



# FTMF: Few-shot temporal knowledge graph completion based on meta-optimization and fault-tolerant mechanism

Luyi Bai<sup>1,2</sup> · Mingcheng Zhang<sup>1</sup> · Han Zhang<sup>1</sup> · Heng Zhang<sup>1</sup>

Received: 15 April 2022 / Revised: 1 July 2022 / Accepted: 29 July 2022 /  
Published online: 4 August 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

## Abstract

Traditional knowledge graph completion mainly focuses on static knowledge graph. Although there are efforts studying temporal knowledge graph completion, they assume that each relation has enough entities to train, ignoring the influence of long tail relations. Moreover, many relations only have a few samples. In that case, how to handle few-shot temporal knowledge graph completion still merits further attention. This paper aims to propose a framework for completing few-shot temporal knowledge graph. We use self-attention mechanism to encode entities, use cyclic recursive aggregation network to aggregate reference sets, use fault-tolerant mechanism to deal with error information, and use similarity network to calculate similarity scores. Experimental results show that our proposed model outperforms the baseline models and has better stability.

**Keywords** Few-shot temporal knowledge graph · Long-tail relations · Meta-optimization

## 1 Introduction

Knowledge graphs (KGs) play an important role in artificial intelligence, and have been applied in many applications such as event forecasting [28], intelligent question answering [24, 39], and social network analysis [40], etc. Knowledge graph is a graph data structure, in which edges represent relations and nodes represent entities. Because knowledge graphs are constructed manually or semi-manually, most knowledge graphs are incomplete. For static knowledge graphs such as WordNet [22] and Freebase [1], most models adopt vector embedding, which vectorizes entities and relations, and graphs them to a low-dimensional continuous space for operation. On the basis of TransE [2], researchers put forward many variant models, such as TransH [37] and TransR [18]. These models have achieved good results in static knowledge graph completion.

---

✉ Luyi Bai  
baily@neuq.edu.cn

<sup>1</sup> School of Computer and Communication Engineering, Northeastern University (Qinhuangdao), Qinhuangdao 066004, China

<sup>2</sup> School of Informatics, University of Leicester, Leicester LE1 7RH, UK

Recently, temporal information begins to appear in knowledge graphs. Researchers expand and add temporal information to form a quadruple of the basis of static knowledge graphs such as ICEWS [3] and GDELT [17]. Similarly, these temporal knowledge graphs are incomplete as well. An incomplete entity knowledge graph can be expressed as  $(?, r, o, t)$ ,  $(s, r, ?, t)$  or  $(s, ?, o, t)$ . In the task of temporal knowledge graph completion, we need to complete the missing entities or relations. Because temporal information in temporal knowledge graph has certain sequence constraints, there are two main ways to complete temporal knowledge graph. One is to use the dynamic time series coding model in neural network to process temporal information. For example, TA-TransE [5] model and TA-DistMult [5] model both use recurrent neural network to serialize temporal information, so that the models can process the invisible time series in the future. The other is to embed temporal information on the basis of static knowledge graph. For example, TTransE [16] model adds vectorization of temporal information on the basis of TransE model and graphs it to the corresponding space for calculation. These methods have achieved good performance in completing temporal knowledge graph.

However, most researches assume that each relation has enough entities to train, ignoring the influence of long tail relation, that is, each relation only has a small number of entities. For example, "competition" relation and "cooperation" relation have a large number of instances in knowledge graph or temporal knowledge graph, but the number of "fuhrer" or "president" relation is very small. In that case, researchers put forward the concept and method of few-shot knowledge graph completion, such as FAAN model [30], Gmatching model [41], MateR model [37], and FSRL [44] model. These models are developed for static knowledge graphs with few samples, and cannot explain temporal knowledge graphs. The encoders they use cannot embed the temporal relation between entities into the models, and the information sharing between fewer entities and the adverse effects caused by wrong information are not taken into consideration. To solve these problems, we propose a new model denoted FTMF. The model uses self-attention mechanism to aggregate the temporal information in the neighborhood to represent entities, uses cyclic automatic aggregation network to aggregate reference machines to enhance interaction ability, and uses fault-tolerant mechanism to reduce the influence of error information in datasets. Finally, similarity network is used to score similarity. The main contributions are described as follows:

- Raising the concept of few-shot temporal knowledge graph completion.
- Constructing the time series neighbor encoder.
- Establishing an aggregation network of cyclic automatic encoders for processing temporal information.
- Proposing a fault-tolerant mechanism to reduce the impact of error information.
- Carrying out experiments and obtaining good performance.

The rest of paper is organized as follows. We introduce the related work in Section 2. After proposing problem formulation in Section 3, Section 4 describes our model in detail. Experimental evaluations are given in Section 5. Section 6 concludes the paper.

## 2 Related work

Concerning on completing few-shot temporal knowledge graph, several categories of approaches are related to our work according to their focuses, including static knowledge graph completion, temporal knowledge graph completion, and few-shot knowledge graph completion.

## 2.1 Static knowledge graph completion

Researchers have put forward many models for static knowledge graph completion. These models can be roughly divided into two categories. The first category is translation model, which mainly vectorizes and maps the knowledge information in static knowledge graph to one or more low-dimensional spaces, and calculates similarity by calculating loss functions. TransE [2] translates the triple  $(s, r, o)$  as vector into the same low-dimensional space. In TransE model, if  $s+r \approx o$  is true, the prediction is correct. However, TransE model only get great performance for 1–1 relation, not good fit 1-N, N-1 and N–N relation. To address this problem, Wang et al. [37] propose TransH, which translates subject vector to the front of the object vector by relation and projects the subject vector and object vector onto a plane associated with the current relation. TransR [18] improves the expression ability of TransE by splitting the entity vector representation space and relation representation space, but it has more parameters than TransE. In order to cut the number of parameters, Ji et al. [10] propose TransD to dynamically obtain the projection matrix of relation by cross product calculation using a vector related to entity and a vector related to relation. Compositional models learn compositional vector representations of entire knowledge graph. The second category is semantic model, which mainly calculates a similarity score through the latent semantics between entity vector and relation vector, and ranks the missing parts according to the calculated similarity score. DistMult [43] uses more flexible linear mapping and assume that entities are represented as vectors, relations are represented as matrices, and think of the relation as a linear change in the vector space. Liu et al. [19] analyze the basic structure of analogical reasoning in knowledge graph, and adds two constraints to the representation of the relational matrix during his learning process to improve the compositional reasoning ability of DistMult. RESCAL [26] adopts a relation weight matrix to interact the latent features of entities, but its function is too simple, which causes it cannot get efficient vector representations. In order to have better representations, NTN [32] proposes a standard neural network layer combining with a bilinear tensor layer, and HoLE [25] uses a circular correlation operation to improve RESCAL model.

## 2.2 Temporal knowledge graph completion

Because temporal knowledge graph has the property of time series constraint, therefore, the completion methods for temporal knowledge graphs are mainly divided into two different types. The first one is to use the dynamic time series coding model in neural network to process temporal information. The difficulty of this kind is how to deal with the invisible time series in the future, because these time series are invisible and cannot be used directly in the training process of the model. To solve this problem, researchers have also thought of ways to solve it. For example, TA-TransE [5] model and TA-DistMult [5] model both use recurrent neural network to serialize time information, so that the model can deal with the invisible time series in the future. In addition, Know-Evolve [33] model proposed by Trivedi et al. is an in-depth evaluation of the structure of knowledge semantic network, in which quaternary coding is multivariate point processing, and entity representation can evolve with time. In addition, RE-Net [13] model selects the neighborhood of aggregated entities as their historical information, and uses recurrent neural network to model the time dependence. Chrono-Transformation [29] model proposes a method of rule mining and graph embedding to deal with temporal information in temporal knowledge graph. In addition, many tensor decomposition and neural network models are also applied to the processing of temporal information.

The second one is to embed the temporal information on the basis of static knowledge graph. This kind of methods mainly expand the temporal information of the static knowledge graph completion models, so that the models have the ability to complete the temporal knowledge graph. These methods mainly add temporal information to calculate the similarity score. The most classical one is that TTransE [16] adds the projection of temporal information and carries out vector calculation on the basis of TransE, and modifies the distance calculation formula  $h+r \approx t$  to  $s+r+t \approx 0$  to complete the temporal knowledge graph. Inspired by TransH, Jiang et al. put forward a new model called HyTE [11], which explicitly combines time in entity relation space by associating each timestamp with its corresponding hyperplane. There are many such ways, with the progress of technology, tensor decomposition and convolution neural network have gradually matured, and is applied to completing temporal knowledge graph. Similar to the proposed DE-Simple [21] model, which uses multi-temporal representation embedding to process temporal information, ConT [15] model uses tensor decomposition to deal with the relevant work, and then ATiSE [7] model is proposed. This model is to analyze the time series, so as to deal with the temporal information accordingly. These models have achieved good completion effect in dealing with temporal knowledge graph. After the proposed attention mechanism, Dysat [20] model came into being. This model adopts the position coding of temporal information by self-attention, thus embedding temporal information into vectors to complete the calculation.

### 2.3 Few-shot knowledge graph completion

In order to obtain good performance, it is often necessary to use a large amount of data to train the model. But in some real knowledge graphs, there are many relations and few entities. There are three meta-learning methods, which are based on metrics, model, and optimization. Its purpose is to use less samples to learn a new task quickly.

GMatching [41] proposes a one-shot relational learning framework, which uses the knowledge extracted from the embedded model and considers the learned embedded and one-hop graph structure to learn matching metrics. MateR [37] proposes a framework to predict the common but challenging link with few shots in KGs, that is, to predict the new triple about a relation only by observing several associated triples. S-shot link prediction enables the model to learn the most important knowledge by transmitting meta-information specific to relation, and the learning speed is faster, which corresponds to relation element and gradient element respectively. Xiong et al. [41] propose a metric-based long tail link prediction method, that is, when there are few sample instances of a certain relation, the subject entity is predicted by the header entity and the relation. In FSRL [44] proposed by Zhang et al., given a small set of reference entity pairs for each relation, learning can effectively infer the matching function of real entity pairs. REFORM [35] proposes a new method for error perception of small sample completion problem and a principle completion framework. Specifically, it constructs problems under the framework of less shot learning, and its goal is to accumulate meta-knowledge among different meta-tasks, and generalize the accumulated meta-knowledge into meta-test tasks, so as to realize the error-aware less shot knowledge graph. MTransH [27] establishes a global phase novel and focused neighborhood aggregator, which accurately integrates the neighborhood semantics of a few shot relations, so that it can filter noisy neighborhoods even when the neighborhoods are extremely sparse. FAAN [30] proposes an adaptive attention network based on adaptive entity and reference representation. Specifically, entities are modeled by an adaptive neighborhood encoder to identify their task-oriented roles, and references are modeled by an adaptive perceptual query aggregator to distinguish their contributions.

