DACHA: A Dual Graph Convolution Based Temporal Knowledge Graph Representation Learning Method Using Historical Relation

LING CHEN, XING TANG, WEIQI CHEN, and YUNTAO QIAN, Zhejiang University
YANSHENG LI and YONGJUN ZHANG, Wuhan University

Temporal knowledge graph (TKG) representation learning embeds relations and entities into a continuous low-dimensional vector space by incorporating temporal information. Latest studies mainly aim at learning entity representations by modeling entity interactions from the neighbor structure of the graph. However, the interactions of relations from the neighbor structure of the graph are neglected, which are also of significance for learning informative representations. In addition, there still lacks an effective historical relation encoder to model the multi-range temporal dependencies. In this article, we propose a dual graph convolution network based TKG representation learning method using historical relations (DACHA). Specifically, we first construct the primal graph according to historical relations, as well as the edge graph by regarding historical relations as nodes. Then, we employ the dual graph convolution network to capture the interactions of both entities and historical relations from the neighbor structure of the graph. In addition, the temporal self-attentive historical relation encoder is proposed to explicitly model both local and global temporal dependencies. Extensive experiments on two event based TKG datasets demonstrate that DACHA achieves the state-of-the-art results.

CCS Concepts: • **Computing methodologies** → *Artificial intelligence*; *Knowledge representation and reasoning*; *Reasoning about belief and knowledge*; *Machine learning*; *Machine learning approaches*; *Learning latent representations*;

Additional Key Words and Phrases: Temporal knowledge graph, representation learning, dual graph convolution

ACM Reference format:
Ling Chen, Xing Tang, Weiqi Chen, Yuntao Qian, Yansheng Li, and Yongjun Zhang. 2021. DACHA: A Dual Graph Convolution Based Temporal Knowledge Graph Representation Learning Method Using Historical Relation. *ACM Trans. Knowl. Discov. Data* 16, 3, Article 46 (October 2021), 18 pages.
https://doi.org/10.1145/3477051

This work was funded by the National Key Research and Development Program of China (No. 2018YFB0505000) and the Fundamental Research Funds for the Central Universities (No. 2020QNA5017).
Authors' addresses: L. Chen (corresponding author), College of Computer Science and Technology, Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies, Zhejiang University, Hangzhou 310027, China; email: lingchen@cs.zju.edu.cn; X. Tang, W. Chen, and Y. Qian, College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; emails: {tangxing, jarvus, ytqian}@cs.zju.edu.cn; Y. Li and Y. Zhang, School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China; emails: {yansheng.li, zhangyj}@whu.edu.cn.

1556-4681/2021/10-ART46 $15.00
https://doi.org/10.1145/3477051

1 INTRODUCTION

Temporal knowledge graphs (**TKGs**) [32], containing temporally annotated knowledge, are regarded as multi-relational graphs, which have played crucial roles in various applications, e.g., international relation prediction [9] and social network analysis [1, 10].

The **Global Database of Events, Language, and Tone** (**GDELT**) [22] and **The Integrated Conflict Early Warning System** (**ICEWS**) [4] are typical event based TKGs, and the events in them are denoted as (head entity, relation, tail entity, and timestamp), abbreviated as (s, r, o, t). For example, for event (United States, make optimistic comment, China, 20180103), "United States" and "China" refer to head entity and tail entity, respectively, "make optimistic statement" refers to relation, and "20180103" refers to timestamp. TKG representation learning embeds relations and entities into a continuous low-dimensional vector space by incorporating temporal information, which is of significance for downstream tasks, e.g., link prediction [32].

Traditional TKG representation learning methods directly model timestamps as corresponding hyperplanes [8], representations [20], or fixed format encodings [13]. These methods simply embed associated timestamps into a low-dimensional vector space, failing to explicitly model the temporal dependency.

Some researchers have tried to introduce sequence models, e.g., **Recurrent Neural Networks** (**RNNs**) [19] and its variants [7, 12] to model the temporal dependency. For example, Know-Evolve [32] learns entity representations by employing RNNs to model historical relations. A historical relation is a sequence of relations augmented with timestamps between a pair of entities. As shown in Figure 1, the historical relation between "United States" and "China" is {(make optimistic comment, 20180103), (impose embargo, boycott, or sanctions, 20180322), (reject economic cooperation, 20180707), etc.}. We can infer that the two countries might further reduce cooperation, as the temporal evolving of the historical relation is of help for portraying "United States" and "China". However, all of the above-mentioned methods neglect the interactions among entities from the neighbor structure of the graph. To deal with this problem, RE-NET [16] learns entity representations by using **Graph Convolution Networks** (**GCNs**) to model entity interactions from the neighbor structure of the graph. Despite the promising results of introducing GCNs, there still exist some limitations needed to be improved.

First, existing methods fail to capture the interactions among relations from the neighbor structure of the graph. However, capturing the correlation between relations, especially historical relations, is of help for predicting the relations between entities in the future. For example, as shown in Figure 1, ignoring the interaction among historical relations from the neighbor structure of the graph, only considering the historical relation between "United States" and "Russia" (the blue line with an arrow), "United States" is likely to further strengthen cooperation with "Russia". Considering the correlation between historical relations of "United States" and "China", as well as "United States" and "Russia" (the blue line and orange line with an arrow), the relation between "United States" and "Russia" is less likely to be positive in the future, as "United States" has adopted a relatively austere economic policy (e.g., "impose embargo, boycott, or sanction" and "reject economic cooperation") to "China" since March, which will affect the relation between "United States" and "Russia".

Second, existing methods ignore the influences of different ranges of temporal dependency. However, information in different ranges indicates distinct relation trends. The local temporal dependency indicates the relation trend in a short time, while the global temporal dependency indicates the overall relation trend in a long time. In addition, the information in different ranges plays different roles, which should not be treated equally. The closer the relations in history are to the relation needed to be predicted, the more important they are. For example, when an event (United States, reject economic operation, China, 20180707) occurs, the relations between "United States"

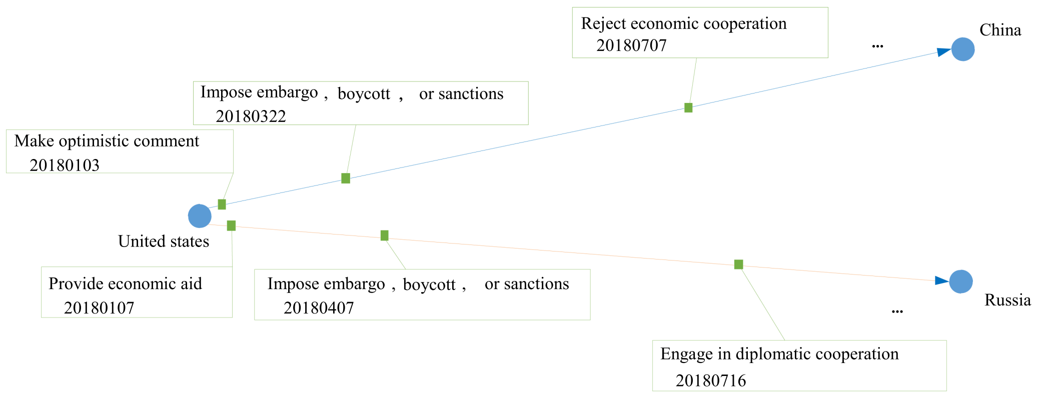
Reject economic cooperation 20180707
China
Impose embargo , boycott , or sanctions 20180322
Make optimistic comment 20180103
United states
Provide economic aid 20180107
Impose embargo , boycott , or sanctions 20180407
Russia
Engage in diplomatic cooperation 20180716

Fig. 1. The illustration of historical relations.

and “China” in the near future are less likely to be positive, and relation “reject economic operation” is more important for predicting the relations between “United States” and “China” in July 2018.

