CRNet: Modeling Concurrent Events over Temporal Knowledge Graph

Shichao Wang[1,2], Xiangrui Cai[1,2(✉)], Ying Zhang[2,3], and Xiaojie Yuan[2,3]

[1] College of Cyber Science, Nankai University, Tianjin, China
wangshichao@dbis.nankai.edu.cn, caixr@nankai.edu.cn
[2] Tianjin Key Laboratory of Network and Data Security Technology, Tianjin, China
[3] College of Computer Science, Nankai University, Tianjin, China
{yingzhang,yuanxj}@nankai.edu.cn

Abstract. Temporal knowledge graph (TKG) reasoning, which aims to extrapolate missing facts in TKGs, is vital for many significant applications, such as event prediction. Previous studies have attempted to equip entities and relations with temporal information in historical timestamps and have achieved promising performance. While ignoring the likelihood that future occurrences would occur simultaneously, they independently forecast the missing data. However, there are complicated connections between future concurrent events that might correlate with and influence one another. Therefore, we propose our **C**oncurrent **R**easoning **Net**work (CRNet) to leverage event concurrency in both historical and future timestamps for TKG reasoning. Specifically, we select the top-k candidate events for each missing event and construct a candidate graph based on the candidate events of all missing events at the future timestamp. The candidate graph connects missing facts by sharing the same entities. Furthermore, we employ a novel relational graph attention network to represent the interactions of candidate events. We evaluate our proposal by the entity prediction task on three well-known public event-based TKG datasets. Extensive experimental results show that our CRNet complete future missing facts with a 15–20% improvement over MRR. (The source code is available at https://github.com/shichao-wang/CRNet-ISWC2022.)

Keywords: Temporal knowledge graph · Temporal reasoning · Concurrent events

1 Introduction

Each fact in the TKGs is a quadruple $(subject, relation, object, timestamp)$. Grouping quadruples by timestamps results in a sequence of KGs. Nodes represent entities in the real world, and the labeled edges represent related events between entities. TKG reasoning attempts to predict missing future facts like (s, r,?, t). Reasoning over TKGs forecasts emerging events, which is helpful for

U. Sattler et al. (Eds.): ISWC 2022, LNCS 13489, pp. 516–533, 2022.
https://doi.org/10.1007/978-3-031-19433-7_30

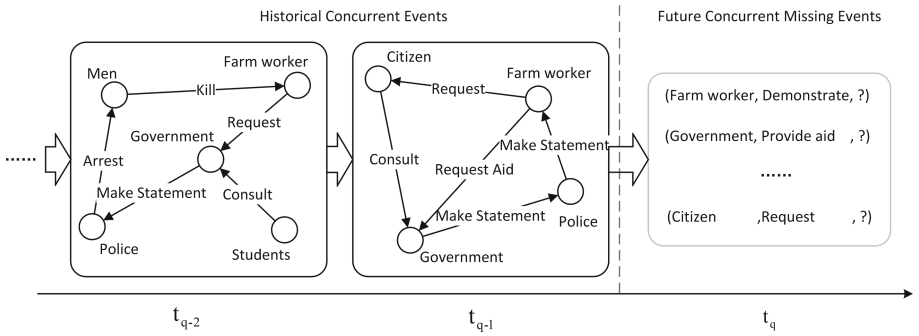

Fig. 1. An illustration of temporal reasoning over TKGs. The concurrent events exist in history and the future.

many real-world applications, including product recommendation [32] and event prediction [13,28].

On the one hand, historical events happened concurrently and are relevant to TKG reasoning. As shown in Fig. 1, for the missing events *(Farmworker, Demonstrate, ?)* a good TKG reasoning method should learn from previous concurrent events *(Government, Make statement, Police)* and *(Police Make statement, Farmworker)*. Previous studies, such as RE-Net [13] and CyGNet [40], attempt to retrieve query-related information from historical events. Some methods such as RE-GCN [19], CEN [17] and EvoKG [24], employ recurrent neural networks (RNNs) to learn a dynamic representation from historical KGs. These methods adopt relational graph convolution networks (RGCNs) to learn the concurrent events at historical timestamps. Limited by the traditional RGCN diagram, which regards the head and tail entities separately, they cannot exploit the complete semantics of event triplets or leverage the different importance of neighbors.

On the other hand, there are also complex dependencies among the concurrent events at future timestamps [21,39], which all the previous studies neglect. As shown in Fig. 1, the prediction results will influence each other. For the query *(Government, Provide aid, ?)*, a possible object would be *Farm worker*, since there are two events *(Farmworker, Request, Government)* and *(Farmworker, Request Aid, Government)* at the previous timestamps. However, when there is the event *(Farmworker, Demonstrate, Government)* at future timestamp. The government would not provide aid to farmworkers somehow, since they are antagonistic to each other. Thus, combining the concurrent events at future timestamp is suitable for real world application and enables predicting missing events.

In this paper, we propose the **C**oncurrent **R**esoning **Net**work (CRNet) for TKG reasoning, which exploits the concurrent events at historical and future timestamps. For the historical concurrent events, we develop a novel relational graph attention network, namely EventRGAT, which passes the complete event message, rather than nodes or edges, to neighbors and aggregates them adap-

tively. We also propose a two-stage framework to model the interactions among future concurrent events. Using true events at future timestamps directly will result in information leakage and is not suitable for real-world application, so we first collect the top-k candidates for every missing event at future timestamp and build the candidate events graph together. Nodes and edges in the candidate graph are the entities and the candidate events. Then, we employ our novel RGAT to encode the interaction among the candidate events and enhance the representations with concurrent dependencies. In summary, our contributions are in three folds:

- We formulate and address the problem of concurrent events for the TKG reasoning in historical and future timestamps, which is fit with the concurrent nature of events and suitable for real world application.
- For the historical concurrent events, we develop EventRGAT to aggregate related events adaptively. For future concurrent events, we propose a two-stage framework, which builds a candidate graph for concurrent missing events, to capture their interactions.
- Extensive experimental results demonstrate that our CRNet achieves significant improvement (15%–20% on MRR) on event-based TKG benchmarks. A thorough case study is carried out to verify the effectiveness of our proposal.

