Complex & Intelligent Systems (2023) 9:7461–7474
https://doi.org/10.1007/s40747-023-01146-9

ORIGINAL ARTICLE

Few-shot temporal knowledge graph completion based on meta-optimization

Lin Zhu[1] · Luyi Bai[1,2] · Shuo Han[1] · Mingcheng Zhang[1]

Received: 18 October 2022 / Accepted: 11 June 2023 / Published online: 4 July 2023

Abstract

Knowledge Graphs (KGs) have become an increasingly important part of artificial intelligence, and KGs have been widely used in artificial intelligence fields such as intelligent answering questions and personalized recommendation. Previous knowledge graph completion methods require a large number of samples for each relation. But in fact, in KGs, many relationships are long-tail relationships, and the existing researches on few-shot completion mainly focus on static knowledge graphs. In this paper, we consider few-shot completion in Temporal Knowledge Graphs (TKGs) where the event may only hold for a specific timestamp, and propose a model abbreviated as FTMO based on meta-optimization. In this model, we combine the time-based relational-aware heterogeneous neighbor encoder, the cyclic automatic aggregation network, and the matching network to complete the few-shot temporal knowledge graph. We compare our model with the baseline models, and the experimental results demostrate the performance advantages of our model.

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

Introduction

Knowledge Graph (KG) is a new concept proposed by Google, which is used to construct multivariate relational data. In recent years, KGs have been widely used in the artificial intelligence field, such as intelligent answers [28, 43] and social network analysis [45]. However, because of the incompleteness of KGs, the performances of tasks related to knowledge graphs will be affected. Like many KGs, such as WordNet [26], Freebase [1], and Google Knowledge Graph [2], these KGs are static and have no temporal information. In previous studies, researchers have proposed many models, such as TransE [3] and its improved models, to complete KGs. Most of these models embed entities and relationships into low-dimensional space and achieve good performance.

In the past few years, knowledge data usually contain abundant temporal information. In this case, many researchers add timestamps to the traditional knowledge graph triples (s, r, o) as temporal knowledge graph (TKG), which is described as quadruples (s, r, o, t). ICEWS [4] and GDELT [21] are two famous temporal knowledge graphs but they are far from complete. The important task of a TKG is to complete quadruples without correct subject–object entity or relational information. For example, the form of an incomplete quadruple may be $(?, r, o, t)$, $(s, r, ?, t)$ or $(s, ?, o, t)$, and we need to infer "$?$" from the quad we already have. Although KG completion model has achieved remarkable results, it has little effect on TKGs completion model. In this case, the completion of TKGs still has a lot of research space, and it also faces great difficulties. With the development of research, researchers add temporal information processing to the model, such as TTransE [19] and TA-TransE [9]. In addition, some models based on recurrent neural networks have also emerged such as RE-Net [16].

In the previously proposed method, the researchers assumed that each relationship has a sufficient number of entities to train for better performance. However, in fact, in TKGs, a large number of relationships have few entity pairs, which is called the long-tail relationship. For example, for a "*s the citizen of*" relationship, there may be thousands of entities, but for a "*is the president of*" relationship, there

✉ Luyi Bai
baily@neuq.edu.cn

1 School of Computer and Communication Engineering, Northeastern University (Qinhuangdao), Qinhuangdao 066004, China

2 Informatics Department, University of Leicester, Leicester LE1 7RH, UK

may be only a few hundred entities. The number of entities corresponding to the two relationships varies greatly, so we should study this situation. The long-tail relationship cannot be ignored in the real world. In this case, several models with few shots, such as GMatching [44], MetaR [5] and FANN [34], were proposed successively. These models are developed for static knowledge graph with few samples, and cannot explain temporal knowledge graph. The encoders they use cannot embed the temporal relation between entities into the models, and the information sharing between fewer entities and the influence of heterogeneous neighborhoods is not considered. After adding temporal information, we propose a relational-aware heterogeneous neighborhood encoder based on temporal information inspired by FSRL [49]. In addition, the one-time learning environment cannot meet the training situation with few samples, so it is necessary to design a new model to realize the interaction between the reference set and temporal information. In addition, we find that the meta-optimizer can be combined with LSTM [12], and LSTM can solve the problem of gradient descent and gradient explosion in training. In addition, combining LSTM update and gradient descent can obtain the optimal parameters of the model better and faster, update the training model through a small number of gradients, realize fast learning of new tasks, and train another neural network classifier through an optimization algorithm in a small number of states.

In this paper, we combine several modules and propose a new model to complete the short shot TKG. The paper makes the following contributions:

- We propose a few-shot completion model, which addresses few-shot completion in temporal knowledge graphs.
- We use timestamp information to enhance the representation of task entities and entity pairs by constructing a time-based relationship-aware heterogeneous neighbor encoder.
- We propose a cyclic automatic encoder aggregation network for TKG.
- We conduct abundant experiments on two public datasets to demonstrate that FTMO outperforms existing state-of-the-art TKG embedding methods and few-shot completion methods.

The rest of the paper is organized as follows. In "Related work", we describe related work. In "Our model", we illustrate the relevant task definitions and the details of the proposed model. Experimental setups and comparative analysis of the experimental results are presented in "Experiments". In "Conclusion", we give a conclusion and possible directions for future improvement.

Related work

Static knowledge graph completion methods

There are two types of static knowledge graph completion models: translation models and others.

On one hand, the translation model transforms relationships and entities into vectors, and calculates the dissimilarity of vectors. Bordes et al. propose the famous TransE [3] model, which interprets the relation vector as the semantic translational operation of the entity vector in the vector space. If $s + r \approx o$ is true, the completion result is correct. However, it only focuses on 1–1 relationship, and is not a good fit for 1–N, N–1 and N–N relationships. To this end, several improved models are taken into account such as TransH [42], TransR [23], and TransD [14], etc. TransH 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 uses the mapping matrix corresponding to the relation to transform entities into different relation semantic spaces to obtain different semantic representations. TransD uses entity-related vectors and relation-related vectors to dynamically obtain the projection matrix of the relation through cross-product computation. On the other hand, one of the most important other models is semantic model, which mainly calculates a similarity score through the latent semantics between entity vector and relation vector, and ranks the completion results according to the calculated similarity score. DistMult [46] proposes a framework, considering entities and relations as low-dimensional vectors and bilinear and/or linear mapping functions. ANALOGY [24] proposes a framework for optimizing the latent representations in the case of the analogical properties of the embedded entities and relations. RESCAL [30] 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. To have better representations, NTN [36] and HolE [29] are proposed to obtain a better vector representations for improvements. On the other hand, MMKRL [25] can utilize multi-modal knowledge effectively to achieve better link prediction and triple classification by summing different plausibility functions and using specific norm constraints. Wang et al. [41] propose the modeling of complex internal logic by integrating the fusion semantic information, which can make the model converge faster. Huang et al. [13] propose local information fusion to join entities and their adjacencies to obtain multi-relational representations. However, these models cannot be applied to temporal knowledge graph directly.

