Enhance Temporal Knowledge Graph Completion via Time-aware Attention Graph Convolutional Network

Haohui Wei, Hong Huang(✉), Teng Zhang, Xuanhua Shi, and Hai Jin

National Engineering Research Center for Big Data Technology and System
Service Computing Technology and Systems Laboratory
Huazhong University of Science and Technology, China
{weihh77,honghuang,tengzhang,xhshi,hjin}@hust.edu.cn

Abstract. Previous works on knowledge graph representation learning focus on static knowledge graph and get fully developed. However, task on temporal knowledge graph is far from consummation because of its late start. Recent researches have shifted to the temporal knowledge graph relying on the extension of static ones. Most of these methods seek approaches to incorporate temporal information but neglect the potential adjacent impact merged in temporal knowledge graphs. Meanwhile, different temporal information of involved facts evoke impact with different extent on the concerned entity, which is always overlooked in the previous works. In our paper, we propose a Time-aware Attention Graph Convolutional Network, named *TAGCN*, for temporal knowledge graph completion. Entity completion can be turned into interactions between entity and associated neighborhood. We utilize a graph convolutional network with a novel temporal attention layer to obtain neighboring information at all timestamps to avoid diachronic sparsity. We conduct extensive experiments on various datasets to evaluate our model performance. The results illustrate that our model outperforms the state-of-the-art baselines on entity prediction.

Keywords: temporal knowledge graph · representation learning · graph neural networks.

1 Introduction

Temporal Knowledge Graph (TKG) stores structured data in quadruples to extract interactions between entities on specific timestamps to represent fact, which is the information of events that have occurred in the reality. Temporal information is a crucial element in the real world because some facts are only valid at some timestamps. Due to its capability to contain rich information, TKG benefits numerous downstream applications, like transaction recommendation, social relation inferring, and event process prediction, all of which are dependent on the quality of the knowledge graph. Nonetheless, TKG is inevitably incomplete and sparse due to corruption and some other irresistible factors, which will

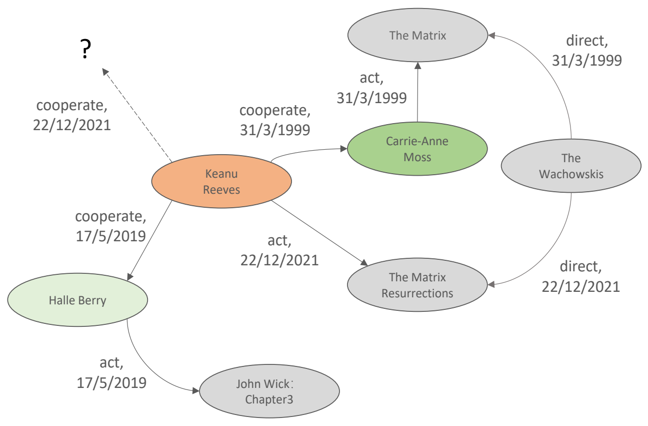


Fig. 1: A toy example for entity prediction in TKG, and the deep green candidate is the most probable entity for the query *(Keanu_Reeves, cooperate, ?, 22/12/2021)*.

undermines the performance of downstream tasks. Therefore, there is a growing need of valid approaches for completion in TKG.

Recently, more and more works pay attention to TKG completion. Majority of researches, like TTransE [1], TNTComplEx [2], neglect rich information that is inherent within the vicinity of entities and limit their semantic capturing because they expand to TKG domain through a straightforward extension from static methods, such as TransE [3], focusing on independent validation of each quadruple. Taking Fig. 1 as an example, our purpose is to predict who *Keanu Reeves* would *cooperate* with on *22/12/2021*. It is obvious that *Carrie-Anne Moss* and *Halle Berry* share the same entity interaction *cooperate* with *Keanu Reeves*. However, the neighborhood of *Carrie-Anne Moss* is quite different from that of *Halle Berry*. Characterizing this would provide abundant information when predicting the missing tail entity in *(Keanu_Reeves, cooperate, ?, 22/12/2021)*. To conclude, neighboring information has large impact on representation learning because learning quadruples independently would lose the huge amount of interactions among entities from the neighboring structure of TKG.

Furthermore, fact in TKG relies on long-term dependency which means fact with long interval may still determine the future prediction. Using the same query in Fig. 1, although *Halle Berry* has closer interactive behavior *cooperate* with *Keanu Reeves* than *Carrie-Anne Moss*, it is not definite that *Halle Berry* has a bigger chance to be the correct prediction in the query *(Keanu_Reeves, cooperate, ?, 22/12/2021)*. But due to the difficulty of simulating complex temporal influence in TKG, some of relevant works, like RE-NET [4] and RTFE [5], only pay more attention to recent fact.

Considering the aforementioned features, it is prominent to comprehend TKG both from the neighbor structure and the temporal dependency of TKG. Therefore, it evokes several challenges when tackling these features. First, *how to extract temporal neighboring information from the neighbor structure of TKG?* The significance of neighborhood encoding has been revealed in several static KG methods [6], but the extension to TKG is difficult due to the additional time dimension. Utilizing the Message Passing Process approaches, it is critical to integrate temporal information with interactions between relations and entities when encapsulating the neighborhood surrounding entities.

The second challenge is *how to encode time to maintain long-term dependency?* Encoding time by separating time snapshots and training through the temporal order may give rise to the same consequence of sequence-based models mentioned above. It is also difficult to comprehend the influence of all time on prediction by simply projecting time into the same space of entity and relation. Therefore, to distinguish different temporal influence on a certain timestamp, it is needed to implement a specific approach to learn the dependency between temporal information and the certain timestamp of the concerned query.