2 Related Works

This section first discusses two the difference between reasoning over temporal knowledge graph and the static knowledge graph. Then, we review the temporal knowledge graph reasoning under two different settings, e.g., interpolative and extrapolative.

2.1 Static Knowledge Graph Reasoning

The static knowledge graph reasoning aims to predict the missing facts in the KG. Recent researches focus on learning the low-dimensional representation for entity and relations in KGs to solve the problem. The representation learning methods can be categorized into translational and semantic-matching. The translational models, such as TransE [1] and its variants [20,29,34], measure the distance between the head and tail entities in the subspace translated by the relation. RESCAL [23], DistMult [37], NTN [26] and ConvE [5] are semantic matching methods, which measure the plausibility of facts by matching the semantics of entities and relations in the vector space. Graph neural networks (GNNs) have also extended for the relational-aware representation learning on KGs, such as R-GCN [30], HAN [33]. However, these methods are developed for static KGs, and they are not capable of modeling the dynamic evolutional patterns in TKGs directly.

Table 1. Important notations and their descriptions.

Notations	Description
G_t, $\mathcal{V}$, $\mathcal{R}$, $\mathcal{E}_t$	Event knowledge graph at timestamp t, and its node set, relation set and events set
$\boldsymbol{h}_i$ $\boldsymbol{h}_j$, $\boldsymbol{r}_k$	Embedding vector for Entity e_i, Entity e_j, and Relation r_k
$\boldsymbol{h}_t$	Embedding vector for Entity e and the matrix at timestamp t
$\boldsymbol{H}_t$, $\boldsymbol{R}$	Embedding matrix for entities at timestamp t and relations
s, r, o	The subject, relation, and object of a event
$\boldsymbol{s}_t$, $\boldsymbol{r}_t$, $\boldsymbol{o}_t$	The subject, relation, and object embedding vector at timestamp t
Q_t, Q_q	The missing events sets at timestamp t and q

2.2 Temporal Knowledge Graph Reasoning

There are two settings for reasoning over TKGs, interpolation, and extrapolation. The interpolative TKG reasoning task assumes that there are missing facts in the historical timestamps. It attempts to completing the missing facts through contextual KGs [6,7,11,12,14,35,36]. For example, Jiang et al. [12] adopt the temporal order of the happening time of facts to constrain the transformation between time-sensitive relations. TimePlex [11] embeds the entities, relations, and timestamps into a uniform compatible space. RTFE [36] treats the sequence of graphs as a Markov chain and tracks the state transition recursively. These methods cannot obtain the representations for entities and relations at future timestamps. Thus, they are not able to tackle the extrapolative TKG reasoning.

On the contrary, the extrapolative reasoning, which this paper focuses on, attempts to predict the facts at future timestamps through historical KGs. These methods can be categorized into two: Query-specific methods and evolution representation learning methods [17]. The query-specific methods retrieve contextual information from the question, such as subject and relation, from the historical KGs. For example, RE-Net [13] aggregates the historical neighbors for the queried subject and predicts its future interactions. CyGNet [40] utilizes the copy mechanism to collect the object distribution given a specific subject and relation. xERTE [8] build the sub-graph from the historical facts for the query. TITer [27] and CluSTeR [18] employ the reinforcement learning to find query-related paths. The evolution representation learning methods update the embedding for every entity and relation based on the historical KGs. RE-GCN [19] learns the evolution representation at a fixed length. CEN [17] extends it for the dynamic lengths. DynamicGCN [3] and Glean [4] enrich the representation with text features.

3 Problem Formulation and Notations

A temporal knowledge graph (TKG) $\mathcal{G} = \{G_1, G_2, \ldots, G_t, \ldots\}$ is a multi-relational directed graph. $G_t = (\mathcal{V}, \mathcal{R}, \mathcal{E}_t)$ denotes a set of events happened at time t, where $\mathcal{V}$ is the set of entities, $\mathcal{R}$ is the set of relations (a.k.a.

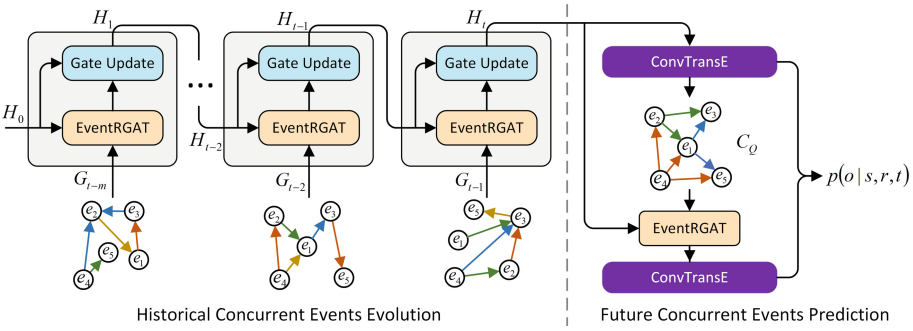

Fig. 2. The overview of our proposed CRNet model architecture. CRNet consists of two parts, e.g., the historical concurrent evolution module and the future concurrent prediction module. Edges in different colors denote different events between entities.

events) and $\mathcal{E}_t$ is the set of facts at timestamp t. A fact in TKG can be represented by a quadruple (s, r, o, t), where $s, o \in \mathcal{V}$, $r \in \mathcal{R}$ represent the subject, object, and relation respectively. The quadruple describes the subject s interacts with the object o with an event of r at time t. For every quadruple in the testing set, the extrapolative TKG reasoning task aims to complete the missing facts $(s, r, ?, t_q)$ and $(?, r, o, t_q)$ with a sequence of historical KGs $G_{q-m:q-1} = \{G_{q-m}, G_{q-m+1}, \ldots, G_{q-1}\}$. Taking the object prediction as an example, the conditional probability of an object given the subject s, relation r, and history $G_{q-m:q-1}$ is $p(o|s, r, G_{q-m:q-1})$. We denote it as $p_i(o|s, r, q)$ in the rest of paper.

In this paper, we conduct the concurrent reasoning over TKGs. Comparing with traditional TKG reasoning our concurrent reasoning diagram considers the concurrent missing events at future timestamp. We denote all the missing facts at future timestamp t_q as $Q_q = \{(s, r)|(s, r, o) \in G_q\}$. The conditional probability for object o given the subject s and relation r is $p(o|s, r, G_{q-m:q-1}, Q_q)$. We denote it as $p_c(o|s, r, q)$ in the rest of paper.