Temporal knowledge graph completion methods

With the growth of the amount of data information, temporal information has been widely considered. In recent years, temporal stamps have been embedded into low-dimensional space, and three elements (s, r, o) have been extended to four elements (s, r, o, t) to complete TKG. Inspired by TransH [42], Dasgupta et al. propose HyTE [15] model which explicitly combines temporal information in the entity-relationship space by associating each temporal stamp with the corresponding hyperplane. TTransE [19] upgrades TransE by improving the scoring function and learns the temporal representations from the temporal text by a recurrent neural network, so TTransE can complete the temporal knowledge graph and obtain great achievements. García-Durán et al. propose TA-TransE [9] and TA-DistMult [9] adding temporal embeddings into its score function to use temporal information to complete the graph. However, in these models, all static temporal information ignores the relevance of the related quad. In addition, the time dependency also needs to be considered. To make good use of the time dependency, Trivedi et al. present Know-Evolve [38], which is an in-depth assessment of the knowledge semantic network structure. RE-Net [16] chooses to aggregate the neighborhoods of entities and applies the recurrent neural network for time dependence. Chrono-Transation [33] deals with temporal information using rule mining and graph embedding operations. However, these models usually assume that enough training quads are provided for all relationships, and the long tail relationship is not considered, which leads to poor performances in the environment with few samples.

Few-shot knowledge graph completion methods

To obtain good performance, a large amount of data is often used to train the model. But in a real knowledge graph, there are relationships with few entities. Meta-learning methods include metrics-based, model-based and optimization-based methods, aiming at fast learning with a small number of samples.

Because long-tail relations are common in KGs, GMatching [44] learns a matching metric by the learned embeddings and one-hop graph structures and proposes a one-shot relational learning model. By observing a few associative triples. However, GMatching assumes that all local neighbors contribute equally to entity embedding, while heterogeneous neighbors may have different influences. It ignores that the interaction between a few reference instances limits the representation ability of reference sets. MetaR [5] studies few-shot link prediction in KGs. It enables the model to learn faster by considering transferring relation-specific meta information. Xiong et al. [44] propose a metric-based approach to link prediction of long-tail relationships with fewer samples. However, the performance of MetaR is affected by the sparsity of entities and the number of tasks, which affects the performances. FSRL [49] can infer the true entity pairs effectively given the set of few-shot reference entity pairs for every relation, which aims at discovering facts of new relations with few-shot references. FSRL does not consider the importance of timestamp information for the completion of temporal knowledge graph. REFORM [40] studies the problem of error-aware few-shot KG completion to accumulate meta-knowledge across different meta-tasks, and propose neighbor encoder module, cross-relation aggregation module, and error mitigation module in each meta-task. MTransH [31] proposes a few-shot relational learning model with the global stage and the local stage. FAAN [34] proposes an adaptive attentional network for few-shot KG completion, which is predictive for knowledge acquisition. However, these methods are mainly aimed at static knowledge graph, and cannot make good use of the timestamp information in temporal knowledge graph.

Our model

In this section, we design a model called FTMO to complete the missing object entities in the dataset of the few-shot temporal knowledge graph, shown in Fig. 1. FTMO mainly includes the following aspects: entity embedding is generated by a time-based heterogeneous neighbor encoder; a small number of reference entity pairs are aggregated by a time-based cyclic automatic encoder to generate reference set embedding; the similarity score between query pair and reference set is calculated by the matching network, and the candidate entities are sorted to obtain the highest ranked entity. Throughout the paper, the main notations are summarized in Table 1.

Few-shot completion task

The representation of TKG can be represented as (s, r, o, t), where s and o represent entities, r represents relationships, and t represents timestamps. The TKG tasks mainly include three types: (1) given the subject entity s, the relationship r and the timestamp t to predict the object entity o: $(s, r, ?, t)$; (2) given the relationship r, the object entity o, and the timestamp t to predict the subject entity s: $(?, r, o, t)$; (3) given the subject entity s, the object entity o, and the timestamp t to predict the relationship r: $(s, ?, o, t)$. In this study, the first case is taken into consideration because we want to complete the determination of the missing object entity in the relationship.

Definition 1 Few-shot TKG completion. Given a few-shot TKG, assuming that the relation r and its few-shot reference entity pairs are known, few-shot TKG task is defined as

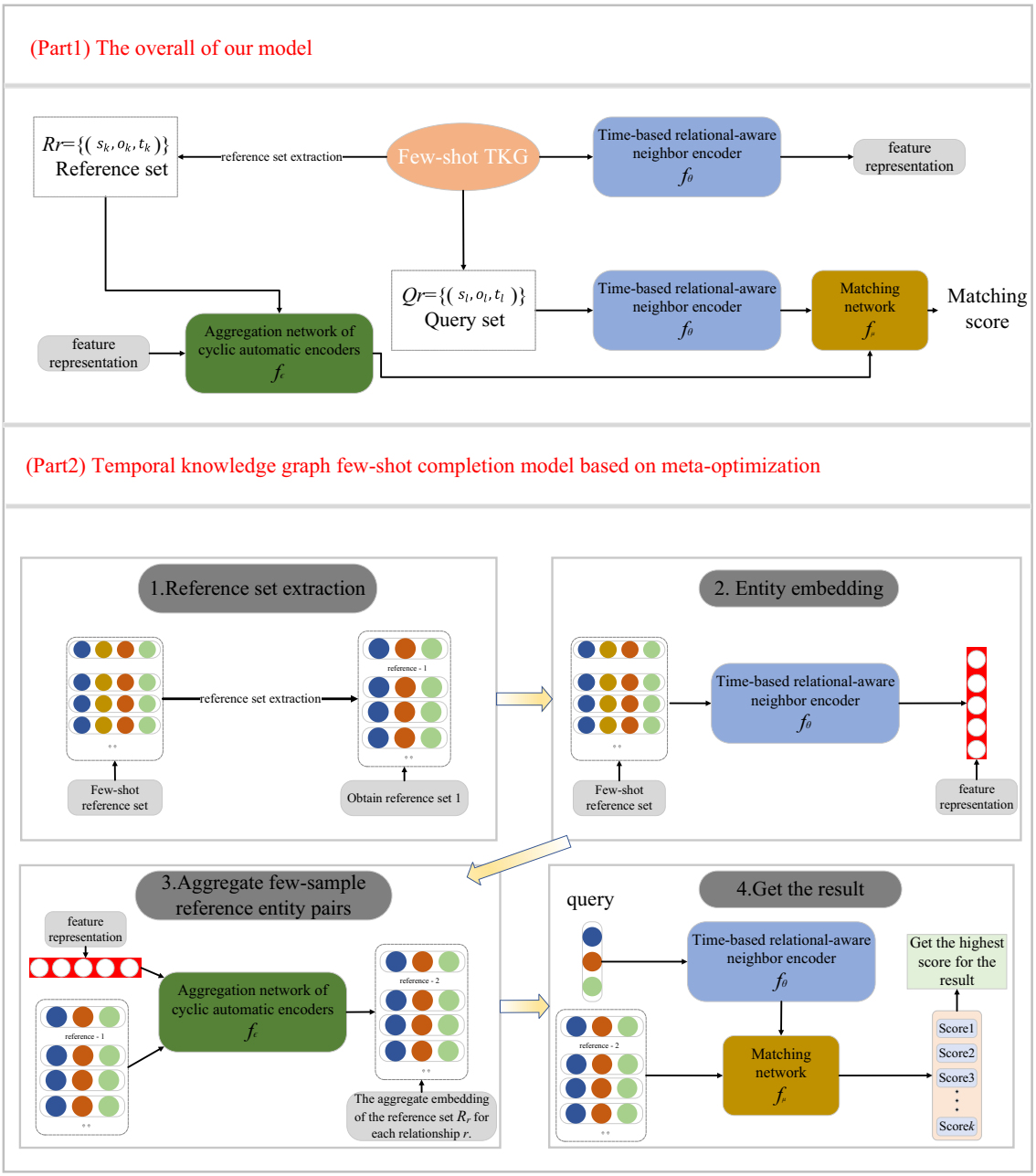

Fig. 1 The framework of FTMO model

designing a machine learning model. The object candidate entities of each new subject entity s are sorted according to the known content, so that the similarity score of the real object entities of the subject entity s is the highest.

