Modeling Precursors for Temporal Knowledge Graph Reasoning via Auto-encoder Structure

Yifu Gao[1] , **Linhui Feng**[1] , **Zhigang Kan**[1] , **Yi Han**[2] , **Linbo Qiao**[1*] and **Dongsheng Li**[1*]
[1]Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha, China
[2]College of Meteorology and Oceanography, National University of Defense Technology
{gaoyifu, linhuifeng, kanzhigang13, hanyi12, qiao.linbo, dsli}@nudt.edu.cn

Abstract

Temporal knowledge graph (TKG) reasoning that infers missing facts in the future is an essential and challenging task. When predicting a future event, there must be a narrative evolutionary process composed of closely related historical facts to support the event's occurrence, namely fact precursors. However, most existing models employ a sequential reasoning process in an auto-regressive manner, which cannot capture precursor information. This paper proposes a novel auto-encoder architecture that introduces a **r**elation-aware **g**raph **a**ttention **l**ayer into **t**ransformer (rGalT) to accommodate inference over the TKG. Specifically, we first calculate the correlation between historical and predicted facts through multiple attention mechanisms along intra-graph and inter-graph dimensions, then constitute these mutually related facts into diverse fact segments. Next, we borrow the translation generation idea to decode in parallel the precursor information associated with the given query, which enables our model to infer future unknown facts by progressively generating graph structures. Experimental results on four benchmark datasets demonstrate that our model outperforms other state-of-the-art methods, and precursor identification provides supporting evidence for prediction.

1 Introduction

The knowledge graph is crucial to many artificial intelligence (AI) applications. Since most knowledge graphs are far from complete, knowledge graph reasoning becomes a critical task and has been extensively studied on static graphs [Yang *et al.*, 2015]. However, the rapidly growing facts on the knowledge graph show complex dynamic characteristics, which creates the need for introducing the concept of temporal knowledge graph (TKG) to infer missing facts. Unlike the static KG, the fact on TKG is composed of a quadruple, i.e., (subject, relation, object, timestamp). Due to the incompleteness and dynamic changes in a graph structure, TKG reasoning is a complex task worth researching.

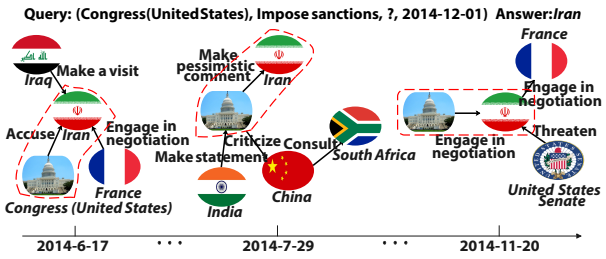


Figure 1: An example of fact precursors related to the query on the ICEWS14 dataset.

Given a TKG from t_0 to t_T, there are two main settings for TKG reasoning, interpolation and extrapolation [Jin *et al.*, 2020]. The former setting aims to learn dynamic representations and infer missing facts for timestamp $t_0<t<t_T$, while the latter attempts to predict future facts for timestamp $t>t_T$, even the corresponding graph snapshots within the time interval are unavailable. A general solution to the extrapolation reasoning problem is to learn the dynamic representation of all historical entities and adopt a temporal point process to predict future facts [Trivedi *et al.*, 2017; Trivedi *et al.*, 2019]. However, they can not model concurrent events occurring within the same time window.

Some recent attempts capture the structural dependencies among concurrent facts and sequential patterns across temporally adjacent facts in an auto-regressive manner. Specifically, RE-NET [Jin *et al.*, 2020] adopts RGCN [Schlichtkrull *et al.*, 2018] to aggregate concurrent events within the same timestamp and summarizes information of the past event sequences with GRU [Cho *et al.*, 2014]. HIP [He *et al.*, 2021] comprehensively transmits historical information in terms of time, structure, and repetition through three novel scoring functions. Instead of encoding related facts for the given query, RE-GCN [Li *et al.*, 2021b] treats TKG as a KG sequence and encodes all historical facts into entity and relation evolutionary representations.

Another line of method argues that only a few historical facts are useful for a specific prediction, which is the most related work with us. CyGNet [Zhu *et al.*, 2021] uses a copy-generation mechanism to model the repetitive historical facts. Nevertheless, only considering 1-hop repetitive paths will lose massive useful query-relevant information. Thus, CluSTeR [Li *et al.*, 2021a] extracts both 1-hop repetitive and

*Corresponding Author

non-repetitive paths related to the query and performs temporal reasoning on the sequence of subgraphs constructed from clues. Furthermore, xERTE [Han *et al.*, 2021] employs a sequential reasoning process over local inference subgraphs and provides interpretability for their predictions. However, they both use heuristic rules to focus on single independent causes close to the current timestamp, making it challenging to discover latent connections between relevant historical facts.