The important mathematical notations are described in Table 1

4 Methodology

This section introduces our proposal, CRNet. Figure 2 depicts the overview of our CRNet, which consists of the historical concurrent evolution module and the future concurrent prediction module. In the historical concurrent evolution module, the evolution embeddings for all entities are learned from historical KGs. In the future concurrent prediction module, we collect the missing facts to build a candidate graph and conduct concurrent prediction.

4.1 Historical Concurrent Events Evolution

To capture the concurrent interactions for entities, we use the historical KG G_t to update the entity embeddings. Give an entity representation $\boldsymbol{h}_i$ at timestamp

t, the adaptive triplet message passing module aims to collect the structural interactions $\boldsymbol{h}_i'$ for it. To obtain the triplet message, we perform a linear transformation over the concatenated triplet (e_i, r_k, e_j) embedding [22].

$$\boldsymbol{t}_{ijk} = \boldsymbol{W}_1 \left[\boldsymbol{h}_i || \boldsymbol{r}_k || \boldsymbol{h}_j\right] \tag{1}$$

where $\boldsymbol{t}_{ijk}$ is the representation for the event triplet (e_i, r_k, e_j). $[\cdot||\cdot]$ is the concatenation operation. $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are the embeddings for e_i and e_j, and $\boldsymbol{r}_k$ for the relation r_k respectively. $\boldsymbol{W}_1 \in \mathbb{R}^{h \times 3h}$ is the learnable parameter matrix. To learn the different importance α_{ijk} for message aggregation, we first adopt a linear transformation parameterized by a vector $\boldsymbol{w}_2$ followed by a LeackyReLU to compute the absolute score for every message, which is similar to the architecture proposed in GAT [31].

$$s_{ijk} = \text{LeackyReLU}\left(\boldsymbol{w}_2 \boldsymbol{t}_{ijk}\right) \tag{2}$$

To get the relative attention value for aggregation, we apply softmax over s_{ijk} shown in Eq. (2).

$$\begin{aligned} \alpha_{ijk} &= \text{softmax}_{jk}(s_{ijk}) \\ &= \frac{\exp\left(s_{ijk}\right)}{\sum_{n \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_{i,n}} \exp\left(s_{inr}\right)} \end{aligned} \tag{3}$$

where $\mathcal{N}_i$ is the neighborhood node set for entity e_i, $\mathcal{R}_{i,n}$ represents the connected relation sets for entity e_i and e_n. The neighbor message is finally adaptively aggregated following Eq. (4).

$$\boldsymbol{h}_i^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{R}_{i,j}} \alpha_{ijk}^l \boldsymbol{t}_{ijk}^l + \boldsymbol{W}_3^l \boldsymbol{h}_i^l\right) \tag{4}$$

where $\boldsymbol{h}_i^l$ is the embedding for e_i learned at l^{th} layer. σ is the RReLU [15] activation function. As suggested in GAT [31], we also employ the multi-head mechanism to collect multiple information from neighborhoods and stabilize the learning process. We employ M independent attention heads to calculate the embeddings from M different subspaces. We average the embeddings from subspaces resulting in the final representation. Note that, there are no parameters shared across heads or layers.

The final interaction information for entity e_i is the aggregated results after L layers, $\boldsymbol{h}_i' = \boldsymbol{h}_i^L$. We treat L and M as empirical hyper-parameters. We denote relational graph attention network above as $\boldsymbol{H}' = \text{EventRGAT}(G, \boldsymbol{H}, \boldsymbol{R})$, where $\boldsymbol{H}, \boldsymbol{R}$ is the embedding matrix for all entities and relations respectively. We will employ it to model the concurrent events again at the future timestamp.

Temporal Evolution. After gathering the interaction information in a specific timestamp t. We need to update the representation for the next timestamp.

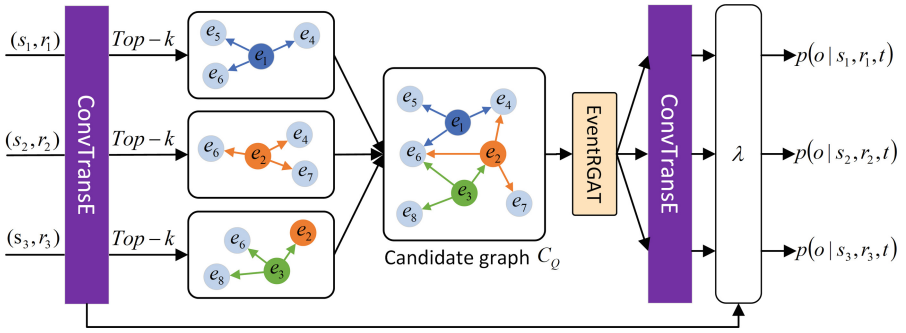

Fig. 3. The architecture of future concurrent events prediction module. There are three concurrent missing events to predict, namely (e_1, r_1), (e_2, r_2), (e_3, r_3). For each missing event, we select top-k (k = 3) candidates to build the candidate graph $\mathcal{C}_Q$. The final predict score is an average of p_c and p_i with the balance term λ.

We denote $\boldsymbol{h}_t$ and $\boldsymbol{h}'_t$ as the entity embedding and interaction information at timestamp t. We employ the gate mechanism to update the entity embedding.

$$\boldsymbol{u} = \sigma\left(\boldsymbol{W}_3\left[\boldsymbol{h}'_t || \boldsymbol{h}_t\right] + \boldsymbol{b}\right) \tag{5}$$

$$\boldsymbol{h}_{t+1} = \boldsymbol{u} \odot \boldsymbol{h}'_t + (\boldsymbol{1} - \boldsymbol{u}) \odot \boldsymbol{h}_t \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function which controls the gate value in vector $\boldsymbol{u} \in \mathbb{R}^d$ ranges 0 to 1. $\odot$ is the vector element-wise dot operation.

