Temporal Knowledge Graph Reasoning with Dynamic Hypergraph Embedding

Xinyue Liu[1], **Jianan Zhang**[1], **Chi Ma**[1], **Wenxin Liang**[1], **Bo Xu**[2,3*], **Linlin Zong**[1]
[1]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province,
School of Software, Dalian University of Technology, China
[2]School of Computer Science and Technology, Dalian University of Technology, China
[3] Key Laboratory of Social Computing and Cognitive Intelligence (Dalian University of Technology),
Ministry of Education, China
{xyliu, wxliang, xubo, llzong}@dlut.edu.cn
nicole9805@163.com, dlut.marx@gmail.com

Abstract

Reasoning over the Temporal Knowledge Graph (TKG) that predicts facts in the future has received much attention. Most previous works attempt to model temporal dynamics with knowledge graphs and graph convolution networks. However, these methods lack the consideration of high-order interactions between objects in TKG, which is an important factor to predict future facts. To address this problem, we introduce dynamic hypergraph embedding for temporal knowledge graph reasoning. Specifically, we obtain high-order interactions by constructing hypergraphs based on temporal knowledge graphs at different timestamps. Besides, we integrate the differences caused by time into the hypergraph representation in order to fit TKG. Then, we adapt dynamic meta-embedding for temporal hypergraph representation that allows our model to choose the appropriate high-order interactions for downstream reasoning. Experimental results on public TKG datasets show that our method outperforms the baselines. Furthermore, the analysis part demonstrates that the proposed method brings good interpretation for the predicted results.

Keywords: Temporal Knowledge Graph, Hypergraph, Reasoning

1. Introduction

Knowledge Graphs (KGs) have abundant information storage and are broadly used for downstream tasks (Zou, 2020) due to their excellent representation of structured knowledge. Knowledge graphs store static facts in the form of triples (s, r, o) in a graphical structure. However, most real-world knowledge is incomplete in that facts constantly change over time. In order to capture the dynamic changes of facts, Temporal Knowledge Graphs (TKGs) with temporal attributes are introduced by extending the triple (s, r, o) into a quadruple (s, r, o, t). Recently, TKGs have received a lot of attention and have been applied to downstream tasks in KG and even other artificial intelligence domains, such as event prediction (Song et al., 2021), knowledge graph reasoning (Li et al., 2022) and question answering (Saxena et al., 2021).

Knowledge graph reasoning that predicts missing facts for incomplete KGs has been widely explored. However, reasoning over Temporal Knowledge Graph (TKG) that predicts facts in the future is still far from resolved. Previous work can model the temporal dynamics with the help of GCN (Kipf and Welling, 2016), RGCN (Schlichtkrull et al., 2017), etc. However, most of the relationships between objects in practical applications cannot be represented by the lower-order model of pairwise connections alone. In fact, the data structures in our real-world applications may go beyond pairwise joins to be even more complex. The Figure 1 shows the historical facts related to a query (Barack-Obama, Makestatement, ?, 2014-12-4) from the ICEWS14 dataset. The answer to the query is China, which is not directly or indirectly linked to the query subject entity, Barack Obama. In this case, the traditional graph structure has limitations in expressing data relevance. However, the high-order relationships can quickly link the related entities, e.g., the red dotted ellipse can connect countries and leaders of countries with common features, making the query subject entity closely related to the answer. The use of high-order relationships becomes very important. Therefore, there are two main challenging problems:

Q1. How to obtain the high-order relationships between objects in temporal knowledge graph reasoning?

Q2. How to represent the high-order relationships in graph neural network for the downstream reasoning?

To address the aforementioned problems, we introduce Dynamic Hypergraph Embedding for Temporal Knowledge Graph reasoning (DHE-TKG). To solve Problem Q1, we introduce the hypergraph structure to obtain high-order relationships in temporal knowledge graphs. Recently, hypergraph has become increasingly popular, which is a structure

*Corresponding author

LREC-COLING 2024, pages 15742–15751

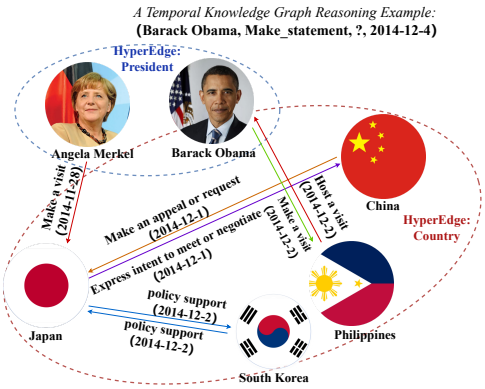

Figure 1: An example of TKG on the ICEWS14.

for describing higher-order interactions and provides a more flexible and natural framework for representing such multi-directional relationships. However, the existing hypergraph structure lacks integration of the information of time into entities on the graph. And in the TKG models, entity features keep evolving with increasing timestamps, where the variation of each moment is important for the prediction of future events, which is ignored by the existing models. So that we design a temporal feature variation module to observe the differences caused by time on the predicted answer features. Target at Problem Q2, we adapt the dynamic meta-embedding technique for hypergraph representation. Specifically, we utilize attention to fuse high-order interaction features in different timestamps of the same entity. Experimental results show that our method outperforms the baseline models.