Training task

Our goal is to design a machine learning model to predict the missing object entities in a few-shot TKG. There are two types of training models with few shots. (1) The first is a metric-based method [18, 27, 35, 39], which can learn effective metrics and the corresponding matching functions in a set of training examples. (2) The second method is based on meta-optimization [7, 8, 20, 22, 32, 47]. Its purpose is to quickly optimize model parameters and give gradients on a small number of shot data instances. Here, we use meta-optimization [32] and add LSTM [12] on this basis, which can learn the precise optimization algorithm in a small number of shot states. The task of few-shot knowledge graph completion is described as follows:

Table 1 The main notations in this paper

Symbol	Meaning	Symbol	Meaning
(s, r, o, t)	The representation of TKG	$\mathcal{T}$	Meta-training set
R_r	The set of relations r	$\mathcal{N}_h$	Time-based encoding heterogeneous neighbors
D_r^{train}	The training set of the relation r	G'	The background knowledge graph
D_r^{test}	The test set of the relation R	σ	Activation unit
Q_{s_i,r,t_i}	The remaining candidate entities set	α_i	Attention weight
$\ominus$	The set of model parameters	$\oplus$	Catenation operator
$\mathcal{L}$	The loss function of r	μ_{ro}, W_{ro}, b_{rt}	Learnable parameters
$\mathcal{AG}$	An aggregation function	n_k, d_k	Hidden states of encoder and decoder
ϵ_{s_k,o_k,t_k}	Representation of (s_k, o_k, t_k) by applying time-based neighbor encoder	RNN_{encoder} RNN_{dencoder}	Recurrent encoder and decoder
k	the size of reference set	RNN_{match}	The LSTM cell
g_t, c_t	The hidden state and the cell state of LSTM	$\mathcal{L}_{\text{rank}}$	The ranking loss
$\mathcal{P}\epsilon_r$	The remaining positive(true) entity pairs	$\mathcal{N}\epsilon_r$	The group of negative (false) entity pairs $\mathcal{N}\epsilon_r$
ξ	The safety margin distance	$\mathcal{S}_{(s_l,o_l,t_l)}$	The similarity score between query pairs (s_l, o_l) and reference set R_r
$\mathcal{L}_{\text{joint}}$	The final objective function	γ	The trade-off factor between $\mathcal{L}_{rank}$ and $\mathcal{L}_{re}$

Given a training task, each relationship $r \in R$ in the temporal knowledge graph should have a corresponding training dataset called D_r^{train}, which contains only few-shot entity pairs about the relation r and a testing dataset called D_r^{test} contains all entity pairs about the relation r. Therefore, given the test queries (s_i, r, t_i) and a small number of reference pairs in the D_r^{train} training, we can sort all the candidate entities and test our model on this basis. In summary, the loss function of r can be defined as $\mathcal{L}_{\ominus}\left(s_i, o_i, t_i | Q_{s_i,r,t_i}, D_r^{\text{train}}\right)$, where $\ominus$ is a collection of all the parameters, and Q_{s_i,r,t_i} represents the remaining candidate entities set.

The process of meta-optimizer is as follows: first, the parameters of meta-learning are initialized, and then n-dimensional learning tasks are cycled. Each task samples $D_r^{\text{meta-train}}$ and $D_r^{\text{meta-test}}$ from D_r^{train} data set, that is, support set and reference set. For each meta-learning task, T quadruples are extracted from $D_r^{\text{meta-train}}$ as reference set, one batch is extracted as query set, and then another query set is obtained by processing its query set, and its matching score is calculated, respectively. After calculating its loss function, LSTM network is used for gradient calculation, and meta-learning parameters are updated. After doing the same for $D_r^{\text{meta-test}}$, update the related parameters of meta-train.

Meta-testing refers to the fact that after sufficient data training, the learning model can be learned spontaneously, and then it can be used to predict the fact that every relationship $r \in R$ in TKG. It should be noted that in the self-learning process of a meta-learning model, every relationship is invisible to the outside world. In addition, in the same pattern as mentioned earlier, every relation r' in meta-testing should also have a few-shot training data $P_{r'}^{\text{train}}$ that contains only few-shot entity pairs about the relation r' and a few-shot testing data $P_{r'}^{\text{test}}$ that contains all entity pairs about the relation r'. The objective of model training can be defined as Eq. (1) as follows:

$$\min_{\ominus} \mathbb{E}_{\mathcal{T}} \left[\sum_{\left(s_i, o_i, t_i, Q_{s_i,r,t_i}\right) \in D_r^{\text{test}}} \frac{\mathcal{L}_{\ominus}\left(s_i, o_i, t_i | Q_{s_i,r,t_i}, D_r^{\text{train}}\right)}{|D_r^{\text{test}}|} \right]. \tag{1}$$

Note that the meta-optimization is performed over the model parameters $\ominus$, whereas the objective is computed using the updated model parameters $\ominus$.where $|D_r^{\text{test}}|$ is the number of quad (s, r, o, t) in D_r^{test}. Details on how to calculate each function section and how to modify and optimize the functionality are discussed in subsequent sections.

Time-based relational aware encoding heterogeneous neighbors

In this section, we propose a time-based relationship-aware heterogeneous neighbor encoder. The local coding structure of explicit graphs performs well in relation prediction [37]. In the previous neighborhood encoders, the average of the encoded features between neighbors is used to embed a given entity. Although embedding with the feature vector average can achieve good performance, this method ignores the different influences that heterogeneous neighbors may bring when calculating feature vectors, and this influence will produce differences in the final results [48]. Because of the existence of temporal elements, we design a relational aware heterogeneous neighbor encoder based on time combined with previous work.

Different from FSRL [49], in this process, we upgrade the matrix from three dimensional to four dimensional. In the neighbor coding process, our model first calculates the temporal information and relationship information once and then calculates the temporal information and entity information in a unified and joint way in the subsequent operation. Given a header entity s, the set of time-based relationship neighbors (*relationship, entity, time*) can be represented as $\mathcal{N}_h = \{(r_i, o_i, t_i) | (s, r_i, o_i, t_i) \in G'\}$, where G' is the background TKG, and r_i, o_i, and t_i are the i-th relation, the corresponding object entity, and the current time point s. Therefore, the time-based heterogeneous neighbor encoder can comprehensively consider the different influences of isomorphic and heterogeneous neighbors $(r_i, o_i, t_i) \in \mathcal{N}_h$, and combine entity and temporal information to calculate the feature vector representation of specific entities. On the basis of this, it can encode $\mathcal{N}_h$ and output a feature representation of s well. The attention module and formula for embedding s are defined as follows:

$$f_\theta(s) = \sigma\left(\sum\nolimits_i z_i\right), \tag{2}$$

$$\alpha_i = \frac{\exp\{\mu_{ro}^{O}\left(\mathcal{W}_{\mathrm{ro}}\left(e_{l_i} \oplus e_{o_i}\right) + b_{ro} + b_{rt}\right\}}{\sum_i \exp\{\mu_{ro}^{O}\left(\mathcal{W}_{\mathrm{ro}}\left(e_{l_i} \oplus e_{o_i}\right) + b_{ro} + b_{rt}\right\}}, \tag{3}$$

$$e_{l_i} = e_{r_i} \oplus e_{t_i}, \tag{4}$$

$$z_i = \alpha_i e_{o_i} e_{t_i}, \tag{5}$$

where σ denotes the activation unit, $\oplus$ represents the concatenation operator, e_{r_i}, e_{o_i}, and $e_{t_i} \in \mathbb{R}^{(d \times d \times 1)}$ are pretrained embeddings of r_i, o_i and t_i, and e_{l_i} represents the variable of the e_{r_i} and e_{t_i} join operations. In addition, $\mu_{ro} \in \mathbb{R}^{(d \times d \times 1)}$, $W_{\mathrm{ro}} \in \mathbb{R}^{(d \times d \times 2d)}$ and $b_{rt} \in \mathbb{R}^{(d \times d \times 1)}$ (d: pretrained embedding dimension) are learnable parameters.

By leveraging the embeddings of entity o_i, relation r_i and temporal stamp t_i to compute α_i and obtain good use of the attention weightα_i, the formulation of $f_\theta(s)$ can consider the different impacts of heterogeneous relational neighbors well. The specific details of the time-based relational aware heterogeneous neighbor encoder are shown in Fig. 2. First, the relationship information and temporal information of the quadruple related to the same subject entity are embedded, then the new variable is embedded with the object entity and temporal stamp, and then the weight factor is calculated with the intermediate quantity. Finally, the characteristic representation of the main entity is calculated.

Aggregation network of cyclic automatic encoders

In this section, we design an aggregator network consisting of recurrent autoencoder aggregators. (s_k, o_k, t_k) can be represented as $\epsilon_{s_k, o_k, t_k} = \left[f_\theta(s_k) \oplus f_\theta(o_k) \oplus f_\theta(t_k)\right]$ by applying the time-based neighbor encoder $f_\theta(s)$ to each entity pair $(s_k, o_k, t_k) \in R_r$. The embedding of R_r can be represented as follows [6, 35]:

$$f_\epsilon(R_r) = \mathcal{AG}_{(s_k, o_k, t_k) \in R_r}\left\{\epsilon_{s_k, o_k, t_k}\right\}, \tag{6}$$

where $\mathcal{AG}$ is an aggregate function for pooling operation and feedforward neural network.

To apply current neural network aggregators in graph embedding [10], we study a cyclic automatic encoder aggregator between a few samples. Specifically, the entity pair embeddings $\epsilon_{s_k, o_k, t_k} \in R_r$ are fed into a recurrent autoencoder sequentially by Eq. (7):

$$\epsilon_{s_1, o_1, t_1} \rightarrow n_1 \rightarrow \cdots \rightarrow n_k \rightarrow d_k \rightarrow \cdots \rightarrow d_1, \tag{7}$$

where k is the size of the reference set.

In Eq. (7), n_k denotes encoding and d_{k-1} denotes decoding, which are both hidden states of the decoder. n_k and d_{k-1} are calculated as Eqs. (8) and (9):

$$n_k = RNN_{\mathrm{encoder}}\left(\epsilon_{s_k, o_k, t_k}, n_{k-1}\right), \tag{8}$$

$$d_{k-1} = RNN_{\mathrm{dencoder}}(d_k), \tag{9}$$

where RNN_{encoder} represents the recurrent encoder and RNN_{decoder} describes the decoder.

The reconstruction loss for optimizing the autoencoder can be measured as Eq. (10):

$$\mathcal{L}_{\mathrm{re}}(R_r) = \sum_k \| d_k - \epsilon_{s_k, o_k, t_k} \|_2^2. \tag{10}$$

The role of $\mathcal{L}_{\mathrm{re}}$ is to merge with relationship-level losses to optimize the representation for each entity pair, thereby improving the model performance.

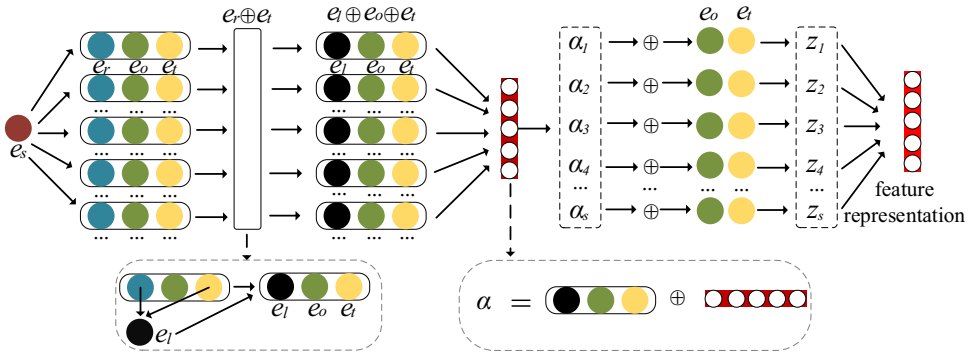

Fig. 2 Time-based relation-aware heterogeneous neighbor encoder

To embed the reference set, the hidden states of the encoder, residual links [11], and attention weights are aggregated and defined as follows:

$$\mathrm{n}'_k = n_k + \epsilon_{s_k, o_k, t_k}, \tag{11}$$

$$\beta_k = \frac{\exp\{\mu_R^T(\mathcal{W}_R n'_k + b_R)\}}{\sum_{k'} \exp\{\mu_R^T(\mathcal{W}_R n'_{k'} + b_R)\}}, \tag{12}$$

$$f_\epsilon(R_r) = \sum_k \beta_k n'_k, \tag{13}$$

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

Compared with FSRL, our model not only improves temporal information processing and the matrix dimension, but also uses a smaller gradient in combination with LSTM, which makes the result better. The cyclic automatic aggregation network for reference set contains encoder part and decoder part, as shown in Fig. 3. The encoder combines the LSTM aggregation of a small number of reference sets and feature representation vectors of entities to generate relations with small sample embeddings. The decoder aggregates the LSTM and aggregates a small number of reference sets and intermediate amounts of feature representation vectors of entities to compute the loss function.

Matching query and reference set

For matching R_r with each (s_l, o_l, t_l) in r, temporal information is considered in the process of the matching network. According to the previous efforts, there are two types of embedding vectors, which are $\epsilon_{s_l, o_l, t_l} = \left[f_\theta(s_l) \oplus f_\theta(o_l) \oplus f_\theta(t_l)\right]$ and $f_\epsilon(R_r)$. For the recurrent processor, we adopt f_μ [11] for multiple step matchings to measure the similarity between these two vectors. The t-th process step can be represented as follows:

$$g'_t, c_t = RNN_{\text{match}}\left(\epsilon_{s_l, o_l, t_l}, \left[g_{t-1} \oplus f_\epsilon(R_r)\right], c_{t-1}\right), \tag{14}$$

$$g_t = g'_t + \epsilon_{s_l, o_l, t_l}, \tag{15}$$

where RNN_{match} is the LSTM cell [12], including the hidden state g_t and the cell state c_t.

