Information Sciences 642 (2023) 119225

Contents lists available at ScienceDirect

Information Sciences

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

T-GAE: A Timespan-aware Graph Attention-based Embedding Model for Temporal Knowledge Graph Completion

Xiangning Hou [a], Ruizhe Ma [b,1], Li Yan [a], Zongmin Ma [a,c,*]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[b] Richard A. Miner School of Computer & Information Sciences, University of Massachusetts Lowell, Lowell, MA 01854, USA
[c] Collaborative Innovation Centre of Novel Software Technology and Industrialization, Nanjing 210023, China

ARTICLE INFO

Keywords:
Temporal Knowledge Graph
Temporal Knowledge Graph Completion
Encoder-Decoder architecture
Long Short-Term Memory Network
Graph Attention Network

ABSTRACT

Temporal knowledge graphs (TKGs) often suffer from incompleteness, leading to an important research issue: *Temporal Knowledge Graph Completion* (TKGC). *Knowledge Graph Embedding* (KGE) methods have proven to be effective in solving this issue. However, most of them handle triples independently and do not capture complex information embedded in the neighborhood topology of central entities. To this end, we propose a ***T**imespan-aware **G**raph **A**ttention-based **E**mbedding Model* named **T-GAE** to tackle the TKGC task. To the best of our knowledge, T-GAE is the first KGE model in which *Graph-Attention-Networks* (GATs) and *Long Short-Term Memory* (LSTM) *Networks* are simultaneously applied to the TKGC task. In essence, our model is an *Encoder-Decoder* architecture, where the encoder consists of an LSTM network and a GAT network. Firstly, we employ LSTM layers to learn new time-aware relational embeddings to incorporate time information. Then, we utilize these time-aware relational embedding and GATs considered as neighborhood aggregators to learn the entity and relational features of the central entity neighborhoods. Thus, T-GAE can capture the interaction features between multi-relational facts and the abundant temporal information in TKGs. As for the decoder, we choose the ConvKB model, which is essentially a scoring function. Our experiments demonstrate that T-GAE significantly outperforms most of the existing baseline methods for TKGC in terms of MRR and Hit@1/3/10.

1. Introduction

Knowledge graphs (KGs) are used to organize, store and manage information in a structured manner. A knowledge graph is regarded as a large multi-relational directed graph where *entities* and *relations* between adjacent entities are considered as *nodes* and *edges*, respectively. Each edge with two nodes in a KG represents a triple of the form (subject, predicate, object), which means the subject and the object are linked by the predicate, e.g., (*Bill Clinton, wasBornIn, Arkansas*). KGs are widely used and play a crucial role in many fields, such as recommendation systems [1], question answering [2], and semantic search [3]. Although KGs have continued to develop in recent years, existing large-scale knowledge bases are often incomplete, indicating that many important facts are missing.

* Corresponding author at: College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.
E-mail address: zongminma@nuaa.edu.cn (Z. Ma).
[1] Co-first author.

https://doi.org/10.1016/j.ins.2023.119225
Received 2 July 2022; Received in revised form 9 April 2023; Accepted 22 May 2023
Available online 26 May 2023

To deal with this problem, many researchers focus on a research direction known as Knowledge Graph Completion (KGC) [4–11]. However, the relational facts of real-world KGs are typically associated with a point in time or a time interval, e.g., (*Bill Clinton, wasBornIn, Arkansas, 1946-8-19*). Therefore, the concept of temporal knowledge graphs (TKGs) was proposed by academics. As shown in Fig. 1, there is a sub-graph of a temporal knowledge graph. Similar to static KGs, TKGs also exist incompleteness. Thus, a novel research direction, TKGC (Temporal Knowledge Graph Completion), which predicts missing facts of given time information, quickly became a new research hotspot.

State-of-the-art (SOTA) KGC methods are known to be primarily KGE models. The key idea of KGE is to map the semantic information of entities and relations into dense low-dimensional vectors and then efficiently compute entities, relations, and complex associations in the low-dimensional space. Recently the employment of KGE approaches to resolve the KGC problem has gained popularity. The classical KGE models for addressing KGC tasks are mainly divided into three types, including *Translational* models [6–8], *Tensor Decomposition* models [4,5,15,16] and *Convolutional Neural Network-based* (CNN) models [17,18]. *Translational* models fully utilize transition-based ideas and a few parameters to learn entity/relation embedding, which does not contain rich semantic information. *Tensor Decomposition* models obtain a low-dimensional vector representation in the form of matrix decomposition, which is relatively easier to train, is faster, and requires fewer hyper-parameters. *CNN*-based models are based on deep learning models, which are trained to obtain more expressive embedding. The main drawback of the former two is that they only focus on a single triple, and little attention is paid to the connection existing in the context of the triple, which cannot capture rich semantic information. Although the latter overcomes the problem of insufficient semantic information and has high parameter efficiency, they still do not take into account interaction among triples.

Most existing KGC approaches are proposed to complete static KGs [12]. However, relational data in real-world KGs are temporal. Static KGs cannot model temporal evolution, meaning that facts in KGs are only valid at a time period. Since static KGC methods completely ignore the time information in KGs, it directly leads to the fact that static KGC methods do not work in real-world scenarios. Some researchers have started to propose temporal knowledge representation methods for TKGC tasks. Several simply temporal extensions of existing well-established static KG representations are proposed [13,14,28,31]. While their architectures are simple and they are easy to be trained, they suffer from several limitations. For example, t-transE in [13] can only capture the temporal order between some relations; HyTE in [14] is not highly interpretable and cannot handle complex contextual associations between entities/relations. Also, some approaches embody temporality through clever modeling of time or propose temporal KG representation learning methods based on complex networks. ChronoR [34] proposes a temporal rotation-based embedding representation model, which can learn a k-dimensional rotational transformation parameterized by relations and time. DyERNIE [32] is a non-Euclidean embedding method that learns to develop embeddings of entities in the product of Riemannian manifolds. TempCaps in [39] proposes a capsule network-based embedding model specifically for the TKGC task. Inspired by capsule networks, TempCaps introduces a novel dynamic routing aggregator to model temporal knowledge graphs and then build entity embedding representations through temporal relationships and neighbor information retrieved by the dynamic routing. These aforementioned methods, however, do not adequately

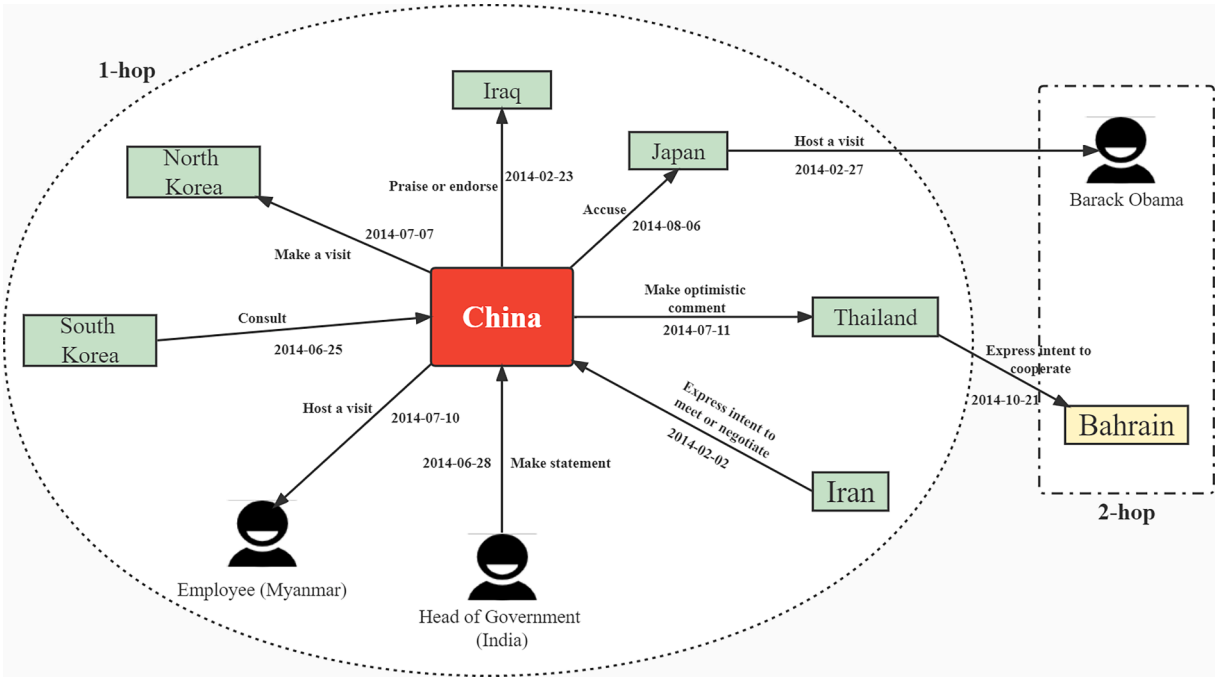

Fig. 1. An example temporal knowledge sub-graph extracted from the ICEWS14 knowledge graph (each relation or interaction between entities is linked to a time stamp, and a collection of interactions forms a multi-relational graph).

represent temporality when dealing with complex and irregular temporal representations, and have insufficient ability to capture the interaction information between facts for TKGC task. Moreover, many of them rely on insufficient datasets like YAGO2, which suffers from the sparsity in the timespan, or a time-augmented variant of Freebase, which adds time to existing facts.