To address the aforementioned problems, we propose **a dual graph convolution network based TKG representation learning method using historical relations** (**DACHA**). DACHA not only considers entity interactions from the neighbor structure of the graph, but also captures historical relation interactions from the neighbor structure of the graph, and leverages multi-range temporal dependencies. The main contributions of our work are summarized as follows:

(1) Propose DACHA, which introduces the dual graph convolution network to learn the representations of entities by capturing the interactions of both entities and historical relations from the neighbor structure of the graph. The primal graph is constructed based on historical relations, and the edge graph is constructed by regarding historical relations as nodes.
(2) Propose a temporal self-attentive historical relation encoder to model temporal information, which can explicitly model both local and global temporal dependencies, and automatically learn the importance of different relations in a historical relation.
(3) Conduct extensive experiments on two real-world datasets, i.e., GDELT18 and ICEWS18. Experimental results show that the proposed method DACHA achieves the state-of-the-art results on the task of link prediction for TKGs.

The rest of this article is organized as follows. Section 2 reviews the existing studies related to this work. Section 3 gives the definitions of related terms and task. Section 4 describes the details of the proposed DACHA. Section 5 presents and discusses the experimental results. Section 6 concludes the article and presents the future work.

2 RELATED WORK

In this section, a review of the related work is presented, which includes static **Knowledge graphs** (**KGs**) representation learning methods, TKG representation learning methods, and **heterogeneous graph representation learning** (**HGLR**) methods.

2.1 Static KG Representation Learning Methods

KGs include a plenty of structured information for entities and relations, which have been successfully applied to various fields, e.g., question answering [27], information extraction [23], document clustering [35], and recommendation [14]. Static KGs usually model multi-relational information with triples represented as (head entity, relation, tail entity), abbreviated as (s, r, o). Static KG representation learning methods, aiming at learning the representations of entities and relations, can be used to efficiently measure the semantic correlations of entities and relations and alleviate sparsity issues in static KGs.

Existing studies on static KGs can be mainly classified into three categories, translation models [3, 24, 30, 31, 36, 38], semantic matching models [26, 33, 37], and GCN based models [2, 25, 28]. TransE [3] is a typical translation model, which regards the relation r as the translation from the head entity s to the tail entity o, and uses $\boldsymbol{s} + \boldsymbol{r} \cong \boldsymbol{o}$ as the optimal objective. TransE performs well with one-to-one relations, but has issues with one-to-many, many-to-one, and many-to-many relations. To address the issue, a lot of improved models based on TransE are proposed [24, 30, 31, 36, 38]. For example, Wang et al. [36] proposed TransH, which differentiates relations by projecting entities into different hyperplanes decided by relations. Lin et al. [24] proposed TransR, which supposes relations and entities belonging to different semantic spaces, and projects entities into relation-specific spaces. RESCAL [26] is a typical semantic matching model, which employs the bilinear model to capture the semantic similarity between entities in each relation. To address the problem of excessive parameters in RESCAL, Yang et al. [37] proposed a simplified model, DistMult, which uses diagonal matrices to represent relations. R-GCN [28] is a typical GCN based model, which learns the weights of different relations, as well as the representations of entities in KGs, by utilizing GCNs to accumulate the representations of neighbor entities. Existing translation models and semantic matching models solely utilize triple, i.e., (s, r, o), while R-GCN exploits the information from the neighbor structure of the graph, which can improve the representations of entities.

2.2 TKG Representation Learning Methods

Event based TKGs include temporally annotated knowledge, which have played crucial roles in various applications, e.g., international relation prediction [9] and social network analysis [1]. The events in them are denoted as (head entity, relation, tail entity, timestamp), abbreviated as (s, r, o, t). TKG representation learning methods, embedding relations and entities into a continuous low-dimensional vector space by incorporating temporal information, can be used to efficiently model the evolving of entities or relations in TKGs.

Some TKG representation learning methods have attracted wide attention [8, 11, 13, 15, 16, 20, 32]. HyTE [8], TTransE [20], and TA-TransE [13] directly model timestamps as corresponding hyperplanes, representations, and fixed format encodings, respectively. These methods simply embed associated timestamps into a low-dimensional vector space, ignoring the temporal dependency. Recently, some researchers have tried to introduce sequence models, e.g., RNNs and its variants, to model the temporal dependency. For example, Trivedi et al. [32] proposed Know-Evolve, which models the occurrence of an event as a temporal point process, and employs RNNs to model historical relations. However, these methods ignore the interactions of entities from the neighbor structure of the graph. To deal with the problem, Jin et al. [16] proposed RE-NET, which learns entity representations by using GCNs to model entity interactions from the neighbor structure of the graph, as well as GRUs to model entity interactions from the neighbor structure of the graph in history. However, RE-NET fails to capture the interactions of relations from the neighbor structure of the graph, especially historical relations. In addition, existing methods ignore the influences of multi-range temporal dependencies.

2.3 Heterogeneous Graph Representation Learning Methods

Heterogeneous graphs, where there are multiple types of nodes or edges, have become ubiquitous in real-world scenarios, e.g., KGs and social networks. HGRL methods aim to learn representations in a continuous low-dimensional vector space while preserving the heterogeneous structures and semantics. To incorporate relation heterogeneity into the graph representation learning, meta-path based HGRL methods have received increasing attention. For example, Wang et al. [39] proposed HAN, which uses a hierarchical attention mechanism to capture the importance between a node and its meta-path based neighbors. MAGNN [40] improves HAN by taking the intermediate nodes in a meta-path into consideration. To model heterogeneous graph structures and node attributes, Zhang et al. [41] proposed HetGNN, which samples heterogeneous neighbors by random walk and aggregates node and type information with specific RNNs. In these methods, meta-paths are manually defined, requiring plenty of domain knowledge. To address this problem, Hwang et al. [42] proposed SELAR, which automatically generates meta-paths without manual labeling and facilitates the link prediction task by introducing an auxiliary task to predict various meta-paths with node representations. However, these methods only model the multiple relations between nodes, which fail to capture the interactions among relations from the neighbor structure of the graph.

3 PRELIMINARIES

In this section, we give some basic definitions of related terms and task, and introduce the technical details of GCNs.

3.1 Problem Formulation

A TKG is a multi-relational graph, which can be represented as a set of events. Formally, a TKG is represented as $G = \{(s, r, o, t) | s, o \in \mathcal{E}, r \in \mathcal{R}, t \in T\}$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and T is the set of timestamps.

A historical relation is a sequence of relations augmented with timestamps between a pair of entities, which can be formulated as $h = \{(r_1, t_1), (r_2, t_2), \ldots, (r_k, t_k), \ldots, (r_I, t_I)\}$, where $t_k < t_{k+1} (1 \leq k < I)$, and I is the length of the historical relation.

Entities and relations in a KG can be represented as nodes and edges in a graph, respectively. In this work, the primal graph and the edge graph are used to model entity interactions and historical relation interactions from the neighbor structure of the graph, respectively.

Let the current moment be t, given events in $[t - T'' + 1 : t]$, the task of link prediction in TKGs is to predict the relations or entities in $[t + 1 : t + T']$.

3.2 Graph Convolution

GCNs [5, 18] are a kind of deep neural networks, which conduct convolution operation on a graph. Formally, the graph convolution operator gc applied in our model is defined as follows:

$$gc\,(\boldsymbol{X}) = \varphi(\tilde{\mathbf{L}}\boldsymbol{X}\theta), \tag{1}$$

where $\boldsymbol{X}$ is the input signal, $\tilde{\mathbf{A}}$ is the adjacent matrix with self-connection, i.e., $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$. $\tilde{\mathbf{L}}$ is the normalized adjacent matrix, with $\tilde{\mathbf{L}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, where $\tilde{\mathbf{D}}$ is the corresponding diagonal degree matrix. φ is a nonlinear activation function, and θ is learnable parameters. One-layer graph convolution can aggregate information from 1-hop neighbors. By stacking the one-layer graph convolutions, we can extend the receptive range of neighbors.