4.2 Future Concurrent Events Prediction

This subsection introduces our two-stage concurrent prediction framework Fig. 3. We first use ConvTransE [25] to predict all entities' probability score p_i.

$$p_i\left(o|s, r, t\right) = \boldsymbol{o}_t \text{ConvTransE}_1(\boldsymbol{s}_t, \boldsymbol{r}_t) \tag{7}$$

where $\boldsymbol{s}_t$, $\boldsymbol{r}_t$, $\boldsymbol{o}_t$ are the corresponding subject, relation and object embedding vectors at timestamp t respectively.

$$\text{ConvTransE}(\boldsymbol{s}, \boldsymbol{r}) = f\left(vec(\boldsymbol{M}(\boldsymbol{s}, \boldsymbol{r}))\boldsymbol{W}_4\right) \tag{8}$$

where $\boldsymbol{M}(\boldsymbol{s}, \boldsymbol{r})$ are aligned output vectors from the convolution kernels. $vec(\cdot)$ converts the feature map matrix into a vector. $f(\cdot)$ denotes the ReLU activation function here.

Candidate Graph Construction. Q_q is the set of concurrent missing events at future timestamp q. For every $(s_i, r_i) \in Q_q$, we select k candidate triplets with highest probability score $p_i(o|s, r, t)$. We then union all queries and their k candidates to build the candidate graph $\mathcal{C}_Q = (\mathcal{V}, \mathcal{R}, \mathcal{E}_Q)$, where $\mathcal{E}_Q$ is the candidate event triplet set for all queries. Thus, it results in $|Q_q| \times k$ edges in the candidate graph, where $|Q_q|$ is the number of missing facts.

Algorithm 1. Batch training procedure of CRNet

Input: Historical knowledge graph sequence $\{G_{t-m}, ..., G_{t-2}, G_{t-1}\}$,
Concurrent missing facts at future timestamp t Q_t
Output: Reasoning score for each query in Q_t

```
Generate Evolutional Embedding from historical KGs. Eq. (4) and (6).
for each (s, r) ∈ Q_t do
    Calculate the prediction score p_i(o|s, r, t) without concurrent context Eq. (7).
    Generate top-k candidate events. E^k_{s,r} with the k highest prediction score.
    Add E^k_{s,r} to candidate graph C_Q.
end for
Enrich entity embeddings with concurrent events based on C_Q Eq. (9).
Calculate the predict score with concurrent context p_c(o|s, r, t) Eq. (10).
Predict missing object by jointing two prediction scores. Eq. (11).
Update model parameters by minimizing cross-entropy loss. Eq. (12).
```

Concurrent Events Prediction. After the candidate graph construction, we employ our novel relational graph attention network EventRGAT to model the interactions among candidate events. The entity representation after future concurrent interactions $\hat{\boldsymbol{H}}$ follows:

$$\hat{\boldsymbol{H}}_t = \text{EventRGAT}(\mathcal{C}_Q, \boldsymbol{H}_t, \boldsymbol{R}) \tag{9}$$

The probability score for entities with concurrent events p_c can be calculated as follows:

$$p_c\left(o|s,r,t\right) = \hat{\boldsymbol{H}}_t \text{ConvTransE}_2(\hat{\boldsymbol{s}}_t, \boldsymbol{r}_t) \tag{10}$$

where $\hat{\boldsymbol{s}}$ is the enhanced entity embedding for subject s. The final probability score is a combination of p_i and p_c with a balance term λ.

$$p(o|s,r,t) = \lambda \cdot p_i(o|s,r,t) + (1-\lambda) \cdot p_c(o|s,r,t) \tag{11}$$

The entity prediction task can be seen as a multi-label classification problem. We employ the cross-entropy loss at future KG G_q:

$$\mathcal{L} = \sum_{(s,r,o) \in G_q} -\log p(o|s,r,t) \tag{12}$$

The training procedure for a batch of data is detailed in Algorithm 1. The training procedure will stop with the early stopping strategy with patience of 5.

5 Experiments

This section demonstrates the effectiveness of our proposal on the TKG reasoning. We first declare our experimental settings in detail, including datasets, baseline methods and evaluation metrics. Secondly, we compare the performance between CRNet and baseline methods on the link prediction and discussed the experimental results. After that, we analyze the influence of important hyperparameters in CRNet. Finally, we carry out a case study to explain the effectiveness intrinsically.

Table 2. Statistics of temporal knowledge graph (TKG) datasets.

Dataset	# Train	# Valid	# Test	# Nodes	# Relations	Granularity
ICEWS18	746,036	91,990	99,090	23,033	256	24 h
ICEWS14	74,845	8,514	7,371	6,869	230	24 h
GDELT	1,734,399	238,765	305,241	7,691	240	15 min

5.1 Experimental Setup

Datasets. We use three real-word event-based TKGs that have been widely used in previous studies: ICEWS18 [2], ICEWS14 [28] and GDELT [16]. Datasets are divided into training (80%), validation (10%) and testing (10%) sets by timestamps following [13]. ICEWS and GDELT are event-based TKGs. Detailed statistics of the aforementioned datasets are listed in Table 2.

Baselines. We compare our proposed method with the following state-of-the-art reasoning methods for temporal knowledge graphs, including

- RE-Net Jin et al. [13] propose an auto-regressive architecture for predicting future missing facts.
- xTERTE Han et al. [8] propose a temporal relational attention network and a reverse representation update strategy to guide the query-specific sub-graph extraction.
- CyGNet Zhu et al. [40] employ a time-aware copy-generation mechanism to identify facts with repetition.
- HIP He et al. [10] develop the historical information passing network to pass information from temporal, structural and repetitive perspectives.
- TANGO Han et al. [9] extends the idea of neural ordinary differential equations (ODEs). TANGO encodes both temporal and structural information into dynamic embeddings.
- TITer Sun et al. [27] define an abstract agent to search the answer from historical KGs. They also design a Dirichlet distribution-based time-shaped reward for reinforcement learning.
- CluSTeR Li et al. [18] propose a clue searching and temporal reasoning two-stage framework to predict future facts with reinforcement learning.
- RE-GCN Li et al. [19] employ a recurrent architecture to learn the evolutional representations of entities and relations following the KG sequence.
- EvoKG Park et al. [24] joint learns the time prediction task and link prediction task in an effective framework.
- CEN Li et al. [17] employ a length-aware decoder and the curriculum learning strategy to mine the complex evolutional pattern from length diversity and time-variability aspects.

