Temporal Knowledge Graph Reasoning based on Graph Convolution Network

Chong Mo
School of Computer Science and Technology
Harbin Institute of Technology (Shenzhen)
Shenzhen, China
1940072595@qq.com

Ye Wang
Peng Cheng Laboratory
Harbin Institute of Technology (Shenzhen)
Shenzhen, China
wangye2020@hit.edu.cn

Yan Jia
Peng Cheng Laboratory
Harbin Institute of Technology (Shenzhen)
Shenzhen, China
jiayan2020@hit.edu.cn

Cui Luo
Peng Cheng Laboratory
Shenzhen, China
luoc@pcl.ac.cn

Ye Ding
School of Cyberspace Security
Dongguan University of Technology
Dongguan, China
dingye@dgut.edu.cn

***Abstract*—Temporal Knowledge Graphs Reasoning that predicts future facts has been widely explored. Previous works attempted to use linear encoding to get recurring facts that appears in temporal knowledge graphs or use graph neural network to aggregate each sequence in temporal knowledge graphs. However, these models cannot well obtain the sequence information and ignore time information for each fact. To get the time dependence when reasoning future facts, we propose a Dynamic Time-aware Graph Convolution Network (DT-GCN), which applies a relation aware GCN to get graph structure for each sequence and a time aware GCN to get time information for each fact, then uses a relation aware recurrent neural network to get sequence patterns in temporal knowledge graphs. Extensive experiments demonstrate that DT-GCN can get better time information and has better performance than baselines in the task of link prediction.**

Keywords—Knowledge Graph, Knowledge Reasoning, Graph Neural Network

I. INTRODUCTION

Knowledge Graphs (KGs) are used to store facts in real world and play an important role in a variety of natural language processing applications [1]. Knowledge graph was proposed by Google in 2012, which is used to improve the ability of search engine and increase the search quality of users. With the development of intelligent information service technology, KGs have been widely used in Intelligent Search, Question Answering, Personalized Recommendation, and other fields [2]. Knowledge in the real world is endless, no single KG can contain all the facts, and incomplete knowledge graphs can lead to a decrease in the efficiency of many tasks. Due to the high cost of manually labeling the missing parts in KGs, predicting the missing parts in KGs becomes a very important task.

Traditionally, KGs can be regarded as multi-relation graphs without time information, each fact in KGs is represented as a triple, which contain subject entity, object entity and the relation between entities. However, each fact in KGs is not always true and may change over time, traditional KGs cannot get time information for each fact. It is necessary to add times into KGs to construct temporal knowledge graphs (TKGs). Fig.1 shows an example of a TKG. Given a TKG from time t_0 to t_T, there are two settings for TKG reasoning.

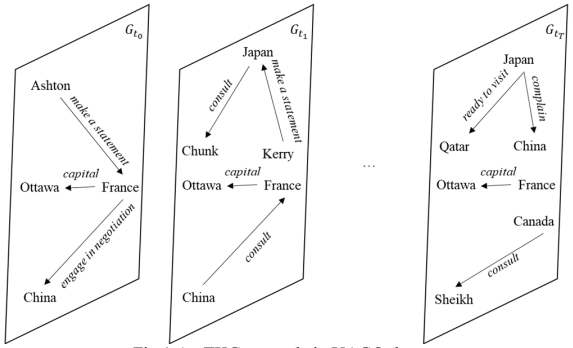

Fig.1 An TKG example in YAGO dataset.

The first is interpolation setting, which target is to predict new facts in time t for $t_0 \leq t \leq t_T$. The second is extrapolation setting, which target is to predict future facts in time t for $t > t_T$ [3]. Although there are few studies in the extrapolation setting, it is also very important. Predicting new facts in future is helpful to understand hidden factors in event, which can be applied in many applications in real world.

Some researchers have studied the extrapolation setting, Know-Evolve[4] and its extension DyRep [5] use temporal point process to model all historical facts, but they cannot model facts that occurred in the same timestamps. RE-Net first gets historical facts that related to the entities in each query, then uses Graph Convolution Network (GCN) to encode them. CyGNet [6] can better identify the repeat facts in history and predict new facts according to them. In summary, existing methods have the following restrictions: (1) mainly focus on repeating facts or concurrent facts, cannot well obtain the sequence information in TKGs. (2) ignore time information for each fact when aggregating graphs. (3) inefficient in encoding entity and relation for each fact.

In this paper, we propose a new method for TKG reasoning based on Relation Graph Convolutional Network (RGCN). The proposed Dynamic Time-aware Graph Convolution Network (DT-GCN) regards a TKG as KG sequences, uses RGCN to aggregate graph structure for each sequence and uses Time-aware GCN to get time information

978-1-6654-5968-6/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDIS55630.2022.00055

for each fact. The main contributions of this paper are as follows:

(1) We propose a new model DT-GCN predict the future facts in TKGs, which can consider graph structure and sequence patterns in TKGs.