4 METHODOLOGY

This section gives the detailed description of the proposed method DACHA. We first give the framework of DACHA, and then present the details of the temporal self-attentive historical relation encoder, dual graph convolution network, and link prediction.

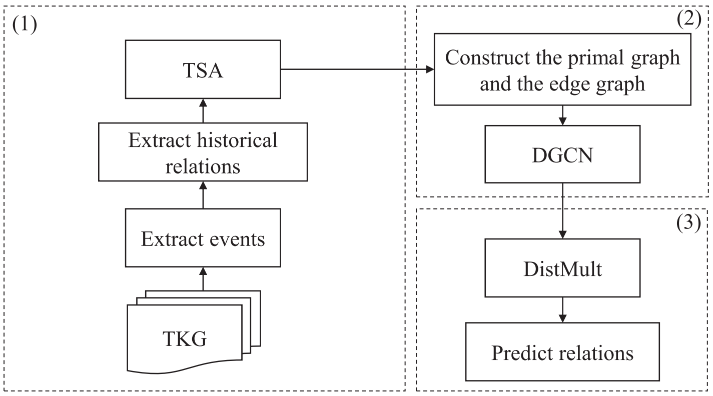
(1)
TSA
Extract historical relations
Extract events
TKG
(2)
Construct the primal graph and the edge graph
DGCN
(3)
DistMult
Predict relations

Fig. 2. The overall flow diagram of DACHA: (1) Temporal dependency modeling; (2) Interaction modeling; and (3) Link prediction. TSA denotes the temporal self-attentive historical relation encoder. DGCN denotes the dual graph convolution network.

4.1 Framework

In this section, we introduce the technical details of the proposed method. The overall flow diagram of DACHA is shown in Figure 2, which consists of three parts: (1) Temporal dependency modeling; (2) Interaction modeling; and (3) Link prediction.

In the temporal dependency modeling part, the temporal self-attentive historical relation encoder is employed to encode historical relations, which can model multi-range temporal dependencies. In the interaction modeling part, the primal graph is constructed based on historical relations, and the edge graph is constructed by regarding historical relations as nodes. The dual graph convolution network contains several primal graph convolution layers and edge graph convolution layers, which are explicitly applied to model the interactions of both entities and historical relations from the neighbor structure of the graph. In the link prediction part, the semantic matching model DistMult is utilized to predict the relations between entities.

4.2 Temporal Self-Attentive Historical Relation Encoder

Equipped with positional encoding or positional mask, self-attention mechanism is widely used in sequence modeling [29, 34, 21]. To encode temporal information, we propose the temporal self-attentive historical relation encoder to leverage multi-range temporal dependencies, which employs positional encoding to consider the relative positions of the relations in the historical relations. Figure 3 shows the structure of the temporal self-attentive historical relation encoder. It consists of intra-block temporal self-attention mechanism and inter-block temporal self-attention mechanism, which can model local and global temporal dependencies, respectively, and automatically learn the importance of different relations or blocks.

We first split the input historical relation h into M blocks with equal length, i.e., $h = [z_1, z_2, \ldots, z_{\mathrm{M}}]$. Each block contains N relations. There is no overlap between blocks, and padding can be applied to the last block if necessary. $r_{m,n}$ denotes the nth relation in block z_m. $t_{m,n}$ denotes the timestamp of relation $r_{m,n}$. The intra-block temporal self-attention mechanism can model the local temporal dependency by learning the importance of different relations in each block, which is formulated as follows:

$$e_{m,n} = \mathbf{W}_1^{\text{intra}} \sigma\left(\mathbf{W}_2^{\text{intra}} \boldsymbol{r}_{m,n} + \mathbf{W}_3^{\text{intra}} \boldsymbol{p}_{m,n} + b_1^{\text{intra}}\right) + b_2^{\text{intra}}, \tag{2}$$

$$a_{m,n} = softmax\,(e_{m,n}) = \frac{\exp(e_{m,n})}{\sum_{n'=1}^{N} \exp(e_{m,n'})}, \tag{3}$$

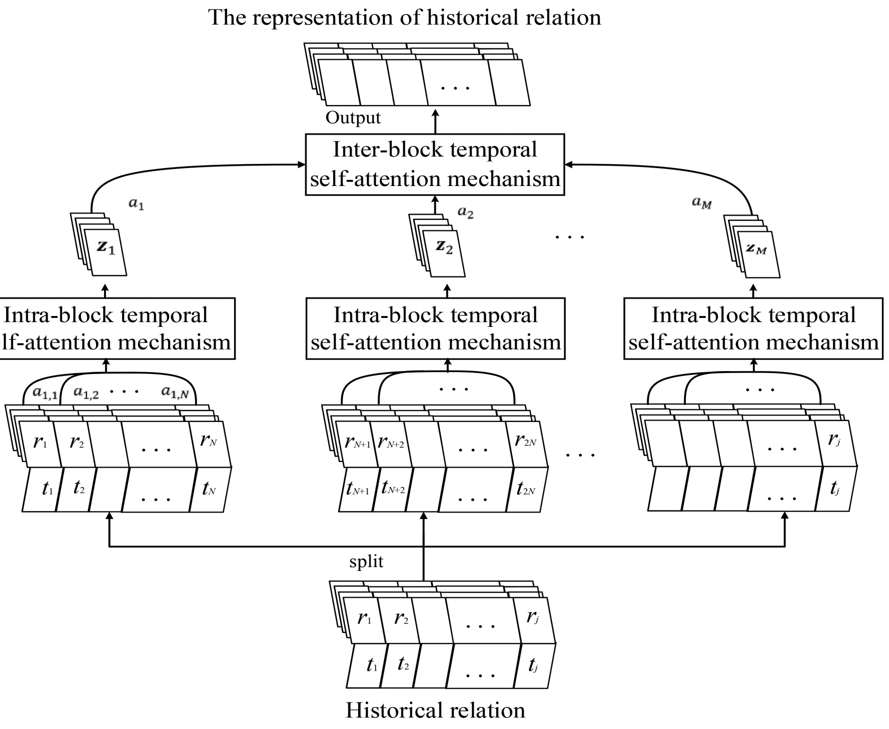
The representation of historical relation
Output
Inter-block temporal self-attention mechanism
a_1
a_2
a_M
z_1
z_2
z_M
Intra-block temporal self-attention mechanism
$a_{1,1}$ $a_{1,2}$ $a_{1,N}$
r_1 r_2 r_N t_1 t_2 t_N
r_{N+1} r_{N+2} r_{2N} t_{N+1} t_{N+2} t_{2N}
r_j t_j
split
Historical relation

Fig. 3. The structure of the temporal self-attentive historical relation encoder.

where $\mathbf{W}_1^{\text{intra}}$, $\mathbf{W}_2^{\text{intra}}$, $\mathbf{W}_3^{\text{intra}}$, b_1^{intra}, and b_2^{intra} are learnable parameters. $\boldsymbol{p}_{m,n}$ is the relative time between each relation in block z_m and relation $r_{m,n}$, i.e., $\boldsymbol{p}_{m,n} = [(t_{m,1} - t_{m,n}), (t_{m,2} - t_{m,n}), \ldots, (t_{m,N} - t_{m,n})]^{\mathrm{T}}$. σ is an activation function. $\boldsymbol{r}_{m,n}$ is the representation of relation $r_{m,n}$, which is randomly initialized and jointly learned during the training process.