To address the abovementioned issues, we propose a temporal KGE method called **T**imespan-aware **G**raph **A**ttention-based **E**mbedding Model (T-GAE) for TKGC. Our architecture is based on an *Encoder-Decoder* structure, with our GAT (Graph-Attention) model, LSTM (Long Short-Term Memory) network, and ConvKB [17] acting as encoders and decoders, respectively. The encoder is composed of two parts: the LSTM network and the GAT network. Firstly, the LSTM network in our architecture can fit sparse and irregular time expressions well. To generate word embedding, we employ character-level architectures, which have been used in language modeling [21] to leverage characters as atomic units, to convert time expressions into token sequences which express the temporal information. The benefit of utilizing a character-level architectures to directly encode time information in relations is that the efficient transfer of information between analogous timestamps is made possible by the use of digits and time modifiers like "occurSince" as atomic tokens. Hence, we can obtain relation representations that carry time information by LSTM networks training with sequences of tokens expressing the time relation and the digits of the timestamp. Secondly, GAT networks in our architecture are employed to capture entities in the multi-hop neighborhood of a given central node, as well as relation features that carry time information. Depending on the difference in importance to the central entity, the GAT network guides attention to both neighbor nodes linked by different edges and relations in the *n-hop* neighborhood of a node. It aggregates not only single-hop neighborhood but also multi-hop neighborhood information for the source node. Then it learns to embed nodes in itself by aggregating information from different neighbor entities and relations features. As the depth of model increases, the number of remote entities grows exponentially, which rapidly increases the computational overhead. The contribution of distant entities shrinks exponentially. There is no denying that multi-layer GAT can aggregate information from n-hop neighborhoods. However, the number of parameters of the GAT network grows rapidly with the number of relations, which can easily lead to overfitting on rare relations or very large datasets. Therefore, it is not recommended to use multi-layer GAT for n-hop neighborhood information aggregation. To solve this problem, we propose a *virtual time-aware edge* inspired by [22]. Finally, we utilize ConvKB as our decoder in our proposed architecture. In essence, it is the *Scoring Function* of our model. We get new entities/relation embedding through the encoder. These new embeddings are used as input to the decoder to get the new triples that may exist.

We outline the following aspects of our contribution:

Temporal information is expressed in KGs in an irregular form. In order to incorporate temporal information inspired by character-level architectures in the field of language modeling, we propose to employ an LSTM network to encode both time expressions and relations into token sequences to produce relation embedding accompanying time information. It is a fusion of temporal information in KGs in a novel, ingenious and unified form.

In contrast to the state-of-the-art static KGE methods, which learn entity/relation embedding independently in a single triple, the proposed T-GAE can learn the entity's graph context by GAT network, guiding different attentions to the different neighbor entity and n-hop relations, which is time-aware. It can capture interactions among triples, time information hidden in relations, and topology information very well.

As for how to handle multi-hop temporal relations of a source entity, we propose a *virtual time-aware edge* concept well to learn information about multi-hop neighborhoods. It makes our model accumulates knowledge that may carry rich time information from n-hop neighbors of a source entity. More than that, we find that the process of learning and aggregating neighborhood information does not take into account its own initial information; we also propose *self-cyclic relation* to incorporate initial semantic information about itself.

We conducted comprehensive experiments to evaluate the efficiency of T-GAE using three large public temporal datasets (ICEWS14, ICEWS05-15, YAGO15k). In terms of typical assessment metrics (e.g., Mean Rank and Hit@1/3/10), the experimental results indicate that the improvements are significant when compared to other state-of-the-art approaches.

The rest of this paper is organized in the following manner. Section 2 presents a summary of related work followed by our thorough methodology in Section 3. Section 4 contains the experimental data results, datasets descriptions, and results analysis, followed by Section 5, which contains our conclusion and future research directions.

2. Related work

Entities and relations are mapped into low-dimensional continuous vectors using Knowledge Graph Embedding (KGE), which has gained lots of attention from researchers in the past few years. We go over *static knowledge graph embedding* and *temporal knowledge graph embedding* in this section.

2.1. Static knowledge graph embedding

A substantial amount of KGE approaches have been proposed for *static knowledge graphs.* These approaches can be roughly grouped into four categories: (1) *translation-based* KGE, (2) *matrix factorization-based* KGE, (3) *CNN-based* KGE, and (4) *graph-based* KGE.

TransE [6] is an early translational-based model that embeds the entities h and t, as well as the relation r, and maps them using the score function: $h + r \approx t$. It aims to minimize the distance between the head entity h and the tail entity t by the relational transition. Since TransE can only perform well on 1-to-1 relationships and cannot be used to handle complex relationships such as 1-to-N, N-to-1, and N-

to-N, many extended models have been proposed to address these problems, such as TransH [7], TransR [8], TransD [23]. *Matrix Factorization*-based models, including DistMult [4], RESCAL [5], HolE [15], ComplEx [24], and SimplE [25], turns relations into linear transformations of entity embeddings imposing on matrices. As for *CNN-based* models, there are primarily two models: ConvE [18] and ConvKB [17]. ConvE and ConvKB reshape entities and relations into multidimensional matrices, model the interactions between entities and relations using convolution, and generate more expressive embedding through feature learning. Finally, we discuss *Graph-based* approaches. DeepWalk [26] is the first to produce vector embeddings of nodes in a network using a random-walk model: a weightless random-walk approach. R-GCN [27], a graph neural network extension, is one of the first ways to employ graph neural networks to learn node embedding by aggregating each entity's neighborhood information, which guides them to equal weights. CompGCN [42], a new framework for merging multi-relational information in graph convolutional networks, systematically exploits the entity-relationship composition operation in knowledge graph embedding, which can jointly embed node information and relation information into multi-relational graphs. However, these KGE methods mentioned above cannot capture the time information of TKGs and they are not applicable to TKGs [12].

2.2. Temporal knowledge graph embedding

There has been a wide range of methods focused on fusing time information into modeling TKGs for resolving the problem of temporal link prediction. KGE-based *Temporal Link Prediction* methods are roughly divided into two categories: interpolation and extrapolation. The interpolation is what we usually call the TKGC problem. Besides interpolation tasks, a few researchers have also studied and proposed some SOTA methods for extrapolation tasks, e.g., RE-NET [35], CyGNet [36] and xERTE [43], which have a common feature of predicting possible events in future moments based on historical information in the temporal knowledge graph in recent years. The first interpretable model for forecasting future links on temporal KGs was created by Han et al. [43] and is called xERTE. A new reverse representation updating strategy and a temporal relational attention mechanism that can maintain the causal features of temporal multi-relational data are both proposed by xERTE to guide the extraction of an surrounding subgraph around the query subject. Moreover, xERTE visualizes the inference process and offers an interpretable inference graph to emphasize key evidence, unlike other models based on black-box embeddings. In this paper, we aim to resolve TKGC tasks that interpolate missing links at observed timestamps.

Most of previous works have been improved by temporalizing the static KEG methods. t-TransE [28] is the first model to learn entity/relation embedding by modeling transitions between time-aware relationships using timing constraints. For instance, "*diedIn*" should be followed by "*wasBornIn*." Thus, TKGFrame [29] is proposed based on the observed patterns which explicitly define the relational chain of TKGs and incorporate it into TKG embeddings. Inspired by the idea of TransH, a KGE approach based on projected-time translation is proposed by HyTE [14], which turns different times into different temporal hyperplanes. It projects entity/relationship embedding onto different temporal hyperplanes to obtain new embedding and then applies the TransE scoring function to calculate the score of the triple at a given time. The new temporally-informed relational embedding is generated through "predicate temporal serialization" and LSTM network processing in TA-DistMult [13], which temporalizes the relations in a triple by integrating the relations with temporal expressions.