(2) DT-GCN considers time information for every fact and adds it into GCN to better obtain the time features.

II. RELATED WORKS

Traditional knowledge graphs can be regarded as static knowledge graphs which not contain time information. Many methods have been applied in static KG reasoning and these methods are used to complete missing facts in knowledge graph. TransE [7] embeds entities and relations in the KG to low dimensional vectors and uses translating distance to get the probability of missing facts. TransH [8] and TransR [9] are extended models of TransE, they use a hyperplane and a new space to project entities and relations to deal with one-to-many, many-to-one, and many-to-many relations which TransE cannot deal with. RESCAL [10] embeds entities and relations to vectors and matrices respectively, then uses semantic matching to get the correlations of entities. DistMult [11] simplifies relation matrix by limiting it to a diagonal matrix and uses bilinear formulation to complete the task of link prediction. ConvE [12] uses Convolution Neural Network to get the interaction between entities and relations.

With the increasing use of Graph Convolution Neural Network(GCN), some models use GCN to get graph structure in knowledge graph. RGCN [13] adds a relation aware part to GCN to assign different weights to all relations in a KG. CompGCN [14] divides relations into several parts, and jointly embeds entities and relation of each part. SACN [15] uses a GCN with weight to utilize learnable relation weight when aggregating graphs and introduces a convolution decoder ConvTransE which can perform better in the task of link prediction for the GCN-based models. However, these models can only be used in static KG, they cannot get time information.

To obtain the time information in KGs, some researchers reason in TKGs. For the interpolation setting, TA-TransE [16] combines relation and timestamp for each fact in TKG to form a predicate token sequence. TTransE [17] also integrates time into the embedding of relation for each fact. HyTE [18] uses a hyperplane to integrates time and relation. TeRo [19] introduces time rotation to embed timestamp into relations and uses the score function like RotatE [20] to predict the missing facts.

For the extrapolation setting, Know-Evolve and DyRep uses temporal point process to model TKGs with continuous time. CyGNet uses a generation-copy network to better get the repeated facts in history. Re-Net uses GCN with relation awareness and GRU to reason concurrent facts of multiple timestamps in TKGs and models the time correlation in the whole-time domain. However, these models cannot well obtain the sequence information in TKGs and ignore time information for each fact.

III. PROBLEM FORMULATION

In this section, we give notations and problem definitions used in DT-GCN. We regard a TKG as a sequence of knowledge graph with timestamp, i.e., $G = \{G_1, G_2, \dots, G_t\}$,

TABLE 1 Symbol Descriptions

Notations	Descriptions
G_t	Sequence in TKG at timestamp t
$\boldsymbol{H}_t, \boldsymbol{R}_t$	Embedding of all entities and relations at timestamp t
$\boldsymbol{R}_t^s$	Embedding of all relations from $\boldsymbol{H}_{t-1}$
$\boldsymbol{H}_t^{\omega,1}$	Embedding of all entities in relation aware GCN
$\boldsymbol{H}_t^{\omega,2}$	Embedding of all entities in time aware GCN

for each KG $G_i = \{E, R, S_i\}, 1 \le i \le t$, E is the set of entities, R is the set of relations, and S_i is the set of facts which occur at timestamp t. A fact in TKC is defined as a quadruple (s, r, o, t), where $s, o \in E$ are subject entity and object entity, $r \in R$ is the relation between two entities, and t is the timestamp of the fact. The symbol descriptions are in Table 1.

In our work, we need to reason facts which occur after the last time t of the temporal knowledge graph. We can transform this problem into link prediction, or more specifically, entity prediction. Entity prediction is predicting the missing object entity of a quadruple $(s, r, ?, t+1)$ or the missing subject entity of a quadruple $(?, r, o, t+1)$. We also assume that the prediction of the facts at timestamp $t+1$ only depends on the historical facts at the latest m timestamps, i.e., $\{G_{t-m+1}, G_t\}$. Therefore, for a quadruple $(s, r, ?, t+1)$, DT-GCN can get the probability vectors of all object entities with the subject entity s, the relation r and the historical KG sequence $G_{t-m+1:t}$, as shown:

$$\boldsymbol{p}(o_{t+1}|s_{t+1}, r_{t+1}) = \boldsymbol{p}(o_{t+1}|\boldsymbol{H}_t, \boldsymbol{R}_t, G_{t-m+1:t}) \quad (1)$$

where $\boldsymbol{H}_t \in \mathbb{R}^{|V|\times d}$ is the embedding matrix of all entities at timestamp t, $\boldsymbol{R}_t \in \mathbb{R}^{|\boldsymbol{V}|\times \boldsymbol{d}}$ is the embedding matrix of all relations at timestamp t, d is the dimension of the embeddings.