To this end, we propose our *Time-aware Attention Graph Convolutional Network* (TAGCN). We decouple temporal knowledge graph completion into two phase, neighboring temporal message aggregation and entity temporal focus attachment. To alleviate the above two challenges, TAGCN is used to encode neighboring information to contextualize the representation of entities. Inspired by self-attention mechanism [7], we devise a novel *temporal self-attention* (TeA) layer to locally extract the temporal influence between timestamps and involved fact on concerned query. The *temporal message aggregation* (TMA) module is used to extract neighborhood structure in TKG. Moreover, a time-aware decoder is applied and uses a simple way to activate different attention to neighboring impact regarding to the time of query. Our contributions can be summarized as follows:

1. We propose TAGCN, which introduces a knowledge graph convolutional network to learn the representation of entity capturing temporal dependency and complex interactions between entities and temporal facts. We decouple temporal knowledge graph completion into two phases, neighboring temporal message aggregation and entity temporal focus attachment.
2. Our work initiates a well-designed temporal self-attention layer, which is leveraged to encode locally temporal impact between target entity and involved facts. Thus, enhancing the estimation of the temporal influence of neighborhood.
3. We conduct extensive experiments on several real-world datasets. Experimental results show that the proposed method has achieved the state-of-the-art results in the task of entity prediction for TKG.

2 Related Works

2.1 Static KG Completion

There have been a large number of researches giving insight into static knowledge graph completion. Traditional KG completion methods map entities and relations into a low-dimensional vector with a score function measuring the possibility of the candidates. They can be classified into three categories in general: translational models, factorization based models, and *convolutional neural network* (CNN) models. TransE [3] is the most well-known method usikkng translation to embed entities and reltions. Following TransE, several methods [8–10] using different mapping methodology come out in succession and achieve better results. Rescal [11] and DistMult [12] are two factorization based models. Simultaneously, ComplEx [13] also projects entities and relations into different spaces using the above-mentioned evaluation function. In this way, these models can further develop the expressiveness beyond the limitation of Euclidean space and learn more complex interaction between entities and relations. Besides these works, ConvE [14] applies convolutional filters to process the vector of entity and relation. Its success ignites further application of other neural networks [15]. However, completion task on TKG reaps poor effect with static methods because of the lack of temporal information processing.

2.2 Temporal KG Completion

Lots of previous works on TKG mainly pay attention to the independent validation of quadruples and lay more focus on the incorporation of static methods and temporal information. The main distinction lies in the representation of timestamps. As mentioned in the above section, TTransE [1] views time as a new element and adds timestamp embedding into the relation embedding to fulfill the conventional score function TransE [3]. HyTE [16] adopts the idea of TransH [8], viewing different hyperplanes as different temporal spaces. Then it projects entities and relations to these hyperplanes and uses static score function to measure the possibility. TA-DistMult [17] learns time embedding by encoding the timestamp string sequence while DE-SimplE [18] leverages diachronic embedding by concatenating it with enduring one. TNTComplEx [2] makes a decomposition of 4 tensors in complex domain, adding the timestamp representation compared with ComplEx [13]. Although these methods extend to TKG successfully, none of them considers the neighborhood information within knowledge graph. RTFE [5] proposes a new training framework to enhance the further boost of conventional methods. T-GAP [19] adopts query-relevant temporal displacement to process the whole TKG, but the time overhead is too high to follow.

3 Proposed Model

3.1 Problem Definition

Temporal knowledge graph $\mathcal{G}$ is composed of quadruples $\mathcal{G} = \{(h, r, t, \tau) \mid h, t \in \mathcal{E}, r \in \mathcal{R}, \tau \in \mathcal{T}\}$, where h refers to head entity, r denotes the relation, t is

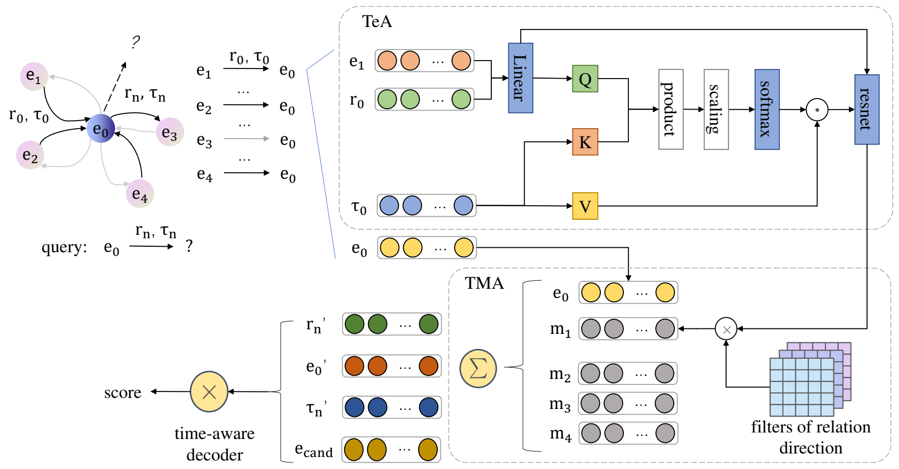


Fig. 2: Overview of our work. Arrows in gray refer to inverse directions, and self loop is omitted for clarity. All entities in $\mathcal{E}$ are considered as candidates.

the tail entity, and τ is the timestamp. $\mathcal{E}$, $\mathcal{R}$ represent the entity set and relation set contained in $\mathcal{G}$, and $\mathcal{T}$ stands for the known timestamps. To distinguish different concept, we use lower-case letters to represent object in dataset, e to represent specific entity in TKG, such as $e_1, e_2 \in \mathcal{E}$ in Fig. 2, $\boldsymbol{z}$ to represent embedding of each object, like $\boldsymbol{z}_r, \boldsymbol{z}_h \in \mathbb{R}^d$ where d is the embedding dimension, and $\boldsymbol{z}_x^\tau$ to represent entity embedding $\boldsymbol{z}_x$ under timestamp τ.