The following models are not simply improved by fusing temporal information of static models. ConT [30] seems to be a Tucker extension that leverages a time-specific tensor rather than the core tensor in Tucker decomposition. Motivated by the diachronic word embedding that was successfully employed in other NLP fields, DE-SimplE [31] is a temporal extension of SimplE applying a diachronic entity embedding function to capture entity features of the same entity at different times. TNTComplEX [33] is a new regularization approach inspired by the 4-order tensor regular decomposition, which is improved by ComplEX by temporalization. ChronoR [34] proposed a temporal rotation-based embedding representation model that learns a relational and temporal k-dimensional rotational transformation parameterized by the time such that the head entity of each fact will fall in the corresponding tail entity near the corresponding tail entity after using the rotational transformation for the head entity of each fact. DyERNIE [32] is a non-Euclidean embedding method that learns developing entity embedding in a product of Riemannian manifolds, with the composition spaces approximated from the sectional curvatures of underlying data using a new perspective to solve TKGC. TeMP (Temporal Message Passing) in [41] designs a two-layer encoder structure: the first layer uses R-GCN to learn the graph structure information; the second layer considers two different structures to learn the temporal information, a GRU-based encoder structure and a transformer attention-like encoder structure. Note that TeMP considers time information and graph topology information, but it does not directly integrate time information into entities or relations. In order to decompose temporal knowledge graphs into fourth-order tensors, Xu *et al.* in [44] presented the TKGE method TeLM, which makes use of multi-vector embeddings and linear temporal regularizers. TeLM uses level 2 geometric algebra's more expressive multi-vector embeddings to model the TKG's entities, relations, and timestamps. The expressiveness of the geometric product is higher than that of the intricate Hermitian operator. Therefore, TeLM uses an asymmetric geometric product to achieve a fourth-order tensor decomposition of a TKG. To learn the representation of temporal knowledge graphs by simultaneous modeling of local and global structural evolution, Zhang *et al.* in [45] presented a temporal knowledge representation learning approach EvoExplore. It means a hierarchical attention-based point-in-time process to generate graph structure in a fine-grained manner and specifically define local structure evolution as the act of creating relationships between entities. EvoExplore suggests using a soft module parameterized by the entity embedding representation to capture the dynamics of TKGs for the evolution of the global structure.

Although these methods perform very well in TKGC tasks, they almost ignore the graphical structure of TKGs and fail to capture associations between facts in the time dimension in the majority of cases. Furthermore, most approaches to learning embedding either focus primarily on entity features or integrate entity and connection characteristics in a disconnected manner [12]. On the contrary, T-

GAE is one of few works for TKGC tasks that capture both complex time information and graphical structure, considering multi-hop and semantically similar relations to assign different weights for neighbor entities in line with importance.

3. The proposed T-GAE model

In this section, we provide our proposed model T-GAE for *Temporal Link Prediction* in TKGs. First, we formally define the essential components of the Temporal Knowledge Graph, expound on the problem of Temporal Knowledge Graph Completion and specify the related notation used throughout this work. Next, we introduce the architecture of T-GAE, which includes the encoder's composition, the roles of the various modules in the model, and the decoder selection. Additionally, the process of parameter learning and training is thoroughly explored.

Definition 1. **(*Temporal Knowledge Graph*).** *After studying many KEG approaches, we found a consensus among them: a KG is considered to be a multi-relational directed graph containing structured facts. Generally speaking, we denote a static KG as* $G = (E, R, F)$*, where* $\boldsymbol{E}$ *and* $\boldsymbol{R}$ *represent the collection of entities and the collection of relations, respectively. So, we consider a TKG obtained by fusing temporal information from static KG would be formalized as* $G = (E, R, T, F)$*, where* $\boldsymbol{T}$ *is the set of timestamps. This way, the total number of probable facts is* $F \in \{E \times R \times E \times T\}$*. The valid facts in a TKG are composed of head entity h, tail entity t, a relation r that links the two entities, and a timestamp* T_n *which means when the fact occurred, e.g., a quadruple* $f = (h, r, t, T_n)$*. In this paper, we refactored quadruples into relational temporal serialized triples, e.g., the new triple* $f = (h, [r{:}T_n], t)$*.*

Definition 2. **(*Temporal Knowledge Graph Completion*).** *Given a TKG whose timestamps vary from* T_0 *to* T_t*, Temporal Knowledge Graph Completion (also known as Temporal Link Prediction) focuses on reasoning new missing facts based on this incomplete TKG at a special given time T, where T is within the time range of* $[T_0, T_t]$*. It can be classified into three types of sub-tasks*:

(i) **Head entity prediction.** It aims to predict a missing head entity given r and t at timestamp T, which we can denote as a prediction query $(?, r, t, T)$.

(ii) **Tail entity prediction.** Similar to *Head entity prediction,* tail entity prediction aims to predict a missing tail entity given h and r at timestamp T, which we can denote as a prediction query $(h, r, ?, T)$.

(iii) **Relation prediction.** Relation prediction aims to predict a missing relation given h and t at timestamp T, which we can denote as a prediction query $(h, ?, t, T)$.

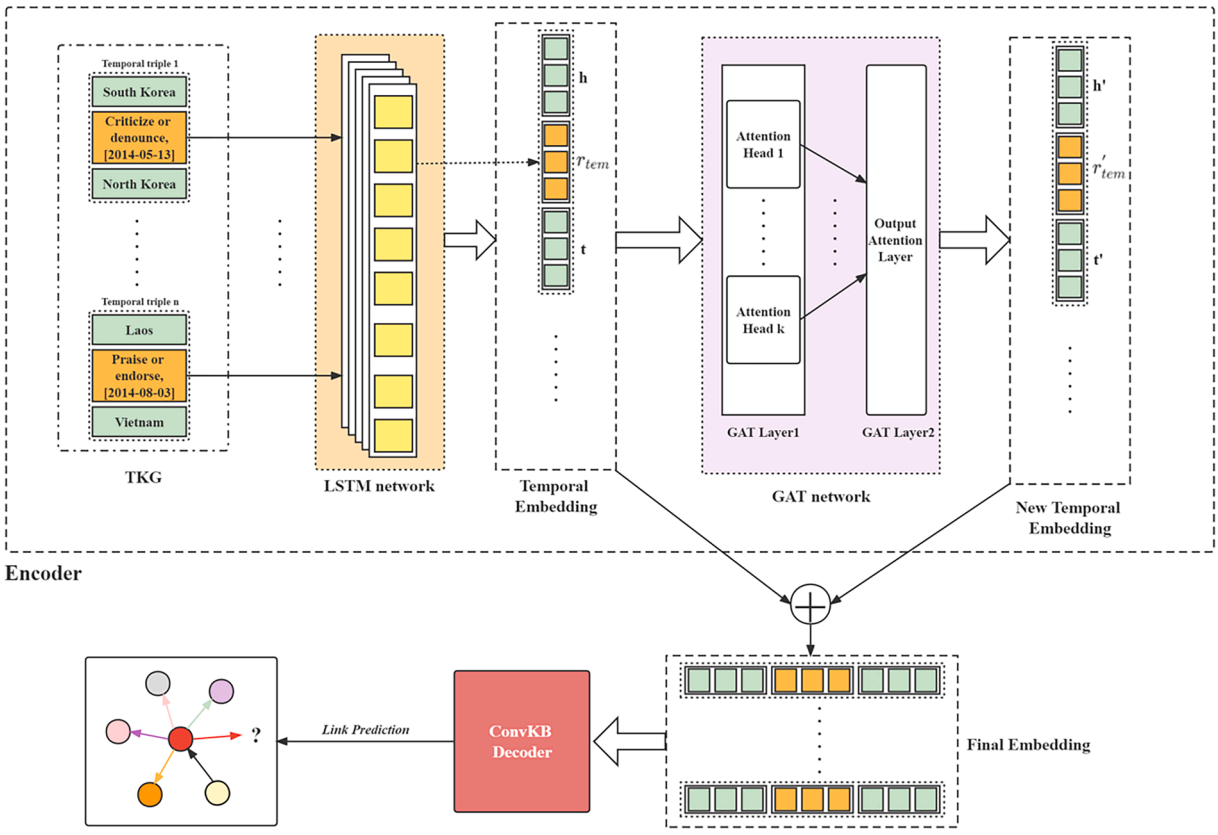

Fig. 2. Overview of the Encoder-Decoder Architecture of T-GAE.

In this work, we focus more on the first two sub-tasks (entity prediction), which we include in the experimental evaluation and the model training process.

3.1. Overview of T-GAE

The fundamental ideas of the model proposed in this work are the combination of temporal dependencies learned from TKGs, and local topological dependencies learned from node neighborhoods. As shown in Fig. 2, our model comprises an encoder and a decoder. The encoder of T-GAE consists of an LSTM network for encoding time-aware relation representation and a GAT network for capturing the neighborhood information. As for the decoder of T-GAE, we choose the ConvKB. In fact, the ConvKB is a score function of our model. Firstly, we obtain new relations embedding with temporal information by the LSTM network in the character-level model. Secondly, we use the original entity embedding representation and the temporal relational embedding representation learned from the LSTM network layer as the input to the graph attention network. Unlike the generic GAT network, we select a GAT network that considers not only the effects of neighbor entities, but also the effects of relationships connected to the central entity on them. As a result, the entity embedding representations learned through this network layer are not only rich in temporal information, but also capture a large amount of graph contextual information. Finally, the decoder takes the entity/relation embedding generated by the encoder as the input and predicts new missing facts. Summarily, we employ a step-by-step training paradigm to train T-GAE, which means that the encoder (LSTM layer and GAT layer) and the decoder (ConvKB) are trained independently. The decoder model is trained to carry out TKGC task after the encoder encodes the entity and temporal relations.