Evaluation Metrics. We evaluate our model on TKG reasoning, which is a link prediction task at future timestamps. We adopt Mean Reciprocal Rank

(MRR), Hits@1, Hits@3 Hits@10 as our evaluation metrics. Note that, the same as previous works, we add reciprocal relation for every quadruple in the dataset, i.e., we add (o, r^{-1}, s, t) for every (s, r, o, t). For each quadruple (s, r, o, t) in the testing set, we predict two facts, e.g.$(s, r, ?, t)$ and $(o, r^{-1}, ?, t)$. We also employ the time-aware filtered setting which removes all the valid facts that appear in the ranking list of time-specific corrupted facts. Taking the query $(s, r, ?, t_1)$ with the answer o_1 and two ground truths (s, r, o_2, t_1), (s, r, o_3, t_2) as an example, under the time-aware setting, we consider the (s, r, o_2, t_1) as the corrupted fact and remove it from the ranking list.

5.2 Implementation Details

There are several empirical hyperparameters in our proposal. For all the entity and relation embeddings, their dimension d is set to 200. We also constrain the embedding vector with $L2$ normalization [38]. The number of layers of the relational graph attention network L is set to 2. The number of attention head M is set to 4. We fix the length of historical length m to 3 over all datasets. We adopt the Adam optimizer with 1e−3 learning rate and 1e−4 weight decay to optimize the model parameters. We employ the grid search algorithm to find the optimal number of candidate k and the balance term λ from the validation set according to MRR. The optimal k are 20,35,10 for ICEWS18, ICEWS14 and GDELT, respectively. The optimal balance term λ are 0.5,0.5,0.9 for ICEWS18, ICEWS14 and GDELT, respectively. We analyze their influence in Sect. 5.4. We use all the missing facts available to conduct the concurrent prediction. We also study its influence in Sect. 5.4.

5.3 Performance Comparison

Table 3 reports the entity prediction results of CRNet and baseline methods on the three event-based TKG datasets. The first group of baselines are query-specific methods, they search context for queries from historical timestamps. They fail to capture the global environment for event evolution, so they obtain a relatively poor performance. The second group consists of methods using reinforcement learning. They design an abstract agent to 'walk' through historical timestamps. The agent usually starts with a query, but 'walks' with a specific strategy, so they will not limit themselves by the query and obtain a better performance. However, reinforcement learning methods require a large number of computational resources and can not fit with large datasets, such as GDELT. The last group of baselines are evolution representation learning methods, which update entities or relations following historical timestamps. They learn entities' interactions from historical concurrent events but fail to capture the concurrent events at the future timestamp. As we can observe, our CRNet outperforms the baselines of all metrics on ICEWS18 and GDELT datasets and achieves an improvement of 14.62% and 19.57% on MRR, respectively. On the ICEWS14, CRNet obtains the best performance on most of the metrics except for Hits@10. CluSTeR searches explicit clues from historical KGs, but is unable to specify

Table 3. The performance of entity prediction with time-aware filtered metrics. Some methods do not report their performance under the time-aware filter setting, we use their public implementation to generate results and denote them with †.

	ICEWS18				ICEWS14				GDELT			
Method	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-Net	28.81	19.05	32.44	47.51	36.93	26.83	39.51	54.78	19.60	12.03	20.56	33.89
CyGNet	24.93	15.90	28.28	42.61	35.05	25.73	39.01	53.55	†18.79	†11.83	†19.84	†32.31
TANGO	28.97	19.51	32.61	47.51	26.25	17.30	29.07	44.18	–	–	–	–
HIP	†29.20	†20.12	†32.27	†47.60	†40.84	†31.60	†40.54	†56.02	†20.08	†12.78	†20.15	†33.62
xERTE	29.31	21.03	33.51	46.48	40.79	32.06	45.67	57.30	–	–	–	–
TITer	29.98	22.05	33.46	44.83	41.73	32.28	46.46	58.44	–	–	–	–
CluSTeR	32.30	20.60	–	55.90	46.00	33.80	–	**71.20**	18.30	11.60	–	31.90
EvoKG	29.28	–	33.94	50.09	27.18	–	30.84	47.76	19.28	–	20.55	34.44
RE-GCN	30.58	21.01	34.34	48.75	40.39	30.66	44.96	59.21	†19.72	†12.46	†20.99	†33.92
CEN	31.50	21.70	35.44	50.59	42.20	32.08	47.46	61.31	†21.16	†13.43	†22.71	†36.38
CRNet	**37.81**	**26.12**	**43.10**	**61.01**	**48.37**	**38.21**	**53.79**	67.79	**25.32**	**15.39**	**27.82**	**44.07**

the most significant clue, thus it achieves high Hits@10 but ordinary Hits@1 or MRR.

5.4 Ablation Studies

To investigate the influence of concurrent event prediction and verify the robustness of our proposal, we conduct several ablation studies for CRNet. We first analyze the influence of important hyperparameters in CRNet, e.g. k and λ. After that, we study the influence of the number of concurrent missing facts.

Influence of k Candidates. k is the number of candidate selected for each missing fact. Figure 4 demonstrates the influence of k ranges from 1 to 50. The metric values reported in the line chart are collected from validation set. As we can observe, for the k ranges from 1 to 10, the performance increase with higher k. The larger k results in more edges in the candidate graph and will have more interactions among candidates. On the other hand, the larger k will more likely to retrieve correct prediction and rank better. However, the larger k does not mean better performance. More candidate facts will lead to a more complex environment for concurrent prediction and decrease the predicting performance. Thus, every datasets have their own optimal k. We choose the optimal k based on the MRR, e.g., 20 for ICEWS18, 35 for ICEWS14 and 10 for GDELT. We think the optimal k is relevant to the scale of dataset, since the GDELT and the ICEWS14 are the largest and smallest dataset, respectively.