The aim of TKGC is to find the missing entity of an incomplete query. Given a query lacking in tail entity, such as $(h, r, ?, \tau)$ where τ is within the observed set $\mathcal{T}$, we hope to learn a mapping function f_{map}: $e \to \mathbb{R}^d$, where $e \in \mathcal{E}$ and d represents the dimension of embedding $d \ll |\mathcal{E}|$ and a score function f_{score} to infer the most probable $t \in \mathcal{E}$ to fulfill the missing component in the query based on the known information of $\mathcal{G}$. What's more, the mapping function f_{map} not only needs to consider the temporal information of each fact but also consolidate neighbor structure.

3.2 Model Overview

In this section, we describe our proposed model TAGCN, which can simultaneously extracts structural information and temporal dependency. Our model is in the architecture of encoder-decoder, and the framework is shown in Fig. 2. Quadruple sets in TKG are mapped into low-dimensional spaces at the beginning. Then, two stages are entailed for entity prediction. With the novel *temporal attention layer* (TeA), we capture the temporal dependency of the neighboring fact locally and dispose with *temporal message aggregation* (TMA) to increase the expressiveness of entities. Last, the time-aware decoder integrates temporal information to model the temporal impact on entity and gives the probability of candidates.

3.3 TAGCN

Preprocess To best improve the connectivity and information transmission efficiency, we allow information in TKG flows along three directions: original, inverse, and self-loop. Among which, self-loop $loop_e$ is constructed for each entity $e \in \mathcal{E}$, quadruple with self-loop is extended with a self-loop relevant timestamp τ_{loop}. Therefore, TKG $\mathcal{G}$ grows into $\mathcal{G}'$,

$$\mathcal{G}' = \mathcal{G} \cup \{(t, r^{-1}, h, \tau) \mid (h, r, t, \tau) \in \mathcal{G}\} \cup \{(e, loop_e, e, \tau_{loop}) \mid e \in \mathcal{E}\}. \tag{1}$$

$\mathcal{R}$ and $\mathcal{T}$ are also extended to $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}_{inv} \cup \{loop_e\}$, $\mathcal{T}' = \mathcal{T} \cup \{\tau_{loop}\}$. Meanwhile, since $d(rel)$ short for the direction of relations is divided into three types, we adopt three different filters, and the relational direction filter is defined as follows:

$$\mathbf{W}_{d(rel)} = \begin{cases} \mathbf{W}_{ori} & rel \in \mathcal{R}, \\ \mathbf{W}_{inv} & rel \in \mathcal{R}_{inv}, \\ \mathbf{W}_{loop} & rel \in \{loop_e\}. \end{cases} \tag{2}$$

Temporal Attention Layer Attention is of great importance in nowadays researches. In order to encode temporal dependency between entity and fact, we utilize self-attention layer for better usage of adequate information among available edge attributes. We compute the implicit attention score when an entity concerns its surrounding neighborhood. We treat neighboring messages as two part for decoupling the structural and temporal information. For a corresponding fact (h, r, t, τ), we use a linear layer to combine the representations of head entity and relation for semantic information $\boldsymbol{m}_{h,r}^{s}$ of the fact:

$$\boldsymbol{m}_{h,r}^{s} = \mathbf{W}_s([\boldsymbol{z}_h | \boldsymbol{z}_r]), \tag{3}$$

where $[\cdot|\cdot]$ denotes concatenate operation, and $\mathbf{W}_s$ is to project the embedding size to the standard dimension space.

Then we use two weight matrices to get the query and key of $\boldsymbol{m}^s$ and $\boldsymbol{z}_\tau$. The intermediate representation $\hat{\boldsymbol{m}}$ refers to the combination of temporal information τ and semantic information of the fact, which could be explored in the following step. Following previous work [7], the scaling operation is employed to alleviate the over inflation of dot products, thus avoiding the extremely small gradient:

$$\hat{\boldsymbol{m}}_{h,r}^{\tau} = \frac{\mathbf{W}_K(\boldsymbol{z}_\tau) \otimes \mathbf{W}_Q(\boldsymbol{m}_{h,r}^{s})}{\sqrt{d}}, \tag{4}$$

where $\mathbf{W}_K$, $\mathbf{W}_Q$ are weight matrices. $\hat{\boldsymbol{m}}_{h,r}^{\tau}$ is an implicit temporal representation of neighborhood. Thus we get the temporal fact attention generated from temporal and semantic representation after softmax. To strengthen the potential impact of fact on specific timestamp, we model the temporal attention by using temporal information and the fact temporal representation:

$$\boldsymbol{m}_{h,r}^{\tau} = \mathrm{softmax}(\hat{\boldsymbol{m}}_{h,r}^{\tau})\mathbf{W}_V(\boldsymbol{z}_\tau). \tag{5}$$

Up to now, we obtain the temporal $\boldsymbol{m}_{h,r}^{\tau}$ information of neighboring fact. To enhance the semantic influence, we utilize the residual network for numerical stability. The neighboring message of the certain fact is formulated as follows:

$$\boldsymbol{m}_f = \mathrm{FCN}(\boldsymbol{m}_{h,r}^{\tau} + \boldsymbol{m}_{h,r}^{s}), \tag{6}$$

where FCN denotes the fully connected net with norm and dropout layer to enhance the generalization. We also use a residual layer to maintain layer wise fact information to deal with some long maintaining facts.