The inner product results between ϵ_{s_l, o_l, t_l} and $f_\epsilon(R_r)$ are employed as the similarity score. The matching network for query pair and reference set is shown in Fig. 4.

The query set and LSTM are combined for embedding, then the reference set and LSTM are combined for calculation, and finally the similarity score is obtained. First, the query set and LSTM are combined for embedding. Second, the reference set and LSTM are combined for calculation. Finally, the similarity score can be obtained.

Target mode training

To test the performance of the model in obtaining a reference set R_r for the query relation r, we select randomly 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, the remaining positive (true) entity pairs can be represented as $\mathcal{P}\epsilon_r = \left\{(s_l, o_l, t_l)|(s_l, r, o_l, t_l) \in G \bigcap (s_l, o_l, t_l) \notin R_r\right\}$, where $\mathcal{P}\epsilon_r$ is the positive entity pairs. On the other hand, the negative (false) entity pairs $\mathcal{N}\epsilon_r = \left\{\left(s_l, o_l^-, t_l\right)|\left(s_l, r, o_l^-, t_l\right) \notin G\right\}$ are created by polluting the tail entities. The ranking loss can be computed by Eq. (16):

$$\mathcal{L}_{\text{rank}} = \sum_r \sum_{(s_l, o_l, t_l)\in \mathcal{P}\epsilon_r} \sum_{(s_l, o_l^-, t_l)\in \mathcal{N}\epsilon_r} \left[\xi + \mathcal{S}_{(s_l, o_l^-, t_l)} - \mathcal{S}_{(s_l, o_l, t_l)}\right]_+, \tag{16}$$

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

The final objective function can be formulated as Eq. (17):

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{rank}} + \gamma \mathcal{L}_{\text{re}}, \tag{17}$$

where γ is the trade-off factor between $\mathcal{L}_{\text{rank}}$ and $\mathcal{L}_{\text{re}}$. γ is a hyperparameter because the final joint loss function cannot

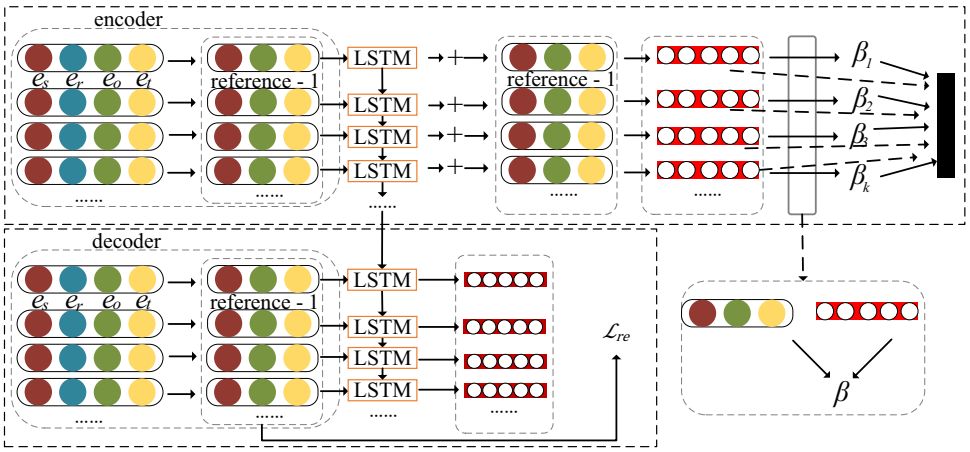

Fig. 3 The cyclic automatic aggregation network for reference set

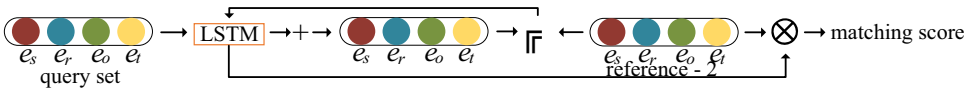

Fig. 4 The matching network for query pair and reference set

be computed directly using the sum of the two partial loss functions. In the experiments, we make the loss function of the two parts an order of magnitude by observing the loss function of the two parts and then finally determine the value of γ.

To minimize $\mathcal{L}_{\text{joint}}$ and optimize the parameters, we deal with each relation as one task and design a batch sampling on the basis of the meta-training. In each training task, few-shot entity pairs and a set of query sets are firstly selected and extracted. Then, a set of negative entity pairs is created by polluting object entities. The feature representation of subject entities, the reconstruction loss for optimizing autoencoder, the challenge and embedding formulation, the ranking loss and the loss function are successively calculated according to the formula proposed in this paper. The optimizer parameters are updated until the task is finished. Finally, an optimal parameter can be returned.

Experiments

Experiment setup

Datasets preprocessing

In the previous knowledge graph completion model, every relation in the existing dataset contained a large number of entity pairs, so it could ensure training accuracy. However, in the real world, there are many relations with a small number of entities. Therefore, to study this kind of relationship called long-tail relationship, each relation should have a small number of entity pairs in a small number of datasets. Therefore, based on the less beat standard [18, 35] and inspired by GMatching [44], we adjust the number of entities in each relationship based on the existing dataset and control the number in a lower range. For example, in a normal TKG dataset, the relationship "*is the president of*" may have approximately 10,000 entity pairs, but in a few-shot dataset, it may only have 50–500 entity pairs. To test the performance of our model, we should adjust the existing TKG dataset accordingly. Therefore, we keep the number of entities per relationship in the dataset within this range and reduce the number of relationships to less than 100. Details of specific dataset processing are described as follows:

In the experiments, ICEWS [4] and GDELT [21] are used for evaluations. The number of entities for one relation maintains between 50 and 500, and the number of relations is controlled under 100. We divide the dataset into the training set, test set and verification set with a ratio of 70: 15: 7. The statistics of ICEWS and GDELT are listed in Table 2.

Baselines

We perform two kinds of baseline models for comparisons. One kind is the vector representation and relational embedding models such as TransE [3], DistMult [46], TTransE [19], TA-TransE [9] and TA-DistMult [9]. The other kind is neighborhood coding models such as RE-Net [16], GMatching [44], MetaR [5] and FSRL [49].

Parameter settings

For GMatching, MetaR, FSRL, and FTMO, the optimal hyper-parameters are listed in Table 3, where n is the embedding dimension, λ is the learning rate, x is the maximum size of ICEWS and GDELT, h is the hidden dimension, q is the

Table 2 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

Table 3 The optimal hyper-parameters for baseline models on both datasets

Model	n	λ	x	h	q	p	a	m	f
GMatching [44]	100	0.001	1000	200	30	2	0.25	5.0	0.0001
MetaR [5]	100	0.001	1000	200	30	2	0.25	5.0	0.0001
FSRL [49]	100	0.001	1000	200	30	2	0.25	5.0	0.0001
FTMO	100	0.001	1000	200	30	2	0.25	5.0	0.0001

Table 4 The optimal hyper-parameters for baseline models on each dataset

Model	ICEWS-Few				GDELT-Few			
	λ	n	B	v	λ	n	B	v
TransE [3]	0.0001	512	512	–	0.0001	512	512	–
DistMult [46]	0.01	512	1024	–	0.01	512	1024	–
TTransE [19]	0.001	512	512	0.0	0.001	256	512	0.0
TA-TransE [9]	0.001	512	512	0.0	0.001	256	512	0.0
TA-DistMult [9]	0.001	512	1024	0.0	0.001	512	1024	0.0
RE-Net [16]	0.001	256	1024	0.5	0.001	256	1024	0.5

maximum local neighborhood number of the heterogeneous neighborhood encoder species, p is the number of steps, a is the weight attenuation, m is the edge distance, and f is the transaction factor. For the other baselines, the optimal hyper-parameters are listed in Table 4, where n is the latitude of vector embedding, B is the batch size of training data, v is the discard probability. Adam optimizer [17] is selected in the process of updating parameters.