The representations of each block are obtained by weighted sum, which can be formulated as:

$$z_m = \sum_{n=1}^{N} a_{m,n} \boldsymbol{r}_{m,n}. \tag{4}$$

The inter-block temporal self-attention mechanism can model the global temporal dependency by learning the importance of different blocks, which is formulated as follows:

$$e_m = \mathbf{W}_1^{\text{inter}} \sigma \left(\mathbf{W}_2^{\text{inter}} z_m + \mathbf{W}_3^{\text{inter}} \boldsymbol{q}_m + b_1^{\text{inter}} \right) + b_2^{\text{inter}}, \tag{5}$$

$$a_m = \operatorname{softmax}(e_m) = \frac{\exp(e_m)}{\sum_{m'=1}^{M} \exp(e_{m'})}, \tag{6}$$

where $\mathbf{W}_1^{\text{inter}}$, $\mathbf{W}_2^{\text{inter}}$, $\mathbf{W}_3^{\text{inter}}$, b_1^{inter}, and b_2^{inter} are learnable parameters. σ is an activation function. $\boldsymbol{q}_m$ is the relative time between the first relation in each block and the first relation in block z_m, i.e., $\boldsymbol{q}_m = [(t_{1,1} - t_{m,1}), (t_{2,1} - t_{m,1}), \ldots, (t_{M,1} - t_{m,1})]^{\mathrm{T}}$.

The representations of each historical relation are obtained by weighted sum, which can be formulated as follows:

$$\boldsymbol{h} = \sum_{m=1}^{M} a_m z_m. \tag{7}$$

4.3 Dual Graph Convolution Network

Chen et al. [6] proposed edge-GCN to conduct convolution operation on the edge graph, which can model the interactions among the edges from the neighbor structure of the primal graph. Inspired by this, we first construct the primal graph based on historical relations and the edge graph by regarding historical relations as nodes. Once we obtain the representations of each historical relation using the temporal self-attentive historical relation encoder and initialize the entity representations, we introduce the dual graph convolution to model the interactions of both entities and historical relations from the neighbor structure of the graph, and thus achieve informative representations.

4.3.1 The Construction of Primal Graph and Edge Graph. To use the significant information contained in historical relations to portray entities, first, we construct the primal graph $G^{\text{primal}} = (V^{\text{primal}}, E^{\text{primal}}, \mathbf{A}^{\text{primal}})$, where V^{primal} is the set of nodes (i.e., entities), E^{primal} is the set of edges (i.e., historical relations), and $\mathbf{A}^{\text{primal}}$ is an adjacency matrix representing the connection between entities. The adjacency matrix $\mathbf{A}^{\text{primal}}$ is formulated as follows:

$$\mathbf{A}_{i,j} = \begin{cases} 1, & there\ is\ a\ historical\ relation\ from\ entity\ s_i\ to\ entity\ s_j \\ 0, & otherwise \end{cases}. \tag{8}$$

To utilize the correlation between historical relations, we construct the edge graph $G^{\text{edge}} = (V^{\text{edge}}, E^{\text{edge}}, \mathbf{A}^{\text{edge}})$ by regarding the edges (i.e., historical relations) in G^{primal} as the nodes in G^{edge}, i.e., $|V^{\text{edge}}| = |E^{\text{primal}}|$. An edge in G^{edge} is defined based on whether a pair of edges in G^{primal} have a connection. Specifically, if two edges in G^{primal} share a node, they will be correlated in G^{edge}. The adjacency matrix $\mathbf{A}^{\text{edge}}$ is formulated as follows:

$$\mathbf{A}^{\text{edge}}{}_{i\to j, u\to v} = \begin{cases} w_{i\to j, u\to v}, & edge\ i \to j\ and\ u \to v\ share\ a\ node \\ 0, & otherwise \end{cases}. \tag{9}$$

Chen et al. [6] regard $\mathbf{A}^{\text{edge}}{}_{i\to j, u\to v} = 1$, if edge $i \to j$ and $u \to v$ share a node. Different from the design of this adjacency matrix, we measure the interaction intensity of historical relations from the neighbor structure of the graph to construct the edge graph. If the shared node has more neighbors in the primal graph, the interaction intensity between edge $i \to j$ and edge $u \to v$ will be weaker, as it is susceptible to other neighbors. Weight $w_{i\to j, u\to v}$ is formulated as follows:

$$w_{i\to j, u\to v} = \begin{cases} \frac{1}{\eta} + \alpha, & \eta > 2 \\ 1, & \eta = 2 \end{cases}, \tag{10}$$

where η is the number of neighbors that the shared node has. α is a hyper-parameter, and $0 < \alpha < 0.5$.

4.3.2 Dual Graph Convolution. With the constructed primal graph and edge graph, as shown in Figure 4, the dual graph convolution contains primal graph convolution layers and edge graph convolution layers, which can explicitly model the interactions of both entities and historical relations from the neighbor structure of the graph.

In the primal graph convolution, we obtain the representations of entities by aggregating the representations of neighbor entities in the primal graph and the representations of historical relations in the edge graph. Similarly, in the edge graph convolution, we obtain the representations of historical relations by aggregating the representations of neighbor historical relations in the edge graph and the representations of entities in the primal graph. The dual graph convolution is formulated as follows:

$$X^{(1)} = \varphi\left(\tilde{\mathbf{L}} X^{(0)} \theta_{\text{node}}{}^{(0)}\right), \tag{11}$$

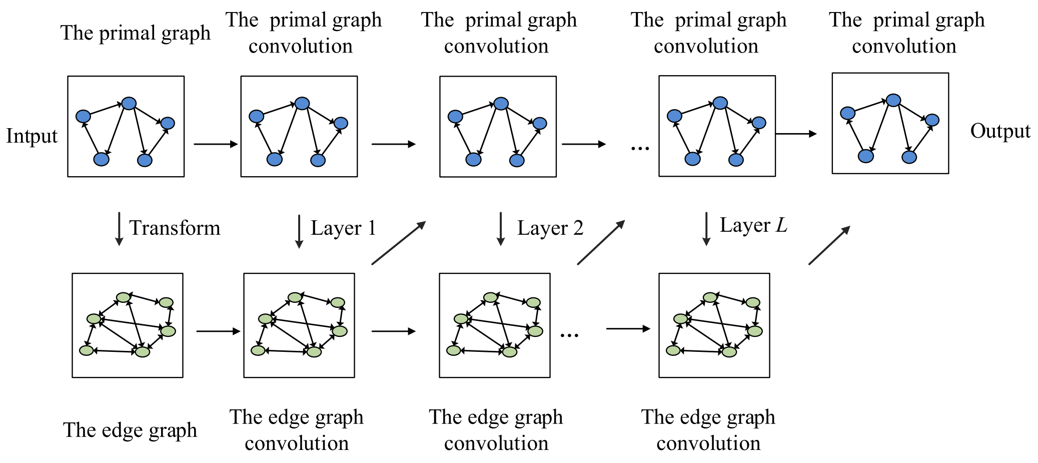
The primal graph
The primal graph convolution
The primal graph convolution
The primal graph convolution
The primal graph convolution
Intput
Output
Transform
Layer 1
Layer 2
Layer L
The edge graph
The edge graph convolution
The edge graph convolution
The edge graph convolution

Fig. 4. The architecture of the dual graph convolution network.

$$X^{(l)} = \varphi\left[\tilde{\mathbf{L}}X^{(l-1)}{\theta_{\text{node}}}^{(l-1)}, \mathbf{M}^{\text{ne}}Y^{(l-1)}{\theta_{\text{en}}}^{(l-1)}\right], \tag{12}$$

$$Y^{(l-1)} = \varphi\left[\tilde{\mathbf{L}}^{\text{edge}}Y^{(l-2)}{\theta_{\text{edge}}}^{(l-2)}, (\mathbf{M}^{\text{ne}})^{\text{T}}X^{(l-1)}{\theta_{\text{ne}}}^{(l-1)}\right], \quad \text{for } l = 2, \ldots, L, \tag{13}$$

where $X^{(0)}$ is the input of the primal graph, which is the randomly initialized representations of entities and jointly learned during the training process. $Y^{(0)}$ is the input of the edge graph, which is the representations of historical relations, obtained by using the temporal self-attentive historical relation encoder. L is the layer of the dual graph convolution. $X^{(l)}$is the output of the lth primal graph convolution layer, and $Y^{(l-1)}$ is the output of the $(l-1)$ th edge graph convolution layer. $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{L}}^{\text{edge}}$ are the normalized adjacent matrix of the primal graph and the edge graph, respectively. θ_{node}, θ_{en}, θ_{edge}, and θ_{ne} are learnable parameters, and $[\cdot,\cdot]$ is the concatenation operation. $\mathbf{M}^{\text{ne}}$ is the incidence matrix that encodes the connections between nodes and edges, which is defined as: ${\mathbf{M}^{\text{ne}}}_{i,i\to j} = {\mathbf{M}^{\text{ne}}}_{i,k\to i} = 1$, and 0 otherwise, where $i, j, k \in V^{\text{primal}}$ and $i \to j, k \to i \in E^{\text{primal}}$.