Temporal Message Aggregation Module With the process of TeA layer, we obtain the input message by modeling the temporal dependency with semantic and temporal information when concerning the central entity e. To improve the representation, we adopt the aggregation and take the relation direction into account. The new entity feature is computed by combining all incoming messages to e:

$$\boldsymbol{z}_e = \sum_{f \in \mathcal{N}_e} \mathbf{W}_{d(r)} \boldsymbol{m}_f, \tag{7}$$

where N_e is the neighboring facts with e as the tail entity, and the facts are all in the original temporal knowledge graph $\mathcal{G}$.

After conducting TAGCN, we use the output as the ultimate representations of entities. These representations well integrate the temporal dependency on certain facts with interactions between entities and the involved facts from the neighboring structure of the knowledge graph.

3.4 Time-aware Decoder

Using the encoder, we obtain representations with perception of the whole TKG. When considering one query, the temporal effect on entity should be activated in decoder for better comprehension. To implement conventional decoders, we combine the entity and timestamp embedding with weight matrix $\mathbf{W} \in \mathbb{R}^{2d \times d}$, which enables the entity embedding to comprehend diversity of fact influence on different timestamps:

$$\boldsymbol{z}_h^{\tau} = \mathbf{W}[\boldsymbol{z}_h | \boldsymbol{z}_\tau], \tag{8}$$

where $\boldsymbol{z}_h$ and $\boldsymbol{z}_t$ are the embeddings of head entity and timestamp selected from embedding matrix according to the query.

In our work, we choose ConvE [14] as decoder to estimate probability for quadruples. ConvE employs convolutional and fully-connected layers to model the interactions between entities and relations input. It stacks the embeddings of head entity and relation, uses convolution operation to acquire the score of quadruple as follows:

$$\boldsymbol{p}_{(h,r,t,\tau)} = \mathrm{ReLU}(vec(\mathrm{ReLU}([\boldsymbol{z}_h^\tau|\boldsymbol{z}_r] * \omega))\mathbf{W})\mathbf{E}^{\mathrm{T}}, \tag{9}$$

where vec$(\cdot)$ represents flattening tensor into vector, $*$ is the convolution operation, ω is the convolutional filter, and $\mathbf{W}$ denotes a parameter matrix to project the flattened result to the embedding dimension. $\mathbf{E}$ is the entity embedding matrix without temporal embedding, so that all entities are treated as candidates.

The model is trained using standard cross entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i\in\mathcal{G}}(t_i \cdot \log(p_i) + (1-t_i)\log(1-p_i)), \tag{10}$$

where t_i is the gold label of fact i while p_i is the inferred probability. At last, we train our model using the optimizer Adam.

4 Experimental Setup

4.1 Datasets

In our work, we evaluate the proposed model on several public datasets for TKG completion, namely, ICEWS14, ICEWS05-15, YAGO11k, and Wikidata12k. iCEWS14 and ICEWS05-15 both come from the *Integrated Crisis Early Warning System* (ICEWS)[1]. ICEWS14 covers facts occurred in 2014, while ICEWS05-15 collects facts occurred between 2005 and 2015. ICEWS datasets are stored in the form of (h, r, t, τ). Wikidata12k and YAGO11k are subsets of Wikidata[2] and YAGO[3], formatted as $(h, r, t, \tau_{start}, \tau_{end})$. Following the same data splitting strategy in HyTE [16], we discretize them into each snapshot. The details of above-mentioned datasets are listed in Table 1.

4.2 Evaluation Metrics

Generated from reality, facts may evolve over time. Two quadruples may appear sharing same factors, like (h, r, t, τ) and (h, r, t', τ). However, results can be flawed once one quadruple end up with testing ones, while the other is from the training set. To avoid the misleading of these corrupted facts, we remove from the dataset the corrupted facts, corresponding to the former work [18]. Thus all metrics in our work are filtered ones. *Mean reciprocal rank* (MRR), Hits@1, Hits@3, and Hits@10 are formally used in knowledge graph entity prediction to compare the performance against other baselines. In our work, we evaluate

[1] https://dataverse.harvard.edu/dataverse/icews
[2] https://www.wikidata.org
[3] https://yago-knowledge.org/

Table 1: Statistics of datasets. $|\mathcal{E}|, |\mathcal{R}|$, and $|\mathcal{T}|$ are the total number of entities, relations, and timestamps. Meanwhile, #train, #test, and #valid refer to the quadruple numbers of train, test, and valid set respectively.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	$\|\mathcal{T}\|$	#train	#test	#valid	granularity
ICEWS14	7,128	230	365	72,826	8,941	8,963	1 day
ICEWS05-15	10,488	251	4,017	368,962	46,275	46,092	1 day
YAGO11k	10,623	10	60	203,858	21,763	21,159	1 year
Wikidata12k	12,254	24	77	239,928	18,633	17,616	1 year