2. RELATED WORK

2.1. Temporal KG Reasoning and Link Prediction

Real-world data often evolves over time and the model dealing with temporal knowledge graph embeddings comes into being. Some of the studies add the part dealing with temporal features to the KG embedding model. For example, TTransE (Leblay and Chekol, 2018) extends TransE (Bordes et al., 2013) by encoding timestamps as translations into score functions. TA-DistMult (García-Durán et al., 2018) uses recurrent neural networks (RNN) to learn temporal-aware representations of relations based on DistMult (Yang et al., 2014). TComlEx and TNTComplEx (Lacroix et al., 2020) are based on ComplEx (Trouillon et al., 2016) and are inspired by the canonical decomposition of the fourth-order tensor.

In addition, some studies also model temporal features in TKG. Know-Evolve (Trivedi et al., 2017) and DyREP (Trivedi et al., 2019) use temporal point processes to model concurrent facts to estimate conditional probabilities to predict future facts; Glean (Deng et al., 2020) incorporates constructed word graphs to summarize the text of events by reasoning in future facts. RENET (Jin et al., 2019) uses RNN-based event encoders to model interactions between entities and uses GCN (Kipf and Welling, 2016) to learn multi-hop structural information. CyGNet (Zhu et al., 2020) uses sequential copy networks to model repetitive facts, exploiting historical information. RE-GCN (Li et al., 2021) learns the evolutional representations of entities and relations to predict future facts and learns the static properties of entities by building additional static graphs.

However, the above approach can only deal with pairwise relationships between entities and has limitations in terms of higher-order relationships.

2.2. Hypergraph Neural Networks

In the real world, the relationship between entities exists not only between two entities but even beyond pairwise associations, which is the reason (Dengyong Zhou and Jiayuan Huang and Bernhard Schölkopf, 2006) first introduced the concept of hypergraph learning to model higher-order relationships for semi-supervised classification and node clustering. A hypergraph is a graph-based extension where each hyperedge can connect multiple nodes and is used to address the case where multiple nodes exist with the same connection and can better exploit the correlation of higher-order data. Extensions to the hypergraph construction method by (Huang et al., 2009) include the use of k-NN and search radius methods. HGNN (Feng et al., 2018) proposes the hyperedge convolution operation to represent the correlation between data better. Based on the problem that the hypergraph structure cannot change dynamically with the features, DHGNN (Jiang et al., 2019) dynamically updates the hypergraph structure on each layer. Additionally, Chien proposes a generic HGNN (Feng et al., 2018) framework that covers most of the HGNN pproaches. HyperSAGE (Arya et al., 2020) uses a two-level neural message-passing strategy instead of transforming the hypergraph structure into a graph. HyperGCN (Yadati et al., 2018) trains GCN (Kipf and Welling, 2016) on hypergraphs based on the spectral theory of hypergraphs.

3. METHODOLOGY

3.1. Notation and Task Definition

A temporal knowledge graph can be viewed as a series of static knowledge graph snapshots under different timestamps, denoted by G =

$\{G_1, G_2, ..., G_t\}$. G_t consists of the set of events $D_t = (S, R, O)$ occurring at time t, where S is the set of subject entities, R is the set of relations, and O is the set of object entities. Each event in the event set D_t can be defined as a quaternion (s, r, o, t) occurring at timestamp t, where s is the set of subject entities, r is the set of relations, and o is the set of object entities.

The goal of TKG reasoning is to predict the missing object entity o_q of the query $(s_q, r_q, ?, t_q)$ given the history graph $G_{0:t_{q-1}}$, or to predict the missing relation r_q in the query $(s_q, ?, o_q, t_q)$. In addition, we add the inverse quaternion (o, r^{-1}, s, t) of each quaternion (s, r, o, t) to the dataset as well.

3.2. The Model

The proposed model DHE-TKG consists of two parts in general, namely encoder and decoder. The encoder is used to map entities and relations on the graph for each timestamp to their low-dimensional embeddings, and the decoder scores the answer embeddings for the degree of match of the query. We focus on the encoder part.

As shown in Figure 2, the encoder consists of evolutionary embedding, dynamic hypergraph embedding and embedding fusion. Evolutionary embedding encodes according to the historical fact development, and the entity embedding evolves continuously with time, and the closer the fact occurs to the query time, the greater the influence on the result. Dynamic hypergraph embedding uses the hypergraph structure to encode higher-order interactions. The embedding fusion combines the evolutionary embedding with the dynamic hypergraph embedding to achieve a balance between low-order interactions and high-order interactions.

