%	40.03%	54.62%
CyGNet	24.99%	28.57%	43.55%	34.62%	38.85%	53.12%	35.42%	40.23%	54.43%
RE-NET	26.18%	29.82%	44.39%	35.74%	40.15%	54.85%	36.85%	41.86%	57.63%
RE-GCN	27.50%	31.18%	46.56%	37.72%	42.51%	58.82%	38.22%	43.12%	59.95%
TKGE	28.84%	32.46%	48.74%	39.73%	44.91%	62.79%	39.63%	44.33%	62.33%
Model	WIKI			YAGO			GDELT		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
HyTE	25.48%	29.14%	37.74%	14.49%	39.85%	46.96%	6.65%	7.51%	19.03%
TTransE	20.46%	23.98%	33.05%	26.14%	36.25%	47.62%	5.42%	4.86%	15.27%
TA-DistMult	26.45%	31.37%	38.88%	44.89%	50.71%	61.21%	10.45%	10.46%	21.67%
RGCRN	28.67%	31.47%	38.59%	43.74%	48.54%	59.65%	18.64%	19.79%	32.43%
CyGNet	30.89%	33.87%	41.25%	46.62%	52.55%	61.12%	18.12%	19.13%	31.43%
RE-NET	30.88%	33.82%	41.39%	46.74%	52.75%	61.85%	19.85%	20.86%	33.83%
RE-GCN	39.83%	44.38%	53.85%	52.32%	65.61%	75.82%	19.22%	20.46%	33.28%
TKGE	43.94%	49.67%	60.06%	57.95%	71.96%	82.71%	19.77%	20.58%	32.67%

KG at each timestamp t and the temporal evolution information of the TKG. This improvement demonstrates that the accuracy of evolutionary embedding generated from TKGE has significantly improved. Secondly, the temporal models for the extrapolation setting, such as CyGNet, RE-NET, and RE-GCN, have good performance. The result verifies the effect of the repetitive patterns and the direct neighbors on the entity prediction task. Among them, the RE-GCN is the most efficient method. Compared with CyGNet and RE-NET, RE-GCN considers the static structure information in each timestamp t and the dynamic temporal information over time and can obtain satisfactory evolutionary representations. Thirdly, our proposed method TKGE performs very well on most datasets. It is noteworthy that compared with other datasets, the experiments result in GDELT are not as superior as others. After further analysis, the reason for this phenomenon may be that there are many abstract entities that don't specify concrete entities (e.g., teacher and school). Specifically, with the time interval increasing, the performance gap between the last two rows in the Table 2 is becoming larger. For WIKI and YAGO, because the long time interval determines the data at each timestamp t having more static structural information, it is helpful to improve the performance of the entity prediction task. Since the TKGE method simultaneously models the static structural and dynamic temporal information, it is efficient for link prediction tasks in TKG.

Relation Prediction. Due to the space limitation, we only compare TKGE with some typical TKG methods and show the performance on MRR. Specifically, we

Table 3. Relation prediction performance on different datasets.

Model	ICEWS18	ICEWS14	ICEWS05-15	WIKI	YAGO	GDELTA
RGCRN	37.16%	38.07%	38.29%	88.89%	90.11%	18.57%
RE-GCN	39.49%	39.74%	38.55%	95.62%	95.15%	19.12%
TKGE	41.52%	41.43%	38.74%	97.92%	97.26%	19.71%

Table 4. TKGE with and without structural self-attention or temporal self-attention.

Model	ICEWS18	ICEWS14	ICEWS05-15	WIKI	YAGO	GDELTA
TKGE	28.84%	39.73%	39.63%	43.94%	57.95%	19.77%
TKGE-NS	27.63%	38.22%	36.98%	39.81%	54.72%	19.02%
TKGE-NT	27.23%	37.57%	34.45%	39.59%	53.08%	18.87%

select the RGCRN and RE-GCN, because we can use them for relation prediction tasks directly. As illustrated in Table 3, we can observe that the performance of the proposed model TKGE is always better than other baselines in all the datasets. The superiority of the TKGE demonstrates that our designed structural and temporal self-attention module can generate more accurate evolutionary entity and relation representations. Compare with the entity prediction task, the performance gap on the relation prediction task is smaller. This is probably because the number of relations was limited, making relation prediction tasks easier than entity prediction tasks. For example, due to the number of relation on WIKI and YAGO being 24 and 10, the experiment results on both dataset is superior to other datasets.

5.3 Utility of Structural and Temporal Self-attention (RQ2)

To demonstrate the effectiveness of our designed structural and temporal self-attention module we compare TKGE with its variants TKGE-NS and TKGE-NT. Two variants represent the model removing the structural self-attention module and the model removing temporal self-attention modules, respectively. As shown in Table 4, we can see that the performance of TKGE-NS and TKGE-NT are lower than TKGE in all datasets. This result verifies the validity of the structural and temporal self-attention module.

5.4 Hyper-Parameter Studies (RQ3)

Due to the space limitation, we only analyze the effect of the number of heads h in multi-head structural and temporal self-attention, since this number plays a crucial role to model TKG. Specifically, except for the measured parameters, we keep other parameters fixed for fairness. We analyze entity prediction tasks on datasets ICEWS14 and YAGO individually. As illustrated in Table 5, we find that multi-head self-attention can improve the performance of TKGE effectively.