IV. OVERVIEW OF DT-GCN

In this section, we introduce our proposed model DT-GCN. DT-GCN consists of two parts, structure module and sequence module, where the former uses GCN to get graph structure in temporal knowledge graph, and the later uses RNN to get sequence patterns in temporal knowledge graph. Fig.2 shows the how dynamic embedding for all entities and relation update in DT-GCN.

A. Structure Module

The structure module consists of a relation-aware GCN and a time-aware GCN. We extend the fact set S_t and relation set R in each sequence G_t with corresponding inverse relations, i.e., $S_t' = S_t \cup \{(o, r^{-1}, s, t)|(s, r, o, t) \in S_t\}$ and $R' = R \cup R_{inv}$, where $R_{inv} = \{r^{-1}|r \in R\}$ is the inverse relation set. For a sequence G_t of the TKG, we can get the embedding $\boldsymbol{r}_t \in \mathbb{R}^d$ for each relation and $\boldsymbol{h}_t \in \mathbb{R}^d$ for each entity. The relation aware GCN updates an entity embedding at layer $l+1$ from the embedding of its corresponding entities and relations at layer $l \in [0, \omega-1]$, as in:

$$\boldsymbol{h}_{o,t}^{l+1} = f\Big(\frac{1}{c_o}\sum_{(s,r),(s,r,o)\in S_t} \boldsymbol{W}_{dir(r)}^l(\boldsymbol{h}_{s,t}^l + \boldsymbol{r}_t) + \boldsymbol{W}_{self-loop}^l \boldsymbol{h}_{o,t}^l\Big) \quad (2)$$

where $\boldsymbol{h}_{s,t}^l, \boldsymbol{h}_{o,t}^l, \boldsymbol{r}_t$ are the embeddings of entities o, s and relation r in l^{th} layer at timestamp t, $\boldsymbol{h}_{o,t}^{l+1}$ is the embedding

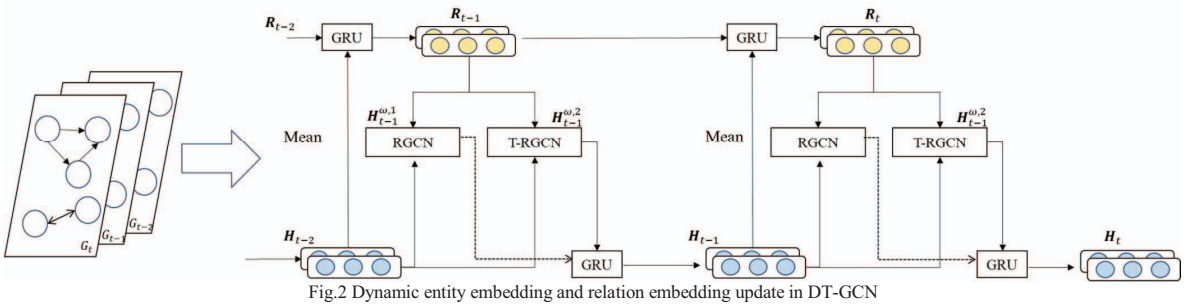

Fig.2 Dynamic entity embedding and relation embedding update in DT-GCN

of entity o in $l+1^{th}$ layer at timestamp t. $\boldsymbol{W}^{l}_{dir(r)} \in \mathbb{R}^{d\times d}$ is a relation-type specific matrix for different relation $r \in R$ and $\boldsymbol{W}^{l}_{self-loop} \in \mathbb{R}^{d\times d}$ is a self-loop matrix in l^{th} layer, $f(\cdot)$ is the ReLU activation function, and c_o is the relation weight for entities, equal to indegree of entities. For the matrix embedding $\boldsymbol{W}^{l}_{dir(r)}$, we define it for different relations, as in:

$$\boldsymbol{W}_{dir(r)} = \begin{cases} \boldsymbol{W}_O, & r \in R \\ \boldsymbol{W}_I, & r \in R_{inv} \end{cases} \tag{3}$$