3.2.1. Evolutionary embedding

We use the embedding evolution encoder to capture the structural dependencies between entities. To follow the factual evolution, the embedding evolution encoder uses RGCN as the basic module to input the node embedding of the previous moment, perform the embedding update of each time step graph G_t by RGCN, and update the node embedding representation of the current timestamp using GRU. Normally, RGCN aggregators are defined as follows:

$$h_{o,t}^{l+1} = \sigma(\sum_{r \in R} \sum_{j \in N_o^r} (\frac{W_r^l h_{o,t}^l}{|N_o^r|}) + W_s^l h_{s,t}^l) \quad (1)$$

where $h_{o,t}^l$ and $h_{s,t}^l$ denote the l-th layer embedding of entities at timestamp t. W_s^l and $W_r^l$are the learnable matrices of the l-th layer. N_o^r denotes the set of neighboring entities of s connected by relation r, then $|N_o^r|$ plays a normalizing role. σ is the activation function.

To maintain historically useful information and to obtain information with more distant timestamps, we use GRU for updating entity embedding:

$$\hat{H}_t = GRU(\hat{H_{t-1}}, H_{t-1}^{RGCN}) \quad (2)$$

where $\hat{H_{t-1}}$ and $\hat{H}_t$ are the entity embeddings at t - 1 and t. H_{t-1}^{RGCN} is the entity embedding after RGCN aggregation at $t-1$.

For relational updates, we also use GRU:

$$r_t = [pooling(H_{t-1,v_{r,t}}||r)] \quad (3)$$

$$R_t = GRU(R_{t-1}, R_t') \quad (4)$$

where $v_{r,t}$ is all entities with r as a relation at time t. $||$ denotes the concatenation operation. R_t' consists of r_t of all relations. R_{t-1} and R_t denote the relation embedding at $t-1$ and t.

3.2.2. Dynamic hypergraph embedding

Dynamic hypergraph embedding has two modules, one is to aggregate all node features in hyperedge e to obtain this hyperedge feature. The other one aggregates the features of all the hyperedges containing the current node v to obtain the features of node v. We effectively distinguish the same entities with different timestamps based on the characteristics of the temporal knowledge graph and aggregate them by a dynamic meta-embedding approach.

(1) Hypergraph Construction. A hypergraph can be represented as $G_h = (V_h, E_h)$, where $V_h = v_1, v_2, ..., v_n$ denotes the set of nodes and $E_h = e_1, e_2, ..., e_n$ denotes the set of hyperedges. Unlike the general graph, a hyperedge can hold several different nodes, indicating that these nodes share common features to establish the interaction of higher-order data. In particular, when the number of nodes in the hyper edge is 2, the hypergraph becomes a simple graph.

Let $X = (x_1, x_2, ..., x_n)$ be the set of n d-dimensional vectors of hypergraph inputs, where $x_i = h_{i,t}$ is the feature vector of the i-th node. In each layer, we use the KNN method for hypergraph construction, where the k-1 nearest neighbors of each vertex u form a hyperedge together with that vertex.

(2) Location Node Aggregation. The location node aggregation module is used to aggregate the features of the nodes in the hyperedge to the hyperedge containing those nodes. As shown in Figure 3, since the hypergraph is not able to distinguish the order of the input history graphs, in order to enable the model to take advantage of the development order of the history graphs, we inject some information about the absolute position of the history graphs. For this purpose, we add the location node