After conducting dual graph convolution, we use $X^{(L)}$, the output of the dual graph convolution, as the ultimate entity representations for link prediction, which are integrated representations considering historical relations as well as the interactions of both entities and historical relations from the neighbor structure of the graph.

4.4 Link Prediction

Following Schlichtkrul et al. [28], we employ DistMult [37] to predict the relations or entities between entities. Specifically, we use the score function of DistMult as our score function, which is formulated as follows:

$$f(s, r, o) = (\boldsymbol{s}^T \mathbf{M}_i \boldsymbol{o}), \tag{14}$$

where $\boldsymbol{s}$ and $\boldsymbol{o}$ are the representations of head entity and tail entity, respectively, which correspond to the rows of $X^{(L)}$. $\mathbf{M}_i$ is the diagonal matrix corresponding to a relation, which is defined as: $\mathbf{M}_i = \text{diag}(\mathbf{W}\boldsymbol{r}_i)$, where diag() is a diagonal matrix transformation function that transforms a vector to a diagonal matrix, $\mathbf{W}$ is a learnable parameter, and $\boldsymbol{r}_i$ is the representation of a relation, which is jointly learned in the temporal self-attentive historical relation encoder. After conducting the temporal self-attentive historical relation encoder and dual graph convolution, the temporal information and interaction information from the neighbor structure of the graph have been encoded into the representations of entities and relations.

Table 1. The Statistics of Datasets

Dataset	#Relation	#Entity	#Train set	#Validation set	#Test set	Time granularity
GDELT18	240	7,691	1,734,399	238,765	305,241	15 mins
ICEWS18	256	23,033	373,018	45,995	49,545	24 hours

We model the link prediction task as a ranking problem. Taking relation prediction as an example, the relations between entities can be predicted by calculating the score function. For each observed positive sample (s, r, o), the negative samples are generated by randomly replacing r. A cross-entropy loss function is used as our training objective, which is formulated as follows:

$$Loss = -\sum_{D} ylog\left[Sigmoid\left(f\left(s,r,o\right)\right)\right] + (1-y)\,log\left[1 - Sigmoid\left(f\left(s,r,o\right)\right)\right], \tag{15}$$

where D is the set of positive samples and negative samples from the history with the length of T'', *Sigmoid* is the logistic sigmoid function, and $y \in \{0, 1\}$ is the label. For positive samples y is set to 1, and 0 otherwise.

5 EXPERIMENTS

In this section, we present the experiments to empirically evaluate the proposed method DACHA. First, we introduce the two datasets used in the experiments. Second, we give the details of evaluation metrics and parameter settings. Then, we present and discuss the experimental results. Finally, we give the complexity analysis and sensitivity analysis.

5.1 Datasets

We evaluate our method on two real-world datasets, i.e., GDELT18 and ICEWS18, which are collected from GDELT [22] with the time granularity of 15 minutes, and ICEWS [4] with the time granularity of 24 hours, respectively. GDELT18 and ICEWS18 contain the records of events, which are collected from Jan 1st 2018 to Jan 31st 2018 and from Jan 1st 2018 to Oct 31st 2018, respectively. The time steps of two datasets are set as 1 day. Both datasets are split in chronological order with 80% for training, 10% for validation, and 10% for testing. Detailed statistics of the datasets are shown in Table 1.

5.2 Experimental Settings

We evaluate our method on the task of link prediction. Let the current moment be t, given evens in $[t - T'' + 1 : t]$, the task of link prediction in TKGs is to predict the relations or entities in $[t + 1 : t + T']$. In experiments, we aim to predict the relations or entities over one day or one week in the future, given the events in the last two weeks, i.e., T'' =14 and T' =1 or 7. For the link prediction in a future horizon (i.e., specified test triples with relations or entities missing), we fill the missing parts with each relation or entity in a TKG to obtain candidate triples. Then, we compute the scores of candidate triples by Equation (14) and obtain a list in ascending order.

The number of layers in the dual graph convolution network is set to 2. The number of blocks in the temporal self-attentive historical relation encoder is set to 5. The number of hidden units is set to 100. We use Adam [17] to optimize the parameters of DACHA. The epoch number is set to 100. The initial learning rate is set to 0.01 with a decay rate of 0.6 per 10 epochs. The batch size is set to 64. The dimension of representations is set to 200. α (see Section 4.3.1) is set to 0.4. For the dual convolution network, the nonlinear activation function φ is the ReLU function. For the temporal self-attentive historical relation encoder, the activation function σ is the tanh function. We adopt a sliding window training approach with the time step of 1 day. **Mean Reciprocal Rank (MRR)** and Hits@k are used as evaluation metrics. MRR denotes the average of the reciprocal ranks of correct relations. Hits@k denotes the percentage of correct relations ranked at top k. Higher MRR and Hits@k indicate better performance.

5.3 Baselines

To demonstrate the effectiveness of DACHA, we compare it with other static KG representation learning methods and TKG representation learning methods. The detailed descriptions and experimental settings of these methods are as follows:

Static KG representation learning method:

TransE [3] regards relations as the translation operations between head entities and tail entities.

DistMult [37] uses diagonal matrices to represent relations.

R-GCN [28] learns the weights of different relations, as well as the representations of entities in KGs, by utilizing GCNs to accumulate the representations of neighbor entities.

For static KG representation learning methods, i.e., TransE, DistMult, and R-GCN, we ignore temporal information and use triples to conduct experiments. The dimension of representations is set to 200. The negative sampling ratios in TransE, DistMult, and R-GCN are set to 1, 1, and 10, respectively. The learning rates in TransE, DistMult, and R-GCN are set to 0.01, 0.1, and 0.01, respectively.

TKG representation learning method:

TTransE [20] models timestamps as corresponding representations. In experiments, the dimension of representations is set to 200, the negative sampling ratio to 1, and the learning rate is set to 0.01.

HyTE [8] models timestamps as corresponding hyperplanes. In experiments, we use every timestamp as a hyperplane. The dimension of representations is set to 128, the negative sampling ratio is set to 5, and the learning rate is set to 0.001.

TA-TransE [13] models timestamps as fixed format encodings. In experiments, the fixed format encoding is year, month, and day for the ICEWS18 dataset and year, month, day, hour, and minute for the GDELT18 dataset. The batch size is set to 1,024, the dimension of representations is set to 200, the negative sampling ratio is set to 1, and the learning rate is set to 0.001.

Know-Evolve [32] models the occurrence of an event as a temporal point process, and employs RNNs to model historical relations. In experiments, the batch size is set to 200, the dimension of representations is set to 100, and the learning rate is set to 0.0005.

RE-NET [16] learns entity representations by using GCNs to model entity interactions from the neighbor structure of the graph, as well as GRUs to model entity interactions from the neighbor structure of the graph in history. In experiments, the batch size is set to 1,024, the dimension of representations is set to 200, the negative sampling ratio is set to 10, and the learning rate is set to 0.001.

For all the baselines, on the task of link prediction, we compute the scores of candidate triples by their own score functions.