TAGCN in tail and head entity prediction respectively. The test set is represented by s_{test}, MRR is defined as $\mathrm{MRR_x} = \frac{1}{|\mathrm{s_{test}}|} \sum_{(\mathrm{h,r,t,}\tau) \in \mathrm{s_{test}}} \frac{1}{\mathrm{rank_x}}$, where $x \in \{h, t\}$, and we calculate both rankings of head entity $rank_h$ and tail entity $rank_t$, consisting with the evaluation method of previous work. Meanwhile, Hits@n, $n = 1, 3, 10$, is defined as $\mathrm{Hits@n_x} = \frac{1}{|\mathrm{s_{test}}|} \sum_{(\mathrm{h,r,t,}\tau) \in \mathrm{s_{test}}} \mathbb{I}(\mathrm{rank_x} \leq \mathrm{n})$, where $\mathbb{I}(\cdot)$ is an indicator function equaling to one if the condition holds, and zero otherwise.

4.3 Baselines

To show the competitiveness of our model, we make a comparison with numbers of temporal and static KG completion models. In order to meet the requirements of static models, we ignore the time information when training these models. T-GAP [19] is not set as our baseline because of the high time overhead, and it only uses tail prediction as its results in the paper, while the tail prediction performance is usually better than the head one. Sequence-based models like RE-NET [4] and RE-GCN [20] are excluded because they are assigned for extrapolation task.

Static baselines are listed as follows:

- *TransE* [3]. In TransE, entities and relations are mapped into the same embedding space, and use translation method to infer the tuples.
- *DistMult* [12]. With the same projection as TransE, this work uses factorization method to calculate the score of each tuple.
- *ComplEx* [13]. ComplEx proposes a method based on complex representation. It divides the embeddings of entities and relations into real and imaginary parts. Lastly it adopts a complex multiplication, and maintains the real part as the final score.
- *ConvE* [14]. This method construct the score function with the same projection as TransE, and uses convolution operation to obtain the interactions between entities and relations.

Temporal baselines are listed as follows:

- *TTransE* [1]. TTransE is an extension of TransE with an additional dimension of timestamp, it uses a linear add operation to encode the temporal information into relation.

- *HyTE* [16]. Using the same projection as TTransE, but HyTE projects all entities and relations to the space of timestamps by linear transition associated with timestamps.
- *TA-DistMult* [17]. TA-DistMult treats timestamp as a string, and maps each character to a vector. When dealing with a tuple, the temporal relation embedding $z_{r,\tau}$ is generated by feeding r and characters in τ as a sequence into a LSTM to encode temporal information.
- *DE-SimplE* [18]. Focusing on temporal encoding, DE-SimplE puts forward a novel entity embedding function by considering that entities are combined with temporal and static features, using diachronic embeddings to estimate the probability of the incomplete tuples.
- *TNTComplEx* [2]. This work focuses on the integration of timestamp with ComplEx, and projects timestamps the same way as other objects in ComplEx. To be more clear, TNTComplEx embeds temporal features through linear representation in complex space.
- *RTFE* [5]. RTFE is a framework to enhance the performance of existing methods. It chooses a method $\hat{f}$, and trains along time after pretraining with setting the embedding of former snapshot $\mathcal{G}_{n-1}$ as the initial embedding of the latter one $\mathcal{G}_n$, where snapshots are separated on account of timestamps.

4.4 Implementation Details

We conduct all experiments of our model and baselines using Pytorch on a Intel(R) Xeon(R) Gold 5117 CPU, Tesla V100 GPUs, and 250GB Memory server. The software of experiment environment is Ubuntu 18.04 with CUDA 11.4. We evaluate our model with setting the learning rate of Adam as 0.001, batch size as 512 in ICEWS05-15 and 128 in other datasets. Moreover, the initiate embedding dimension is set as 100 for both entities and timestamps while output dimension is set as 200, label smooth in two datasets of ICEWS is set as 0 and 0.01 in the other datasets. We use the released code of baselines and the parameters in baselines are set as their default settings.

4.5 Results and Comparison

In this part, we show our performance against other baselines and make an analysis on the results.

Table 2 demonstrates entity prediction results comparing with other models on the popular datasets for TKG completion task. Different from previous works, we use respective head and tail entity prediction to evaluate for rigorous comparison. For saving space, H@1, H@3, and H@10 are used to replace Hits@1, Hits@3, Hits@10 in this section. Our work shows outstanding performance against other baselines on ICEWS14 and ICEWS05-15. TAGCN delivers an increment of 2% on MRR on these two datasets. Although we still observe that TAGCN does not always achieve the best results from Table 2 on YAGO11k, we achieve the best results in mean metrics. Further analysis shows that this attributes to the

Table 2: Entity prediction results on several popular datasets. The best results of each metric are in bold, and the scond ones are underlined. The percent sign is omitted for all data.