In addition, attention mechanisms are applied to capture fine-grained semantics of entities and references for better representation. P-INT [42] is able to infer and utilize the expressive encoding of the relation between two entities at the path level. In addition, P-INT can capture fine-grained matches, and can calculate path interactions instead of simply mixing interactions between each entity pair. MateP [12] can extract patterns with high performance through a module called convolution pattern learner, and then accurately measure the effectiveness of triples by matching queries and referring to patterns. FTAG [23] proposes a one-time learning framework, which is used to predict links in temporal knowledge graphs with few samples. FTAG model uses a self-attention mechanism to effectively deal with the time interaction modeling between different entities, and then the model uses a similarity calculation network to calculate the similarity score between a given query set and a (one-time) example.

## 2.4 Discussion

Different from the static knowledge graph completion methods, our model uses a temporal neighbor encoder to complete entity embedding, considering the impact of temporal information on the completion task. Different from the temporal knowledge graph completion methods, our model takes the long tail relation in the temporal knowledge graph into account, and trains the model based on meta-learning to solve the problem that there are only a few samples in some relations. Few-shot knowledge graph completion methods does not consider the information sharing among fewer entities and the adverse effects caused by wrong information. In our model, self-attention mechanism is used to aggregate temporal information, a cyclic automatic aggregation network is used to aggregate reference machines to enhance interaction ability, and fault-tolerant mechanism is used to reduce the impact of error information in datasets, so as to improve the completion performance on the few-shot temporal knowledge graph.

## 3 Problem formulation

### 3.1 Few-shot temporal completion task

The representation of temporal knowledge graph is a quaternary that can be described by  $(s, r, o, t)$ , where  $s$  and  $o$  represent entities,  $r$  represents relations, and  $t$  represents timestamps. In the task of temporal knowledge graph completion, there are mainly two kinds of tasks: completing the missing entities  $s$  or  $o$ , and completing the missing relations  $r$  between entities. In this paper, we study the first task to complete the missing objective entity  $o$ .

### 3.2 Few-shot temporal knowledge graph training

The purpose of training is to construct and train a model with only a few marked instances for each relational class, so as to complete the temporal knowledge graph with a few samples. The goal of meta-learning is to learn quickly from a few instances of the same concept and gain the ability to continuously adapt to more concepts. After defining the task requirements, meta-learning with the ability of fast training and self-learning with few samples is very suitable for our needs. Therefore, we use the meta-learning method based on optimizer to establish a multi-module framework to complete the task of temporal knowledge graph with few

samples. In this meta-learning framework, we have a large number of task sets, and we need to merge the task sets. In this context, each task corresponds to a corresponding scene, and the number of samples is very small. In this way, we can use the information between different tasks and share information, which can solve the problem of missing information caused by scarce data. By using embedded shared information, we can solve the negative problem caused by less data information in less sample data to a certain extent, improve the usability of each task information, and improve the performance of temporal knowledge graph completion with less samples.

First of all, we have a bunch of tasks, which are divided into training set  $D_r^{train}$  and test set  $D_r^{test}$ . In the training set  $D_r^{train}$ , only entity pairs with few samples of relation  $R$  are included, while in the test set  $D_r^{test}$ , all entity pairs of relation  $R$  are included. Then we define the loss function of relation  $r$  as the following form:

$$\mathcal{L}_\Theta(s_i, o_i, t_i | Q_{s_i, r, t_i}, D_r^{train}) \tag{1}$$

where  $\Theta$  is a collection that represents all the parameters in the model and  $Q_{s_i, r, t_i}$  is the remaining candidate entities set.

Then we define the objective function of the model as the following form:

$$\min_{\Theta} \mathbb{E}_{\mathcal{T}} \left[ \sum_{(s_i, o_i, t_i, Q_{s_i, r, t_i}) \in D_r^{test}} \frac{\mathcal{L}_\Theta(s_i, o_i, t_i | Q_{s_i, r, t_i}, D_r^{train})}{|D_r^{test}|} \right] \tag{2}$$

where  $|D_r^{test}|$  is the number of quad  $(s, r, o, t)$  in  $D_r^{test}$ .

In the following Sect. 3, we will talk about how to calculate and optimize the above functions to achieve our results.

### 3.3 Related settings

The problem of temporal knowledge graph completion with few samples mainly comes from the problem of scarce training data. In the previous researches on static knowledge graph completion, the training framework with few samples based on meta-learning can deal with the problem of static knowledge graph completion very well. On this basis, we further extend it to the task of temporal knowledge graph completion with few samples.

Firstly, given a temporal knowledge graph, we divide the relations into two groups according to the frequency of occurrence, which are frequent relations and sparse relations. Since the goal of our paper is temporal knowledge graph completion with few samples, we use sparse relations to construct the task set needed for our model training. In the task set, each relation has its own corresponding test set  $D_r^{test}$  and training set  $D_r^{train}$ . A few-shot temporal knowledge graph completion task is always defined for a specific relation. During prediction, there is usually more than one quadruple to complete in a task. Under a given support set, we call the set of all quadruplets to be predicted as the query set. The query set and support set are represented as follows:

$$D_r^{test} = \{s_i, r, o_i, t_i | s_i, o_i \in E\} \tag{3}$$

$$D_r^{train} = \{s_i, r, o_i, t_i | s_i, o_i \in E\} \tag{4}$$

where  $E$  stands for entity collection, which contains all the entities in the dataset.

The support set  $D_r^{train}$  contains a tagged instance of a task set. In each training scenario, a relation and a quad containing the relation are selected randomly to form the support set used by our training. We use a time-dependent approach to pick quads for the query set. When building a query set, the quad of the query set is limited by the distance between them and the timestamp in the supporting set, which is expressed as follows:

$$D_r^{train} = \{s_i, r, o_i, t_i | s_i, o_i \in E, t_i \in [t_i - y, t_i + y]\} \tag{5}$$

where  $t_i$  represents the timestamp of the query instance in the current query set, and  $y$  represents the distance from its corresponding timestamp in the supporting set. The support set  $D_r^{train}$  in Eq. 5 represents the set of training tuples before and after a specific time point. The representation is shown in Figure 1, and the solid line with arrows represents the temporal knowledge graph quadruple at the current timestamp.

The ultimate goal of model training is to optimize the score by using the loss function of each scenario training, so that entities are ranked according to the similarity score in the query set, and the higher the ranking entity object should be the real entity we need. In the whole task set, we divide all the relations into three mutually exclusive sets to ensure that there is no overlap between their relations. In order to ensure the accuracy and reliability of our experiment, we also require that the timestamps in the three sets are different from each other. The representation is shown in Figure 2.

In the whole training process, all entities and relations are invisible to the outside world. That is to say, given a temporal knowledge graph as the background and expressed as  $G' = (s, r, o, t | s, o \in E)$ ,  $E$  as the entity set and  $r$  as the relation, they are closed and visible internally in the whole training process. In this work, the background knowledge graph  $G'$  is a subset of the temporal knowledge graph  $G$ , and we remove the quadruplets used for training and testing.

### 4 Model

In this section, we propose a model named FTMF to complete the few-shot temporal knowledge graph. FTMF model includes time series neighbor encoder module, cyclic automatic coding aggregator module, fault-tolerant mechanism module, and similarity network module. The framework of FTMF model is shown in Figure 3. Firstly, all connected entities of a unified relation under a unified timestamp are encoded by neighbors, and the feature vector representation of an entity’s temporal neighborhood is outputted. After that, the reference set is recursively aggregated by a cyclic automatic aggregator. The relation is improved by fault-tolerant mechanism, and a query network is outputted. Finally, the similarity score of the reference set and the query set is calculated by using the similarity network.

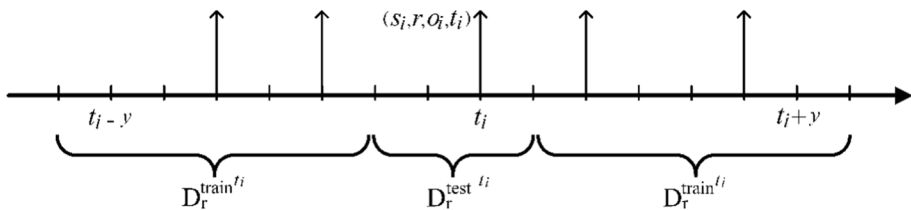
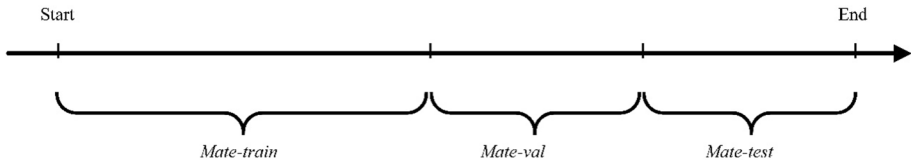


Figure 1 The construction of query set and support set





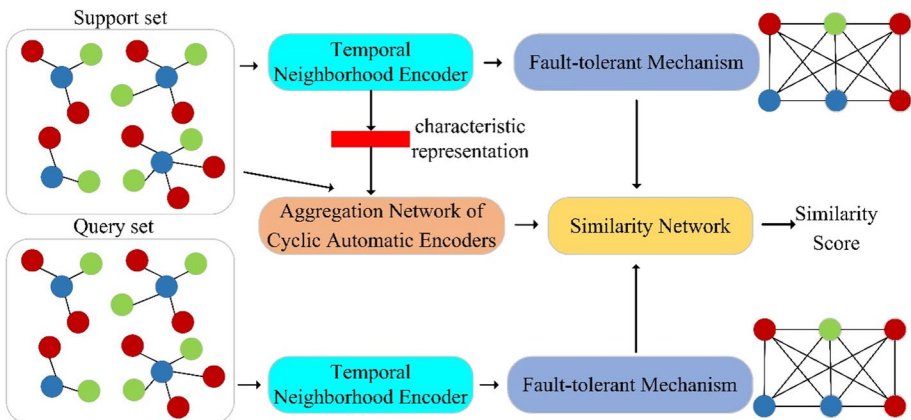
**Figure 2** The representation of training set, verification set and test set on timeline

### 4.1 Time series neighbor encoder

In this subsection, we propose a new neighbor encoder that can compute the representation of neighbor events to improve the representation of topic entities. Entity embedding based on relational information has been proposed and applied many times. It is proved that the local coding structure of explicit graph has good performance in relation prediction and can be applied to temporal knowledge graph completion. In the previous neighbor encoders, such as GMatching [41], Xiong et al. propose a neighbor encoder to enhance the embedding of entities by their one-hop neighbors; FSRL [44] designs a relation-aware heterogeneous neighbor encoder by considering the different influences of relational neighbors, and then encodes the features of the entity pair. All of them adopt static encoding mode. Although these methods can achieve good performance, it is obviously not suitable for our requirements. On this basis, we design a new encoder that combines snapshot aggregation and continuous aggregation to represent the neighbor encoding of a given entity under a certain time stamp.