3.2. Encoder of T-GAE model

In contrast to most existing representation learning methods for KGs, we study representation learning for TKGs. The most important thing is time processing and how to incorporate temporal information with relations. This section first utilizes a novel way of processing time. Then, we explain how to combine the processed temporal information with relations and how it applies to a GAT network that considers relations and captures rich graph contextual information to obtain a new embedding with temporal and contextual information.

3.2.1. Time processing

Generally, the basic building blocks of TKGs are quadruples, e.g., $f = (h, r, t, T)$. Most facts of TKGs are coupled with time information which time expression may be irregular. Hence, we propose factorizing a given (possibly incomplete) timestamp into a sequence of temporal tokens to overcome this problem. For instance, the time expression like {####-##-##} consists of up to 8 characters, and each character is a number from 0 to 9. As shown in Fig. 3, a *Time Vocabulary* is designed as time decomposition.

There are 34 temporal tokens in Time Vocabulary. The year, month, and day are represented as the suffix "*y*," "*m*," and "*d*," respectively. The year contains 4 digits, and each one is a token from "0y" to "9y". Similar to the year, the month and day contain 1 and 2 digits, respectively. Considering the practicalities in daily life, the year is divided into 12 months, with each month having a maximum of 31 days. Furthermore, we may extract a sequence of relation tokens for each quadruple that always includes the connection type token and a temporal modifier token that may exist, such as "*occurSince*" or "*occurUntil*." Finally, we easily obtain the sequence of temporal relation token r_{tem} by concatenating relation and temporal sequence. A TKG would be composed of a set of triples in the format $f = (h, r_{tem}, t)$, where r_{tem} contains both relational information and rich time information. Table 1 lists a few examples of facts after temporalizing and corresponding temporal relation tokens. Here, We assume the ID of relation "isCitizenOf," "Criticize or denounce," and "isMarriedTo" are 7, 0, and 11, respectively. Next, these sequences of r_{tem} are tokens and the input for the LSTM network to generate new relation embedding.

3.2.2. Learning Time-aware Representations by LSTMs

A long short-term memory (LSTM) network [20] is a type of temporal recurrent neural network (RNN), which is specially designed to solve the long-term dependency problem of the general RNNs by utilizing structures called gates to forget or add information to the cell state based on the input sequences. Thus, because of its structural characteristics, LSTM is well suited for solving many sequential tasks (e.g., *language modeling, machine translation*) and performs better than the general RNNs, especially when large amounts of data are available. The architecture of LSTM is shown in Fig. 4, and the related calculation formulas of LSTM are given in Equations (1)-(6).

$$f_t = \sigma(W_f \cdot [h_{n-1}, x_t] + b_f) \tag{1}$$

Temporal token Type		Corresponding Relation ID
Year:	0y 1y 2y 3y 4y 5y 6y 7y 8y 9y	0-9
Month:	01m 02m 03m 04m 05m 06m 07m 08m 09m 10m 11m 12m	10-21
Day:	0d 1d 2d 3d 4d 5d 6d 7d 8d 9d	22-31
Temporal Modifier:	Occursince Occuruntil	32-33

Fig. 3. Time Vocabulary.

Table 1
Three examples of facts after temporalizing and corresponding temporal relation token.

Facts	Temporal Relation Token	Corresponding ID sequence
(Poul_Anderson, isCitizenOf, Denmark)	[isCitizenOf]	[7]
(South Korea, Criticize or denounce, North Korea, 2014-05-13)	[Criticize or denounce,2y,0y,1y,4y,05m,1d,3d]	[0,2,0,1,4,14,23,25]
(Gale_Anne_Hurd, isMarriedTo, Brian_De_Palma, occursSince,1991)	[isMarriedTo,occursSince,1y,9y,9y,1y]	[11,32,1,9,9,1]

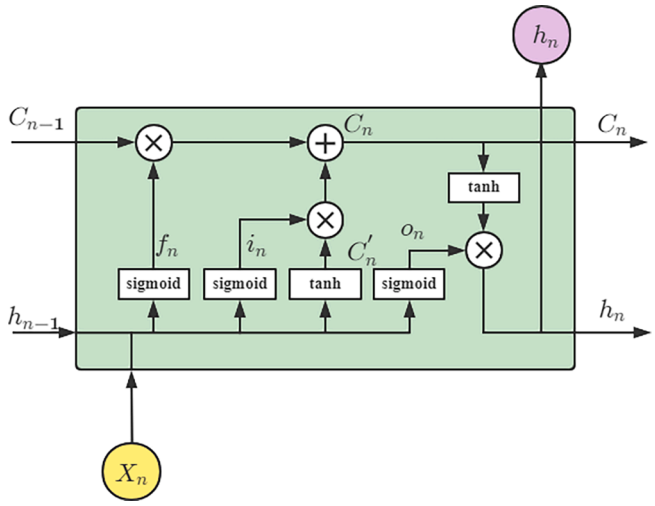

Fig. 4. The Architecture of an LSTM Cell.

$$i_t = \sigma(W_i \cdot [h_{n-1}, x_t] + b_i) \tag{2}$$

$$C'_n = \tanh(W_C \cdot [h_{n-1}, x_t] + b_c) \tag{3}$$

$$C_n = f_n * C_{n-1} + i_n * C'_n \tag{4}$$

$$O_n = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5}$$

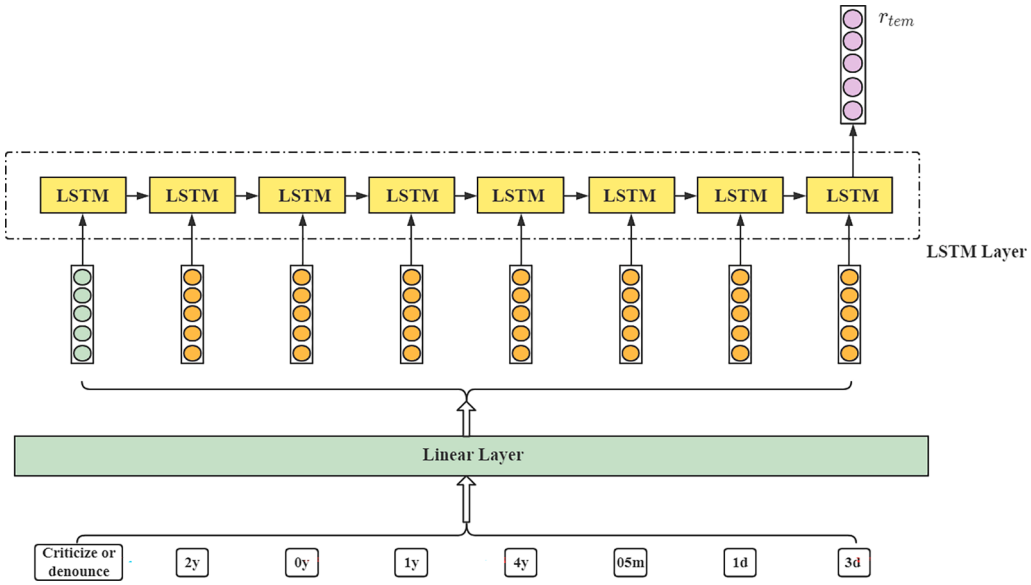

Fig. 5. A Concrete Example of Learning Temporal Relation Embedding with LSTMs.

$$h_t = O_t * \tanh(C_t) \tag{6}$$

To transport information through and control the cell state, LSTM employs gates, namely the forget gate f_n, the input gate i_n, and the output gate o_n. $x_n \in R^d$ is denoted as the input of LSTM at the *n-th* element of temporal relation token embedding, C_n and h_n represent the output of the hidden and cell state of the LSTM at each element of temporal relation token. Here, σ and *tanh* are, respectively, *sigmoid* and *tanh* activation functions, and we suppose the dimension of entity/relation embedding is d. Next, we give a concrete example to elaborate on the LSTM learning time-aware relational embedding process.

As shown in Fig. 5, we first employ a linear layer to turn every token of the temporal relation sequence into a vector whose dimension is set as d. Here, N is the length of the sequence. According to the corresponding sequence length N, we apply N LSTM units in the LSTM layer to process it. Next, the LSTM layer takes the embedding after the mapping process as input. Finally, according to the characteristics of the LSTM structure, we know that the hidden state of the last LSTM cell is used as the output of the whole LSTM layer, which is what we need for r_{tem}. That is, $e^r_{tem} = h_N = LSTMs(r_{tem})$. At this moment, the relation embedding e^r_{tem} carries extensive time information and the semantic information of the relation itself. Thus, the GAT network, which considers relations are vital, takes the initial entity embedding and e^r_{tem} learned from the LSTM layer as input to generate new embedding.