Evaluation index

We use the hit ratio (Hits@1, Hits@5, and Hits@10) and the mean reciprocal rank (MRR) to evaluate the performances.

Experimental results

Comparisons with baselines

In this group of experiments, performance comparisons with baselines on ICEWS-Few and GDELT-Few are presented in Table 5, where the pre/post scores describe experimental results from the validation/test set. In Table 5, the best experimental results are shown in bold, and the best experimental results of the comparative baselines are underlined.

From Table 5, we can make the following conclusions.

(1) In two different comparison models, we can clearly see that the results of the model using graph neighborhood coding are better than those of the relational embedding method, which shows that neighborhood coding can better deal with heterogeneous relational entities, and the method combined with the matching network can better represent and embed entities, thus enhancing the performance of entity completion.
(2) Our model achieves the best performance in all the evaluation parameters, which proves the effectiveness of our model, indicating that preprocessing relational and temporal attributes and the combined use of heterogeneous neighborhood encoder and cyclic autoencoder aggregation network can complete the work of few-shot entities well.

Comparisons over different relations

In this group of experiments, we perform comparative experiments over different relations to evaluate the validity and stability. The comparisons are performed between FSRL and FTMO over ICEWS-Few and GDELT-Few. The experimental results are shown in Tables 6 and 7, where the pre/post scores represent the experimental scores of FSRL and FTMO, respectively.

Table 5 Performance comparisons on ICEWS-Few and GDELT-Few

Model	ICEWS-Few				GDELT-Few			
	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10	MRR
TransE [3]	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 [46]	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 [19]	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 [9]	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 [9]	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 [16]	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 [44]	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
MetaR [5]	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 [49]	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
FTMO	**0.361/0.223**	**0.523/0.446**	**0.573/0.562**	**0.435/0.346**	**0.358/0.157**	**0.507/0.319**	**0.546/0.400**	**0.413/0.267**

Best experimental results are shown in bold, and the best experimental results of the comparative baselines are underlined

Table 6 The results of FSRL and FTMO for each relation in dataset ICEWS-Few

RelationId	Hits@1	Hits@5	Hits@10	MRR
1	0.899/0.968	0.976/0.988	1.000/1.000	0.983/0.972
2	0.016/0.051	0.029/0.228	0.083/0.465	0.034/0.147
3	0.288/0.965	0.418/0.972	0.431/0.983	0.344/0.981
4	0.055/0.476	0.109/0.431	0.147/0.378	0.120/0.421
5	0.073/0.069	0.156/0.216	0.176/0.378	0.135/0.198
6	0.203/0.376	0.528/0.526	0.587/0.653	0.345/0.478
7	0.530/0.493	0.698/0.703	0.881/0.867	0.598/0.615
8	0.142/0.168	0.558/0.606	0.678/0.698	0.314/0.397
9	0.645/0.598	0.609/0.701	0.728/0.831	0.587/0.552
10	0.054/0.070	0.135/0.223	0.208/0.331	0.095/0.127

Table 7 The results of our model and FSRL for each relation in dataset GDELT-Few

RelationId	Hits@1	Hits@5	Hits@10	MRR
1	0.923/0.986	0.949/0.990	1.000/1.000	0.989/0.992
2	0.035/0.069	0.098/0.314	0.165/0.537	0.045/0.198
3	0.321/0.980	0.459/0.988	0.512/0.992	0.378/0.990
4	0.067/0.541	0.186/0.630	0.209/0.629	0.192/0.597
5	0.088/0.071	0.198/0.256	0.223/0.459	0.172/0.271
6	0.196/0.335	0.488/0.500	0.539/0.570	0.333/0.509
7	0.601/0.555	0.706/0.724	0.901/0.888	0.700/0.719
8	0.157/0.189	0.598/0.676	0.666/0.727	0.420/0.498
9	0.536/0.623	0.645/0.736	0.780/0.891	0.653/0.748
10	0.036/0.085	0.110/0.196	0.214/0.369	0.106/0.237

From Tables 6 and 7, we can make the following conclusions.

(1) From the results, we can see that the variance values of the two models we used are higher in different relationships. This is because in the evaluation process, the size of candidate sets is different for different relationships, so it is normal to have a large variance. As seen in the table, relationships with smaller candidate sets score relatively higher.
(2) It can be observed that our model has strong robustness for different relationships, which shows that our model has strong stability and can handle abnormal situations. The results show that in most cases, our model performs better for most relationships.

Ablation experiments

In this group of experiments, we perform ablation experiments to evaluate the impact of each module in FTMO from three viewpoints: without the time-based relational-aware heterogeneous neighbor encoder (W1), without the cyclic autoencoder (W2), and without the matching network (W3). We replace the relationship-aware heterogeneous neighbor encoder with an embedded average pool layer covering all neighbors in W1. In W2, the cyclic automatic encoder aggregator network is replaced by an average pool operation. In W3, LSTM is canceled and the inner product between the query embedding and reference embedding is used as the similarity score. The evaluations are performed over ICEWS-Few and GDELT-Few. The experimental results are shown in Tables 8 and 9, where the pre/post scores indicate the experimental results from the validation/test set. Different experimental results can be observed from Tables 8 and 9, and the performance differences indicate their impact of different modules in FTMO.

Stability

Impact of few-shot size. The main task of this paper is to investigate TKG completion with small samples, so we study the influence of the size of K. The few-shot size describes the size of K, and K is the size of the hit ratio (Hits@). We conduct experiments on FTMO, FSRL, and GMatching. The evaluations are performed over ICEWS-Few and GDELT-Few, and the experimental results are shown in Figs. 5 and 6. It can be observed that the completion performance of FTMO, FSRL, and GMatching improve with the increase of K value, and the performance of FTMO is always higher than that of FSRL and GMatching. It shows that FTMO has better stability and robustness when completing few-shot TKG, and is more suitable for completing few-shot TKG.

Computational complexity analysis

The time cost of FTMO mainly comes from neighbor encoder, aggregation and matching modules. The computational complexity of neighbor coding is expressed as $O(|R||E|d)$, where $|R|$ is the maximum number of neighbor relationships of task relationship r, $|E|$ is the number of neighbor entities of task entities involved in the training process, and d is the embedded representation dimension in the experiments. For the aggregation of FTMO, the time cost of updating the entity representation is $O(|E|Ld)$, where L is the number of aggregation layers, d is the embedding dimension. The computational complexity of the matching processor is $O(|R|(|E| + |T|)$, where the embedding representation of the input entity pair is $O(|R||E|)$ and $|T|$ is the number of neighbor timestamps of task entities involved in the training process. The computational complexity of timestamp information between entities is $O(|R||T|)$. The initial representation complexity of entity, relationship and timestamp information in the dataset is$O(|g|d)$, where $|g|$ is the number of tuples in the training set in the temporal knowledge graph. Because there are N iterations in the training process, the total computational complexity of the model is $O(N(|R||E|d + |E|Ld + |R|(|E| + |T|) + |g|d))$. By analyzing the computational complexity of the model in the training process, we found that compared with the baseline, we increase the processing of time information, which leads to an increase in computational complexity. When the number of neighbors facing the task relationship increases, the time complexity is further improved, that is, it takes a lot of time to train large-scale datasets, so the computational efficiency of the model will become lower for training large-scale datasets.