In order to better represent the composition structure of entity, relation and temporal information, we uniformly express the set of neighbors (*relation, entity, time*) of a given header entity  $s$  as  $\mathcal{N}_h = \{(r_i, o_i, t_i) | (s, r_i, o_i, t_i) \in G'\}$ , where  $G'$  is the background temporal knowledge graph,  $r_i$ ,  $o_i$ , and  $t_i$  represent the  $i$ -th relation and the corresponding object entity and current temporal point of  $s$ , respectively.

Given the primary entity  $h$ , we define  $\mathcal{N}(h)$  as the set of all adjacent entities connected to the entity  $h$  with relation  $r$  at time  $t$ . Then the adjacent coding mainly consists



**Figure 3** The framework of FTMF model



of snapshot aggregation and continuous aggregation. Snapshot aggregation can encode a single-hop neighborhood at a given timestamp  $t$ , while continuous aggregation can generate a temporal neighborhood representation based on the previous timestamp  $t$ .

Snapshot aggregation mainly aggregates the local neighborhood information of a given specific timestamp  $t$ , and the aggregation mode and representation form are as follows:

$$\eta(\mathcal{N}(s)) = \sigma\left(\frac{1}{C_{s_r}} \sum \sum (W^T [e_r : e_s] + b)\right) \tag{6}$$

$$x^h = [\eta(\mathcal{N}(s)) : e_s] \tag{7}$$

where  $\sigma$  is a nonlinear activation factor function,  $C_{s_r}$  is a normalized factor,  $W \in R^{2d \times d}$  and  $b \in R^{d \times 1}$  represent Learnable parameters,  $e_r$  and  $e_s$  represent relations and entities respectively,  $[:]$  represents a concatenation.

Continuous aggregation can aggregate the previous  $l$ -step  $\{t-l, \dots, t-2, t-1\}$  timestamps into a snapshot sequence. Here we use an attention-based encoder and decoder model to model the sequence of events. The encoder part is used to encode the information and capture the time-dependent information in the sequence of events effectively. The function of this part is mainly completed by the attention layer and the position-wise layer.

The attention layer mainly projects the input sequence into a query and a set of key-value vectors. The specific way and representation form are as follows:

$$MultiHead(Q, K, J) = [head_1 : \dots : head_m]W^O \tag{8}$$

$$head_i = Attention(QW_i^Q, KW_i^K, JW_i^J) \tag{9}$$

$$Attention(Q, K, J) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)J \tag{10}$$

where  $W^O, W_i^Q, W_i^K$ , and  $W_i^J$  are parameter matrices, and  $W^O \in R^{2d \times hd_v}$ ,  $W_i^Q \in R^{2d \times d_q}$ ,  $W_i^K \in R^{2d \times d_k}$ ,  $W_i^J \in R^{2d \times d_j}$ .

In order to be suitable for our model, we add the coding work of the corresponding position in the input embedding of the model, so that our model can be applied to sequence order. The purpose of using position coding is to add the relative position information or absolute position information of each element to the input sequence of the model. The specific representation form is as follows:

$$PE_{(pos,2f)} = \sin\left(\frac{pos}{10000^{\frac{2f}{2d}}}\right) \tag{11}$$

$$PE_{(pos,2f+1)} = \cos\left(\frac{pos}{10000^{\frac{2f}{2d}}}\right) \tag{12}$$

where  $pos$  represents the position of the corresponding sequence, and  $f$  represents the dimension of the corresponding sequence.

The position-wise layer is a fully connected feedforward neural network, which transforms functions into matrix operations, and uses multi-layer networks to carry out iterative operations for many times, which are applied to each sequence in the same form.

$$FNN(x_T) = \max(0, x_T W_1 + b_1) W_2 + b_2 \tag{13}$$

In the encoder, we take the neighborhood snapshot representation sequence  $x = \{x_{t-1}, \dots, x_{t-2}, x_{t-1}\}$ , the number of layers of the feedforward neural network and the number of attention headers as the input of the encoder. By calculation, the neighbor snapshot sequence  $x$  of the input is finally mapped to a time-aware sequence output. The specific representation form is as follows:

$$output = [p_{t-1}, \dots, p_{t-2}, p_{t-1}] \tag{14}$$

$$p = Attention(x, num_{head}, num_{layers}) \tag{15}$$

where  $p$  represents the corresponding sequence output,  $num_{head}$  and  $num_{layers}$  represent the number of attention heads and the number of layers of the feedforward neural network respectively.

Therefore, we can calculate the neighborhood representation sequence of the main entity  $s$  at time  $t$ , and the representation is as follows:

$$\mathbb{N}(s) = \sigma([p_{t-1} : \dots : p_{t-2} : p_{t-1}] W^*) \tag{16}$$

where  $W^* \in R^{2dl \times d_{output}}$  is a parameter matrix,  $[:]$  represents a concatenation, and  $\sigma$  is a non-linear activation factor function.

The model diagram of the time series neighbor encoder is shown in Figure 4. All connected entities with the same relation at the same point in time first pass through the snapshot aggregation network, then act as input to the continuous aggregation network, and the result of the output is evaluated with the parameter matrix to finally output the feature representation of the entity.

### 4.2 Cyclic automatic coding aggregator

In this subsection, we design an aggregator network of cyclic automatic encoder aggregator to perform aggregation embedding for each relation. Because the existing model does not have the ability to deal with small sample instances interactively, we need to design a module to effectively formulate the aggregation embedding of reference set  $R_r$  for each relation  $r$  and complete the embedding of temporal information, so as to improve the performance of the model.

We can obtain the representation of  $(s_k, o_k, t_k)$  in the form of  $e_{s_k, o_k, t_k} = [\mathbb{N}(s_k) \oplus \mathbb{N}(o_k)]$ , by applying the neighbor encoder  $\mathbb{N}(s)$  to each entity pair  $(s_k, o_k, t_k) \in R_r$ . Learning to use reference set representation of entity pairs with few shots is a great challenge, because it needs to effectively model the interaction between different entity pairs and accumulate their expressive ability on this basis. We define the embedding of  $R_r$  by aggregating representations of all entity pairs in  $R_r$  [4, 31] as follows:

$$f_c(R_r) = \mathcal{AG}_{(s_k, o_k, t_k) \in R_r} \{e_{s_k, o_k, t_k}\} \tag{17}$$

where  $\mathcal{AG}$  is an aggregate function. In the whole model, it plays a role in pooling operation and feedforward neural network.

Aiming at the application of recurrent neural network aggregator in graph embedding and getting good results [6], we design a cyclic automatic encoder aggregator to deal with the interaction between few samples. Specifically, the entity pair embeddings  $e_{s_k, o_k, t_k} \in R_r$  are sequentially fed into a recurrent autoencoder by:

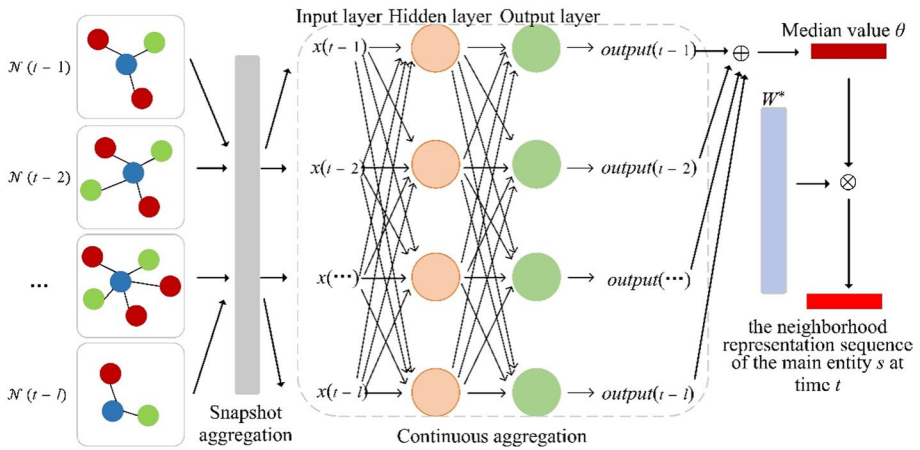


Figure 4 The diagram of the time series neighbor encoder

$$\epsilon_{s_1, o_1, t_1} \rightarrow n \rightarrow \dots \rightarrow n \rightarrow d_k \rightarrow \dots \rightarrow d_1 \tag{18}$$

where  $k$  is the size of reference set.

Both  $n_k$  and  $d_{k-1}$  are hidden states of the decoder.  $n_k$  stands for encoding,  $d_{k-1}$  stands for decoding, and  $n_k$  and  $d_{k-1}$  are calculated as follows:

$$n_k = RNN_{encoder}(\epsilon_{s_k, o_k, t_k}, n_{k-1}) \tag{19}$$

$$d_{k-1} = RNN_{decoder}(d_k) \tag{20}$$

where  $RNN_{encoder}$  represents recurrent encoder and  $RNN_{decoder}$  represents decoder.