3.2.3. GATs considering time-aware relations

A TKG is viewed as a large-scale multi-relational heterogeneous network in this study. A graph convolutional network (GCN) [37] gathers information from a source entity's neighbors and assumes that all of the source entity's neighbors contribute equally to the information-passing process, which may limit the model's ability to capture local topological features. [19] presented graph attention networks (GAT) - establishing an attention mechanism [38] in graph convolutional networks (GCNs), which presents an attention mechanism that weights the features of surrounding nodes, where each source entity's neighbors are different. Different surrounding nodes of each source entity have different levels of importance. The weights assigned to neighboring node attributes are fully determined by the node features and are entirely independent of the graph structure.

Despite the fact that most GATs have performed well in studies on a variety of basic tasks, such as *Node Classification*, they all have one shortcoming: they all omit relational features, which are a crucial part of the triples and a fundamental building block of KG. As a result, they are much less effective in a KG. If the relations connecting two identical 1-hop neighbors of a source entity are different, it is evident that the two identical entities are of differing importance to that source entity. According to their affiliation links, surrounding entities in KG play distinct functions. As a result, the GAT network applied in our work considers not only neighbor node features but also relation features before aggregating them to obtain the desired embedding representation of the source entities. Following that, we review the parameters and formulas that make up this GAT network.

In order to generate a new embedding of entity e_h, we need to learn the representation of each temporal triple associated with entity h. Assuming the existence of $f = (h, r_{tem}, t)$, we first concatenate the head entity embedding e_h, tail entity embedding e_t, and temporal relation embedding e^r_{tem} to create a new vector. Then, we learn the representation of this triple c_{hrt} by employing linear transformation, as shown in Equation (7).

$$c_{htr} \rightarrow = W_c[e_h||e_t||e_r] \tag{7}$$

Here c_{hrt} is the representation of the triple $f = (h, r_{tem}, t)$, and e_h, e_t, e^r_{tem}, are the initial embedding of the head entity h, the temporal relation r_{tem}, and the tail entity t, respectively. Wc is a linear transformation matrix.

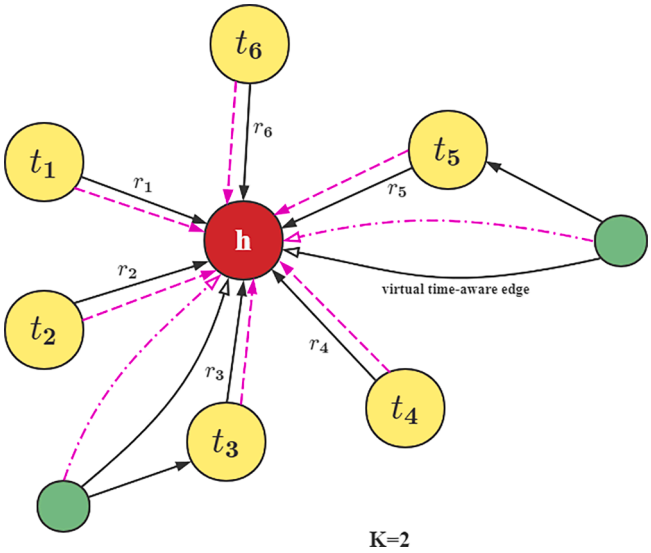

Fig. 6. The Process of Learning and Aggregating GATs.

To extract the attention values of triples, we apply a linear transformation by using a parametrized weight matrix W_b and, subsequently, the *LeakyRelu* nonlinear function in Equation (8).

$$b_{htr} = \text{LeakyReLU}(\text{W}_\text{b}\text{c}_{\text{htr}}\rightarrow) \tag{8}$$

Here b_{hrt} denotes the importance level of each triple $f = (h, r_{tem}, t)$ related to entity h.
As is shown in Equation (9), the normalization function *softmax* is used to get its relative weight values.

$$w_{htr} = soft\max_{tr}(b_{htr}) = \frac{\exp(b_{htr})}{\sum_{t\in N_h}\sum_{r\in R_{ht}}\exp(b_{htr})} \tag{9}$$

Here N_h represents the collection of neighboring nodes of head entity h; R_{ht} is the collection of relations linked head entity h and tail entity t. Next, we produce a new embedding of entity h using the obtained relative attention values. This new embedding is obtained by adding up each triple representation that has been weighted by the relative attention values, as shown in Equation (10):

$$\overrightarrow{e'_h} = \sigma(\sum_{t\in N_h}\sum_{r\in R_{ht}}\alpha_{htr}c_{htr}\rightarrow) \tag{10}$$

Moreover, we employ the *Multi-head Attention Mechanism* (MHA) first introduced and applied by [55] to generate a new expressive embedding. MHA helps stabilize the learning process and improve the description of neighborhood information. In this work, we also employ K-independent layers of the attention mechanism to learn the embedding of head entity h. Finally, the new embedding learned by each attention layer is averaged to generate a new embedding of head entity h, as shown in Equation (11) and Fig. 6:

$$\overrightarrow{e'_h} = \sigma(\frac{1}{K}\sum_{k\in K}(\sum_{t\in N_h}\sum_{r\in R_{ht}}\alpha^k_{htr}\overrightarrow{c^k_{htr}})) \tag{11}$$

We also observe that the process of learning and aggregating neighborhood information does not take into account its own initial information. Therefore, we also propose *self-cyclic relation* to incorporate initial semantic information about itself. As is shown in Equation (12), We add the initial entity embedding information to the entity embedding obtained from the last attention layer.

$$e^{final}_h = e_h + \overrightarrow{e'_h} \tag{12}$$

Our GAT layer learns not only information about 1-hop neighbor entities but also information about multi-hop neighbor nodes as well as relations. We handle multi-hop temporal relations of a source entity by applying *virtual time-aware edges* to accumulate knowledge that may carry rich time information from n-hop neighbors of a source entity. In essence, *virtual time-aware edges* are directed relational paths in a multi-relational graph. However, the importance of multi-hop entities to the central entity decreases exponentially with increasing distance from the central entity as well as the computational complexity and cost explode with increasing order. Therefore, combined with the actual situation, we only consider at most the 2-hop neighborhood. There is an example illustrating the process of learning new embedding, including a multi-head attention mechanism and virtual time-aware edges in Fig. 6.

Finally, as for the problem of how to train the encoder reasonably, the key idea in our proposed model of training the encoder using *hinge-loss* is inspired by the scoring function of TransE. Thus, we try to train this model for learning entity/temporal relation embedding by minimizing the loss function L_e with L_1-norm dissimilarity measure as shown in Equation (13):

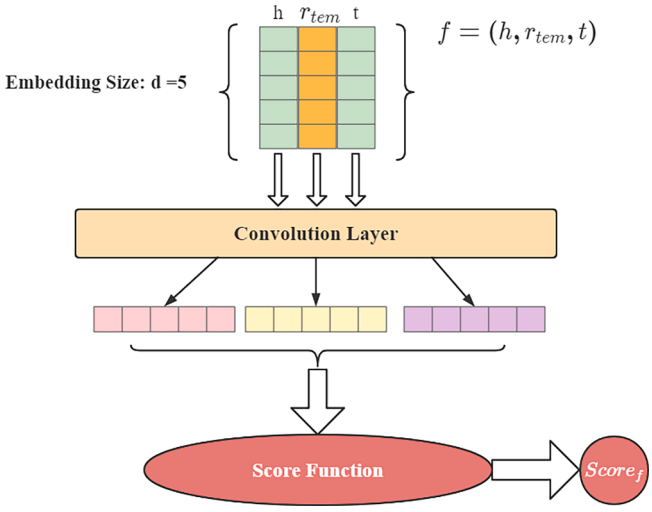

Fig. 7. The Process of Learning in the Decoder.

$$L_e = -\sum_{f_{hrt}\in G}\sum_{f'_{hrt}\in G'} \min((||h + r_{tem} - t||_1 - ||h' + r'_{tem} - t'||_1 + \chi), 0) \tag{13}$$

where χ denotes a margin hyper-parameter that is greater than 0. As is shown in Equation (14), G' represents a set of negative triples by randomly replacing h, r, or t of valid triples in G.

$$G' = \left\{ f^l_{ht} \middle| h' \in (E-h), f^l_{h,t} \notin G \right\} \cup \left\{ f^l_{ht} \middle| t' \in (E-t) \right\}, f^l_{ht} \notin G \tag{14}$$