For fairness, all compared methods follow the same experimental protocol and the hyperparameters are all well-tuned. To avoid the influence of random factors, all methods are evaluated five times and the average results are reported. To statistically measure the significance of performance differences, pairwise t-tests at 95% significance level are conducted between DACHA and compared methods.

5.4 Comparisons with Static KG and TKG Representation Learning Methods

To demonstrate the effectiveness of DACHA, we compare it with static KG and TKG representation learning methods. Table 2 presents the average MRR, Hits@1, Hits@3, and Hits@10 of DACHA and baseline models on both datasets on the task of relation prediction over one day and one week

Table 2. The Performances of DACHA and the Compared Methods on the Task of Relation Prediction

	Method	GDELT18				ICEWS18			
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
	TransE	18.44 ± 0*	3.17 ± 0*	19.79 ± 0*	31.75 ± 0.01*	19.72 ± 0*	2.91 ± 0*	23.15 ± 0*	34.33 ± 0.01*
	DistMult	20.78 ± 0.01*	10.86 ± 0*	27.13 ± 0*	40.17 ± 0*	20.94 ± 0*	15.96 ± 0.03*	30.25 ± 0*	43.72 ± 0.01*
	R-GCN	22.85 ± 0*	17.69 ± 0.02*	29.32 ± 0*	43.67 ± 0*	25.08 ± 0*	19.77 ± 0.02*	33.48 ± 0*	45.73 ± 0*
1	TTransE	11.04 ± 0*	1.83 ± 0*	19.25 ± 0.02*	29.56 ± 0.02*	11.96 ± 0*	3.97 ± 0*	12.79 ± 0.01*	24.33 ± 0*
Day	HyTE	19.47 ± 0.01*	5.69 ± 0*	23.86 ± 0.01*	40.43 ± 0*	21.85 ± 0*	6.86 ± 0*	25.64 ± 0*	41.86 ± 0*
	TA-TransE	19.62 ± 0*	3.58 ± 0*	20.77 ± 0*	38.79 ± 0*	21.79 ± 0*	3.84 ± 0*	32.18 ± 0*	43.34 ± 0*
	Know-Evolve	23.09 ± 0*	11.40 ± 0*	31.48 ± 0*	46.37 ± 0*	25.38 ± 0*	14.75 ± 0*	33.75 ± 0.02*	49.87 ± 0*
	RE-NET	42.12 ± 0*	35.72 ± 0*	45.96 ± 0*	51.59 ± 0.01*	42.25 ± 0.01*	33.81 ± 0*	44.98 ± 0*	52.72 ± 0*
	DACHA	46.78 ± 0	39.72 ± 0.01	49.89 ± 0	57.92 ± 0	47.87 ± 0	40.91 ± 0.02	50.47 ± 0	58.69 ± 0
	TransE	17.55 ± 0.02*	2.93 ± 0*	18.27 ± 0*	30.17 ± 0*	19.08 ± 0.02*	2.41 ± 0*	22.28 ± 0.01*	32.23 ± 0*
	DistMult	19.68 ± 0.01*	10.07 ± 0.02*	25.26 ± 0.01*	39.09 ± 0.01*	20.11 ± 0.02*	14.16 ± 0*	28.01 ± 0*	42.02 ± 0.01*
	R-GCN	22.35 ± 0*	16.39 ± 0*	28.28 ± 0*	42.17 ± 0*	23.48 ± 0.02*	18.75 ± 0*	32.03 ± 0*	44.13 ± 0*
1	TTransE	10.74 ± 0*	2.31 ± 0*	10.25 ± 0*	24.36 ± 0.01*	11.26 ± 0*	3.01 ± 0*	11.92 ± 0*	23.43 ± 0.02*
Week	HyTE	19.07 ± 0*	4.55 ± 0*	23.16 ± 0.01*	39.03 ± 0*	21.01 ± 0*	5.96 ± 0*	24.78 ± 0*	40.86 ± 0*
	TA-TransE	19.12 ± 0.01*	2.78 ± 0*	19.37 ± 0*	37.15 ± 0.02*	21.09 ± 0*	3.57 ± 0*	31.77 ± 0*	42.58 ± 0*
	Know-Evolve	22.28 ± 0*	11.08 ± 0*	30.19 ± 0*	45.75 ± 0*	24.08 ± 0*	13.37 ± 0*	32.01 ± 0*	48.17 ± 0.01*
	RE-NET	40.78 ± 0*	33.34 ± 0*	44.96 ± 0.02*	52.07 ± 0*	43.05 ± 0.01*	35.61 ± 0*	46.28 ± 0*	54.11 ± 0*
	DACHA	43.02 ± 0	37.35 ± 0	48.06 ± 0.01	56.62 ± 0	46.97 ± 0.01	38.99 ± 0.01	49.78 ± 0.01	57.06 ± 0

(mean±std). * Indicates that DACHA is Statistically Superior to the Compared Method (Pairwise *t*-test at 95% Significance Level).

Table 3. The Performances of DACHA and the Compared Methods on the Task of Entity Prediction, in Which "Head" Means Predicting Head Entity and "Tail" Means Predicting Tail Entity

	Method	Head				Tail			
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
	TransE	15.88 ± 0*	2.03 ± 0*	17.69 ± 0*	27.57 ± 0.01*	16.38 ± 0.01*	2.97 ± 0*	21.68 ± 0.02*	31.23 ± 0.05*
static	DistMult	10.88 ± 0.03*	10.11 ± 0.03*	23.76 ± 0.02*	36.49 ± 0*	18.14 ± 0.04*	12.76 ± 0.03*	25.37 ± 0*	40.08 ± 0.02*
	R-GCN	20.44 ± 0*	14.42 ± 0*	25.28 ± 0.03*	40.17 ± 0*	20.88 ± 0*	17.19 ± 0.03*	30.08 ± 0.03*	42.43 ± 0*
	TTransE	10.64 ± 0*	1.78 ± 0*	17.75 ± 0.03*	22.66 ± 0.02*	10.56 ± 0.01*	2.74 ± 0*	17.89 ± 0.02*	20.87 ± 0.03*
	HyTE	17.17 ± 0.03*	4.09 ± 0.02*	22.16 ± 0.01*	37.23 ± 0*	20.05 ± 0*	4.75 ± 0*	22.38 ± 0*	40.16 ± 0.03*
temporal	TA-TransE	18.32 ± 0*	3.67 ± 0*	20.07 ± 0*	35.49 ± 0*	21.59 ± 0*	3.74 ± 0*	32.07 ± 0*	42.98 ± 0*
	Know-Evolve	23.19 ± 0*	10.40 ± 0*	30.48 ± 0*	45.97 ± 0*	22.08 ± 0*	14.29 ± 0*	30.32 ± 0*	46.17 ± 0*
	RE-NET	37.42 ± 0*	30.54 ± 0.01*	41.26 ± 0.02*	50.67 ± 0*	40.25 ± 0.02*	33.87 ± 0.01*	43.38 ± 0*	51.09 ± 0.03*
	DACHA	40.31 ± 0.01	34.42 ± 0	45.26 ± 0	53.72 ± 0	44.37 ± 0	36.78 ± 0	46.45 ± 0	54.76 ± 0

(mean±std) * Indicates that DACHA is Statistically Superior to the Compared Method (Pairwise *t*-test at 95% Significance Level).

in the future. Table 3 presents the average MRR, Hits@1, Hits@3, and Hits@10 of DACHA and baseline models on dataset GDELT18 on the task of entity prediction over one week in the future. From the results we can observe the following phenomena:

(1) R-GCN performs better than static KG representation learning methods, i.e., TransE and DistMult, as well as TKG representation learning methods, i.e., TTransE, HyTE, and TA-TransE, which indicates the effectiveness of R-GCN to learn the representations of entities by modeling entity interactions from the neighbor structure of the graph.