Combined with the above information, we define the reconstruction loss for optimizing autoencoder as:

$$\mathcal{L}_{re}(R_r) = \sum_k \|d_k - \epsilon_{s_k, o_k, t_k}\|_2^2 \tag{21}$$

where  $\epsilon_{s_k, o_k, t_k}$  is the embedding of entity pair. Under the action of recurrent neural network aggregator, we get the decoding vector  $d_k$ . The role of  $\mathcal{L}_{re}$  is to merge with relation-level losses to optimize the representation for each entity pair, thereby improving the performance of the model.

Next, we embed the reference set. We aggregate all the hidden states of the encoder, and add residual links [8] and attention weights to further expand the reference set. We define  $f_e(R_r)$  as follows:

$$n'_k = n_k + \epsilon_{s_k, o_k, t_k} \tag{22}$$

$$\beta_k = \frac{\exp\{\mu_R^T(\mathcal{W}_R n'_k + b_R)\}}{\sum_{k'} \exp\{\mu_R^T(\mathcal{W}_R n'_{k'} + b_R)\}} \tag{23}$$

$$f_e(R_r) = \sum_k \beta_k n'_k \tag{24}$$

where  $\mu_R \in \mathbb{R}^{(d \times d \times 1)}$ ,  $\mathcal{W}_R \in \mathbb{R}^{(d \times d \times 2d)}$ ,  $b_R \in \mathbb{R}^{(d \times d \times 1)}$  ( $d$ : pre-trained embedding dimension).

The processing of the aggregation network of cyclic automatic encoders is shown in Figure 5. We first use the steps in Eq. (18) to input the embedding of the entity pair into the cyclic automatic encoder aggregator, and combine the loss through the action of the encoder and decoder in Eq. (19) and Eq. (20) to obtain the final loss. The representation of the final reference set is then obtained through the aggregate processing of the hidden state units. The cyclic automatic encoder is mainly composed of two parts, namely encoder and decoder. The encoder combines the LSTM aggregation of a small number of reference sets and the entity’s feature representation vector generation relation with a small sample embedding. The decoder combines the LSTM aggregation of a small number of reference sets and the intermediate quantities of the entity’s feature representation vectors to calculate the loss function.

### 4.3 Fault-tolerant mechanism

Errors are common in temporal knowledge graphs with few samples, and will cause troubles to applications. In the process of completing the temporal knowledge graph with few samples, because the number of supported instances in each meta-training task (each relation) is extremely limited, it cannot support enough training to ensure the integrity. Therefore, even if there is a small amount of error information in the support set, it may have a great adverse impact on the information sharing and information utilization among different elements. In that case, it will affect the integrity of the temporal knowledge graph with few samples and the performance of the by-election model. In previous studies, we propose a new inter-neighbor encoder which can generate neighborhood information. In the process of entity and time information embedding, it can complete information embedding well and reduce the influence of periodic errors. However, due to data reasons, there will inevitably be some wrong information in the support set, so there may still be wrong query instances in the query set.

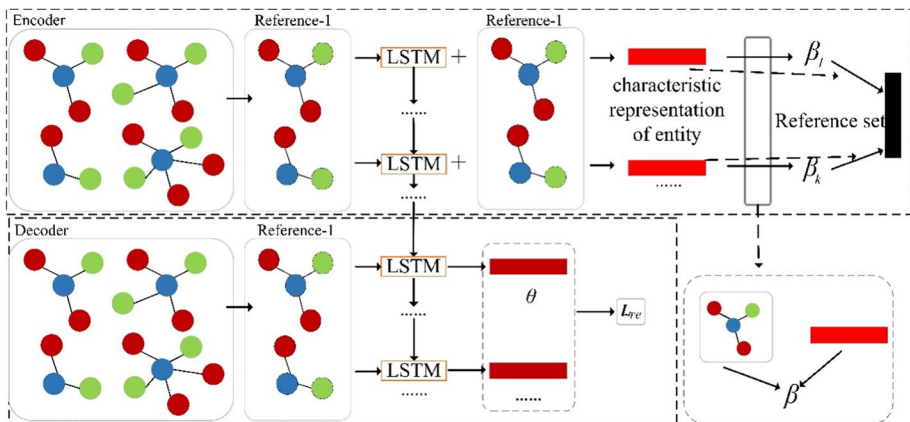


Figure 5 The cyclic automatic aggregation network for reference set

In the model, we use Graph Convolution Neural Network (GCN) [14] to calculate each different query instance, and generate the corresponding confidence weights of different relations, so as to reduce the inevitable error influence caused by the support set and improve the performance of the model. In detail, since incorrect information is inevitable in the support set, the levels of different relations should be different, and we should divide them accordingly. For example, in the support set, if there are a large number of entity pairs with error information in a relation, we need to reduce the confidence of query entity pairs belonging to the relation, that is, the relation is unreliable. Therefore, in this way, we need to set a confidence weight for each relation in the support set based on the instance information that each relation has. Specific to the model, in the support set, we need to measure the impact of different instances on specific query instances, and build a query-oriented graph, in which different support instances are represented by nodes and their intimacy is represented by edges. Therefore, the graph structure is different for different query instances, so we can apply it to different query instances, which has strong flexibility and adaptability. In the process of building the query graph, we first need to embed each node, that is, different query instances. The specific embed representation is as follows:

$$v_a = \phi_v(e_{sq} \oplus e_{sa} \oplus (e_{sa} + e_{sq}) \oplus (e_{sa} \odot e_{sq})) \tag{25}$$

$$V = (v_1, v_2, \dots, v_n) \in R^{n \times d} \tag{26}$$

where  $e_{sa}$  is the embedding of the supporting instance of step  $a$  in the supporting set and  $e_{sq}$  is the embedding of a specific query instance in the query set.  $\phi_v$  is a fully connected layer that graphs link input to a new embedding space.  $V$  is the final embedding matrix of a different node. In addition,  $\oplus$  represents the link factor, and  $\odot$  represents the product operation between elements.

Through the above calculation, we can interact and model between different supporting instances in the support set and specific query instances in the query set, so as to establish a graph embedded node matrix that can retrieve novelty.

After node matrix calculation, we need to process the similarity matrix of different nodes. The processing method is to use a fully connected layer to calculate the similarity matrix of different nodes in the query graph and normalize it by row. The specific calculation and processing methods are as follows:

$$[A]_{ij} = \phi(v_i)^T \phi(v_j) \tag{27}$$

$$\tilde{A} = \text{softmax}(A) \tag{28}$$

where  $[A]_{ij}$  represents the data information of  $i$ -th row and  $j$ -th column in matrix  $A$ .

Each row in matrix  $A$  is normalized by adapting softmax function. After that, in order to measure the credibility of each supporting instance in the support set to the query instance, we adopt a GCN layer with remaining links for calculation processing, and the specific calculation processing method is as follows:

$$\text{confidence} = \text{sigmoid}((V + \tilde{A}VW_v)W_u) \in R^n \tag{29}$$

where  $W_v \in R^{d \times d}$  and  $W_u \in R^d$  are learnable parameters,  $\tilde{A}VW_v$  can propagate information through different nodes in query-oriented graph, and  $V$  can be regarded as a residual link [36].

Because the node graph is a fully connected graph, it's not necessary to spend extra layers to calculate, so we only need one layer to propagate all information. After calculating the *sigmoid*

function, we can get a length vector, in which each element represents the confidence score of the support instance in the support set for a specific query instance. In this case, we need to obtain the maximum value on each row of the new matrix, thus generating the credibility weight of a specific query instance for each relation. The specific calculation method is as follows:

$$Confidence = reshape(confidence) \in R^{n \times d} \tag{30}$$

$$[w]_i = \max\{[w]_{i1}, [w]_{i2}, \dots, [w]_{id}\} \tag{31}$$

where  $[w]_i$  represents the confidence weight of the  $i$ -th relation for a specific query instance.

Therefore, we can quantify the reliability of the relation between specific query instances, which can reduce the inevitable errors caused by data problems in temporal knowledge graphs.

After completing the above steps, we use the concept of energy function [2, 34, 38] to calculate a unique energy fraction for each relation in the support set as follows:

$$rescore_i = -Energy(e_{sq}, r_i) = -\sigma(W_e e_{sq})^T \sigma(r_i) \tag{32}$$

where  $Energy$  represents the energy function,  $e_{sq}$  and  $r_i$  represent the advance and the  $i$ -th relations in the querying machine,  $W_e$  represents a trainable weight matrix, and  $\sigma$  is an activation function.

Finally, in order to get the distribution probability of each relation, we jointly calculate the above energy score and credibility weight, and the specific calculation method is as follows:

$$probability'_i = \frac{\exp(rescore_i)}{\sum_k^n \exp(rescore_k)} [w]_i \tag{33}$$

$$probability_i = \frac{probability'_i}{\sum_k^n probability'_i} \tag{34}$$

After that, the loss of each query instance in the query set can be calculated by cross entropy loss in the process of meta-training, and the specific calculation method is as follows:

$$\mathcal{L}_r = -\frac{1}{|D_r^{test}|} \sum_i^{|D_r^{test}|} \sum_j^n be_i^j \ln(probability_i^j) \tag{35}$$

where  $be_i^j$  indicates whether the  $i$ -th query instance belongs to the  $j$ -th relation, and there are only two values of  $be$ : if it belongs, the value of  $be$  is 1, otherwise, the value of  $be$  is 0.  $probability_i^j$ , which indicates the class allocation probability that the  $i$ -th query instance in the support set is allocated to the  $j$ -th relation category, specifically refers to the class allocation probability that the  $i$ -th query instance is allocated to the  $j$ -th relation category.

The query graph of fault-tolerant mechanism is shown in Figure 6. Firstly, the entities in query set and support set are processed into node matrix through bucket layer, and then the confidence level is calculated by GCN and neural network, and the confidence level is taken as edge. Finally, a fully connected query graph is formed.