Metrics	MRR		H@1		H@3		H@10		MRR		H@1		H@3		H@10	
	t	h	t	h	t	h	t	h	t	h	t	h	t	h	tail	head
	ICEWS14								**ICEWS05-15**							
TransE	33.4	29.4	17.2	12.0	45.3	38.1	67.2	61.0	34.2	30.8	17.5	13.3	45.7	40.3	68.4	62.6
DistMult	50.7	37.1	36.4	28.6	52.1	46.5	73.1	59.9	47.1	44.1	36.9	30.7	55.1	47.9	71.9	66.3
ComplEx	50.7	37.7	42.4	37.6	45.7	40.1	68.9	64.1	49.5	42.9	36.8	32.2	54.9	49.7	72.7	66.1
ConvE	51.2	40.8	37.2	31.0	54.2	49.4	74.1	65.7	48.9	44.5	37.3	31.1	56.2	49.6	73.1	68.1
TTransE	27.2	23.8	11.1	3.7	43.7	37.2	62.1	57.9	32.0	22.2	10.0	6.8	42.8	38.6	64.1	59.1
HyTE	30.5	25.7	12.8	7.3	45.1	36.7	65.4	63.5	33.9	28.1	15.2	8.1	46.9	40.5	70.5	65.8
TA-DistMult	49.1	46.3	38.4	34.2	46.7	40.9	71.1	66.1	50.2	44.6	38.3	30.9	50.1	44.9	75.8	69.8
DE-SimplE	52.1	47.5	40.1	37.3	59.7	52.9	75.5	66.9	54.3	48.7	43.2	37.8	60.3	56.5	74.2	69.8
TNTComplEx	59.1	53.9	54.1	40.1	66.1	57.7	78.1	69.9	62.1	58.3	54.7	45.9	66.8	59.2	77.3	70.7
RTFE	62.1	56.5	56.8	43.8	68.4	58.8	78.9	70.0	64.2	61.2	56.7	53.1	70.4	61.0	81.7	76.9
TAGCN	**63.7**	**59.1**	**58.3**	**49.9**	**70.4**	**61.8**	**80.1**	**71.9**	**66.3**	**62.0**	**57.2**	**53.4**	**71.2**	**65.7**	**85.1**	**77.3**
	Wikidata12k								**YAGO11k**							
TransE	21.1	17.3	12.4	8.2	21.2	17.0	42.4	25.2	14.2	6.4	1.9	0.7	18.2	9.2	36.2	13.4
DistMult	24.1	20.1	13.8	10.1	27.1	20.3	48.1	44.3	17.2	13.4	12.4	8.8	19.7	12.7	33.3	20.1
ComplEx	25.6	21.2	14.9	10.0	28.3	22.1	47.7	39.3	18.1	14.5	12.8	8.2	17.5	13.5	34.4	21.8
ConvE	24.3	20.7	14.8	10.6	25.9	18.7	42.1	39.7	15.8	11.8	10.1	7.3	15.4	11.6	33.2	17.4
TTransE	21.1	13.3	11.4	7.8	22.8	14.0	40.9	24.9	14.0	7.1	3.2	0.8	22.3	7.7	34.6	15.6
HyTE	21.2	15.1	11.5	7.7	24.1	13.5	40.9	27.1	14.0	6.5	2.8	0.7	21.5	8.0	35.2	11.7
TA-DistMult	24.5	18.9	14.1	10.3	25.7	20.7	47.1	42.7	18.4	14.2	13.1	9.5	22.9	10.5	39.7	19.3
DE-SimplE	31.3	19.7	17.9	11.7	27.4	22.8	55.6	42.6	17.3	13.5	10.1	6.5	31.9	14.0	33.8	19.6
TNTComplEx	45.1	26.7	28.7	24.3	57.2	30.0	66.9	49.6	31.7	18.3	21.9	11.8	34.5	16.3	50.9	26.8
RTFE	52.5	37.3	41.9	29.3	58.7	41.8	68.3	54.8	32.5	18.3	23.4	14.2	35.3	16.9	51.9	**27.6**
TAGCN	**53.4**	**37.3**	**43.2**	**29.5**	**59.5**	**41.9**	**70.3**	**55.4**	**33.4**	**18.5**	**23.7**	**14.3**	**35.7**	**17.1**	**53.1**	27.4

continuity of these two datasets. YAGO11k contains facts with their starting and ending time. This characteristic accommodates for the snapshot separation of RTFE. To be more specific, the continuity makes the timestamps between the time span have a better understanding of the maintaining facts. While our model only aggregate the temporal information after the preprocessing of temporal data, our training strategy brings out the fluctuations. To conclude, the properties of dataset have a large impact on the processing procedure.

Meanwhile, it is also interesting to find that methods for static knowledge graph are better than the previous temporal knowledge graph [1,16]. From our analysis, it is explicit that encoding temporal information into the same space of entities with linear function cannot help the integration. It is crucial to improve the expressiveness, and the temporal effect should be emphasized.

Table 3: Ablation experiments on ICEWS14. TAGCN is implemented with TeA layer, 1-layer GCN, and temporal modified decoder.

Method	MRR	H@1	H@3	H@10
TMA + time-aware decoder	56.0	43.1	61.3	72.6
Only time-aware decoder	54.8	42.8	60.7	71.3
TAGCN with conventional decoder	48.1	37.8	55.7	70.7
TAGCN	61.8	54.1	66.1	76.0

4.6 Ablation Study

To examine the effect of our model, we further conduct ablation study on each module. The results are displayed in Table 3. For saving place, we use mean metrics of head and tail entity predictions as the final results in the further experiments. Consisted of encoder-decoder framework shown in the Fig. 2, we assign the ablation study as follows:

Firstly, we hide the TeA layer and only use a linear function to integrate timestamp, relation, and head entity embeddings as the neighbor message to update all representations of entities, it is obvious that performance on ICEWS14 degrades about 10% in H@1. The result attests that our proposed temporal information integration method has a better comprehension on facts impact and encodes temporal dependency more effectively.