Table 4. The Comparison of DACHA and Variants Without the Dual Graph Convolution Network

Method	GDELT18		ICEWS18	
	MRR	Hits@10	MRR	Hits@10
DACHA-single graph	41.88 ± 0.02*	52.93 ± 0*	42.89 ± 0.02*	53.87 ± 0*
DACHA-meta-path	42.25 ± 0*	54.09 ± 0.03*	44.36 ± 0*	55.27 ± 0*
DACHA-unweighted graph	45.01 ± 0.01*	56.21 ± 0.02*	46.12 ± 0*	57.17 ± 0.01*
DACHA	46.78 ± 0	57.92 ± 0	47.87 ± 0	58.69 ± 0

(mean±std) * Indicates that DACHA is Statistically Superior to the Compared Method (Pairwise *t*-test at 95% Significance Level)

(2) Know-Evolve and RE-NET perform better than TTransE, HyTE, and TA-TransE, which indicates the effectiveness of modeling the temporal dependency. RE-NET performs better than Know-Evolve and R-GCN, which indicates the effectiveness of RE-NET to learn the representations of entities by modeling entity interactions from the neighbor structure of the graph and the temporal dependency.

(3) DACHA performs the best, which further justifies the advantages of introducing the dual graph convolution to model the interactions of both entities and historical relations from the neighbor structure of the graph, and the temporal self-attentive historical relation encoder to model the multi-range temporal dependencies.

5.5 Ablation Experiments

5.5.1 The Effectiveness of the Dual Graph Convolution Network. To verify the effectiveness of introducing the dual graph convolution network to model the interactions of both entities and historical relations from the neighbor structure of the graph, we compare DACHA with three variants, including: (1) DACHA-single graph, which ignores the edge graph convolution and only utilizes a two-layer GCN to learn the representations of entities. (2) DACHA-meta-path, which ignores the edge graph convolution, automatically generates some meta-paths between entities, and utilizes a two-layer GCN to learn the representations of entities. (3) DACHA-unweighted graph, which replaces the edge graph in DACHA with the original edge graph in [6] that is an unweighted graph.

Table 4 shows the average MRR and Hits@10 of DACHA and its variants on the task of relation prediction over one week in the future, from which we can observe the following phenomena: (1) DACHA-meta-path performs better than DACHA-single graph, which might be because DACHA-meta-path captures the multiple relations between entities by modeling meta-paths. (2) DACHA-unweighted graph performs better than DACHA-meta-path, which might be because DACHA-unweighted graph considers the interactions of historical relations from the neighbor structure of the graph. (3) DACHA performs better than DACHA-unweighted graph, which shows the effectiveness of capturing the interaction intensity of historical relations from the neighbor structure of the graph. By measuring the interaction intensity to construct the edge graph, the dual graph convolution network of DACHA models the interactions of both entities and historical relations from the neighbor structure of the graph more effectively.

5.5.2 The Effectiveness of the Temporal Self-Attentive Historical Relation Encoder. To further verify the effectiveness of proposing the temporal self-attentive historical relation encoder to model multi-range temporal dependencies, we compare DACHA with two variants, including: (1) DACHA-self attention, which utilizes the vanilla self-attention mechanism to encode historical relations, and differentiates the effects of different relations in a historical relation, ignoring

Table 5. The Comparison of DACHA and Variants Without the Temporal Self-Attentive Historical Relation Encoder

Method	GDELT18		ICEWS18	
	MRR	Hits@10	MRR	Hits@10
DACHA-RNN	$42.49 \pm 0^*$	$53.27 \pm 0.03^*$	$44.66 \pm 0^*$	$55.48 \pm 0^*$
DACHA-self attention	$45.77 \pm 0^*$	$56.03 \pm 0^*$	$46.31 \pm 0.01^*$	$56.98 \pm 0^*$
DACHA	46.78 ± 0	57.92 ± 0	47.87 ± 0	58.69 ± 0

(mean±std) * Indicates that DACHA Is Statistically Superior to the Compared Method (Pairwise t-test at 95% Significance Level).

multi-range temporal dependencies. (2) DACHA-RNN, which utilizes RNNs to encode historical relations and considers the information of different ranges equally.

Table 5 shows the average MRR and Hits@10 of DACHA and its variants on the task of relation prediction over one week in the future, from which we can observe the following phenomena: (1) DACHA-self attention performs better than DACHA-RNN, which might be because DACHA-self attention can differentiate the effects of different relations in a historical relation. (2) DACHA performs better than DACHA-self attention. The results verify the effectiveness of the temporal self-attentive historical relation encoder, which can explicitly model both local and global temporal dependencies, and automatically learn the importance of different relations in a historical relation.

5.6 Complexity Analysis

The time complexity of DAHCA consists of the main three modules. For the temporal self-attentive historical relation encoder, the time complexity is $\Theta(D \times N)$, where D is the dimension of representations, N is the length of historical relation. For the dual graph convolution network, the time complexity of the primal graph convolution is $\Theta(T'' \times |E| \times D^2 + T'' \times |E| \times D)$, where T'' is the length of history and $|E|$ is the number of edges in the primal graph; the time complexity of the edge graph convolution is $\Theta(T'' \times |V| \times D^2)$, where $|V|$ is the number of nodes in the primal graph. Assuming that the numbers of nodes and edges in the primal graph are the same, the total time complexity of the dual graph convolution network is $\Theta(T'' \times |E| \times D^2)$. For the link prediction module, the time complexity is $\Theta(D)$.

5.7 Sensitivity Analysis

We investigate the influence of several important parameters on the model performance, including the number of layers in the dual graph convolution network and the number of blocks in the temporal self-attentive historical relation encoder.

To explore the impact of the number of layers in the dual graph convolution network, we vary it from 1 to 5. The experimental results on dataset GDELT18 are shown in Figure 5, from which we can find that with the increase of the number of layers, first the model performance increases and then decreases slightly. When the number is 2, the model achieves the highest Hits@1, Hits@3, Hits@10, and MRR.

For the number of blocks in the temporal self-attentive historical relation encoder, we vary it from {1, 5, 10, 15} in the experiment. The experimental results on dataset GDELT18 are shown in Figure 6, from which we can find similar tendencies. As the number of blocks in the temporal self-attentive historical relation encoder increases, the model performance ascends at the beginning and then descends gradually. When the number of blocks in the temporal self-attentive historical relation encoder is 5, the model achieves the best performance. The reason may be that the larger the number of blocks is, the more parameters are needed to learn, which may cause overfitting.

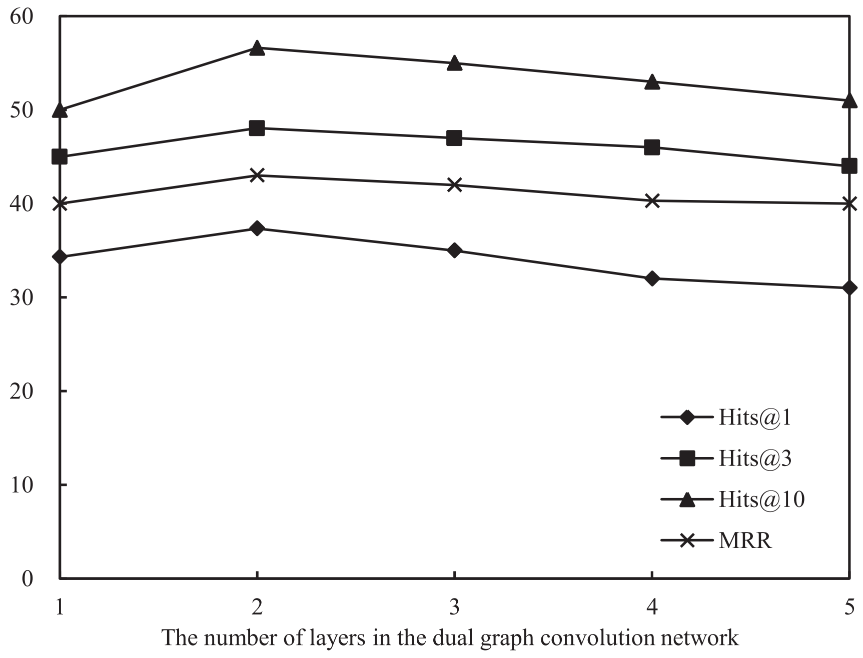

Fig. 5. The impact of the number of layers in the dual graph convolution network.