### 4.4 Similarity network

In this subsection, we will present how to efficiently match the reference set  $R_r$  with each query pair  $(s_r, o_r, t_r)$  in the set of all query pairs of relation  $r$ . We add temporal information processing in the matching network, which makes the similarity score calculated by the matching

network more accurate. Based on the previous work, we can obtain two embedding vectors  $\epsilon_{s_l, o_l, t_l} = [f_\theta(s_l) \oplus f_\theta(o_l) \oplus f_\theta(t_l)]$  and  $f_e(R_r)$  respectively by applying the time-based relational aware heterogeneous neighbor encoder  $f_\theta$  and the reference set aggregator  $f_e$  to the query entity pair  $(s_l, o_l, t_l)$  and the reference set  $R_r$ . We adopt a recurrent processor  $f_\mu$  to perform multiple steps matching, in order to measure the similarity between two vectors. We define the  $t - th$  process step as follows:

$$gradient'_t, c_t = RNN_{match}(\epsilon_{s_l, o_l, t_l}, [ggradient_{t-1} \oplus f_e(R_r)], c_{t-1}) \tag{36}$$

$$gradient_t = gradient'_t + \epsilon_{s_l, o_l, t_l} \tag{37}$$

where  $RNN_{match}$  is the LSTM [9] cell, and it includes the input  $\epsilon_{s_l, o_l, t_l}$ , the hidden state  $gradient_t$  and the cell state  $c_t$ .  $gradient_T$  is the last hidden state after  $T$  “processing” step, and what it does is to refine embedding of query pair  $(s_l, o_l, t_l) : \epsilon_{s_l, o_l, t_l} = gradient_T$ .

In order to make a good calculation for the subsequent ranking optimization process, we use their inner product results between  $\epsilon_{s_l, o_l, t_l}$  and  $f_e(R_r)$  as their similarity score. The detailed flow of the matching network is shown in Figure 7. First, the query set and LSTM are combined to embed, then the reference set and LSTM are combined to calculate, and finally the similarity score is obtained.

### 4.5 Target mode training

In order to acquire the reference set  $R_r$  for the query relation  $r$ , we randomly sample a set of few positive (true) entity pairs  $\{(s_k, o_k, t_k) | (s_k, r, o_k, t_k) \in G\}$ . After that, we define the remaining positive (true) entity pairs as  $\mathcal{P}e_r = \{(s_l, o_l, t_l) | (s_l, r, o_l, t_l) \in G \cap (s_l, o_l, t_l) \notin R_r\}$  and use  $\mathcal{P}e_r$  as positive entity pairs. In addition, we contaminate the object entities and create a group of negative (false) entity pairs  $\mathcal{N}e_r = \{(s_l, o_l^-, t_l) | (s_l, r, o_l^-, t_l) \notin G\}$ . Thus, we can formulate the ranking loss as:

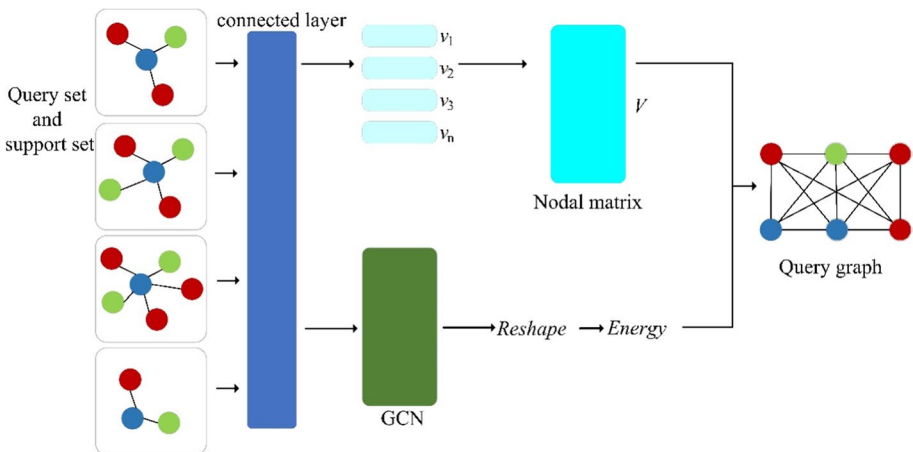


Figure 6 The query graph of fault-tolerant mechanism



$$\mathcal{L}_{rank} = \sum_r \sum_{(s_l, o_l, t_l) \in \mathcal{P}e_r} \sum_{(s_r, o_r, t_r) \in \mathcal{N}e_r} \left[ \xi + \mathcal{S}_{(s_l, o_l, t_l)} - \mathcal{S}_{(s_r, o_r, t_r)} \right]_+ \tag{38}$$

where  $[x]_+ = \max[0, x]$  is standard hinge loss, and  $\xi$  is the safety margin distance,  $\mathcal{S}_{(s_l, o_l, t_l)}$  and  $\mathcal{S}_{(s_r, o_r, t_r)}$  are the similarity scores between query pairs  $(s_l, o_l / o_l^-, t_l)$  and reference set  $R_r$ .

By taking advantage of the reconstruction loss  $\mathcal{L}_{re}$  of reference set aggregator, we can define the final objective function as follows:

$$\mathcal{L}_{joint} = \mathcal{L}_{rank} + \gamma \mathcal{L}_{re} + \mathcal{L}_r \tag{39}$$

where  $\gamma$  is trade-off factor between  $\mathcal{L}_{rank}$  and  $\mathcal{L}_{re}$ .

In order to minimize  $\mathcal{L}_{joint}$  and optimize the model parameters, we treat each relation as a task. We design a batch sampling based on meta-training procedure. Current temporal knowledge graphs such as GDELTA and ICEWS can play a huge role in question answering and personalized recommendation. The long-tail phenomenon in such knowledge graphs is also very important. In some relations, there are only a small number of samples, which increases the difficulty of knowledge graph reasoning. In order to better complete the training of the model, for a specific knowledge graph, we first divide the dataset according to its size and the degree of long-tail problems, and then sample the reference set and query set from the selected experimental dataset. The construction of the background graph and the training of the pre-trained temporal knowledge graph embedding are completed before model training. After that, we will complete the training of the model according to the process shown in Algorithm 1. Firstly, the input part includes the meta-learning task set  $\mathcal{T}$  of the relation part, the background  $TKG G'$ , the pre-training embedding of a few temporal knowledge graphs and three original model parameters. When the training task is not completed, the relation in the meta-learning task set is shuffled first, then the entity pairs with small samples are selected as the reference set, and then a new time-based quadruple is created by using the existing quadruple (lines 01–04). For  $L$  in each training task, the model first selects a few-shot entity pair as a reference set, and extracts a set of query sets, and then generates a set of negative entity pairs for experiments by polluting object entities. Then, according to the proposed formulas (lines 05–13), the feature vector representation of the temporal neighborhood of the subject entity is calculated in turn. The reconstruction loss of the optimized automatic encoder, the challenge of embedding and formulation, the sorting loss, the loss of calculating the fault-tolerant relation, and finally the loss function of the whole model are calculated. After that, the model needs to update the optimizer parameters according to the calculation results until all tasks are completed (lines 14–15). Finally, the model needs to return an optimal set of model

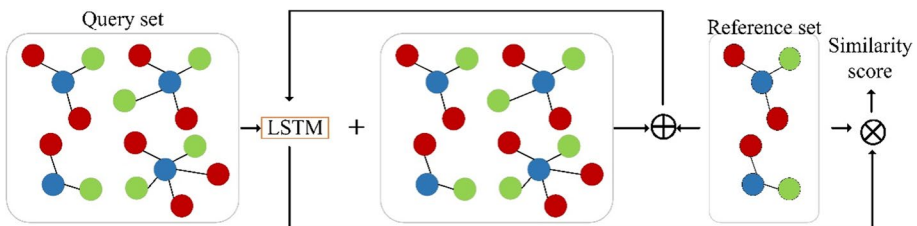


Figure 7 The matching network for query pair and reference set

parameters (line 18) based on the descent of the gradient as the model calculates. The new parameter set will be used as an optimal parameter for training new tasks.

### Algorithm 1 FTMF Meta-Training

---

**Input:** Meta-training task(relation) set  $\mathcal{T}$ ,  
 Background  $TKG G'$ ,  
 Pre-trained  $TKG$  embeddings,  
 Initial model parameters  $\epsilon, \theta$  and  $\mu$

01 **while** not done do  
 02   Shuffle tasks(relations) in  $\mathcal{T}$   
 03   Sample few-shot entity pairs as reference set  
 04   Contaminate the object entity of  $(s_i, r, o_i, t_i)$  to create  $(s_i, l_i, o_i, t_i)$   
 05   **for**  $\mathcal{T}_i \in \mathcal{T}$  do  
 06     Sample few-shot entity pairs as reference set  
 07     Sample a batch of query entity pairs  $(s_i, o_i, t_i)$   
 08     Pollute the object entity of  $(s_i, o_i, t_i)$  to  $get(s_i, o_i^-, t_i)$   
 09     Calculate the feature representation of  $s_i$  by Equation (16)  
 10     Calculate the reconstruction loss for optimizing autoencoder  $\mathcal{L}_{re}(R_r)$  by Equation (21)  
 11     Calculate the challenge and formulate the embedding  $f_\epsilon(R_r)$  by Equation (24)  
 12     Calculate the loss of query instances in query set  $\mathcal{L}_r$  by Equation (35)  
 13     Calculate the ranking loss  $\mathcal{L}_{rank}$  by Equation (38)  
 14     Accumulate the loss function  $\mathcal{L}_{joint}$  by the Equation (39)  
 15     Update parameters by Adam (adagrade) optimizer  
 16   **end**  
 17 **end**  
 18 **return** Optimal model parameters  $\epsilon^*, \theta^*$  and  $\mu^*$

---

## 5 Experiment

### 5.1 Experimental setup

#### 5.1.1 Datasets pre-processing

In the experiments, we use two public datasets. One is based on ICEWS [3] and the other is based on GDELTA [17]. Inspired by the thought of Gmatching model, we process ICEWS and GDELTA to meet the few-shot criteria. In addition, we follow the dataset partition setting method proposed by Xiong et al. [41], in which the relations are selected with less than 500 but more than 50 triples as the few-shot task. We keep the number of entities per relation between 50—500 by extracting the relation between the number of conforming standards in the whole dataset. Then we control the number of relations under 100. For each set of entities of the relation, we divide the number of entities in the training set, test set and verification set into a ratio of 70: 15: 7. Table 1 lists the statics of the two datasets.