All the matrix embeddings will be initialized at the first layer, and the relation aware GCN get all entity embeddings based on the facts which occurred between them at timestamp t, DT-GCN represents it as $\boldsymbol{H}^{\omega,1}_t$. In order to get time information for each fact in the sequence G_t, DT-GCN applies a time aware GCN. According to the dynamic entity embedding $\boldsymbol{H}_{t-1}$, the time aware GCN updates an entity embedding at layer $p+1$ ($p \in [0, \omega-1]$) can be defined in the below:

$$\boldsymbol{h}^{p+1}_{o,t} = g\Big(\sum_{(s,r),\exists(s,r,o)\in S_t} \mathcal{T}_{(s,r,o,t)} \cdot \boldsymbol{W}^{p}_1 \boldsymbol{h}^{p}_{s,t} + \boldsymbol{W}^{p}_2 \boldsymbol{h}^{p}_{o,t}\Big) \tag{4}$$

where $\boldsymbol{h}^{p}_{s,t}$ and $\boldsymbol{h}^{p}_{o,t}$ are the embeddings of entities o, s in p^{th} layer at timestamp t, $\boldsymbol{h}^{p+1}_{o,t}$ is the embeddings of entities o in $p+1^{th}$ layer at timestamp t, $\boldsymbol{W}^{p}_1 \in \mathbb{R}^{d\times d}$ is a weight matrix and $\boldsymbol{W}^{p}_2 \in \mathbb{R}^{d\times d}$ is a self-loop matrix in l^{th} layer. $\mathcal{T}_{(s,r,o,t)}$ is time weight for facts in G_t, equal to the last time each fact occurred minus the last time the entity s appeared, and then take the logarithm. $g(\cdot)$ is the tanh activation function. Finally, the time aware GCN get all entity embeddings based on the facts which occurred at timestamp t, DT-GCN represents it as $\boldsymbol{H}^{\omega,2}_t$.

B. sequence module

DT-GCN first set dynamic entity embedding $\boldsymbol{H}_t$ and dynamic relation embedding $\boldsymbol{R}_t$ as random initial embeddings $\boldsymbol{H}_0$ and $\boldsymbol{R}_0$. For each sequence G_t, DT-GCN gets a structural relation embedding $\boldsymbol{R}^{s}_t$ from the entity embedding $\boldsymbol{H}_{t-1}$ at timestamp $t-1$ according to the entities related to relations appear in G_t, and then applies mean pooling operation in $\boldsymbol{R}^{s}_t$. The dynamic relation embedding can be updated by RNN, as in:

$$\boldsymbol{R}_t = GRU(\boldsymbol{R}_{t-1}, \boldsymbol{R}^{s}_t) \tag{5}$$

For the update of dynamic entity embedding, DT-GCN concatenates the two embeddings $\boldsymbol{H}^{\omega,1}_t$and $\boldsymbol{H}^{\omega,2}_t$ from relation aware GCN and time aware GCN, and then use RNN to get the sequence information, as in:

$$\boldsymbol{H}_t = GRU(\boldsymbol{H}_{t-1}, [\boldsymbol{H}^{\omega,1}_t; \boldsymbol{H}^{\omega,2}_t]) \tag{6}$$

where ; represents concatenate operation.

C. Score Function

Previous works on KG reasoning show that the convolutional decoder performs better. Thus, DT-GCN uses ConvTransE as the decoder. According to the dynamic entity embedding $\boldsymbol{H}_t$ and the dynamic relation embedding $\boldsymbol{R}_t$, the probability of entity prediction can be defined as:

$$\boldsymbol{p}(o_{t+1}|\boldsymbol{H}_t, \boldsymbol{R}_t, G_{t-m+1:t}) = \boldsymbol{\sigma}(\boldsymbol{H}_t \otimes ConvTransE(\boldsymbol{s}_t, \boldsymbol{r}_t)) \tag{7}$$

where $\boldsymbol{s}_t$ and $\boldsymbol{r}_t$ are the embeddings of entity s and relation r, $ConvTransE(\cdot)$ consists of two convolutional layers and a linear layer, σ is the sigmoid function, $\otimes$ represents Hadamard product. In order to calculate the final loss, DT-GCN also get the probability of relation prediction, similar as the entity prediction, the probability of relation prediction can be defined as:

$$\boldsymbol{p}(r_{t+1}|\boldsymbol{H}_t, \boldsymbol{R}_t, G_{t-m+1:t}) = \boldsymbol{\sigma}(\boldsymbol{R}_t \otimes ConvTransE(\boldsymbol{s}_t, \boldsymbol{o}_t)) \tag{7}$$

where $\boldsymbol{s}_t$ and $\boldsymbol{o}_t$ are the embeddings of entity s and entity o.

D. Parameter Learning

The task of link prediction can be regards as multi-classification task, DT-GCN uses cross-entropy to calculate the loss, as in:

$$L = -(\sum \lambda \cdot log\, \boldsymbol{p}(o_{t+1}) + (1-\lambda) \cdot log \boldsymbol{p}(r_{t+1})) \tag{8}$$

where $log\boldsymbol{p}(o_{t+1})$ and $log\boldsymbol{p}(r_{t+1})$ are the entity probability vector $\boldsymbol{p}(o_{t+1}|\boldsymbol{H}_t, \boldsymbol{R}_t, G_{t-m+1:t})$ and the relation probability vector $\boldsymbol{p}(r_{t+1}|\boldsymbol{H}_t, \boldsymbol{R}_t, G_{t-m+1:t})$, λ is the parameter which controls the loss terms, $\sum(\cdot)$ represents the sum of losses for all quadruples.