Defects analysis

Our model combines a time-based relational-aware heterogeneous neighbor encoder, cyclic automatic encoder aggregation network, and matching network to complete few-shot TKG. Although FTMO has better stability and is more suitable for completing few-shot TKG, there are several limitations:

(1) Datasets limitations: Although the model achieves good performance on two datasets, our model is suitable for specific temporal datasets with few samples. When applied to other datasets, the dataset needs to be processed into a few-shot temporal dataset with only a few relationships and a few entity pairs.
(2) Method limitation: In the process of time-based relational-aware neighborhood coding, the completeness of temporal knowledge graph is improved by processing relational information and temporal information first. Because the data in the dataset do not

Table 8 The results of ablation Experiment for our model and FSRL in dataset ICEWS-Few

Experiment	Hits@1	Hits@5	Hits@10	MRR
W1	0.108/0.169	0.322/0.365	0.441/0.478	0.244/0.256
W2	0.288/0.198	0.489/0.442	0.522/0.519	0.370/0.311
W3	0.268/0.213	0.449/0.398	0.550/0.488	0.359/0.317
FTMO	**0.361/0.223**	**0.523/0.446**	**0.573/0.562**	**0.435/0.346**

Best experimental results are shown in bold

Table 9 The results of ablation Experiment for our model and FSRL in dataset GDELT-Few

Experiment	Hits@1	Hits@5	Hits@10	MRR
W1	0.155/0.103	0.399/0.207	0.487/0.335	0.301/0.239
W2	0.301/0.116	0.457/0.265	0.509/0.369	0.366/0.235
W3	0.283/0.126	0.456/0.281	0.520/0.372	0.379/0.244
FTMO	**0.358/0.157**	**0.507/0.319**	**0.546/0.400**	**0.413/0.267**

Best experimental results are shown in bold

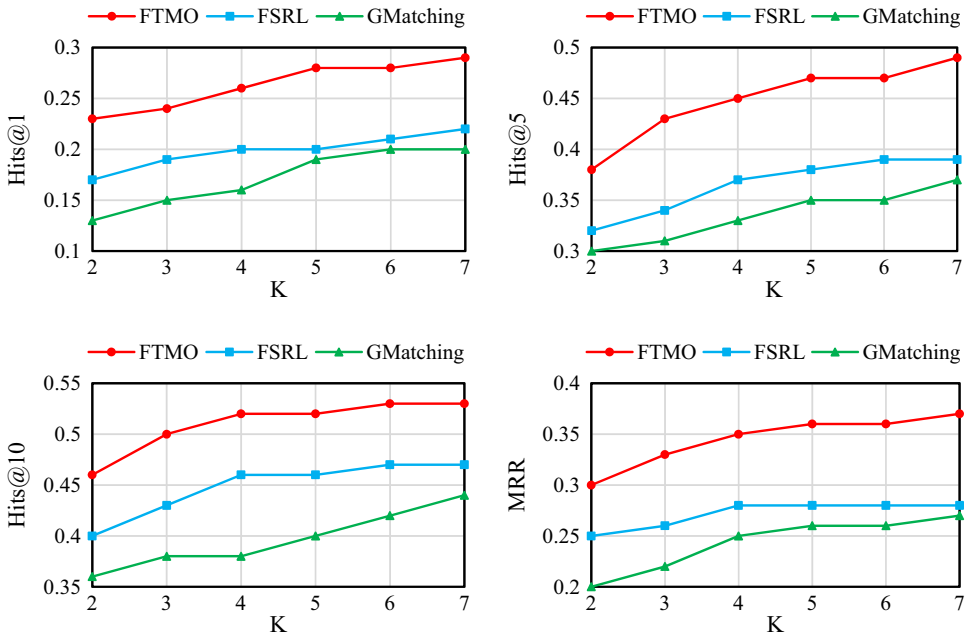

Fig. 5 Impact of few-shot size K in dataset ICEWS-Few

necessarily contain temporal information, the embedding ability of entities and relationships is unknown for the data without temporal information, which may have an uncertain impact on the final results.

Conclusion

In this paper, we first extracted few-shot datasets from two common datasets according to the rule of few-shot samples and proposed an innovative small relational model to solve the problem of few-shot TKG completion. The proposed model combines the time-based relational-aware heterogeneous neighbor encoder, cyclic automatic encoder aggregator network and matching network, and obtains good results through experiments. We performed experiments on two few-shot datasets, and the results are superior to those of the existing baseline methods. The completion ability of our model significantly improved in comparative experiments, up to 12% in ICEWS-Few dataset and up to 18% in GDELT-Few dataset. In addition, we also conducted ablation experiments and K-size analysis experiments, and the results show the effectiveness of each module for model performance and the stability of our model for entity completion.

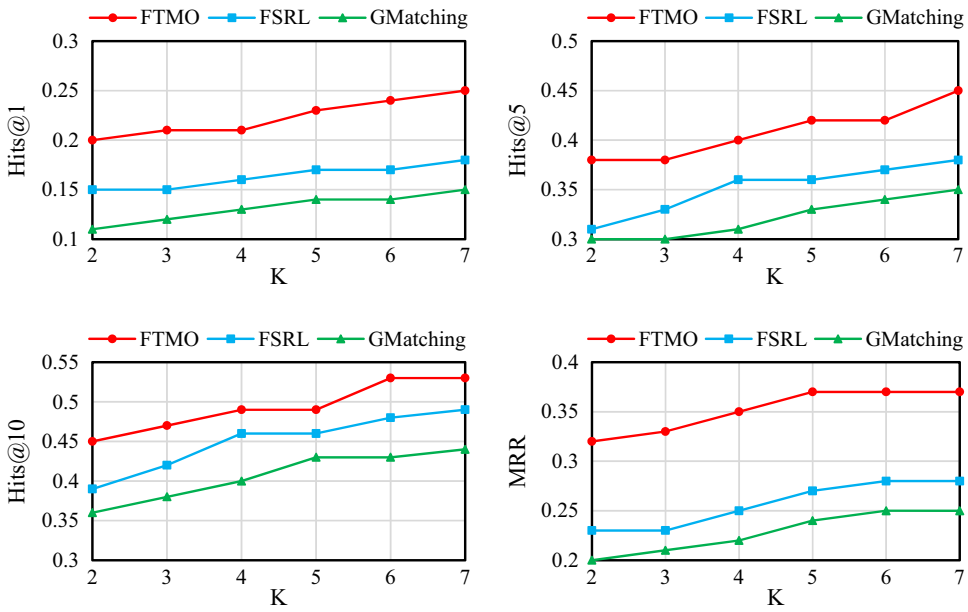

Fig. 6 Impact of few-shot size K in dataset GDELT-Few

Due to the complex structure of the neural network of FTMO model and the consideration of timestamp information, the tensor in FTMO is higher than that in baselines, which leads to the increase of computational complexity. Although the high tensor increases the complexity, it also makes the effect of entity embedding and aggregation of entity pairs in FTMO model higher than that in baselines, so that the performances of FTMO model with few-shot completion is higher than that of the baselines. In the future, we will consider these issues and make optimizations. In addition, there are other studies on few-shot TKG completion in the future. For example, we can combine entity attributes or text descriptions to improve entity embedding quality, or we can consider the relationship between different timestamps of the same triple when processing temporal information. These improvements may further improve the model performances.