#### 5.1.2 Baseline methods

In the structure of our model, the vector representation of entities and temporal neighbor encoder are involved. Some models in related work have similar structures. Therefore, in the

selection of baseline models, we choose the models with better performances on the target dataset and the latest model as the experimental comparison models. In this subsection, we mainly introduce two kinds of baseline models for comparisons.

**Vector representation and relational embedding model.** This kind of model is mainly to embed the entity or the relation through modeling the relation structure. We adopt the following models for comparative experiments: TransE [2], DistMult [43], TTransE [16], TA-TransE [5] and TA-DistMult [5]. The parameter settings of all experimental datasets are exactly the same as the pre-processed few-shot datasets we used.

**Neighborhood coding model.** This kind of model combines graph local neighborhood encoder and matching network to learn entity embedding and predict new fact relations. We adopt the following models for comparative experiments: RE-Net [13], GMatching [41], MateR [37] and FSRL [44]. The parameter settings of all experimental datasets are exactly the same as the pre-processed few-shot datasets we used.

### 5.1.3 Experimental parameter settings

In order to further improve the performance of the model, we carry out a pre-training process for the data before formal training. Considering all kinds of factors, we choose Complex as the pre-training input. For our model, we make some parameter optimization and the main parameter settings are as follows: (i) The embedding dimension  $n$  of the two datasets is uniformly set to 100. (ii) LSTM is used as the reference aggregator and matching processor. The hidden dimension  $h$  of LSTM is consistently set to 200. (iii) For two datasets, the maximum local neighborhood number of the heterogeneous neighborhood encoder species  $q$  is 30. (iv) In the process of updating model parameters, we choose Adam optimizer. (v) For both datasets, we set the number of steps  $p$  in the matching cycle in the network to 2. (vi) The initial learning rate  $\lambda$  is 0.001, and the weight attenuation  $a$  is 0.25. (vii) The edge distance  $m$  in the objective function is set to 5.0 and the transaction factor  $f$  is set to 0.0001. (viii) In the construction of entity candidate sets, we set the maximum size  $x$  of the two datasets to 1000.

For other models, the original optimal parameters may lose their optimal performance due to the change of datasets. Therefore, we reproduce all other models to determine the optimal parameters when they achieve the optimal performance, and the results obtained are all optimal results. For models Gmatching, MateR, and FSRL, the optimal parameter settings are the same as our model. The specific optimal parameter list of each model is shown in Table 2.

For the other baseline models used in the experiments, the specific optimal parameters are shown in Table 3. In Table 3,  $\lambda$  represents the learning rate, and its candidate set is  $\{0.01, 0.001, 0.0001\}$ .  $n$  represents the latitude of vector embedding, and its candidate set is  $\{128, 256, 512\}$ .  $B$  represents the batch size of training data, and its candidate set is  $\{256, 512, 1024\}$ .  $v$  represents the discard probability, and its candidate set is  $\{0.1, 0.3, 0.5\}$ . In addition, we retain the original parameter settings for each models' specific parameters.

### 5.1.4 Experimental evaluation index

In order to evaluate the performances of our model and compare with other models, we use some specific indicators to evaluate the results. We use the relations and entities in the training

**Table 1** Statistics of ICEWS-Few and GDELT-Few

Dataset	#Ents	#Quads	#Relations	#Tasks	#Times
ICEWS-Few	7531	391,936	253	102	632
GDELT-Few	4570	270,858	225	90	485

data, so that the model has the ability of self-learning. On this basis, we use the verification set and the test set to evaluate the model, so as to optimize the performance of the model. We use the hit ratio (Hits@) and the mean reciprocal rank (MRR) to compare the performances. In the selection of hit ratio, we chose the following three hit ratio: Hits@1, Hits@5, and Hits@10.

## 5.2 Experimental comparisons

### 5.2.1 Experimental comparison with baselines

Verification and test performance comparisons on ICEWS-Few and GDELT-Few are presented in Table 4. In all experimental results, the pre/post scores represent experimental data from the validation/test set, respectively. The best results of all the experiments are shown in bold, and the best results of the comparative experiments are underlined.

As shown in Table 4, for a clearer comparison of the experimental results, they are presented in Figure 8 as well. The figure on the left shows the test results of each model on ICEWS-Few and the figure on the right shows the test results of each model on GDELT-Few. The performances of different models correspond to the data parts of different colors. From Figure 8, we can clearly compare the performance of different methods under the same data set. We can draw the following conclusions:

- i). The completion performances of the models using neighbor coding are higher obviously. It proves that using neighbor coding can solve the disadvantage of insufficient entity embedding representation. The dataset used in the experiment is temporal knowledge graph dataset. The experimental results show that the performance of temporal knowledge graph completion method is better than that of static knowledge graph completion method, so the temporal information in temporal knowledge graph completion task is very important. Moreover, we can better represent entities by processing time series, and finally improve the embedding ability of entities by improving the representation form of entities, thus improving the performance of the model, which shows that neighbor coding is more suitable for entity embedding.
- ii). Among all the results, FTMF has better performances, which directly shows that the combination of time series encoder, cyclic recursive aggregation network, fault-tolerant mechanism and similarity network can enhance the representation ability of entities to a greater extent, and at the same time reduce the adverse effects caused by error information in temporal knowledge graph with few samples. It can further improve the completion performance of the model.

**Table 2** The optimal hyper-parameters for baseline models on both datasets

Model	$n$	$\lambda$	$x$	$h$	$q$	$p$	$a$	$m$	$f$
Gmatching [41]	100	0.001	1000	200	30	2	0.25	5.0	0.0001
MateR [37]	100	0.001	1000	200	30	2	0.25	5.0	0.0001
FSRL [44]	100	0.001	1000	200	30	2	0.25	5.0	0.0001
FTMF	100	0.001	1000	200	30	2	0.25	5.0	0.0001

### 5.2.2 Comparison over different relations

In order to better verify the validity and stability of our model, we set up comparative experiments with different relations, where relationId represents a class of relations in a dataset. In this group of experiments, we not only validate the overall performances of all relations, but also evaluate the performances of each relation in the test dataset. The comparative models are FTMF and FSRL. The datasets used in the experiments are ICEWS-Few and GDELT-Few, and the experimental evaluation indexes used in the experiment are the same as before. The experimental results are listed in Table 5 and Table 6. The pre/post experimental scores represent the scores of FTMF and FSRL respectively.

It can be observed from Table 5 and Table 6 that the value of variance is large. It can be explained by the fact that the size of candidate sets corresponding to different relations is also different. The experimental results show that the performance of our model is much better than that of FSRL on some specific relations. It can be explained by the fact that temporal information is very important for completion task on some task relations. In our model, the combination of time series encoder and cyclic recursive aggregation network can effectively utilize temporal information, solve the disadvantage of insufficient entity embedding representation, and improve the model performances. In addition, we can see that the less relations, the higher the scores of each index. The scores of FTMF are higher than FSRL in most cases. At the same time, it can be observed that FTMF is more stable, has higher fault tolerance, and is more competent for the temporal knowledge graph completion.

### 5.3 Ablation study

Our model is a joint learning framework composed of multiple neural network modules, so the existence of each module has a certain impact on the performance of the model. Therefore, we perform ablation experiments to evaluate the influence of the four different modules. The symbolic representations of ablation experiments are presented in Table 7. The datasets used in the experiment are ICEWS-Few and GDELT-Few, and the experimental evaluation indexes used in the experiment are the same as before. Table 8 and Table 9 report the results of ablation experiments on ICEWS-Few and GDELT-Few. The meaning of the "Bold" entries is to mark the best result. In all experimental results, the pre/post scores represent experimental data from the validation/test set, respectively.

**Without time series neighbor encoder (W1)** This group of experiments are conducted to verify the effect of time series neighbor encoder. We replace it with an embedded average pool

**Table 3** The optimal hyper-parameters for baseline models on each dataset

Model	ICEWS-Few				GDELT-Few			
	$\lambda$	$n$	$B$	$v$	$\lambda$	$n$	$B$	$v$
TransE [2]	0.0001	512	512	-	0.0001	512	512	-
DistMult [43]	0.01	512	1024	-	0.01	512	1024	-
TTransE [16]	0.001	512	512	0.0	0.001	256	512	0.0
TATransE [5]	0.001	512	512	0.0	0.001	256	512	0.0
TA-DistMult [5]	0.001	512	1024	0.0	0.001	512	1024	0.0
RE-Net [13]	0.001	256	1024	0.5	0.001	256	1024	0.5

**Table 4** Verification and test performance comparisons on ICEWS-Few and GDELT-Few

Model	Dataset: ICEWS-Few				Dataset: GDELT-Few			
	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10	MRR
TransE [2]	0.064/0.145	0.127/0.264	0.208/0.321	0.115/0.188	0.164/0.081	0.241/0.096	0.306/0.117	0.281/0.129
DistMult [43]	0.071/0.158	0.145/0.291	0.202/0.329	0.126/0.213	0.186/0.076	0.265/0.101	0.324/0.133	0.334/0.125
TTransE [16]	0.126/0.196	0.281/0.345	0.315/0.380	0.256/0.217	0.215/0.088	0.396/0.135	0.429/0.207	0.350/0.153
TA-TransE [5]	0.117/0.190	0.284/0.351	0.333/0.378	0.255/0.222	0.214/0.093	0.385/0.146	0.428/0.213	0.349/0.152
TA-DistMult [5]	0.131/0.198	0.294/0.337	0.356/0.379	0.273/0.246	0.223/0.114	0.397/0.140	0.437/0.235	0.351/0.167
RE-Net [13]	0.178/0.177	0.337/0.361	0.487/0.471	0.318/0.330	0.276/0.156	0.406/0.241	0.461/0.342	0.366/0.229
GMatching [41]	0.256/0.204	0.436/0.398	0.493/0.483	0.347/0.298	0.267/0.147	0.400/0.253	0.441/0.341	0.353/0.230
MateR [37]	0.250/0.208	0.425/0.365	0.510/0.509	0.386/0.280	0.265/0.143	0.415/0.270	0.432/0.361	0.352/0.221
FSRL [44]	0.281/0.202	0.443/0.377	0.508/0.514	0.377/0.298	0.279/0.145	0.448/0.264	0.459/0.355	0.367/0.225
FTMF	<b>0.343/0.244</b>	<b>0.551/0.439</b>	<b>0.594/0.577</b>	<b>0.451/0.355</b>	<b>0.366/0.228</b>	<b>0.557/0.398</b>	<b>0.608/0.467</b>	<b>0.432/0.273</b>