3.3. Decoder

In our architecture, we choose ConvKB [17] as the decoder due to its advancement and high efficiency. ConvKB uses a convolutional neural network to capture global relationships and transitional properties between entities and relations in knowledge bases, advancing state-of-the-art models. As illustrated in Fig. 7 (the embedding size $n = 5$, the number of convolutional *filters* $= 3$, and the activation function $\sigma =$ ReLU), each triple (h, r, t) in ConvKB is treated as a three-column matrix T, with each column vector representing a triple element. Next, this three-column matrix T is sent into a convolution layer, which employs numerous filters to generate various feature maps. The input triple is concatenated from these feature maps into a single feature vector. In order to obtain a score, the feature vector is multiplied by a weight vector using a dot product. Thus, we use ConvKB as our decoder compared with other decoders (score function), such as MLP decoder. Formally, the ConvKB score function is defined in Equation (15).

$$S_{(h,r_{tem},t)} = concat\{\sigma([e_h, e^{tem}_r, e_t] * \Omega)\} \cdot W \tag{15}$$

In this equation, *concat* represents the operation of concatenating; Ω and W are two shared parameters, which are independent of h, r_{tem}, and t; * denotes a convolution operator. By minimizing the loss function L_d with L_2 regularization on the weight vector of the model, we utilize the *Adam optimizer* to train ConvKB in Equation (16).

$$L_d = \sum_{(h,r_{tem},t)\in\{G\cup G'\}} \log(1 + \exp(l_{(h,r_{tem},t)} \cdot S_{(h,r_{tem},t)})) + \frac{\lambda}{2}||w||^2_2 \tag{16}$$

$$l_{(h,r_{tem},t)} = \begin{cases} 1 & (h, r_{tem}) \in G \\ -1 & (h, r_{tem}, t) \in G' \end{cases} \tag{17}$$

Note that w denotes the weight vector of the decoder, and λ denotes a margin hyper-parameter.

4. Experiment

In this section, we evaluate the proposed model's performance on the temporal link prediction task using a temporal knowledge base and discuss the model's advantages and disadvantages when compared to the baseline KGE methods on different datasets. The temporal link prediction task can be viewed as a *multi-classification* task, with each classification corresponding to a (head/tail) entity in a candidate set accompanied by a score and normalized for ranking purposes. The link prediction task is essentially a sorting problem in which the set of potential (head/tail) entities is sorted. First, we go over the basics of the experiment, including information on the temporal datasets and benchmark comparison methodologies employed, as well as the evaluation criteria. Next, the experimental results of T-GAE with different datasets will be thoroughly compared and analyzed. Finally, we will conduct ablation studies of T-GAE to further illustrate the model's efficiency and robustness.

4.1. Experiment setup

4.1.1. Datasets

We conduct experiments to evaluate the performance of the proposed model using three temporal knowledge graphs, ICEWS14, ICEWS05-15, and YAGO15k. Table 2 summarizes a few statistics of used TKG benchmarks. The columns include the dataset's name, the total number of entities and relations, total timestamps, the total number of facts in the train/validation/test set, and the time span of datasets.

A large number of political facts with time stamps are kept in the *Integrated Crisis Early Warning System* (ICEWS), e.g., the temporal fact *(Barack Obama, Demand, Iran, 2014-07-17)*. The two sub-datasets, ICEWS05-15 and ICEWS14, are extracted from ICEWS by time interval, with timestamps for every fact with a temporal granularity of 1 day. Evidently, ICEWS14 and ICEWS05-15 contain all facts

Table 2
Statistics of the used TKG benchmarks.

Datasets	#Entities	#Relations	#Timestamps	#Train	#Valid	#Test	#Time Span
ICEWS14	7,128	230	365	72,826	8941	8963	2014
ICEWS05-15	10,488	251	4017	368,926	46,275	46,092	2005–2015
YAGO15K	15,403	34	198	110,441	13,815	13,800	1513–2017

from 2014-01-01 to 2014-12-31 and from 2018-01-01 to 2018-10-31, respectively. They create a new temporal data set successfully. In order to create YAGO15K, the entities in FB15K were linked with those in YAGO, which contain temporal information. The final data set contains all facts that have been successfully aligned. Since YAGO does not have temporal information for all facts, this data set is temporally incomplete and more challenging. YAGO15K contains the following examples:

- (Mel_Brooks, hasWonPrize, Grammy_Award)
- (Colin_Miller_(soccer), playsFor, Heart_of_Midlothian_F.C., occursSince, "1995-##-##")
- (Gale_Anne_Hurd, isMarriedTo, Brian_De_Palma, occursUntil, "1993-##-##")

4.1.2. Baseline methods

In order to demonstrate the effectiveness of T-GAE, we compare our model with a broad spectrum of state-of-the-art baselines, which contain both static and temporal KGE methods for the TKGC task, all of which were introduced in Section 2. From the static KG embedding methods, we choose four methods, including TransE [6], DistMult [4], SimplE [25], ComplEx [24], and RotatE [16]. All these methods omit the importance of temporal information. As for the existing state-of-the-art temporal KG embedding methods, we select several of them as the temporal baselines, such as TTransE [28], HyTE [14], TA-DistMult [13], ConT [30], TNTComplEx [33], DE-SimplE [32], ChronoR [34], TeMP-SA [41] and TempCaps [39]. It is worth noting that we do not choose RE-NET [35] and CyGNet [36] as temporal benchmark methods because they are designed for addressing extrapolation tasks rather than TKGC tasks.

4.1.3. Implementations details

We conduct all of our experiments in PyTorch, and all are trained on a Tesla V100 with 32 GB GPU memory. We employ a step-by-step training paradigm to train T-GAE, which means that the encoder (LSTM layer and GAT layer) and decoder are trained independently. The ConvKB decoder model is trained to carry out the TKGC task after our encoder has been taught to encode the information of entity and temporal relations. For baseline models, we apply the parameter settings in the respective articles to initialize all of the baselines, then turn them on our datasets for optimal performance for a fair comparison. In particular, we explicitly removed the available time information in these datasets when we used the static model on the temporal benchmark and only used the static triples to match the static methods. As for T-GAE, we employed an Adam optimizer with a weight decay rate of *1e-5* and *5e-6* for the encoder and decoder, respectively. Additionally, our basic experimental setup is as follows: the learning rate $l = 1e\text{-}3$, batch size $b = 4096$, the initial entity and relation embedding size $d = 100$, negative sampling ratio $n = 2$, and dropout rate $drop\_out = 0.3$ for the ICEWS14 and ICEWS05-15 datasets. In the final output layer, we set both the entity and the relation embedding size to 200. For YAGO15k dataset, we obtained $d = 100$, $l = 1e\text{-}3$, $b = 1024$, $n = 2$, $drop\_out = 0.2$.

4.2. Evaluation protocol

In the two sub-tasks of temporal link prediction, head entity prediction and tail entity prediction focus on predicting a quadruple (*h, r, t, T*) with *h* or *t* missing at a given time T, such as (*h, r,?, T*) and (*?, r, t, T*). In our experimental setup, since T-GAE is developed for the TKGC problem rather than for extrapolation, the train/validation/test set ratio was randomly split and sampled to be roughly 80%/10%/10%. In our experiment, two precise evaluation criteria refined metrics are reported to evaluate the performance of each model on all test data, including Mean Reciprocal Rank (MRR) and Hits@1/3/10. Their definitions are as follows:

- **Mean Reciprocal Rank:** Mean Reciprocal Rank (MRR) represents the reciprocal of the average ranking of the correct entities predicted by the model among the candidate entities. The MRR is more stable than Mean Rank (MR), which is heavily influenced by a single incorrect prediction. The larger the MRR value, the higher the effectiveness of the model.

Table 3
Experimental results on ICEWS14, ICEWS05-15 and YAGO15k datasets under the filter setting.

Models	ICEWS14			ICEWS05-15			YAGO15K		
	MRR	Hits@3	Hits@10	MRR	Hits@3	Hits@10	MRR	Hits@3	Hits@10
TransE	28.0	–	63.70	29.4	–	66.3	29.6	–	46.8
DistMult	43.9	–	67.2	45.6	–	69.1	27.5	–	43.8
SimplE	45.8	51.6	68.7	47.8	53.9	70.8	–	–	–
ComplEx	47.0	54.0	71.0	49.0	55.0	73.0	36.0	36.0	54.0
RotatE	41.8	47.8	69.0	30.4	35.5	59.5	–	–	–
TTransE	25.5	–	60.1	27.1	–	61.6	32.1	–	51.0
HyTE	29.7	41.6	65.5	31.6	44.5	68.1	–	–	–
TA-DistMult	47.7	36.6	68.6	47.4	–	72.8	29.1	–	47.6
ConT	18.5	20.5	31.50	16.4	18.9	27.20	–	–	–
TNTComplEx	60.72	65.92	77.17	66.64	71.82	81.67	35.94	36.84	53.75
DE-SimplE	52.6	59.2	72.5	51.3	57.8	74.8	–	–	–
TeMP-SA	60.7	68.4	**84.0**	68.0	76.9	**91.3**	–	–	–
ChronoR(k = 2)	62.53	66.88	77.31	67.50	72.29	82.03	36.62	37.92	53.79
TempCaps	48.9	54.4	67.9	52.1	57.6	70.5	–	–	–
T-GAE (ours)	**66.84**	**71.94**	80.87	**69.61**	**77.07**	86.23	**60.40**	**62.0**	**67.78**

$$MRR = \frac{\sum_{(h,r_{tem},t)\in test}\frac{1}{rank(h,r_{tem},t)}}{|test|}$$

- **Hit@N:** Hit@N denotes the percentage of correct entities predicted by the model in the top N rankings after scoring in descending order of the corresponding scores. Similar to MRR, The larger the Hit@N value, the higher the prediction accuracy of models.

$$Hit@1/3/10 = \frac{\sum_{(h,r_{tem})\in test} val(rank(h,r_{tem},t)\leqslant 1/3/10)}{|test|}$$

Similar to earlier works, we also applied the filtered setting introduced in detail in [40]. When computing evaluation metrics, if corrupt triples already appeared in one of the train/validation/test sets, they would be removed from these sets. Results on the three datasets reported in earlier related works are the best numbers reported in their respective papers. All models are trained three times for each evaluation metric, and we present the average evaluation metric for each model. As shown in Table 3, the best result of the KGE models is in **bold**, and the second best of the KGE models is underlined.