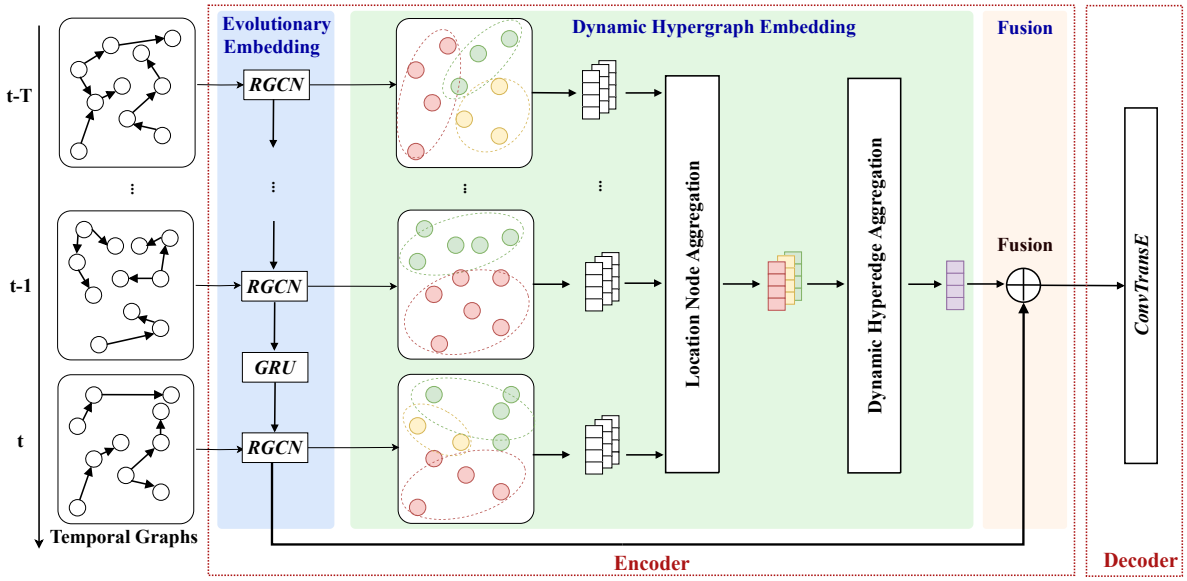

Figure 2: Overview of DHE-TKG. DHE-TKG takes the history graph from time step $t-T$ to t as input. Evolutionary embedding captures the low-order structural dependencies of entities through the input history graph time order. Dynamic hypergraph embedding captures the higher-order dependencies of entities. The final answer is obtained by balancing the low-order and high-order representations.

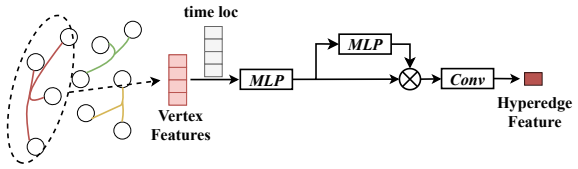

Figure 3: Location node aggregation.

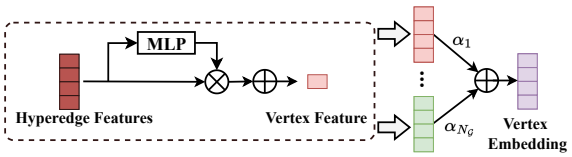

Figure 4: Dynamic hyperedge aggregation.

aggregation module, which adds location information to the node features of each history graph input as a distinction between entities under different timestamps and gives them location order information. The features of the history graph are mapped to a new dimensional space by multilayer perceptron (MLP). Next, we model the input feature matrix with MLP to model the attention matrix and then use a convolutional layer used to compress the features, thus enabling the aggregation of nodes on multiple history graphs to super edges:

$$X_t = W_{loc}(H_t||t) + b_{loc} \tag{5}$$

$$\alpha_{trans} = f(X_t) \tag{6}$$

$$x_e = conv(\alpha_{trans} \cdot f(X_t)) \tag{7}$$

where t is the t-th history graph of the input. W_{loc} and b_{loc} are learnable parameters.

(3) Dynamic hyperedge aggregation. Dynamic hyperedge aggregation module obtains the hypergraph embedding by aggregating the hyperedge features containing node v to update the representation of nodes in the current hypergraph. As shown in Figure 4, the attention score of each hyperedge is obtained using MLP and the node features are aggregated by attention summation.

$$w = softmax(x_e W + b) \tag{8}$$

$$x_u = \sum_{i=0}^{|C(u)|} w^i x_e^i \tag{9}$$

where $C(u)$ denotes the number of hyperedges where node u is located. It is worth noting that the node representations at this point contain the same entities with different timestamps.

Since queries from different datasets have different sensitivities to the input graphs with different timestamps, we use dynamic attention to learn the focus of different datasets on queries. We combine the entity features with different timestamps by obtaining a weighted sum to get the updated entity features X.

$$X = \sum_{t=0}^{N_{\mathcal{G}}} \alpha_{v,t} x_u^{v,t} \tag{10}$$

where $\alpha_{v,t} = g(\{x_u^{v,t}\}_{v=1}^{N_v})$ is the scalar weight ob-

tained by self-attention. Specifically,

$$\alpha_{v,t} = \phi(W_a \cdot x_u^{v,t}) + b \tag{11}$$

where W_a and b are learnable parameters. ϕ is the sigmoid function in this paper due to its good performance. $N_{\mathcal{G}}$ is the number of input history graphs.