V. EXPERIMENTS

A. Experimental Settings

In our work, we use five datasets for the experiment, ICEWS18, ICEWS14, GDELT, YAGO and WIKI. The first two datasets are from the Integrated Crisis Early Warning System [21], GDELT is from the Global Database of Events, Language and Tone [22]. The form of facts in ICEWS18, ICEWS14 and GDELT is (s, r, o, t), where t is the timestamp of the fact, and the form of facts in YAGO and WIKI is $(s, r, o, [t_s, t_e])$, where t_s is the starting timestamp and t_e is the ending timestamp. We also divide all datasets into training, validation and testing sets with a proportion of 80%, 10% and

TABLE 2 Statistics of datasets

Datasets	Entities	Relations	Training	Validation	Testing	Time interval
ICEWS18	23,033	256	373,018	45,995	49,545	24 hours
ICEWS14	6,869	230	74,845	8,514	7,371	24 hours
WIKI	12,554	24	539,286	67,538	63,110	1 year
YAGO	10,623	10	161,540	19,523	20,026	1 year
GDELT	7691	240	1,734,399	238,765	305,241	15 mins

10% by timestamps following. Table 2 summarizes the statistics of these datasets.

B. Evaluation Metrics

We use two metrics to evaluate our model, Mean Reciprocal Rank (MRR) and Hits@{1, 3, 10} (the proportion of correct test cases that are ranked within top 1/3/10). We report all experimental results on the raw datasets. Some filter settings [23-24] only filter out the facts at query timestamp t, instead of deleting these corrupted facts, it may cause our model views some facts that occur at previous timestamp as correct facts at timestamp t. Thus, we use raw setting in all experiments when evaluating.

C. Baselines

We compare our model with static KG reasoning models and TKG reasoning models. Static KG reasoning models include DistMult, RGCN, ConvE and RotatE. TKG reasoning models include interpolation models HyTE, TTransE, TA-DistMult and extrapolation models Know-Evolve, DyRep, CyGNet and RE-Net.

D. Experimental Settings

All experiments are evaluated on an Nvidia A100 80GB PCIe GPU. Adam is adopted for parameter learning with the learning rate of 0.001. We set the dimension of entity embedding and relation embedding is 200. The length of history for ICEWS18, ICEWS14, WIKI, YAGO and GDELT are all set to 5. The number of relation aware GCN and time aware GCN layers is 2 and the dropout rate for each layer is 0.2. For ConvTransE, it contains two convolution layers and a linear layer, and the dropout rate for each layer also is 0.2. For the learning parameter λ, we set it as 0.5. All experiments are carried out under the same conditions, and we only use the historical facts in TKGs to predict multi-step facts in the future.

E. Experimental Results

The experimental results on the entity prediction task are showed in Table 3 and Table 4. Tables 3 shows that DT-GCN have better performance than baselines on the ICEWS14 dataset. Compared with static KG models, DT-GCN considers temporal information in TKGs, so DT-GCN gets better results. Compared with the three temporal models for the interpolation setting, DT-GCN can get better results because DT-GCN considers temporal sequences in TKGs, it can get dynamic entity embedding and dynamic relation embedding for each timestamp. For models applied in extrapolation setting, DT-GCN also has advantages on the entity prediction task. Specifically, CyGNet uses linear method directly to encode for repeated facts in TKGs, DT-GCN adds structural dependency on concurrent facts. RE-Net verifies the importance of GCN for obtaining structural dependency. Based on the use of GCN, DT-GCN adds a relation aware GCN and time aware GCN to get time dependency, so DT-GCN performs better than RE-Net. For the ICEWS18 dataset, RE-Net performs better than DT-GCN, because facts in ICEWS18 dataset have static information that is ignored by DT-GCN.

Table 4 shows that DT-GCN have better performance than baselines on the WIKI, YAGO and GDELT. For the two datasets WIKI and YAGO, static KG models and interpolation temporal models perform poorly, because the time interval for the two datasets is one year, they cannot get accurate entity representation from TKGs. We also report the experimental result of DT-GCN with the ground truth events (GT), which means the true facts at predicted timestamp will be added to the historical facts after DT-GCN reasons on the timestamp is completed. It also illustrates the portability of our model, which can combine the new facts to predict the probability of future facts. We can see that the result of DT-GCN w.GT is better than raw results, because the entity representation will be inaccurate when the time step increasing during the reasoning period. For the GDELT dataset, all the models perform poorly, the MRR score of DT-GCN is 19.63. That's because many entities are abstract concepts in GDELT dataset, it may cause temporal reasoning more difficult, and all the models can only get similar partial facts.