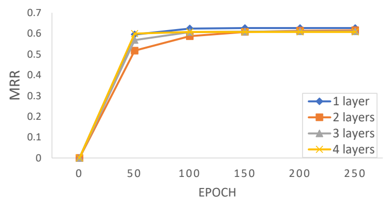
Secondly, we construct a model only utilizing the time-aware decoder to train for representation learning. From Table 3, we can find that the performance is behind the best performance of TAGCN, which denotes that the temporal encoder could definitely capture neighboring structure information and lift the expressive capability. Meanwhile, the accuracy reduces slightly comparing with the former study, we can interpret that temporal dependency in TKG has greater impact than adjacent information.

Lastly, to analyse the importance of our decoder, we resort to a new structure of TAGCN with a conventional decoder. In other words, we use the original ConvE [14] as the decoder, thus overlooking the temporal impact on entity. The result in Tabel 3 implies that the modification on decoder helps the entity concentrates on the within-time facts impact and chooses the most appropriate candidate.

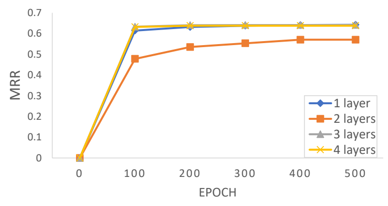
4.7 Parameter Analysis

Besides, to evaluate our model, we analyze important parameters in TAGCN: number of aggregation layers l, the embedding dimension d, and temporal integration variant operation ope. We conduct these experiments on ICEWS14 and ICEWS05-15.

Number of Layers l. Considering multi-hop neighbors could enhance full graph perception and increase the global understanding. The results are shown



(a)



(b)

Fig. 3: Experimental results of influence with layer numbers, we conduct this experiment on ICEWS14(a) and ICEWS05-15(b). The X-axis represents epochs, while the Y-axis is the MRR of entity prediction on each dataset.

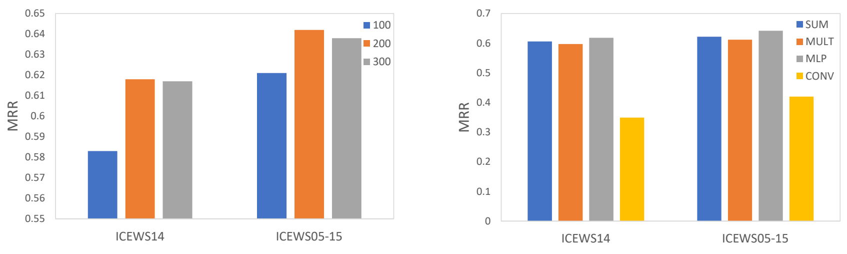
in Fig. 3. We show how the layer of aggregation can influence TAGCN. Experimental results in Fig. 3 indicate that training with two layers undermines the accuracy in both datasets. More training layers only bring little gain at the cost of time overhead and mainly accelerate the convergence in the early stage of the model.

The Embedding Dimension *d*. It is obvious that the expressiveness is related to the embedding dimension d. As shown in Fig. 4(a), TAGCN does not need too little or too superfluous embedding dimension to capture the features in TKG. Therefore, the proper size of embedding dimension is around 200.

Temporal Integration Operation *ope*. Further, we evaluate the effectiveness of TAGCN with different operations, such as linear calculations and convolution operation, to integrate timestamp and entity embeddings in the decoder. The experimental results are listed in Fig. 4(b), Sum and $Mult$ have better performance than $Conv$, which denotes that too complex approach will sabotage the representation and integration. To balance the performance and the complexity, we use MLP as temporal integration operation.

4.8 Further Analysis

As mentioned in the former section, we hypothesize that the temporal change in the relation is too small. Considering the example mentioned in Section 1, the meaning and usage of relation *cooperate* has been maintained for a long time. However, the characteristics of figures change with time. It is obvious that *Keanu Reeves* was the actor in different movie in *31/3/1999* and *17/5/2019*. So the temporal change in entity is explicitly more significant than that in relation. To



(a) (b)

Fig. 4: Comparison of different parameters: the embedding dimension(a), temporal integration operation(b). The X-axis represents different datasets, while the Y-axis is the MRR of entity prediction.

Table 4: Further analysis of temporal embedding

Method	MRR	H@1	H@3	H@10
Temporal embedding on both relations and entities	60.0	52.1	64.3	74.6
Temporal embedding on relations only	59.8	51.8	64.7	73.3
Temporal embedding on entities only	61.8	54.1	66.1	76.0

verify this point, we employ the temporal embedding in three approaches respectively to compare the accuracy. This experiment is conducted on ICEWS14. The result is shown in Table 4.

From Table 4, we can confirm that decoder with only temporal embedding on entities outperforms the other approaches. Temporal embedding only on entities is 2% ahead of that on relations. Temporal embeddings on both entities and relations is 1.8% behind because the temporal changes in relations mislead the representation learning for entity prediction. This experiment corroborates the hypothesis we suggest: entity is more sensitive to temporal impact while relation may evolve at a very low rate, and forcing relations to give reflect to temporal information will only degrade the performance. Thus only using static representations to model relations is more sufficient.

4.9 Time Prediction

The visualization of time prediction on three facts is shown in Table 5. For all timestamps in the time set $\mathcal{T}$, we calculate the probability of the object entity among all candidates. For a more intuitive comparison, we reserve the non-zero position in the same order of magnitude. We only pick the top three scores for display to save place.