Influence of the Balance Term λ. λ is the balance term between $p_i(o|s,r,t)$ and $p_c(o|s,r,t)$. The larger λ lead our CRNet to predict missing facts more on concurrent context $p_c(o|s,r,t)$. We evaluate the effectiveness with λ in a range of 0.0, 0.1, 0.3, 0.5, 0.7, 0.9 and 1.0. $\lambda = 0.0$ and $\lambda = 1.0$ are two special cases, in which CRNet predict the missing facts purly by $p_i(o|s,r,t)$ or

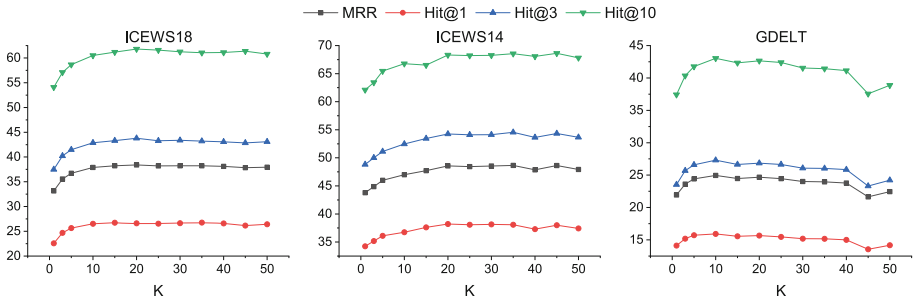

Fig. 4. The influence of the number of candidates k on three event-based TKG datasets. The black, red, blue, and green line represent MRR, Hits@1, Hits@3, and Hits@10. (Color figure online)

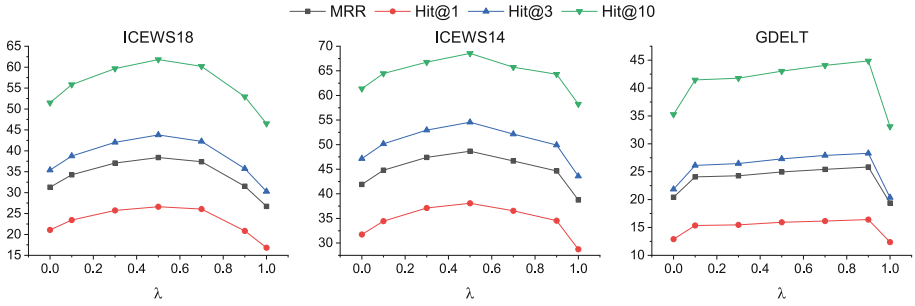

Fig. 5. Influence of balance term λ on three event-based datasets. The black, red, blue, and green line represent MRR, Hits@1, Hits@3, and Hits@10. (Color figure online)

$p_c(o|s,r,t)$. Figure 5 illustrates the influence of different λ. As we can observe from Fig. 5, ICEWS14, ICEWS18 and GDELT have their own optimal λ. Since the ICEW14 and ICEWS18 share similar collecting procedure, they have the same $\lambda = 0.5$. GDELT obtain the best performance with $\lambda = 0.9$. The metric values with $\lambda = 0.0$ and $\lambda = 1.0$ obtain a relatively poor performance comparing with any joint prediction model, which means our proposed concurrent context $p_c(o|s,r,t)$ complement with $p_i(o|s,r,t)$ well. However, the pure predict metrics of $p_c(o|s,r,t)$ are worse than $p_i(o|s,r,t)$. This means that our candidate graph not only create the interactions between future missing facts, but also introduce some distractive information. We leave this problem in our future work.

Influence of the Number of Concurrent Missing Facts. As we introduced in Sect. 4.2, we build our candidate sub-graph from the concurrent missing fact set Q_q. Therefore, the number of missing facts affects the scale of candidate sub-graph, and influence the performance further. Since the number of the concurrent missing facts varies from datasets and future timestamps, we split the missing facts into several partitions, i.g., 1, 2, 3, 4 and 5, to analysis how the number of

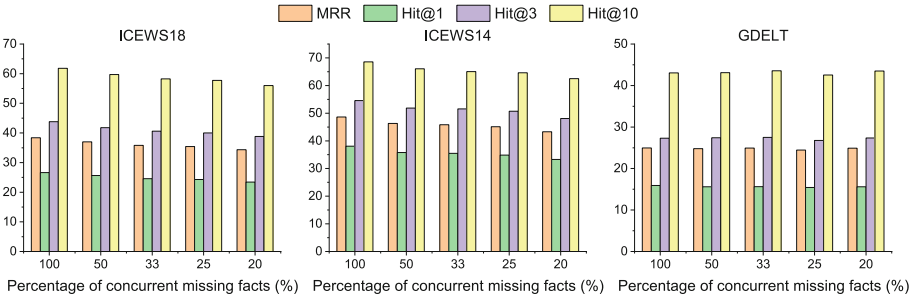

Fig. 6. MRR and Hit@3 performance with different ratio of concurrent missing facts at ICEWS18, ICEWS14 and GDELT datasets. All the metrics are obtained from validation set.

concurrent missing facts affect the performance. Figure 6 illustrates the performance difference from different ratio of concurrent missing facts. In ICEWS18 and ICEWS14 datasets, all metric values drop with smaller scale of concurrent missing facts significantly. On the contrary, the performance of different number of missing facts are almost the same and the best performance is obtained with 33% missing facts in GDELT dataset. This is because the GDELT has a relatively low performance and has more concurrent missing facts comparing with the other datasets. Thus, GDELT samples can not benefit from concurrent missing facts well.

5.5 Case Studies

To evaluate the effectiveness of concurrent missing events, we visualize 4 typical cases in the testing set of ICEWS14 in Fig. 7. More concretely, we group the missing events in the same topic and compare the prediction results of CRNet with RE-GCN, which can not leverage the concurrent context.

In case 1, there are two highly related missing events, *(China, Express intent to meet, ?)* and *(Japan Express intent to meet, ?)*. The traditional methods will easily predict the events with object *South Korea*, because there are two related events, such as *(South Korea, Make statement, China)* and *(South Korea, Make statement, Japan)* events in historical context. The *Express intent to meet* is seen as a evolution result of *Make statement*. However, the *Express intent to meet* event usually happens to each other. Our CRNet can discover the relationship between concurrent events *(China Express intent to meet, ?)* and *(Japan Express intent to meet, ?)*, and make correct predictions.