Compared with the results of baselines, DT-GCN has achieved 14% MRR improvement than RE-Net for ICEWS14 dataset, 18% MRR improvement than RE-Net for WIKI dataset, 16% MRR improvement than RE-Net for YAGO dataset. The results show that DT-GCN is more capable than baselines for TKG reasoning. Furthermore, DT-GCN gets the embedding of all entities and relations during the encoding process, so DT-GCN can be trained faster than CyGNet and RE-Net. Specifically, DT-GCN performs 2.7 times faster than RE-Net, and DT-GCN consumes fewer computing resources. Therefore, DT-GCN is more efficient than extrapolation temporal models.

F. Ablation Studies

To verify whether each part of DT-GCN works in the reasoning process, we set some experiments for ablation studies using the ground truth history on the testing sets. We report all experimental results of DT-GCN w.GT in Table 5 and Table 6.

As we have mentioned in the last section, DT-GCN uses a relation aware GCN to get graph structure in TKGs. To show how the relation aware GCN affects the experimental results of DT-GCN, we remove relation aware GCN from DT-GCN, and use random initial embedding of entities and relation to replace the embeddings generated by the relation aware GCN. The result denoted as -r-GCN w.GT, as shown in Table 5 and Table 6. We can see that the experimental results will get worse than DT-GCN w.GT in the ICEWS18, ICEWS14 and GDELT datasets, because facts in the three datasets have stronger structure. The experimental results are equal to DT-GCN w.GT in the YAGO and WIKI datasets, because facts in the two datasets have long time interval, structure is not important. It also shows that the significance of relation aware GCN for TKG reasoning.

TABLE 3 The results of entity prediction task on ICEWS18 and ICEWS14 with raw metrics

Model	ICEWS18				ICWES14			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
DistMult	13.86	5.61	15.22	31.26	9.72	8.92	10.09	22.53
RGCN	15.05	8.13	16.49	29.00	15.03	10.53	16.12	31.47
ConvE	22.56	14.26	25.41	41.67	21.64	17.32	23.16	38.37
RotatE	11.63	6.47	12.31	28.03	9.79	7.50	9.37	22.24
HyTE	7.41	3.10	7.33	16.01	7.72	6.72	7.94	20.16
TTransE	8.44	1.85	8.95	22.38	10.86	10.41	12.72	23.65
TA-DistMult	15.62	8.45	17.09	32.21	11.29	8.94	11.60	23.71
Know-Evolve	7.41	6.82	7.87	14.76	16.81	13.62	18.63	29.20
DyRep	7.82	6.75	7.73	16.33	17.54	14.08	19.87	30.34
CyGNet	24.98	15.54	28.58	43.54	22.74	17.74	27.54	41.62
RE-Net	**26.62**	**16.43**	**30.27**	**45.57**	23.85	10.83	14.63	42.58
DT-GCN	25.05	16.05	28.10	43.86	**28.87**	**21.35**	**33.26**	**47.52**
DT-GCN w.GT	28.22	18.23	32.10	48.01	33.62	26.70	39.03	51.96

TABLE 4 The results of entity prediction task on YAGO, WIKI and GDELT with raw metrics

Model	WIKI			YAGO			GDELT			
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@1	H@3	H@10
DistMult	27.96	32.45	39.51	44.05	49.70	59.94	8.61	3.91	8.27	17.04
RGCN	13.96	15.75	22.05	20.25	24.01	37.30	12.17	7.40	12.37	20.63
ConvE	26.41	30.36	39.41	41.31	47.10	59.67	18.43	14.79	19.57	32.25
RotatE	26.08	31.63	38.51	42.08	46.77	59.39	3.62	0.52	2.26	8.37
HyTE	25.40	29.16	37.54	14.42	39.73	46.98	6.69	0.01	7.57	19.06
TTransE	20.66	23.88	33.04	26.10	36.28	47.73	5.53	0.46	4.97	15.37
TA-DistMult	26.44	31.36	38.97	44.98	50.64	61.11	10.34	4.44	10.44	21.63
Know-Evolve	10.54	13.08	20.21	5.23	5.63	10.23	15.88	10.33	15.69	22.28
DyRep	10.41	12.06	20.93	4.98	5.54	10.19	16.25	9.27	16.45	22.86
CyGNet	30.77	33.83	41.19	46.72	52.48	61.52	18.05	11.13	19.11	31.50
RE-Net	30.87	33.55	41.27	46.81	52.71	61.93	19.60	**12.03**	**20.56**	33.89
DT-GCN	**37.88**	**42.68**	**52.94**	**56.22**	**62.53**	**73.15**	**19.63**	11.80	19.73	**34.21**
DT-GCN w.GT	49.96	56.20	67.31	60.84	67.54	79.53	21.02	12.94	21.80	35.18