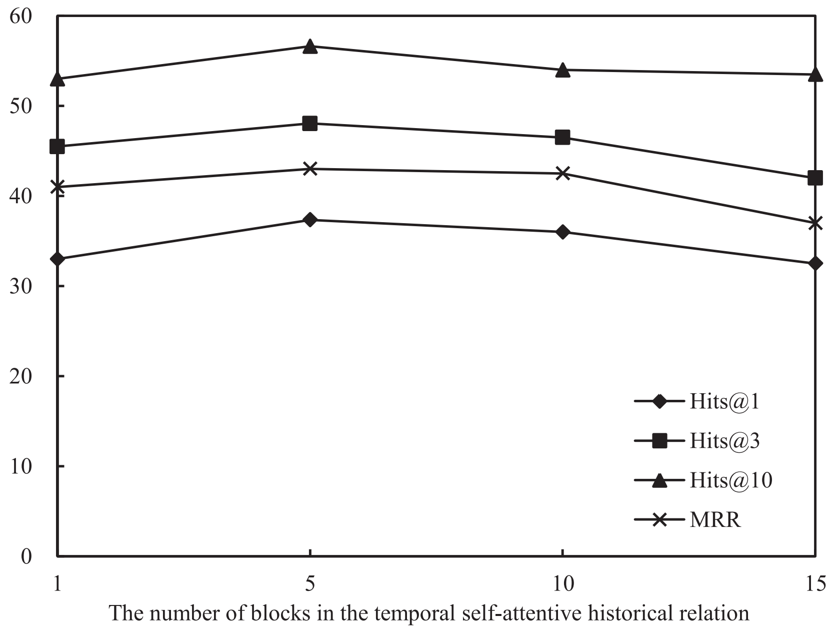

Fig. 6. The impact of the number of blocks in the temporal self-attentive historical relation encoder.

6 CONCLUSIONS AND FUTURE WORK

We propose a dual graph convolution network based temporal knowledge graph representation learning method using historical relations. Specifically, the dual graph convolution network is proposed to capture the interactions of both entities and historical relations from the neighbor structure of the graph. The primal graph is constructed according to historical relations, and the edge graph is constructed by regarding historical relations as nodes. In addition, the temporal self-attentive historical relation encoder is proposed to effectively leverage the multi-range temporal dependencies. On two events based TKG datasets, our model achieves an average of 4.43% and 3.44% improvement in relation prediction and entity prediction task, respectively, than the state-of-the-art results. The following conclusions can be derived from the ablation experiments results: (1) The dual graph convolution network can effectively model the interactions of both entities and historical relations from the neighbor structure of the graph. (2) The introduced temporal self-attentive historical relation encoder can effectively capture multi-range temporal dependencies.

In the future, we will investigate how to discover implicit correlations between entities that have not occurred in an event in history. We also plan to utilize the hierarchical information of entities and relations to improve the learning process of representations.

REFERENCES

[1] Martin Atzmueller. 2019. Onto model-based anomalous link pattern mining on feature-rich social interaction networks. In *Companion Proceedings of the 27th World Wide Web Conference*. 1047–1050.

[2] Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4387–4392.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. 2787–2795.

[4] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS coded event data. *Harvard Dataverse*.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations*.

[6] Zhengdao Chen, Xiang Li, and Joan Bruna. 2019. Supervised community detection with line graph neural networks. In *Proceedings of the 7th International Conference on Learning Representations*.

[7] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing*. 1724–1734.

[8] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 23th Conference on Empirical Methods in Natural Language Processing*. 2001–2011.

[9] Denis Degterev, Kristina Badrutdinova, and Anna Stepanova. 2017. Interconnections among the United States, Russia and China: Does Kissinger's American leadership formula apply? *International Organizations Research Journal* 12, 1 (2017), 81–109.

[10] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. 2011. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data* 5, 2 (2011), 1–27.

[11] Cristóbal Esteban, Volker Tresp, Yinchong Yang, Stephan Baier, and Denis Krompaß. 2016. Predicting the co-evolution of event and knowledge graphs. In *Proceedings of the 19th International Conference on Information Fusion*. 98–105.

[12] Chang Gao, Junkun Yan, Shenghua Zhou, Pramod K. Varshney, and Hongwei Liu. 2019. Long short-term memory-based deep recurrent neural networks for target tracking. *Information Sciences* 502, 2 (2019), 279–296.

[13] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 23rd Conference on Empirical Methods in Natural Language Processing*. 4816–4821.

[14] Xiaobo Guo, Wenfang Lin, Youru Li, Zhongyi Liu, Lin Yang, Shuliang Zhao, and Zhenfeng Zhu. 2020. DKEN: Deep knowledge-enhanced network for recommender systems. *Information Sciences* 540, 5786 (2020), 263–277.

[15] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016. Towards time-aware knowledge graph completion. In *Proceedings of 26th International Conference on Computational Linguistics*. 1715–1724.

[16] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *Proceedings of the 25th Conference on Empirical Methods in Natural Language Processing*. 6669–6683.

[17] Diederik P. Kingma, and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations.*

[18] Thomas N. Kipf, and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations.*

[19] Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. 2011. Recurrent neural network based language modeling in meeting recognition. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association.* 1045–1048.

[20] Julien Leblay, and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the 26th World Wide Web Conference.* 1771–1776.

[21] John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. 2019. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data* 13, 6 (2019), 1–25.

[22] Kalev Leetaru, and Philip A. Schrodt. 2013. GDELT: Global data on events, location, and tone, 1979-2012. *ISA Annual Convention* 2, 4 (2013), 1–49.

[23] Peipei Li, Haixun Wang, Hongsong Li, and Xindong Wu. 2018. Employing semantic context for sparse information extraction assessment. *ACM Transactions on Knowledge Discovery from Data* 12, 5 (2018), 1–36.

[24] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 20th Conference on Empirical Methods in Natural Language Processing.* 705–714.

[25] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* 4710–4723.

[26] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning.* 809–816.

[27] Chen Qiao and Xiao Hu. 2020. A neural knowledge graph evaluator: Combining structural and semantic evidence of knowledge graphs for predicting supportive knowledge in scientific QA. *Information Processing and Management* 57, 6 (2020), 102309.

[28] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the 15th European Semantic Web Conference.* 593–607.

[29] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *Proceedings of the 6th International Conference on Learning Representations.*

[30] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the 7th International Conference on Learning Representations.*

[31] Xing Tang, Ling Chen, Jun Cui, and Baogang Wei. 2019. Knowledge representation learning with entity descriptions, hierarchical types, and textual relations. *Information Processing and Management* 56, 3 (2019), 809–822.

[32] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-Evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning.* 3462–3471.

[33] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning.* 2071–2080.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 5998–6008.

[35] Chenguang Wang, Yangqiu Song, Dan Roth, Ming Zhang, and Jiawei Han. 2016. World knowledge as indirect supervision for document clustering. *ACM Transactions on Knowledge Discovery from Data* 11, 2 (2016), 1–36.

[36] Zhen Wang, Jianwen Zhang, and Jianlin Feng. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence.* 1112–1119.

[37] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representations.*

[38] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence.* 3065–3072.

[39] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *Companion Proceedings of the 27th World Wide Web Conference.* 2022–2032.

[40] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Companion Proceedings of the 28th World Wide Web Conference*. 2331–2341.
[41] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.
[42] Dasol Hwang, Jinyoung Park, Sunyoung Kwon, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J. Kim. 2020. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. In *Proceedings of the 34th Conference on Neural Information Processing Systems*.

Received December 2020; revised May 2021; accepted July 2021