3.2.3. Embedding fusion

Since the impact of evolutionary embedding and dynamic hypergraph embedding may be different, we measure their influence by the hyperparameter α. The final entity features are shown below:

$$H_t = (1-\alpha)\hat{H}_t + \alpha X \tag{12}$$

3.2.4. Decoder

Regarding the temporal knowledge graph prediction task a scoring function is usually used to measure the plausibility of the prediction results. Previous studies have shown that a GNN with convolutional score functions has good performance on the temporal knowledge graph prediction task. Therefore, we choose ConvTransE as the decoder of the model, using $ConvTransE(\cdot)$ to represent it. Thus the entity prediction probability is represented as follows:

$$p(o|H_t, R_t, s, r) = \sigma(H_{t-1} ConvTransE(s_{t-1}, r_{t-1})) \tag{13}$$

Similarly, the relationship prediction probability is represented as follows:

$$p(r|H_t, R_t, s, o) = \sigma(R_{t-1} ConvTransE(s_{t-1}, o_{t-1})) \tag{14}$$

where σ is the sigmoid function.

3.3. Training

The entity o prediction obtained from the query $(s, r, ?, t)$ to obtain entity predictions can be considered as a multi-classification task, where each class corresponds to an object entity. Similarly, the relation r prediction obtained from the query $(s, ?, o, t)$ to obtain a prediction of the relation r can also be considered as a multi-classification task, where each class corresponds to one relation. To learn the representation of weights and entities with relations, we use binary cross entropy as a loss function to obtain the optimal solution by minimizing the cross-entropy loss during the training period. Therefore, the equations for entity prediction loss L^e and relationship prediction loss L^r as:

$$L^e = \sum_{(s,r,o,t)\in\mathcal{G}} y_t^e log p(o|H_t, R_t, s, r) \tag{15}$$

$$L^r = \sum_{(s,r,o,t)\in\mathcal{G}} y_t^r log p(o|H_t, R_t, s, o) \tag{16}$$

where y_t^e are the label vectors for the two tasks, in which the element is 1 if the facts occur, otherwise 0. Based on the above loss function, the final loss function is defined as:

$$L = \lambda L^e + (1-\lambda) L^r \tag{17}$$

where λ is the hyper-parameter from 0 to 1 to balance the different losses.

4. Experiment

4.1. Experimental Setup

4.1.1. Datasets

We use five typical TKG datasets for evaluating our model, namely ICEWS14, ICEWS05-15, ICEWS18, WIKI, and YAGO. The first three datasets are from the Integrated Crisis Early Warning System (ICEWS) containing socio-political events from 2014, 2005 to 2015, and 2018, respectively, with a time granularity of days. WIKI and YAGO are two knowledge bases containing facts with temporal information, and we use a subset with a temporal granularity of years. We split the dataset into 80%/10%/10% for training/ validation/ testing. The Table 1 provides the statistics of these datasets.

4.1.2. Evaluation Metrics

Our experiments are used for the knowledge graph reasoning task with future timestamps, including entity prediction $(s, r, ?, t)$ and relation prediction $(s, ?, o, t)$ two subtasks. Mean Reciprocal Rank (MRR), Mean Rank (MR), and HITS@1/3 are used as evaluation metrics. Without loss of generality, only the experimental results under the original setup are reported. The reason for this is that for the quadruple query $(s, r, ?, t1)$ the missing object answer is o_1. It is assumed that (s, r, o_2, t_2) exists in the training set. For the previous study, its filtering settings would ignore temporal information, leading the model to incorrectly assume that o_2 is also the correct answer. In fact (s, r, o_2) may not occur at t_1. Thus, the filtering setting can lead to over-optimistic experimental effects and may result in incorrect higher rankings.

4.1.3. Baselines

We compared our model with static KG inference models and TKG reasoning models. For the static KG reasoning model, we chose three kinds of baselines.

(1) Typical static models. Including DistMult (García-Durán et al., 2018), Complex (Trouillon et al., 2016), ConvE (Dettmers et al., 2017) and RotatE (Sun et al., 2018). Note that these static

Table 1: Statistics of the datasets

Datasets	Entities	Relations	Train	Valid	Test	Time gap	Snapshot numbers
ICEWS18	23033	256	373018	45995	49545	24 hours	365
ICEWS14	7128	230	74845	8514	7371	24 hours	365
ICEWS05 - 15	7128	230	63685	13823	13222	24 hours	4017
WIKI	12554	24	539286	67538	63110	1 years	232
YAGO	10623	10	161540	19523	20026	24 hours	189

Table 2: Entity prediction on ICEWS18.

Model	MRR	HITS@1	HITS@3
DistMult	13.86	5.61	15.22
Complex	15.45	8.04	17.19
ConvE	22.81	13.63	25.83
RotatE	14.53	6.47	15.78
HyTE	7.41	3.10	7.33
TTransE	8.44	1.85	8.95
TA-DistMult	16.42	8.60	18.13
RGCRN	23.46	14.24	26.62
CyGNet	24.98	15.54	28.58
Re-Net	26.17	16.43	29.89
RE-GCN	29.16	19.14	33.05
DHE-TKG	**29.23**	**19.15**	**33.31**

Table 3: Entity prediction on ICEWS14.