TABLE 5 The results for ablation studies on ICEWS18 and ICEWS14 with raw metrics

DT-GCN	ICEWS18				ICWES14			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
DT-GCN w.GT	28.22	18.23	32.10	48.01	33.62	26.70	39.03	51.96
-r-GCN w.GT	27.34	17.72	30.94	46.82	32.10	25.41	37.82	50.29
-t-GCN w.GT	28.22	18.23	32.10	48.01	33.62	26.70	39.03	51.96

TABLE 6 The results for ablation studies on WIKI, YAGO and GDELT with raw metrics

DT-GCN	WIKI			YAGO			GDELT			
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@1	H@3	H@10
DT-GCN w.GT	49.96	56.20	67.31	60.84	67.54	79.53	21.02	12.94	21.80	35.18
-r-GCN w.GT	49.96	56.20	67.31	60.84	67.54	79.53	19.54	11.41	19.71	33.19
-t-GCN w.GT	45.74	50.91	62.04	55.82	60.73	73.31	20.51	12.04	20.42	33.92

As we have mentioned in the last section, DT-GCN uses a time aware GCN to get graph structure in TKGs. To show how the time aware GCN affects the experimental results of DT-GCN, we remove time aware GCN from DT-GCN, and use random initial embedding of entities and relation to replace the embeddings generated by the time aware GCN. The result denoted as -t-GCN w.GT, as shown in Table 5 and Table 6. We can see that the experimental results will get worse than DT-GCN w.GT in the YAGO and WIKI datasets, because facts in the two datasets have stronger timeliness. The experimental results are equal to DT-GCN w.GT in the ICEWS18 and ICEWS14 datasets, because the two datasets have weak time dependency. We can also observe that the experimental results become worse that GDELT dataset, because facts in GDELT datasets are abstract, neither structural information nor time information is particularly obvious. It also shows that the significance of time aware GCN for TKG reasoning.

In addition, we also use linear decoder to replace ConvTransE, the result is worse than DT-GCN w.GT. the experimental results of this part are not shown in this paper. The experiment shows that the convolution decoder is more suitable for DT-GCN, because the convolution decoder can better get entity features. The dimension of entity embedding and relation embedding is important factor affecting the experimental results, we set the dimension as 50, 100, 200, 400, 1000. When the embedding dimension is less than 200, the experimental results basically do not change, but when the embedding dimension is 400 or 1000, the experimental results will significantly decrease. To find a history length which can fit all the datasets, we set the history length as 1, 2, 5, 10. The experimental results show that DT-GCN has the best performance when the history length is 5.

TABLE 7 Several case studies used to illustrate how to inference in TKGs

Historical facts before t-1	Historical facts at t-1	Query at t	Answer
Catherine Ashton, make a statement, **France** **France**, engage in negotiation, China	China, consult, **France**	**France**, consult, ?	China
Canada, consult, Sheikh Hashina Wajed **Canada**, refuse to ease economic sanctions, North Korea	**Canada**, impose embargo, North Korea	**Canada**, make an appeal or request, ?	Iran
Japan, consult, Chunk Hagel John Kerry, make statement, **Japan** **Japan**, complain officially, China	**Japan**, ready to visit, Qatar	**Japan**, make a visit, ?	Qatar

G. Case Studies

We provide several case studies in Table 8. The first case (France, consult, ?, t) from ICEWS14 illustrates how historical facts affect the predict results. We can see that France had a series of actions toward China in history and China had consulted with Canada at timestamp $\boldsymbol{t-1}$, so we inference Canada may consult with China at timestamp $\boldsymbol{t}$, and that's exactly the correct answer for the query. The second case (Canada, make an appeal or request, ?) shows that historical facts may not play a role in reasoning. Canada had several negative actions towards North Korea in history, so we inference that Canada may make an appeal or request to North Korea at timestamp $\boldsymbol{t}$, but the right answer is Iran, which not appear in history. The problem happens a lot in reasoning, and it is also one of the reasons why reasoning is inaccurate.

The third case (Japan, make a visit, ?) demonstrate how historical facts at timestamp $\boldsymbol{t-1}$ affect the reasoning result at timestamp $\boldsymbol{t}$. We can see that the answer for this query is only correlated to the facts at timestamp $\boldsymbol{t-1}$, the situation is more likely to happen in YAGO and WIKI because the time interval is long in the two datasets. If we pay more attention to historical facts before timestamp $\boldsymbol{t-1}$, the predicted results for timestamp $\boldsymbol{t}$ may be inaccurate, that's why the experimental results can be better in YAGO datasets.