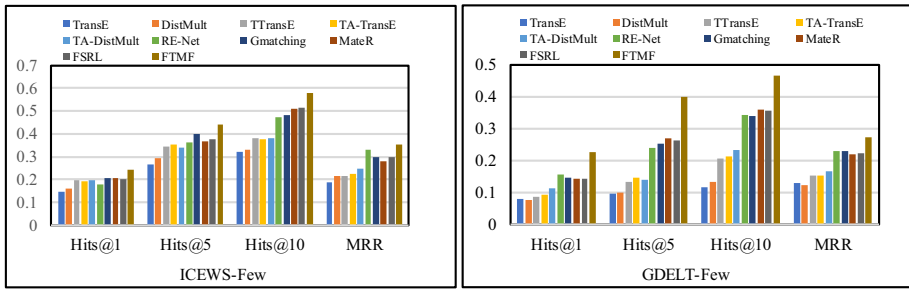


Figure 8 Test performance comparisons on ICEWS-Few and GDELT-Few

layer covering all neighbors. It can be seen from Table 8 and Table 9 that the performance of the model decreases when the relational aware heterogeneous neighbor encoder is lost.

**Without cyclic autoencoder (W2)** This group of experiments are conducted to verify the effect of cyclic automatic encoder aggregator network. We replace the cyclic automatic encoder aggregator network with an average pool operation. According to the experimental results in Table 8 and Table 9, it can be seen that our model has better performances.

**Without Fault-tolerant mechanism (W3)** This group of experiments are conducted to verify the influence of fault tolerance mechanism. We removed the fault tolerance mechanism, which means that all information, whether correct or not, will participate in the calculation. It can be observed from Table 8 and Table 9 that our model has better performances.

**Without matching network (W4)** This group of experiments are conducted to verify the effect of matching network on model performance. We cancelled LSTM and use the inner product between query embedding and reference embedding as the similarity score. We can observe that our model has better performance, which indicates that the circular matching network has a good performance in calculating the correlation between queries and references.

### 5.4 Stability experiments

In this subsection, we study the influence of size  $K$ . The few-shot size represents the size of  $K$  and  $K$  represents the size of reference set. We perform experiments on FTMF, FSRL, and GMatching model, and set different  $K$  values for these three models. The datasets used in the experiments are ICEWS-Few and GDELT-Few, and the experimental evaluation indexes used in the experiment are the same as before. Experimental results are shown in Figure 9 and Figure 10.

It can be observed from Figure 9 and Figure 10 that FTMT, of FSRL, and GMatching model have good completion performances with the increase of reference set size. It can be explained by the fact that the number of selectable entities is increasing when the reference set becomes larger. The loss function will be more accurate when recursive processing of the reference set, which is conducive to improving the score ranking of entities in the candidate set. At the same time, the performance of FTMF is always better than the other two models. This also shows that our proposed model has good ability in completing the few-shot temporal knowledge graph.



**Table 5** The results of FTMF and FSRL for each relation on ICEWS-Few

RelationId	Hits@1	Hits@5	Hits@10	MRR
1	0.981/0.899	0.993/0.976	1.000/1.000	0.987/0.983
2	0.168/0.016	0.326/0.029	0.535/0.083	0.324/0.034
3	0.973/0.288	0.986/0.418	0.991/0.431	0.985/0.344
4	0.495/0.055	0.478/0.109	0.421/0.147	0.459/0.120
5	0.118/0.073	0.276/0.156	0.421/0.176	0.306/0.135
6	0.408/0.203	0.579/0.528	0.708/0.587	0.596/0.345
7	0.555/0.530	0.786/0.698	0.897/0.881	0.772/0.598
8	0.305/0.142	0.708/0.558	0.774/0.678	0.532/0.314
9	0.653/0.645	0.772/0.609	0.895/0.728	0.763/0.587
10	0.125/0.054	0.378/0.135	0.527/0.208	0.342/0.095

**Table 6** The results of FTMF and FSRL for each relation on GDELT-Few

RelationId	Hits@1	Hits@5	Hits@10	MRR
1	0.991/0.923	0.996/0.949	1.000/1.000	0.995/0.989
2	0.178/0.035	0.399/0.098	0.605/0.165	0.387/0.045
3	0.990/0.321	0.994/0.459	0.998/0.512	0.994/0.378
4	0.065/0.067	0.693/0.186	0.752/0.209	0.682/0.192
5	0.159/0.088	0.349/0.198	0.478/0.223	0.336/0.172
6	0.408/0.196	0.562/0.488	0.641/0.539	0.576/0.333
7	0.603/0.601	0.786/0.706	0.906/0.901	0.793/0.700
8	0.251/0.157	0.772/0.598	0.806/0.666	0.568/0.420
9	0.678/0.536	0.792/0.645	0.917/0.780	0.784/0.653
10	0.175/0.036	0.293/0.110	0.468/0.214	0.287/0.106

**Table 7** Symbolic representations

Symbol	Explanation	Substitute
W1	Without time series neighbor encoder	Embedded average pool layer
W2	Without cyclic autoencoder	Average pool operation
W3	Without Fault-tolerant mechanism	Value of $be$ is 1
W4	Without matching network	Inner product between query and reference embedding

## 5.5 Defects analysis

We combine time series neighbor encoder, cyclic recursive automatic aggregation network, fault-tolerant mechanism, and similarity network to complete the task of few-shot temporal knowledge graph. Although the proposed model has achieved good performance, there are still some limitations:

- i). Limitations of datasets: the dataset we use is a small sample dataset that has been processed. Therefore, when applied to other datasets, the datasets should be processed accordingly to form the sample datasets.

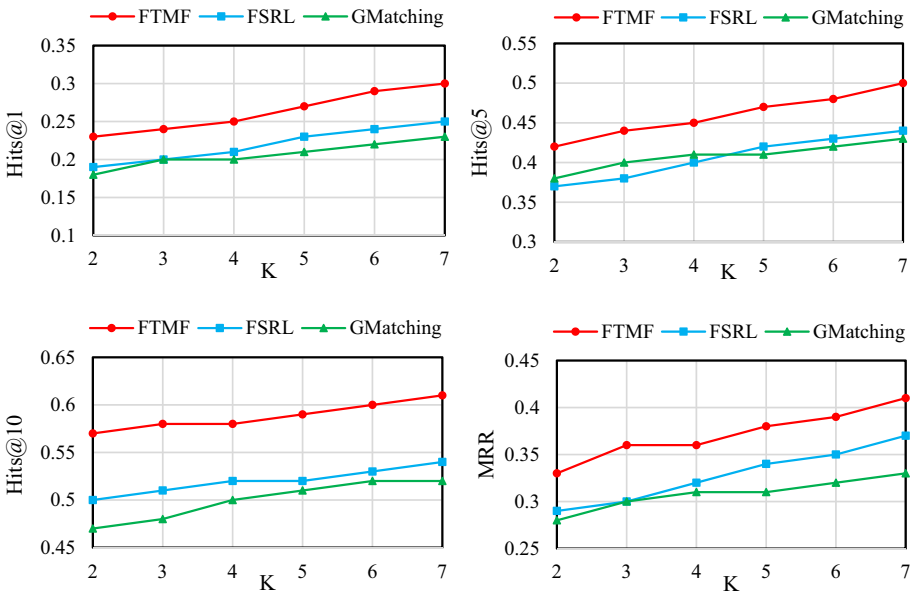
**Table 8** The results of ablation experiment on ICEWS-Few

Experiment	Hits@1	Hits@5	Hits@10	MRR
W1	0.121/0.177	0.340/0.366	0.467/0.487	0.266/0.283
W2	0.297/0.200	0.493/0.417	0.531/0.533	0.398/0.324
W3	0.303/0.213	0.512/0.420	0.547/0.545	0.425/0.329
W4	0.301/0.210	0.499/0.411	0.527/0.540	0.401/0.325
FTMF	<b>0.343/0.244</b>	<b>0.551/0.439</b>	<b>0.594/0.577</b>	<b>0.451/0.355</b>

**Table 9** The results of ablation experiment on GDELT-Few

Experiment	Hits@1	Hits@5	Hits@10	MRR
W1	0.167/0.142	0.412/0.258	0.501/0.376	0.326/0.255
W2	0.300/0.156	0.467/0.279	0.521/0.390	0.383/0.261
W3	0.311/0.191	0.499/0.320	0.545/0.425	0.399/0.266
W4	0.311/0.185	0.428/0.305	0.536/0.399	0.395/0.262
FTMF	<b>0.366/0.228</b>	<b>0.557/0.398</b>	<b>0.608/0.467</b>	<b>0.432/0.273</b>

ii). Limitation of model: we propose a FTMF model for neighbor encoders of temporal sequences which requires a unified relation with connected entities at the same time. If the number of connected entities is small, it may affect the quality of entity embedding. In addition, the goal of fault-tolerant mechanism is to reduce the impact of error information on entity interaction, so the less error information in dataset, the contribution of fault-tolerant mechanism will be reduced.