Model	MRR	HITS@1	HITS@3
DistMult	20.32	6.13	27.59
Complex	22.61	9.88	28.93
ConvE	30.30	21.30	34.42
RotatE	25.71	16.41	29.01
HyTE	16.78	2.13	24.84
TTransE	12.86	3.14	15.72
TA-DistMult	26.22	16.83	29.72
RGCRN	33.31	24.08	36.55
CyGNet	34.68	25.35	38.88
Re-Net	35.77	25.99	40.10
TAE	35.80	26.09	40.17
RE-GCN	39.26	29.23	43.94
DHE-TKG	**40.02**	**30.13**	**44.99**

baseline methods are used without considering the temporal information in the input.

(2) Temporal models under the interpolation setting. Including HyTE (Dasgupta et al., 2018), TTransE (Leblay and Chekol, 2018), and TA-DistMult (García-Durán et al., 2018).

(3) TKG reasoning model. Including RGCRN (Zhao et al., 2022), CyGNet (Zhu et al., 2020), RE-NET (Jin et al., 2019), RE-GCN (Li et al., 2021), TAE (Duan et al., 2022) and HGAT (Shao et al., 2023).

To be fair, all methods in this paper do not use the static graph.

4.1.4. Implementation Details

The embedding dimension d is set to 200 on all datasets. The number of RGCN layers is set to 2. Dropout is set to 0.2. The value of the hyperparameter is determined based on the MRR performance of each validation set, where the hyperparameter α is set to 0.4 and 0.1 for the ICEWS05-15 and ICEWS18 datasets, respectively, and to 0.2 for the other three datasets. The model parameters were initialized using Xavier and then optimized with a learning rate of 0.01 using the Adam optimizer. For the hypergraph construction stage, the number of vertices in a k-NN hyperedge is set to 8. For ConvTransE, we follow the relevant settings mentioned in RE-GCN.

Table 4: Entity prediction on ICEWS05-15.

Model	MRR	HITS@1	HITS@3
DistMult	19.91	5.63	27.22
Complex	20.26	6.66	26.43
ConvE	31.40	21.56	35.70
RotatE	19.01	10.42	21.35
HyTE	16.05	6.53	20.20
TTransE	16.53	5.51	20.77
TA-DistMult	27.51	17.57	31.46
RGCRN	35.93	26.23	40.01
CyGNet	35.46	25.44	40.20
Re - Net	36.86	26.24	41.85
TAE	37.18	26.70	42.34
RE - GCN	44.54	33.55	50.62
DHE-TKG	**45.05**	**34.16**	**50.97**

Table 5: Entity prediction on WIKI.

Model	MRR	HITS@1	HITS@3
DistMult	27.96	32.45	39.51
ComplEx	27.69	31.99	38.61
ConvE	26.03	30.51	39.18
RotatE	26.08	31.63	38.51
HyTE	25.40	29.16	37.5
TTransE	20.66	23.88	33.04
TA-DistMult	26.44	31.36	38.9
RGCRN	28.68	31.44	38.58
CyGNet	30.77	33.83	41.19
RE-Net	30.87	33.55	41.27
TAE	31.27	34.39	41.67
RE-GCN	50.97	57.24	68.40
DHE-TKG	**51.20**	**57.47**	**69.25**

Table 6: Entity prediction on YAGO.

Model	MRR	HITS@1	HITS@3
DistMult	44.05	49.70	59.94
ComplEx	44.09	49.57	59.64
ConvE	41.22	47.03	59.90
RotatE	42.08	46.77	59.39
HyTE	14.42	39.73	46.98
TTransE	26.10	36.28	47.73
TA-DistMult	44.98	50.64	61.11
RGCRN	43.71	48.53	56.98
CyGNet	46.72	52.48	61.52
RE-Net	46.81	52.71	61.93
TAE	47.03	52.82	62.88
RE-GCN	62.65	70.54	82.04
DHE-TKG	**62.93**	**71.00**	**82.72**

Table 7: Relation prediction

Model	RGCRN	REGCN	DHE-TKG
ICEWS18	37.14	39.00	**39.42**
ICEWS14	38.04	39.24	**40.11**
ICEWS05-15	38.37	**39.48**	39.44
WIKI	88.88	98.13	**98.18**
YAGO	90.18	**94.18**	94.07

4.2. Results

4.2.1. Results on Entity Prediction

Table 2-Table 6 shows the experimental results of our model relative to the baseline approach for the entity prediction task. Our model demonstrates its effectiveness on the ICEWS dataset as well as the WIKI and YAGO datasets. Specifically, our model significantly outperforms the static models because it captures temporal dynamics and can effectively use temporal information. Moreover, our model outperforms those temporal methods. For the temporal approach baseline, CyGNet, RE-NET, and RE-GCN take into account the fact of adjacent timestamps and show strong performance in experiments. However, unlike these individual methods for modeling low-dimensional interactions, our model uses hypergraphs to capture higher-order interaction information and is able to discover potential connections and common features among entities.