For case 2, there are two opposite events *Make visit* and *Host visit*, which usually happen concurrently at the same timestamp. In the historical context, *Envoy (US)* makes a visit to the *South Korea*, and the *South Korea* also host a visit for *Envoy (US)* as a response. When it comes to the missing events *(South Korea, Host visit, ?)* at a future timestamp, previous methods will fill the object with *Envoy (US)* according to the historical context. On the contrary,

Case 1

Historical context	Concurrent missing events	RE-GCN	CRNet
(South Korea, Make statement, China)	(China, Make statement, ?)	Japan	Japan
(South Korea, Make statement, Japan)	(China Express intent to meet, ?)	South Korea	Japan
	(Japan Express intent to meet, ?)	South Korea	China
	(Japan Diplomatic cooperation, ?)	South Korea	China

Case 2

Historical context	Concurrent missing events	RE-GCN	CRNet
(Envoy (US), Make visit, South Korea)	(North Korea, Make visit, ?)	South Korea	South Korea
	(South Korea, Make statement, ?)	Envoy (US)	Envoy (US)
(South Korea Host visit, Envoy (US))	(South Korea Host visit, ?)	Envoy (US)	North Korea

Case 3

Historical context	Concurrent missing events	RE-GCN	CRNet
-	(Protester, Protest, ?)	Police	Police
	(Protester, Demonstrate, ?)	Police	Police
	(Police, Use military force, ?)	Protester	Protester
	(Police, Repress, ?)	Protester	Protester
	(Protester, Make a request, ?)	Police	Student
	(Student, Express intent to yield, ?)	Protester	Police

Case 4

Historical context	Concurrent missing events	RE-GCN	CRNet
-	(Government, Intent to diplomatic coop., ?)	Citizen	Citizen
	(Government, Expel individuals, ?)	Citizen	Citizen
	(Citizen, Accuse, ?)	Government	Government
	(Citizen, Make request, ?)	The Judiciary	The Judiciary
	(Citizen, Appeal economic aid, ?)	Government	The Judiciary

Fig. 7. Four typical cases in the testing set of ICEWS14.

our CRNet considers the concurrent future event, such as *(North Korea, Make visit, ?)*, and predict them jointly with concurrent context. With the help of concurrent context *(North Korea, Make vist, South Korea)*, CRNet completes the missing event *(North Korea, Make visit, ?)* with the object *North Korea*.

The latter two cases are two emergencies, in which historical context cannot provide enough information to model the actors' behavior concretely.

In case 3, there is a conflict between the protester and the police. RE-GCN can predict the events between the police and the protester by transferring knowledge learned from previous conflicts. However there is a new participant *Student* in the happening conflict (obtained from *(Student, Express intent to yield, ?)*. RE-GCN limits itself with the participant of *Police* and *Protester* and cannot leverage the relationship between *Student* and *Protester*, which exists in the concurrent context.

In case 4, previous studies intend to predict the missing event *(Citizen, Appeal economic aid, ?)* with *Government*, since *Citizen* usually reach out to *Government* for help according to previous events. However, the *Citizen* and the

Government are in poor relationship which can be learned from the concurrent events. The *Citizen* are not likely to request economic aid from the *Government*.

In summary, concurrent missing events at future timestamps are important for TKG reasoning. Our proposal can mine the relationship between concurrent missing events and complete missing events more accurately.

6 Conclusion

We formulate and address the problem of concurrent events int TKG reasoning task in historical and future timestamps. Our proposal, CRNet, is consisted of two parts. For the historical concurrent events, we propose a novel relational graph attention network, EventRGAT, to model the interactions among events in a specific timestamp. For the future concurrent events, we propose a two-stage frame work, in which we build a candidate graph and model the interactions among future candidate events. Extensive experiments on three event-based TKG benchmarks demostrate the effectiveness of our CRNet. We also investigate into cases to study the influence of concurrent missing facts. The results indicate the concurrent context at future timestamp is informative for predicting missing events.

Supplemental Material Statement: Source code for our proposal is attached with the submission on EasyChair and will be available to public after acceptance. The datasets we used is adopt from the repository of RE-GCN. and have been submitted in the supplemental material. The raw data used to generate Table 3, Fig. 4, Fig. 5, and Fig. 6 are attached on EasyChair.

Acknowledgements. We would like to thank all anonymous reviewers for their insightful comments. This research is supported by the NSFC-General Technology Joint Fund for Basic Research (No. U1936206), the NSFC-Xinjiang Joint Fund (No. U1903128), the Natural Science Foundation of China (No. 62002178, No. 62172237), and the Fundamental Research Funds for the Central Universities (No. 63223046).

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NeurIPS, vol. 26. Curran Associates, Inc. (2013)
2. Boschee, E., Lautenschlager, J., O'Brien, S., Shellman, S., Starz, J., Ward, M.: ICEWS Coded Event Data (2015). https://doi.org/10.7910/DVN/28075
3. Deng, S., Rangwala, H., Ning, Y.: Learning dynamic context graphs for predicting social events. In: KDD, pp. 1007–1016. KDD 2019. Association for Computing Machinery, New York, NY, USA, July 2019. https://doi.org/10.1145/3292500.3330919
4. Deng, S., Rangwala, H., Ning, Y.: Dynamic knowledge graph based multi-event forecasting. In: KDD, pp. 1585–1595. ACM, Virtual Event CA USA, August 2020. https://doi.org/10.1145/3394486.3403209

5. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, pp. 1811–1818. AAAI 2018/IAAI 2018/EAAI 2018. AAAI Press, 2 Feb 2018
6. García-Durán, A., Dumančić, S., Niepert, M.: Learning sequence encoders for temporal knowledge graph completion. In: EMNLP, pp. 4816–4821. Association for Computational Linguistics, Brussels, Belgium, October 2018. https://doi.org/10.18653/v1/D18-1516
7. Han, Z., Chen, P., Ma, Y., Tresp, V.: DyERNIE: dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion. In: EMNLP, pp. 7301–7316. Association for Computational Linguistics, Online, November 2020. https://doi.org/10.18653/v1/2020.emnlp-main.593
8. Han, Z., Chen, P., Ma, Y., Tresp, V.: Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In: ICLR, 28 September 2020
9. Han, Z., Ding, Z., Ma, Y., Gu, Y., Tresp, V.: Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 8352–8364. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.emnlp-main.658
10. He, Y., Zhang, P., Liu, L., Liang, Q., Zhang, W., Zhang, C.: HIP network: historical information passing network for extrapolation reasoning on temporal knowledge graph. In: IJCAI, vol. 2, pp. 1915–1921, 9 August 2021. https://doi.org/10.24963/ijcai.2021/264
11. Jain, P., Rathi, S., Mausam, Chakrabarti, S.: Temporal knowledge base completion: new algorithms and evaluation protocols. In: EMNLP, pp. 3733–3747. Association for Computational Linguistics, Online, November 2020. https://doi.org/10.18653/v1/2020.emnlp-main.305
12. Jiang, T., et al.: Encoding temporal information for time-aware link prediction. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 2350–2354. Association for Computational Linguistics, Austin, Texas, November 2016. https://doi.org/10.18653/v1/D16-1260
13. Jin, W., Qu, M., Jin, X., Ren, X.: Recurrent event network: autoregressive structure inference over temporal knowledge graphs. In: EMNLP, pp. 6669–6683. Association for Computational Linguistics, November 2020. https://doi.org/10.18653/v1/2020.emnlp-main.541
14. Jung, J., Jung, J., Kang, U.: Learning to walk across time for interpretable temporal knowledge graph completion. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 786–795. KDD 2021. Association for Computing Machinery, New York, NY, USA, August 2021. https://doi.org/10.1145/3447548.3467292
15. Khalid, M., Baber, J., Kasi, M.K., Bakhtyar, M., Devi, V., Sheikh, N.: Empirical evaluation of activation functions in deep convolution neural network for facial expression recognition. In: 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), pp. 204–207, July 2020. https://doi.org/10.1109/TSP49548.2020.9163446
16. Leetaru, K., Schrodt, P.A.: GDELT: global data on events, location, and tone, 1979–2012. In: ISA Annual Convention, vol. 2, pp. 1–49. Citeseer (2013). https://www.gdeltproject.org/

17. Li, Z., et al.: Complex evolutional pattern learning for temporal knowledge graph reasoning. In: ACL, 20 March 2022
18. Li, Z., et al.: Search from history and reason for future: two-stage reasoning on temporal knowledge graphs. In: ACL, pp. 4732–4743. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.acl-long.365
19. Li, Z., et al.: Temporal knowledge graph reasoning based on evolutional representation learning. In: SIGIR, pp. 408–417. ACM, 11 July 2021. https://doi.org/10.1145/3404835.3462963
20. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 2181–2187. AAAI 2015. AAAI Press, 25 January 2015
21. Matsubara, Y., Sakurai, Y., Faloutsos, C., Iwata, T., Yoshikawa, M.: Fast mining and forecasting of complex time-stamped events. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 271–279. KDD 2012. Association for Computing Machinery, New York, NY, USA, August 2012. https://doi.org/10.1145/2339530.2339577
22. Nathani, D., Chauhan, J., Sharma, C., Kaul, M.: Learning attention-based embeddings for relation prediction in knowledge graphs. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4710–4723. Association for Computational Linguistics, Florence, Italy, July 2019. https://doi.org/10.18653/v1/P19-1466
23. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, pp. 809–816. ICML 2011, Omnipress, 28 June 2011
24. Park, N., Liu, F., Mehta, P., Cristofor, D., Faloutsos, C., Dong, Y.: EvoKG: jointly modeling event time and network structure for reasoning over temporal knowledge graphs. In: WSDM, 16 February 2022. https://doi.org/10.1145/3488560.3498451
25. Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B.: End-to-End structure-aware convolutional networks for knowledge base completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 3060–3067, 17 July 2019. https://doi.org/10.1609/aaai.v33i01.33013060
26. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems, vol. 26. Curran Associates, Inc. (2013)
27. Sun, H., Zhong, J., Ma, Y., Han, Z., He, K.: TimeTraveler: reinforcement learning for temporal knowledge graph forecasting. In: EMNLP, September 2021
28. Trivedi, R., Dai, H., Wang, Y., Song, L.: Know-Evolve: deep temporal reasoning for dynamic knowledge graphs. In: ICML, pp. 3462–3471. PMLR, 17 July 2017
29. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, pp. 2071–2080. ICML 2016, JMLR.org, 19 June 2016
30. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.: Composition-based multi-relational graph convolutional networks. In: ICLR, 25 September 2019
31. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., Bengio, Y.: Graph Attention Networks. ICLR (2018). https://doi.org/10.17863/CAM.48429
32. Wang, R., et al.: RETE: retrieval-enhanced temporal event forecasting on unified query product evolutionary graph. In: WWW, February 2022

33. Wang, X., et al.: Heterogeneous graph attention network. In: The World Wide Web Conference, pp. 2022–2032. WWW 2019. Association for Computing Machinery, 13 May 2019. https://doi.org/10.1145/3308558.3313562
34. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, pp. 1112–1119. AAAI 2014, AAAI Press, 27 July 2014
35. Xu, C., Chen, Y.Y., Nayyeri, M., Lehmann, J.: Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings. In: NAACL, pp. 2569–2578. Association for Computational Linguistics, Online, June 2021. https://doi.org/10.18653/v1/2021.naacl-main.202
36. Xu, Y., et al.: RTFE: a recursive temporal fact embedding framework for temporal knowledge graph completion. In: NAACL, pp. 5671–5681. Association for Computational Linguistics, Online, June 2021. https://doi.org/10.18653/v1/2021.naacl-main.451
37. Yang, B., Yih, S.W.T., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of the International Conference on Learning Representations (ICLR) 2015 (2015)
38. Yang, M., Meng, Z., King, I.: FeatureNorm: L2 feature normalization for dynamic graph embedding. In: ICDM (2020). https://doi.org/10.1109/ICDM50108.2020.00082
39. Zhao, L.: Event prediction in the big data era: a systematic survey. ACM Comput. Surveys **54**(5), 94:1–94:37 (2021). https://doi.org/10.1145/3450287
40. Zhu, C., Chen, M., Fan, C., Cheng, G., Zhang, Y.: Learning from history: modeling temporal knowledge graphs with sequential copy-generation networks. In: AAAI, vol. 35, pp. 4732–4740, May 2021