4.3. Performance Comparison on TKG

In this section, we report and analyze the performance of our proposed model T-GAE compared with the existing baseline methods. Table 3 illustrates the results generated by different models for the temporal link prediction task on ICEWS14, ICEWS05-15, and YAGO15k datasets. The results show that the T-GAE performs significantly better than other baseline approaches by improving on nearly all datasets, which strongly validates the superiority of our model. In terms of link prediction MRR, the T-GAE consistently outperforms these baselines, including static KGE methods and temporal KGE methods. In terms of the Hits@3 metric, the T-GAE is better than or on an equal level with previous studies. As for the Hits@10 metric, one of the most important metrics, our proposed model does not reach SOTA results in ICEWS14 and ICEWS05-15. We believe this is due to the fact that the number of relations and timestamps in the two datasets is significantly higher than that of YAGO15k, leading to higher time complexity and time smoothness. However, the Hits@10 results of our model are also convincing, which became the second-best model in ICEWS14 and ICEWS05-15 datasets and achieved SOTA performance in YAGO15k. For example, our model achieves a Hits@10 of 80.87, while the Hits@10 metric of the best-performed model TeMP-SA is 84.0% on the ICEWS14 dataset.

In YAGO15k datasets, the timestamp in quadruples is not just a time point but a time expression with a time modifier, such as "occursSince" and "occursUntil". The YAGO15k dataset's characteristics make many baseline KGE approaches either experimentally underperform or even fail to achieve the temporal link prediction task. As shown in Table 3, the experimental performance of TA-DistMult, TNTComplEx, and ChronoR is particularly poor. DE-SimplE and ConT are unable to produce results using the YAGO15k dataset because they cannot handle the timestamps in this form. In addition, DE-SimplE only considers the evolution of entities over time and ignores the effect of time on the relationships between entities. A large number of parameters makes ConT inefficient and lead to overfitting the model, so the results on the ICEWS datasets are not ideal. Furthermore, we observe an important phenomenon: both TA-DistMult and DE-SimplE outperform the respective static models DistMult and SimplE on the two ICEWS datasets, which largely reflects the importance of temporal information for the TKGC task. However, both of them consider triples independently and do not consider the context information, leading to poor performance. Our T-GAE consistently outperforms them in terms of MRR, Hits@3, and Hits@10, which verifies the superiority of fusing graph context information in TKGC.

On the ICEWS14 datasets, our model outperforms the baseline models on MRR and Hit@3. For MRR and Hit@3, our proposed model improves over the second-best model by 4.31% and 3.54%, respectively. On the ICEWS05-15 dataset, the results of T-GAE are much better than other baselines. For instance, the MRR metric of T-GAE performs better than any of the basic methods and improves by 1.61% over the second-best model (TeMP-SA). As for Hit@3, T-GAE also achieves the SOTA result and improves by 0.17% over TeMP-SA. On the YAGO15k dataset, since the format of expressing time in this dataset is entirely different from that on ICEWS datasets, only a few methods can handle such a time data format and get corresponding results, which are not very well. However, the proposed model outperforms other baseline methods by far, achieving the SOTA in all metrics, as shown in Table 3.

4.4. Ablation study

As introduced in the methods section, the encoder of T-GAE consists of an LSTM network, which deals with temporal information in TKGs, and a GAT network, which enriches entity features by capturing central neighborhood information. In this section, we aim to analyze and discuss the contribution of each module to the overall model and to help better understand the effectiveness of each

Table 4
Influence of the n-hop neighborhood information and temporal information for temporal link prediction.

Model	ICEWS14			
	MRR	Hits@1	Hits@3	Hits@10
T-GAE w/o n-hop	66.50	58.17	71.93	81.12
T-GAE w/o time	60.82	48.39	70.30	81.07
T-GAE	**66.84**	**58.95**	**71.94**	**80.87**

module by constructing ablation experiments. To do so, we generate two variants of T-GAE by adjusting the use of time information and n-hop neighborhood information and compare the performance of each variant on the two ICEWS14 datasets: (1) whether or not we apply time information in TKGs; (2) the application of n-hop neighborhood information. The results of the ablation studies are presented in Table 4 and Fig. 8.

As illustrated in Fig. 8, *T-GAE w/o time* is one of the T-GAE variants, which removes the LSTM layer and cannot handle the temporal information in the TKGs. *T-GAE w/o time* performs particularly poorly in the MRR, Hits@1, and Hits@3 metrics. The MRR metric shows a significant dip in *T-GAE w/o time* compared to T-GAE, down 6%. Similar to the MRR metric, *T-GAE w/o time* decreased by 10.6% and 1.6%, respectively, for the Hits@1 and Hits@3 metrics. Hence, we can conclude that the absence of an LSTM layer to incorporate temporal information and obtain new relational embeddings carrying temporal information affects the experimental results significantly from the experimental data in Table 4, which illustrates the importance of the temporal process of the LSTM layer for T-GAE. The comparison of experimental results strongly validates the effectiveness of applying LSTM layers to our model.

Table 4 illustrates the experimental results of the model under different orders of neighborhood aggregators. We run experiments on all datasets employing a 2-hop neighborhood aggregator. From the results in Table 4, one of the variants of T-GAE named *T-GAE w/o n-hop* is unable to outperform our T-GAE on ICEWS14 datasets. Removing the n-hop neighborhood information can result in a decrease in MRR as well as a decrease in other metrics. For the MRR metric, using a single-hop neighborhood aggregator can lead to a drop of 0.34% compared to our original model T-GAE. In addition, the Hit@1 result of this variant decreased by 0.78%. The Hit@3 and Hit@10 results for this variant are at the same level as the T-GAE, which indicates that both our T-GAE and this variant can adequately aggregate important neighborhood information associated with the central node, including the influence of neighboring nodes and relations. This result indirectly proves our inference that the more distant the relationship from the central node, the less the neighbor nodes contribute to it. It also shows the correctness of our aggregation of only 2-hop neighborhood information after considering the computational overhead and time complexity. Thus, we can conclude that it is helpful to learn to predict facts by aggregating information from multi-hop neighborhoods.

Table 5 illustrates the effect of two important parameters (*embedding size* and *number of head attention*) on link prediction results, and also presents the time cost overhead in the corresponding cases. We evaluate parameter sensitivity on the datasets ICEWS14 and YAGO15k. As shown in Table 5, the number of attentional heads of the MHA has a significant influence on the final experimental results, both for the metric MRR and Hits@3. On the dataset ICEWS14, after the number of heads is increased from 1 to 2, the metric MRR gets a huge boost, by 24.81%, and the value of the metric Hits@3 also gets a boost, by improved by 8.6%. This result demonstrates the effectiveness of the MHA mechanism, which can help T-GAE improve the aggregation process of graph context information to generate more accurate and expressive embedding. Also, we observe in Table 5 that the time overhead cost of T-GAE increases rapidly as the number of heads increases, and the time overhead increases by 73.6% as the number of heads increases from 1 to 2. Table 5 reveals that embedding size is also an important parameter that affects the link prediction results. *Embedding size* represents, to some extent, the expressiveness of the entity/relation embedding and feature information. On the dataset YAGO15k, it can be found that increasing the embedding size of T-GAE from 50 to 100 gives a small improvement in MRR and Hits@3, but in terms of time overhead, the latter increases by 41.5% - costing 1.62 h more time. Increasing the embedding size has some improvement on the final experimental results, but the effect is more limited, leading to a significant unnecessary time cost. The time cost of T-GAE is influenced

Fig. 8. Effect of Time Information for the TKGC Task.

Table 5
Effect of important parameters on link prediction results and the corresponding time costs.