4.2.2. Results on Relation Prediction

As shown in the Table 7, we selected the temporal model baselines that can be used for relational prediction for comparison. DHE-TKG outperforms most of the baselines. The main reason is that entity prediction and relationship prediction complement each other and enhance each other. The multi-task learning scheme used in this paper effectively combines the goals of the two tasks and improves the effectiveness of the model in each task.

4.3. Ablation Study

In this subsection, we studied the contribution of different model components of our model through ablation experiments. We compare the performance by tuning the model components on ICEWS14 and the results are shown in Table 8.

Table 8: Ablation studies on ICEWS14.

Method	MRR	Hits@1	Hits@3
-loc	39.38	29.12	44.56
+mean	39.74	29.78	44.87
DHE-TKG	**40.02**	**30.13**	**44.99**

4.3.1. Location Information Component

We compare adding location information with no location information, which is denoted by -loc in the Table 8. The performance without adding location information decrease, which is in line with our expectation for the location information component to keep the input graph from losing order information in the DHE-TKG.

4.3.2. Dynamically Weighted Components

We use the averaging method instead of the dynamic weighting method, which is denoted by +mean in the Table 8. The results in Table 8 show that the dynamic weighting component has the positive impact on the results, which indicates that the input history graph does not have an equal impact on the inference.

4.4. Analysis of Gating in DHE-TKG

In this section, we explore the impact of different activation functions (Gating) in the dynamic weighting module by conducting experiments on the dataset ICEWS14, as shown in Table 9. Overall, the sigmoid function performs better. One reason is that the sigmoid function can give the model more options to make decisions with higher confidence for the current sample and can divide the differences between all options.

Table 9: Gating analysis

Method	MRR	Hits@1	Hits@3
ReLU	39.64	29.49	44.42
Tanh	39.76	29.93	44.44
GELU	39.63	29.66	44.57
Sigmoid	**40.02**	**30.13**	**44.99**

4.5. Visualization of Weight Distribution

In order to observe the effect of history graphs with different timestamps on answer inference for different datasets, we have displayed the dynamic

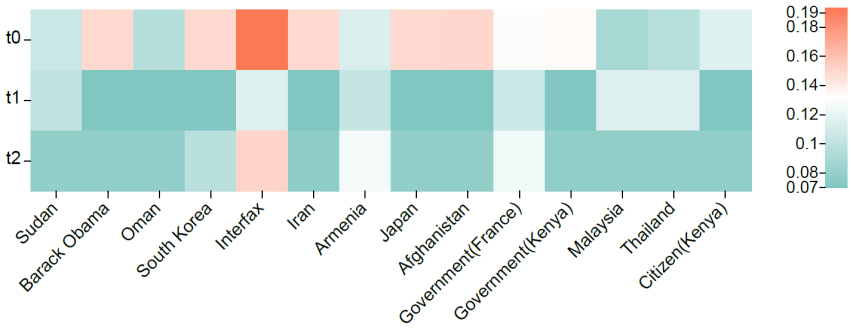

Figure 5: Visualization example of dynamic attention.

weights. We took the input time step of 3 as an example and selected some entities that successfully reasoned and built a heat map of the dynamic weight distribution on the ICEWS14 dataset. As shown in Figure 5, it can be seen that different dynamically weighted components enable different entities to dynamically determine the weights at the three timestamps. And it can be observed that most of the entities on the ICEWS14 dataset choose weights that are more biased towards the timestamp with the forward input.

4.6. Sensitivity Analysis

We explore the importance of the evolutionary embedding versus the dynamic hypergraph embedding through experiments that adjust the magnitude of the hyper-parameter α in Equation 12. We report the change in the performance of our model on the ICEWS14 dataset and ICEWS05-15. As shown in Figure 6, which shows the change in α from 0 to 1 performance of the model. This demonstrates that neither ignoring low-order interaction information, nor high-order interaction information can make a valid inference. This demonstrates the necessity of blending low-order information with higher-order information. By adjusting α, the best balance of low-order and high-order information can be achieved.

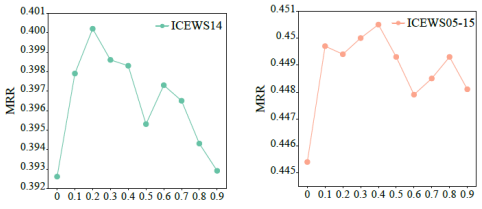

Figure 6: Performance of DHE-TKG with α variation on ICEWS14 and ICEWS05-15 dataset.