Acknowledgments The work was supported by the National Natural Science Foundation of China (61402087), the Natural Science Foundation of Hebei Province (F2022501015), and the Fundamental Research Funds for the Central Universities (2023GFYD003).

Data availability The data sets used and/or analysed during the current study available from the corresponding author on reasonable request.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Bollacker K, Evans C, Paritosh P et al (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of SIGMOD, 1247–1250
2. Bordes A, Glorot X, Weston J, Bengio Y (2014) A semantic matching energy function for learning with multi-relational data. Mach Learn 94(2):233–259
3. Bordes A, Usunier N, Garcia-Duran A et al (2013) Translating embeddings for modeling multi-relational data. In: Proceedings of Neural information processing systems, 2787–2795
4. Boschee E, Lautenschläger J, O'Brien S et al (2015) Ward MD ICEWS coded event data. Harvard Dataverse 12
5. Chen M, Zhang W, Zhang W et al (2019) Meta relational learning for few-shot link prediction in knowledge graphs. In: Proceedings of empirical methods in natural language processing & international joint conference on natural language processing, 4216–4225
6. Conneau A, Kiela D, Schwenk H et al (2017) Supervised learning of universal sentence representations from natural language inference data. In: Proceedings of empirical methods in natural language processing, 670–680
7. Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of international conference on machine learning,1126–1135
8. Finn C, Xu K, Levine S (2018) Probabilistic model-agnostic meta-learning. In: Proceedings of neural information processing systems, 9537–9548
9. García-Durán A, Dumančić S, Niepert M (2018) Learning sequence encoders for temporal knowledge graph completion. In:

Proceedings of empirical methods in natural language processing, 4816–4821
10. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 1025–1035
11. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778
12. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
13. Huang J, Lu T, Zhu J et al (2022) Multi-relational knowledge graph completion method with local information fusion. Appl Intell 52(7):7985–7994
14. Ji G, He S, Xu L et al (2015) 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), 687–696
15. Jiang T, Liu T, Ge T et al (2016) Encoding temporal information for time-aware link prediction. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2350–2354
16. Jin W, Zhang C, Szekely P, Ren X (2019) Recurrent event network for reasoning over temporal knowledge graphs. CoRR, vol. http://arxiv.org/1904.05530
17. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the international conference on learning representations (ICLR)
18. Koch G, Zemel R, Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: Proceedings of international conference on machine learning deep learning workshop
19. Leblay J, Chekol MW (2018) Deriving validity time in knowledge graph. In: Proceedings of the web conference, 1771–1776
20. Lee Y, Choi S (2018) Gradient-based meta-learning with learned layerwise metric and subspace. In: Proceedings of international conference on machine learning. PMLR, 2927–2936
21. Leetaru K, Schrodt PA (2013) GDELT: global data on events, location, and tone. In: Proceedings of ISA annual convention, 1–49
22. Li Z, Zhou F, Chen F et al (2017) Meta-SGD: learning to learn quickly for few shot learning. CoRR. http://arxiv.org/1707.09835
23. Lin Y, Liu Z, Sun M et al (2015) Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of Twenty-ninth AAAI conference on artificial intelligence, 2181–2187
24. Liu H, Wu Y, Yang Y (2017) Analogical inference for multi-relational embeddings. In: Proceedings of international conference on machine learning, 2168–2178
25. Lu X, Wang L, Jiang Z et al (2022) MMKRL: a robust embedding approach for multi-modal knowledge graph representation learning. Appl Intell 52(7):7480–7497
26. Miller GA (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41
27. Mishra N, Rohaninejad M, Chen X et al (2018) A simple neural attentive meta-learner. In: Proceedings of international conference on learning representations (poster), 845–861
28. Nguyen GH, Lee JB, Rossi RA et al (2018) Continuous-time dynamic network embeddings. In Proceedings of the web conference, 969–976
29. Nickel M, Rosasco L, Poggio T (2016) Holographic embeddings of knowledge graphs. In: Proceedings of 30th AAAI Conference on Artificial Intelligence, 1955–1961
30. Nickel M, Tresp V, Kriegel HP (2011) A three-way model for collective learning on multi-relational data. In: Proceedings of international conference on machine learning, 809–816
31. Niu G, Li Y, Tang C et al (2021) Relational learning with gated and attentive neighbor aggregator for few-shot knowledge graph completion. In: Proceedings of ACM special interest group on information retrieval, 213–222
32. Ravi S, Larochelle H (2017) Optimization as a model for few-shot learning. In: Proceedings of international conference on learning representations
33. Sadeghian A, Rodriguez M, Wang DZ et al (2016) Temporal reasoning over event knowledge graphs. In: Proceedings of workshop on knowledge base construction, reasoning and mining, 6669–6683
34. Sheng J, Guo S, Chen Z et al (2020) Adaptive attentional network for few-shot knowledge graph completion. In: Proceedings of empirical methods in natural language processing, 1681–1691
35. Snell J, Swersky K, Zemel RS (2017) Prototypical networks for few-shot learning. In: Proceedings of neural information processing systems, 4077–4087
36. Socher R, Chen D, Manning CD et al (2013) Reasoning with neural tensor networks for knowledge base completion. In: Proceedings of advances in neural information processing systems, 926–934
37. Sung F, Yang Y, Zhang L et al (2018) Learning to compare: relation network for few-shot learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 1199–1208
38. Trivedi R, Dai H, Wang Y et al (2017) Know-evolve: deep temporal reasoning for dynamic knowledge graphs. In: Proceedings of International conference on machine learning, 3462–3471
39. Vinyals O, Blundell C, Lillicrap T et al (2016) Matching networks for one shot learning. In: Proceedings of Neural information processing systems, 3630–3638
40. Wang S, Huang X, Chen C et al (2021) REFORM: error-aware few-shot knowledge graph completion. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 1979–1988
41. Wang H, Jiang S, Yu Z (2020) Modeling of complex internal logic for knowledge base completion. Appl Intell 50(10):3336–3349
42. Wang Z, Zhang J, Feng J et al (2014) Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence, 1112–1119
43. Xiong C, Merity S, Socher R (2016) Dynamic memory networks for visual and textual question answering. In: Proceedings of international conference on machine learning. PMLR, 2397–2406
44. Xiong W, Yu M, Chang S et al (2018) One-shot relational learning for knowledge graphs. In: Proceedings of Empirical Methods in Natural Language Processing, 1980–1990
45. Xiong C, Zhong V, Socher R (2017) Dynamic coattention networks for question answering. In: Proceedings of international conference on learning representations (poster)
46. Yang B, Yih W, He X et al (2015) Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of international conference on learning representations (poster)
47. Yao H, Wei Y, Huang J et al (2019) Hierarchically Structured Meta-learning. In: Proceedings of international conference on machine learning. PMLR, 7045–7054
48. Zhang C, Song D, Huang C et al (2019) Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, 793–803
49. Zhang C, Yao H, Huang C et al (2020) Few-shot knowledge graph completion. In: Proceedings of the AAAI conference on artificial intelligence, 3041–3048