CONCLUSION

In this paper, we propose a new method DT-GCN for temporal reasoning, which applies a relation aware GCN to get structure information and a time aware GCN to get time information in temporal knowledge graphs. DT-GCN cannot only obtain the structure information and sequence patterns of the temporal knowledge graphs, but also obtain the time information for each fact in the graph. Thus, temporal reasoning can be gotten with score functions based on the dynamic entity embedding and the dynamic relation embedding. Experimental results show that DT-GCN has better performance than baselines on three datasets, it also shows that DT-GCN has significant advantages and superiority in temporal reasoning.

ACKNOWLEDGEMENT

This work is supported in part by the National Natural Science Foundation of China under grant No.61976051 and the Major Key Project of PCL(Grant No.PCL2021A09，PCL2021A02, PCL 2022A03).

REFERENCES

[1] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge Graph Embedding: A Survey of Approaches and Applications," IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 12, pp. 2724-2743, Dec 2017.

[2] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, and H. Xiong etal, "A Survey on Knowledge Graph-Based Recommender Systems," IEEE Transactions on Knowledge and Data Engineering, 2020.

[3] W. Jin, M. Qu, X. Jin, and X. Ren, "Recurrent event network: Autoregressive structure inference over temporal knowledge graphs ," Conference on Empirical Methods in Natural Language Processing, 2020, pp. 6669-6683.

[4] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," International Conference on Machine Learning, 2017, pp. 3462-3471.

[5] R. Trivedi, M. Farajtabar, P. Biswai, and H. Zha, "Dyrep: Learning representations over dynamic graphs," International Conference on Learning Representations, 2019.

[6] C. Zhu, M. Chen, C. Fan, G. Cheng and Y. Zhan, "Learning from History: Modeling Temporal Knowledge Graphs with Sequential Copy-Generation Networks," International Conference on Learning Representations, 2021.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," Advances in Neural Information Processing Systems, 2013.

[8] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," AAAI Conference on Artificial Intelligence, 2014.

[9] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," AAAI Conference on Artificial Intelligence, 2015.

[10] M. Nickel, V. Tresp, and H.P. Kriegel, "A three-way model for collective learning on multi-relational data," International Conference on Machine Learning, 2011.

[11] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," International Conference on Learning Representations, 2015.

[12] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," AAAI Conference on Artificial Intelligence, 2018.

[13] M. Schlichtkrull, T. Kipf, P. Bloem, R. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," European semantic web conference, 2018, pp. 593-607.

[14] S. Vashishth, S. Sanyal, V. Nitin and P. Talukdar, "Composition-based multi-relational graph convolutional networks," International Conference on Learning Representations, 2019.

[15] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," AAAI Conference on Artificial Intelligence, vol. 33, pp. 3060-3067, 2019.

[16] A. Garcia-Duran, S. Dumancic, and M. Niepert, "Learning sequence encoders for temporal knowledge graph completion," Conference on Empirical Methods in Natural Language Processing, 2018.

[17] J. Leblay, and M.W. Chekol, “Deriving validity time in knowledge graph,” International World Wide Web Conferences Steering Committee, 2018.

[18] S. Dasgupta, S.N. Ray, and P. Talukdar, “Hyte: Hyperplane-based temporally aware knowledge graph embedding,” Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2001-2011.

[19] C. Xu, M. Nayyeri, F. Alkhoury, H.S. Yazdi, and J. Lehmann, “TeRo: A time-aware knowledge graph embedding via temporal rotation, “ International Conference on Computational Linguistics, 2020.

[20] Z. Sun, Z. H. Deng, J. Y. Nie and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” International Conference on Learning Representations, 2019.

[21] Z. Y. Liu, M. S. Sun, Y. K. Lin and R. Xie, “Research Progress in Knowledge Representation Learning,” Computer Research and Development, Vol 53, pp. 247-261, 2016.

[22] K. Leetaru and P.A. Schrodt, “GDELT: Global data on events, location, and tone, 1979–2012, ” ISA annual convention, Vol. 2, No. 4, pp. 1-49, 2013.

[23] Z. Han, Y. Ma, Y. Wang, S. Gunnemann, and V. Tresp, “Graph Hawkes Neural Network for Forecasting on Temporal Knowledge Graphs,” Automated Knowledge Base Construction, 2020.

[24] P. Jain, S. Rathi, and S. Chakrabari, “Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols,” Conference on Empirical Methods in Natural Language Processing, pp. 3733-3747, 2020.