4.7. Case Study

We study a query (*France*, *Make_statement*, ?, 338) from the ICEWS14 test set. As shown in Figure 7, DHE-TKG predicts this query by entering the history graphs with a time step of 3. The left side of the figure shows a subgraph consisting of selected facts related to the query for timestamps 335 through 337. For the sake of direct observation, we omit the relationships in the graphs. We observe that subject **France** in the query is not directly related to the answer **Iran** in the facts of the input history graphs. However, frequent interactions between **Iran** and Iraq occur within these three timestamps. The right side of the figure shows the higher-order interactions between countries established by the hypergraph, and we observe that **France**, **Iran**, and **Iraq** are all connected by the hyperedges. DHE-TKG successfully reasons the query answer **Iran** which occurs at 338 timestamp. Compared with RE-GCN, we find that RE-GCN does not reason about this query because of the lack of capturing higher-order interactions.

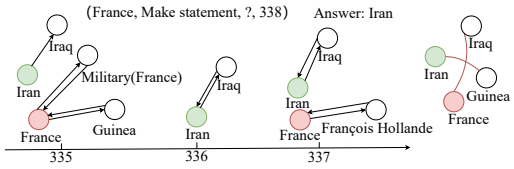

Figure 7: An example of DHE-TKG capturing higher-order information on the ICEWS14 dataset.

5. Conclusion

We propose a new TKG representation learning model for temporal reasoning called DHE-TKG. DHE-TKG effectively captures higher-order correlations by combining low-order and high-order information for representation learning. It also incorporates temporal location information to fusion node features dynamically to determine the importance of the input history graph. The experimental results show the significant advantages of our model for entity prediction and relationship prediction on TKG. Moreover, it is applicable to complex data and is able to incorporate higher-order data correlations into representation learning.

6. Acknowledgements

This work is supported by the Social Science Planning Foundation of Liaoning Province under Grant L21CXW003.

7. References

Devanshu Arya, Deepak K. Gupta, Stevan Rudinac, and Marcel Worring. 2020. Hypersage: Generalizing inductive representation learning on hypergraphs. *ArXiv*, abs/2010.04558.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Pratim Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Conference on Empirical Methods in Natural Language Processing*.

Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Dengyong Zhou and Jiayuan Huang and Bernhard Schölkopf. 2006. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. In *AAAI Conference on Artificial Intelligence*.

Hao Duan, Haoyu Jin, Kang Chen, Shaochong Du, Tao Fang, and Hong Huo. 2022. An effective time-aware encoder for temporal knowledge graph reasoning. In *Proceedings of the 5th International Conference on Machine Learning and Natural Language Processing*, pages 81–87.

Yifan Feng, Haoxuan You, Zizhao Zhang, R. Ji, and Yue Gao. 2018. Hypergraph neural networks. In *AAAI Conference on Artificial Intelligence*.

Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Conference on Empirical Methods in Natural Language Processing*.

Yuchi Huang, Qingshan Liu, and Dimitris N. Metaxas. 2009. Video object segmentation by hypergraph cut. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1738–1745.

Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. 2019. Dynamic hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2019. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *Conference on Empirical Methods in Natural Language Processing*.

Thomas Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. *ArXiv*, abs/2004.04926.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*.

Yujia Li, Shiliang Sun, and Jing Zhao. 2022. Tirgn: Time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In *International Joint Conference on Artificial Intelligence*.

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutional representation learning.

Apoorv Saxena, Soumen Chakrabarti, and Partha P. Talukdar. 2021. Question answering over temporal knowledge graphs. In *Annual Meeting of the Association for Computational Linguistics*.

M. Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. In *Extended Semantic Web Conference*.

Pengpeng Shao, Jiayi He, Guanjun Li, Dawei Zhang, and Jianhua Tao. 2023. Hierarchical graph attention network for temporal knowledge graph reasoning. *Neurocomputing*, 550:126390.

Xin Song, Haiyang Wang, Kang Zeng, Yujia Liu, and Bin Zhou. 2021. Kat-gcn: Knowledge-aware attention based temporal graph convolutional network for multi-event prediction. In *International*

Conference on Software Engineering and Knowledge Engineering.

Zhiqing Sun, Zhihong Deng, Jian-Yun Nie, and Jian Tang. 2018. Rotate: Knowledge graph embedding by relational rotation in complex space. *ArXiv*, abs/1902.10197.

Rakshit S. Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Knowevolve: Deep reasoning in temporal knowledge graphs. *ArXiv*, abs/1705.05742.

Rakshit S. Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Pratim Talukdar. 2018. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Neural Information Processing Systems*.

Bishan Yang, Wen Tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575.

Wei Zhao, Shiqi Zhang, Bing Zhou, and Bei Wang. 2022. Residual graph convolutional recurrent networks for multi-step traffic flow forecasting. *CoRR*, abs/2205.01480.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhan. 2020. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *AAAI Conference on Artificial Intelligence*.

Xiaohan Zou. 2020. A survey on application of knowledge graph. *Journal of Physics: Conference Series*, 1487.