From Table 5, it is explicit that the true timestamp has the highest score in the second fact and among the top three scores in the other two facts.

Table 5: Case study of three time prediction on ICEWS14

Gold Facts	Timestamps	Scores
(Merkel,Intent_to_cooperate,Obama,19/03/2014)	03/07/2014	5.1949
	19/03/2014	4.6959
	09/08/2014	3.6903
(Portugal,Consult,European_Central_Bank,24/10/2014)	24/10/2014	3.9880
	21/10/2014	1.7212
	21/02/2014	1.0325
(Juan_Carlos_I,Make_statement,Felipe_de_Borbon,03/06/2014)	03/06/2014	3.6341
	03/06/2014	2.5742
	11/02/2014	1.3227

We collate the dataset and find that in ICEWS14, some facts appear in different timestamps, which is consistent with our results. For example, the fact $(Angela\_Merkel, Express\_intent\_to\_cooperate, Barack\_Obama)$ occurred three times corresponding to listed timestamps in ICEWS14 and same for the other facts. So our model can obviously focus on the appropriate timestamps and improve the decoder performance.

5 Conclusion

In this paper, we propose a time-aware attention graph convolutional network TAGCN for link prediction in TKG. Inspired by self-attention layer in Transformer, we bring out a novel message generator for neighboring temporal message. To accomplish entity prediction in temporal knowledge graph, we decouple this task into two phases, using the encoder to aggregate neighboring semantic and temporal information, and acquire different temporal impact in decoding phase on account of the query. The proposed TeA layer is used to capture the neighboring information of all involved facts when considering the central entity. We conduct abundant experiments on real-world datasets, and the results show that TAGCN achieves best performance. In the future, we will investigate the better approach to encode potential temporal information from a novel angle.

Acknowledgements

The work is supported by the National Key Research and Development Program of China (Grant No. 2020AAA0108501) and the National Natural Science Foundation of China (Grant No. 62172174, 62127808).

References

1. T. Jiang, T. Liu, T. Ge, L. Sha, B. Chang, S. Li, and Z. Sui: Towards time-aware knowledge graph completion. In Proceedings of COLING, pp.1715–1724 (2016)

2. T. Lacroix, G. Obozinski, and N. Usunier: Tensor decompositions for temporal knowledge base completion. In Proceedings of ICLR (2020)
3. A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko: Translating embeddings for modeling multi-relational data. In Proceedings of NeurIPS (2013)
4. W. Jin, M. Qu, X. Jin, and X. Ren: Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In Proceedings of EMNLP, pp.6669–6683 (2020)
5. Y. Xu, H. E, M. Song, W. Song, X. Lv, H. Wang, and J. Yang: RTFE: A recursive temporal fact embedding framework for temporal knowledge graph completion. In Proceedings of NAACL-HLT, pp.5671–5681 (2021)
6. D. Nathani, J. Chauhan, C. Sharma, and M. Kaul: Learning attention-based embeddings for relation prediction in knowledge graphs. In Proceedings of ACL, pp.4710–4723 (2019)
7. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin: Attention is all you need. In Proceedings of NeurIPS (2017)
8. Z. Wang, J. Zhang, J. Feng, and Z. Chen: Knowledge graph embedding by translating on hyperplanes. In Proceedings of AAAI, pp.1112–1119 (2014)
9. G. Ji, S. He, L. Xu, K. Liu, and J. Zhao: Knowledge graph embedding via dynamic mapping matrix. In Proceedings of ACL, pp.687–696 (2015)
10. Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu: Learning entity and relation embeddings for knowledge graph completion. In Proceedings of AAAI, pp.2181–2187 (2015)
11. M. Nickel, V. Tresp, and H.P. Kriegel: A three-way model for collective learning on multi-relational data. In Proceedings of ICLR, pp.809–816 (2011)
12. B. Yang, W.T. Yih, X. He, J. Gao, and L. Deng: Embedding entities and relations for learning and inference in knowledge bases. In Proceedings of ICLR (2015)
13. T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard: Complex embeddings for simple link prediction. In Proceedings of ICML, pp.2071–2080 (2016)
14. T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel: Convolutional 2d knowledge graph embeddings. In Proceedings of AAAI, pp.1811–1818 (2018)
15. S. Vashishth, S. Sanyal, V. Nitin, N. Agrawal, and P. Talukdar: Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In Proceedings of AAAI, pp.3009–3016 (2020)
16. S. Dasgupta, S. Ray, and P. Talukdar: Hyte: Hyperplane-based temporally aware knowledge graph embedding. In Proceedings of EMNLP, pp.2001–2011 (2018).
17. A. García-Durán, S. Dumancic, and M. Niepert: Learning sequence encoders for temporal knowledge graph completion. In Proceedings of EMNLP, pp.4816–4821 (2018)
18. R. Goel, S. Kazemi, M. Brubaker, and P. Poupart: Diachronic embedding for temporal knowledge graph completion. In Proceedings of AAAI, pp.3988–3995 (2020)
19. J. Jung, J. Jung, and U. Kang: Learning to walk across time for interpretable temporal knowledge graph completion. In Proceedings of KDD (2021)
20. Z. Li, X. Jin, W. Li, S. Guan, J. Guo, H. Shen, Y. Wang, and X. Cheng: Temporal knowledge graph reasoning based on evolutional representation learning. In Proceedings of SIGIR (2021)