**Figure 9** Impact of few-shot size  $K$  on ICEWS-Few

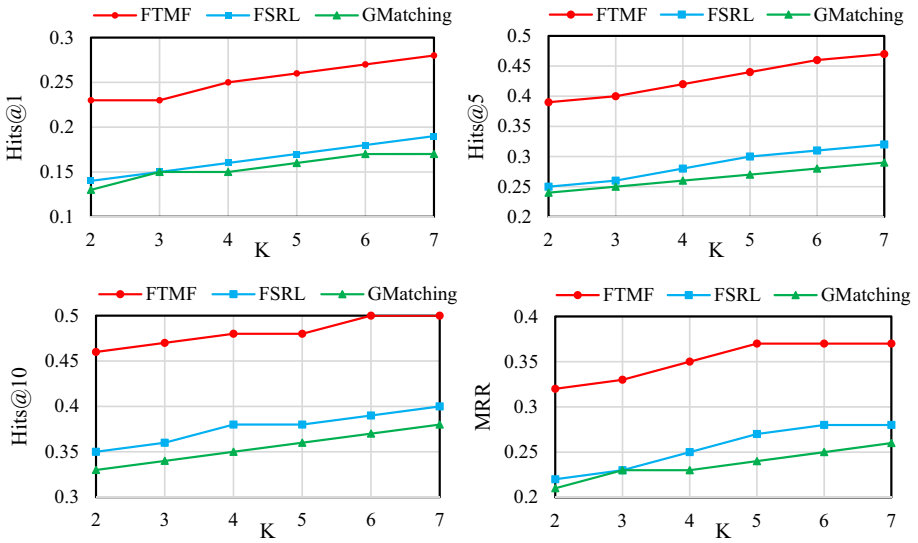


Figure 10 Impact of few-shot size  $K$  on GDELT-Few

## 6 Conclusion

In this paper, we propose a new temporal knowledge graph completion model for the task of short-sample temporal knowledge graph completion. Our model combines time series neighbor encoder to generate the feature representation vector of an entity in time neighborhood. The interaction between reference set instances is modeled by time-based cyclic automatic encoder. Fault-tolerant mechanism is used to reduce the impact of error information in datasets. Finally, we use similarity network to calculate the similarity score between query set and reference set. The experimental results show that our model has achieved remarkable results in completion ability, with the performance reaching 17% on ICEWS-Few dataset and 46% on GDELT-Few dataset respectively. In addition, the experimental results on different relations show that our model has a better stability. The ablation experiments of four modules are also carried out, and each module is indispensable. Finally, we perform the experiments of reference set size. The results show that with the increase of reference set, the performance of the model is also improving, and the performance of our model is always the best one.

Our model mainly focuses on few-shot temporal knowledge graph completion tasks, and there are still some limitations as described in defects analysis Section. In the future work, we plan to extend it to incorporate more contextual information like textual description to improve reasoning performance.

**Author contribution** Luyi Bai: Conceptualization, Methodology, Formal analysis, Funding acquisition, Writing—original draft, Writing—review & editing; Mingcheng Zhang: Investigation, Validation, Formal analysis, Writing—original draft; Han Zhang: Validation, Formal analysis, Writing—original draft; Heng Zhang: Writing—review & editing.

**Funding** The work was supported by the National Natural Science Foundation of China (61402087), the Natural Science Foundation of Hebei Province (F2022501015), the Key Project of Scientific Research Funds in Colleges and Universities of Hebei Education Department (ZD2020402), and in part by the Program for 333 Talents in Hebei Province (A202001066).

**Data availability** Not applicable.

## Declarations

**Ethical approval and consent to participate** Not applicable.

**Human and animal ethics** Not applicable.

**Consent for publication** Not applicable.

**Competing interests** The authors declare no potential conflicts of interest with respect to the research, authorship, and publication of this article.

## References

- Bollacker K.D., Evans, C., Paritosh, P.K., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of Special Interest Group on Management of Data, pp. 1247–1250 (2008)
- Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of Neural Information Processing Systems, pp. 2787–2795 (2013)
- Boschae, E., Lautenschläger, J., O’Brien, S., Shellman, S.M., Starz, J., Ward, M.D.: ICEWS coded event data. *Harvard Dataverse* **12** (2015). <https://doi.org/10.7910/DVN/28075>
- Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: Proceedings of Empirical Methods in Natural Language Processing, pp. 670–680 (2017)
- García-Durán, A., Dumančić, S., Niepert, M.: Learning sequence encoders for temporal knowledge graph completion. In: Proceedings of Empirical Methods in Natural Language Processing, pp. 4816–4821 (2018)
- Hamilton, W.L., Ying, Z.H., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035 (2021)
- Hasanzadeh, A., Hajiramezanali, E., Narayanan, K., et al.: Variational Graph Recurrent Neural Networks. In: Proceedings of the Advances in Neural Information Processing Systems, pp. 10700–10710 (2019)
- He, K., Zhang, X.Y., Ren, S.Q., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
- Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
- Ji, G.L., He, S.Z., Xu, L.L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 687–696 (2015)
- Jiang, T.S., Liu, T.Y., Ge, T., Sha, L., Li, S.J., Chang, B.B., Sui, Z.F.: Encoding temporal information for time-aware link prediction. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 2350–2354 (2016)
- Jiang, Z., Gao, J., Lv, X.: MetaP: Meta Pattern Learning for One-Shot Knowledge Graph Completion. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2232–2236 (2021)
- Jin, W., Zhang, C.L., Szekely, P., Ren, X.: Recurrent event network for reasoning over temporal knowledge graphs. arXiv preprint arXiv:1904.05530 (2019)
- Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations, pp. 1–14 (2017)

15. Kumar, S., Zhang, X., Leskovec, J.: Learning dynamic embeddings from temporal interactions. arXiv preprint arXiv.1812.02289. (2018)
16. Leblay, J., Chekol, M.W.: Deriving validity time in knowledge graph. In: Proceedings of the The Web Conference, pp. 1771–1776 (2018)
17. Leetaru, K., Schrodt, P.A.: GDEL: global data on events, location, and tone. In: Proceedings of ISA Annual Convention, pp. 1–49 (2013)
18. Lin, Y.K., Liu, Z.Y., Sun, M.S., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of Twenty-ninth AAAI Conference on Artificial Intelligence, pp. 2181–2187 (2015)
19. Liu, H.X., Wu, Y.X., Yang, Y.M.: Analogical inference for multi-relational embeddings. In: Proceedings of International Conference on Machine Learning, pp. 2168–2178 (2017)
20. Ma, Y., Tresp, V., Daxberger, E.A.: Embedding models for episodic knowledge graphs. *J. Web Semant.* **59**, 1–26 (2019)
21. Manessi, F., Rozza, A., Manzo, M.: Dynamic graph convolutional networks. *Pattern Recogn.* **97**(107000), 1–16 (2020)
22. Miller, A.: WordNet: A lexical database for English. *Commun. ACM* **38**(11), 39–41 (1995)
23. Mirtaheri, M., Rostami, M., Ren, X., et al.: One-shot learning for temporal knowledge graphs. arXiv preprint arXiv.2010.12144 (2020)
24. Nguyen, G.H., Lee, J.B., Rossi, R.A., Ahmed, N.K., Koh, E., Kim, S.: Continuous-Time Dynamic Network Embeddings. In: Proceedings of the The Web Conference, pp. 969–976 (2018)
25. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proceedings of 30th AAAI Conference on Artificial Intelligence, pp.1955–1961 (2016)
26. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proceedings of International Conference on Machine Learning, pp. 809–816 (2011)
27. Niu, G., Li, Y., Tang, C., et al.: Relational Learning with Gated and Attentive Neighbor Aggregator for Few-Shot Knowledge Graph Completion. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 213–222 (2021)
28. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: Learning representations over dynamic graphs. In: Proceedings of the 7th International Conference on Learning Representations, pp. 1–25 (2019)
29. Sadeghian, A., Rodriguez, M., Wang, D.Z., Colas, A.: Temporal reasoning over event knowledge graphs. In: Proceedings of Workshop on Knowledge Base Construction, Reasoning and Mining, pp. 6669–6683 (2016)
30. Sheng, J., Guo, S., Chen, Z., Yue, J.W., Wang, L.H., Liu, T.W., Xu, H.B.: Adaptive Attentional Network for Few-Shot Knowledge Graph Completion. In: Proceedings of Empirical Methods in Natural Language Processing, pp. 1681–1691 (2020)
31. Snell, J., Swersky, K., Zemel, R.S.: Prototypical Networks for Few-shot Learning. In: Proceedings of Neural Information Processing Systems, pp. 4077–4087 (2017)
32. Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: Proceedings of Advances in Neural Information Processing Systems, pp. 926–934 (2013)
33. Trivedi, R., Dai, H., Wang, Y., Song, L.: Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In: Proceedings of International Conference on Machine Learning. PMLR, pp. 3462–3471 (2017)
34. Trouillon, T., Welbl, J., Riedel, S., et al.: Complex embeddings for simple link prediction. In: Proceedings of the 33rd International Conference on Machine Learning. PMLR, pp. 2071–2080 (2016)
35. Wang, S., Huang, X., Chen, C., et al.: REFORM: Error-Aware Few-Shot Knowledge Graph Completion. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1979–1988 (2021)
36. Wang, X., Girshick, R., Gupta, A., et al.: Non-local neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7794–7803 (2018)
37. Wang, Z., Zhang, J.W., Feng, J.L., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 1112–1119 (2014)
38. Xiao, H., Huang, M., Hao, Y., et al.: Transg: A generative mixture model for knowledge graph embedding. The Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 2316–2325 (2016)
39. Xiong, C.M., Merity, S., Socher, R.: Dynamic Memory Networks for Visual and Textual Question Answering. In: Proceedings of International Conference on Machine Learning. PMLR, 2016, pp. 2397–2406 (2016)
40. Xiong, C., Zhong, V., Socher, R.: Dynamic coattention networks for question answering. In: Proceedings of the 5th International Conference on Learning Representations, pp. 1–14 (2017)

41. Xiong, W., Yu, M., Chang, S., Guo, X.X., Wang, W.Y.: One-shot relational learning for knowledge graphs. In: Proceedings of Empirical Methods in Natural Language Processing, pp. 1980–1990 (2018)
42. Xu, J., Zhang, J., Ke, X., et al.: P-INT: A Path-based Interaction Model for Few-shot Knowledge Graph Completion. In: Proceedings of the Association for Computational Linguistics. EMNLP 2021, pp. 385–394 (2021)
43. Yang, B., Yih, W., He, X.D., Gao, J.F., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of International Conference on Learning Representations, pp. Poster (2015)
44. Zhang, C.X., Yao, H.X., Huang, C., Jing, M., Li, Z.H., Chawla, N.V.: Few-shot knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3041–3048 (2020)

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.