Datasets	Embedding Size	Number of Head Attention (MHA)	MRR	Hits@3	Time Costs
ICEWS14	100	1	42.03	63.34	2.61 h
		2	**66.84**	**71.94**	**4.53 h**
YAGO15k	50	2	59.76	61.59	3.90 h
	100		**60.40**	**62.0**	**4.52 h**

by not only the number of attention heads, but also the embedding size. Both of them increase the complexity of the model to some extent. After considering the time cost overhead, we finally train the model with embedding size = 100 and head = 2 as our optimal settings.

5. Conclusion and future work

In this paper, we propose a novel ***T**imespan-aware **G**raph **A**ttention-based* **E**mbedding model named T-GAE for Temporal Knowledge Graph Completion. T-GAE consists of a digit-level LSTM encoder for incorporating complex time information, a GAT encoder for capturing the complex interaction features between multi-relational facts, and a decoder. To the best of our knowledge, the proposed model is the first KGE method, where GAT and LSTM networks are simultaneously applied to TKGC tasks. We conducted experiments on the ICEWS14, ICEWS05-15, and YAGO15K datasets. The experimental results demonstrate that T-GAE outperforms several state-of-the-art static and temporal KGE methods for TKGC.

For future work, we intend to extend our method to better capture higher-hop relations between entities in our graph-attention model to avoid exponential growth in computing overhead. Furthermore, our model only focuses on dealing with datasets where time is expressed as time points. We will investigate how to incorporate the time interval data into the TKG facts.

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

No data was used for the research described in the article.

References

[1] Y. Cao, X. Wang, X. He, et al., Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences, in: Proceedings of the 2019 World Wide Web Conference, 2019, pp. 151–161.
[2] D. Diefenbach, K. Singh, P. Maret, Wdaqua-core1: a question answering service for rdf knowledge bases, in: Companion Proceedings of the Web Conference 2018, 2018, pp. 1087–1091.
[3] L. Feddoul, Semantics-driven Keyword Search over Knowledge Graphs. In DC@ ISWC. 2020. pp. 17-24.
[4] B. Yang, S.W. Yih, X. He, et al. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In Proceedings of the 2015 International Conference on Learning Representations, 2015. 2015.
[5] M. Nickel, V. Tresp, H.P. Kriegel, A three-way model for collective learning on multi-relational data. Proceedings of the 2011 International Conference on Machine Learning, 2011.
[6] A. Bordes, N. Usunier, A. Garcia-Duran, et al., Translating embeddings for modeling multi-relational data. Advances in neural information processing systems. 2013. 26.
[7] Z. Wang, J. Zhang, J. Feng, et al. Knowledge graph embedding by translating on hyperplanes. In Proceedings of the 2014 AAAI Conference on Artificial Intelligence. 2014. pp. 1112-1119.
[8] Y. Lin, Z. Liu, M. Sun, et al. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the Twenty-ninth AAAI Conference on Artificial Intelligence. 2015. pp. 2181-2187.
[9] R. Jenatton, N. Roux, A. Bordes, et al. A latent factor model for highly multi-relational data. Advances in neural information processing systems, 2012. 25. pp. 3176-3184.
[10] A. Bordes, X. Glorot, J. Weston, Y. Bengio, A semantic matching energy function for learning with multi-relational data, Machine Learning. 94 (2) (2014) 233–259.
[11] R. Socher, D. Chen, C.D. Manning, et al., Reasoning with neural tensor networks for knowledge base completion, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, pp. 926–934.
[12] Y. Dai, S. Wang, N.N. Xiong, W. Guo, A survey on knowledge graph embedding: Approaches, applications and benchmarks, Electronics 9 (5) (2020) 750.
[13] 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.
[14] S.S. Dasgupta, S.N. Ray, T.P. 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.
[15] M. Nickel, L. Rosasco, T. Poggio, Holographic embeddings of knowledge graphs, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 1955–1961.
[16] Z. Sun, Z.H. Deng, J.Y. Nie, et al., RotatE: knowledge graph embedding by relational rotation in complex space. Proceedings of the 2018 International Conference on Learning Representations, 2018.
[17] T.D.N. Dai Quoc Nguyen, D.Q. Nguyen, D. Phung, A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics. 2018. pp. 327-333.

[18] T. Dettmers, P. Minervini, P. Stenetorp, et al., Convolutional 2D knowledge graph embeddings. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. 2018. pp. 1811-1818.
[19] P. Veličković, G. Cucurull, A. Casanova, et al., Graph Attention Networks. Proceedings of the 2018 International Conference on Learning Representations, 2018.
[20] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Computation 31 (7) (2019) 1235–1270.
[21] R.K. Jørgensen, C. Igel, Machine learning for financial transaction classification across companies using character-level word embeddings of text fields, Intelligent Systems in Accounting, Finance and Management. 28 (3) (2021) 159–172.
[22] Y. Lin, Z. Liu, H. Luan, et al., Modeling Relation Paths for Representation Learning of Knowledge Bases. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 705-714.
[23] G. Ji, S. He, L. Xu, et al., 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, 2015, pp. 687–696.
[24] T. Trouillon, J. Welbl, S. Riedel, et al., Complex embeddings for simple link prediction, in: Proceedings of the 2016 International Conference on Machine Learning, 2016, pp. 2071–2080.
[25] S.M. Kazemi, D. Poole, Simple embedding for link prediction in knowledge graphs, Advances in neural information processing systems 31 (2018).
[26] B. Perozzi, R. Al-Rfou, S.S. Deepwalk, Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[27] M. Schlichtkrull, T.N. Kipf, P. Bloem, et al., Modeling relational data with graph convolutional networks, in: Proceedings of the 2018 European Semantic Web Conference, 2018, pp. 593–607.
[28] T. Jiang, T. Liu, T. Ge, et al., Encoding temporal information for time-aware link prediction, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 2350–2354.
[29] J. Zhang, Y. Sheng, Z. Wang, et al., TKGFrame: a two-phase framework for temporal-aware knowledge graph completion, in: Proceedings of the 2020 Asia-Pacific Web and Web-Age Information Management Joint International Conference on Web and Big Data, 2020, pp. 196–211.
[30] Y. Liu, W. Hua, K. Xin, et al., Context-aware temporal knowledge graph embedding, in: Proceedings of the 2020 International Conference on Web Information Systems Engineering, 2020, pp. 583–598.
[31] R. Goel, S.M. Kazemi, M. Brubaker, et al. Diachronic embedding for temporal knowledge graph completion. In Proceedings of the AAAI Conference on Artificial Intelligence. 2020. pp. 3988-3995.
[32] Z. Han, P. Chen, Y. Ma, et al. DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. 2020: 7301-7316.
[33] T. Lacroix, G. Obozinski, N. Usunier, Tensor decompositions for temporal knowledge base completion. Proceedings of the 2019 International Conference on Learning Representations, 2019.
[34] A. Sadeghian, M. Armandpour, A. Colas, et al., ChronoR: rotation based temporal knowledge graph embedding. In Proceedings of the AAAI Conference on Artificial Intelligence. 2021. pp. 6471-6479.
[35] W. Jin, M. Qu, X. Jin, et al. Recurrent Event Network: Autoregressive Structure Inferenceover Temporal Knowledge Graphs. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. 2020: 6669-6683.
[36] C. Zhu, M. Chen, C. Fan, et al., Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In Proceedings of the AAAI Conference on Artificial Intelligence. 2021. pp. 4732-4740.
[37] M. Welling, T.N. Kipf, Semi-supervised classification with graph convolutional networks. Proceedings of the 2016 International Conference on Learning Representations, 2016.
[38] A. Vaswani, N. Shazeer, N. Parmar, et al., Attention is all you need, Advances in neural information processing systems 30 (2017).
[39] G. Fu, Z. Meng, Z. Han, et al., TempCaps: a capsule network-based embedding model for temporal knowledge graph completion, in: Proceedings of the Sixth Workshop on Structured Prediction for NLP, 2022, pp. 22–31.
[40] Jain P, Rathi S, Chakrabarti S. Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. 2020: 3733-3747.
[41] J. Wu, M. Cao, J.C.K. Cheung, et al., TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 5730–5746.
[42] S. Vashishth, S. Sanyal, V. Nitin, et al., Composition-based Multi-Relational Graph Convolutional Networks. Proceedings of the 2020 International Conference on Learning Representations, 2020.
[43] Z. Han, P. Chen, Y. Ma, et al., Explainable subgraph reasoning for forecasting on temporal knowledge graphs. Proceedings of the 2021 International Conference on Learning Representations, 2021.
[44] C. Xu, Y.Y. Chen, M. Nayyeri, et al., Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 2569–2578.
[45] J. Zhang, S. Liang, Y. Sheng, J. Shao, Temporal knowledge graph representation learning with local and global evolutions, Knowledge-Based Systems 251 (2022) 109234.