Table 5. Impact of hyper-parameter h on entity prediction.

Dataset	$h = 1$	$h = 2$	$h = 4$	$h = 8$	$h = 16$
ICEWS14	38.03%	38.17%	38.29%	39.73%	38.65%
YAGO	56.24%	56.49%	57.74%	57.95%	57.23%

In addition, with the parameter h increasing, the performance of entity prediction tasks will increase firstly then reduce after the locally optimal value. In this paper, the best number of attention heads is 8. In general, the multi-head self-attention which can model TKG from many angles is an effective method for entity prediction tasks.

6 Conclusions

We have presented TKGE, a novel approach for embedding TKG. It jointly models both the static structural and temporal information in learning evolutionary representations for entities and relations. Extensive experiments on six real-world datasets demonstrate the effectiveness and rationality of our TKGE method. In this work, we have regarded TKG as a series of static KG snapshots, thus we only explore the discrete-time approach. Since discrete-time approach characterizes temporal information in relatively coarse levels, the learned embeddings may lose some information between snapshots. To address this issue, we plan to extend our TKGE method to explore continuous-time approaches [19–21] to integrate more fine-grained temporal information.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China Youth Fund (No. 61902001), the Open Project of Shanghai Big Data Management System Engineering Research Center (No. 40500-21203-542500/021), the Industry Collaborative Innovation Fund of Anhui Polytechnic University-Jiujiang District (No. 2021cyxtb4), and the Science Research Project of Anhui Polytechnic University (No. Xjky072019C02, No. Xjky2020120). We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

1. Cheng, K., Yang, Z., Zhang, M., Sun, Y.: Uniker: a unified framework for combining embedding and definite horn rule reasoning for knowledge graph inference. In: EMNLP, pp. 9753–9771 (2021)
2. Che, F., Zhang, D., Tao, J., Niu, M., Zhao, B.: Parame: regarding neural network parameters as relation embeddings for knowledge graph completion. In: AAAI, pp. 2774–2781 (2020)

3. Goel, R., Kazemi, S.M., Brubaker, M., Poupart, P.: Diachronic embedding for temporal knowledge graph completion. In: AAAI, pp. 3988–3995 (2020)
4. Mei, H., Eisner, J.: The neural hawkes process: a neurally self-modulating multivariate point process. In: NIPS, pp. 6754–6764 (2017)
5. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: learning representations over dynamic graphs. In: International Conference on Learning Representations (2019)
6. Li, Z., et al.: Temporal knowledge graph reasoning based on evolutional representation learning. In: SIGIR, pp. 408–417 (2021)
7. Jin, W., Qu, M., Jin, X., Ren, X.: Recurrent event network: autoregressive structure inference over temporal knowledge graphs. In: EMNLP, pp. 6669–6683 (2020)
8. Kong, C., Chen, B., Li, S., Chen, Y., Chen, J., Zhang, L.: GNE: generic heterogeneous information network embedding. In: WISA, pp. 120–127 (2020)
9. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
10. Cheng, S., Xie, M., Ma, Z., Li, S., Gu, S., Yang, F.: Spatio-temporal self-attention weighted VLAD neural network for action recognition. IEICE 104-D, pp. 220–224 (2021)
11. Liu, J., Chen, S., Wang, B., Zhang, J., Li, N., Xu, T.: Attention as relation: learning supervised multi-head self-attention for relation extraction. In: IJCAI, pp. 3787–3793 (2020)
12. Xu, Y., Huang, H., Feng, C., Hu, Y.: A supervised multi-head self-attention network for nested named entity recognition. In: AAAI, pp. 14185–14193 (2021)
13. Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B.: End-to-end structure-aware convolutional networks for knowledge base completion. In: AAAI, pp. 3060–3067 (2019)
14. Dasgupta, S.S., Ray, S.N., Talukdar, P.P.: Hyte: hyperplane-based temporally aware knowledge graph embedding. In: EMNLP, pp. 2001–2011 (2018)
15. Leblay, J., Chekol, M.W.: Deriving validity time in knowledge graph. In: WWW, pp. 1771–1776. ACM (2018)
16. García-Durán, A., Dumancic, S., Niepert, M.: Learning sequence encoders for temporal knowledge graph completion. In: EMNLP, pp. 4816–4821 (2018)
17. Schlichtkrull, M.S., Kipf, T.N., an Rianne van den Berg, P.B., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC, vol. 10843, pp. 593–607 (2018)
18. Zhu, C., Chen, M., Fan, C., Cheng, G., Zhang, Y.: Learning from history: modeling temporal knowledge graphs with sequential copy-generation networks. In: AAAI, pp. 4732–4740 (2021)
19. Garg, K., Panagou, D.: Fixed-time stable gradient flows: applications to continuous-time optimization. IEEE Trans. Autom. Control. **66**(5), 2002–2015 (2021)
20. Chien, J., Chen, Y.: Continuous-time attention for sequential learning. In: AAAI, pp. 7116–7124 (2021)
21. Zhang, L., Zhao, L., Qin, S., Pfoser, D., Ling, C.: TG-GAN: continuous-time temporal graph deep generative models with time-validity constraints. In: WWW, pp. 2104–2116 (2021)