In fact, there must have been an evolutionary process associated with the predicted fact along history to suggest the event's occurrence. For example, as shown in Figure 1, when facing the query *(Congress (United States), Impose sanctions, ?, 2014-12-01)*, several historical facts (red dashed lines) involved by the subject entity of given query constitute a narrative fact segment, which helps people capture the evolutionary law behind the *Impose sanctions* event and find the answer *Iran* to the query. We discover this underlying phenomenon and introduce the concept of fact precursors to describe these query-relevant historical facts with either continuous or discontinuous timestamps. However, previous models adopt a sequential reasoning structure, which can not capture the close connections between historical facts. Recently transformer [Vaswani *et al.*, 2017] has been widely used in various fields, such as computer vision and machine translation, thanks to its ability to compute the correlation between arbitrary positions in a sequence. Inspired by this mechanism, we treat the TKG as a static KG sequence and first propose an auto-encoder architecture, rGalT, which employs transformer based on a relation-aware graph attention layer (RGAL) to fully explore the precursor information related to the predicted fact. More specifically, in the intra-graph component, we use RGAL to calculate query-dependent attention scores by considering the entity and relation factors of the given query and guide the model to aggregate neighbor information of the query's interest within the same timestamp. As for the inter-graph part, we adopt a transformer encoder to discover diverse fact segments within the graph snapshot sequence. Finally, the generative method in the inference component enables our model to decode precursor information related to the given query and infer future missing facts.

Overall, this paper makes the following contributions: 1) We introduce the concept of fact precursors in the TKG reasoning task to reveal the underlying evolutionary patterns behind the event. Fact precursors can be viewed as a narrative fact segment composed of several query-relevant historical facts with continuous or discontinuous timestamps. 2) To the best of our knowledge, we are the first to propose an auto-encoder structure, rGalT, to accommodate extrapolation inference over the TKG. Compared to traditional auto-regressive methods, rGalT can more fully capture interactions between fact precursors. 3) Our experiments achieve state-of-the-art performance in four widely used datasets, thus demonstrating the validity of our model. Precursor identification can provide supporting evidence for the results.

2 Related Work

Static KG reasoning. In recent years, we have witnessed increasing interest in knowledge reasoning. Furthermore, there are three main categories: KG embedding-based reasoning, rule-based reasoning, and relation path-based reasoning. KG embedding aims to map the entities and relations into continuous vector space and score the probabilities with embeddings, such as the translating model [Bordes *et al.*, 2013] and the semantic matching model [Yang *et al.*, 2015; Trouillon *et al.*, 2016]. Rule-based methods utilize the instantiation rules [Galárraga *et al.*, 2015] or inject the rules into models [Guo *et al.*, 2018; Zhang *et al.*, 2019], to improve reasoning accuracy. Moreover, relation path-based reasoning finds the target answer on the graph with reinforcement learning [Xiong *et al.*, 2017]. However, static KG reasoning methods ignore the temporal information, which makes them not applicable to temporal knowledge graph reasoning.

Temporal KG reasoning. There are two main settings for TKG reasoning, interpolation and extrapolation. The former setting [Jiang *et al.*, 2016; Dasgupta *et al.*, 2018; García-Durán *et al.*, 2018] aims to learn dynamic representations and infer missing facts for the historical timestamps. In contrast, this paper focuses more on the latter setting, which attempts to predict facts in the future. Previous models [Trivedi *et al.*, 2017; Trivedi *et al.*, 2019] consider all historical facts and adopt a temporal point process to infer the occurrence of future facts. However, they can not model concurrent facts at the same timestamps. Accordingly, some earlier attempts consider structural and temporal patterns over the TKG. RE-NET [Jin *et al.*, 2020] uses an aggregator and GRU to transmit information of the past event sequences sequentially. HIP [He *et al.*, 2021] passes historical information selectively from temporal, structural, and repetitive perspectives. Instead of facing each query, RE-GCN [Li *et al.*, 2021b] encodes all historical facts into entity and relation evolutionary representation for future fact prediction. Besides, some recent attempts extract some related historical information for each query. CyGNet [Zhu *et al.*, 2021] proposes a copy-generation method to model repetitive facts with the same entity and relation as each query. CluSTeR [Li *et al.*, 2021a] uses reinforcement learning to search for 1-hop repetitive and non-repetitive facts related to the query and constructs them into temporal subgraphs for reasoning. Moreover, xERTE [Han *et al.*, 2021] conducts a sequential reasoning process on query-relevant subgraphs that are dynamically expanded by an iterative sampling of temporal neighbors and temporal relational attention propagation. The inference graph can be seen as a graphical explanation. However, most of the above models use an auto-regressive approach, which makes it challenging to capture the underlying evolutionary process behind the occurrence of events.

3 Our Model

In this section we start with the notation and task definition, then we provide an overview in Section 3.2 and explain each module from Section 3.3 to 3.5.

3.1 Notations and Task Definition

A temporal knowledge graph (TKG) $\mathcal{G}$ can be viewed as a multi-relational, directed graph with timestamped edges between nodes (entities), which can be formalized as a sequence

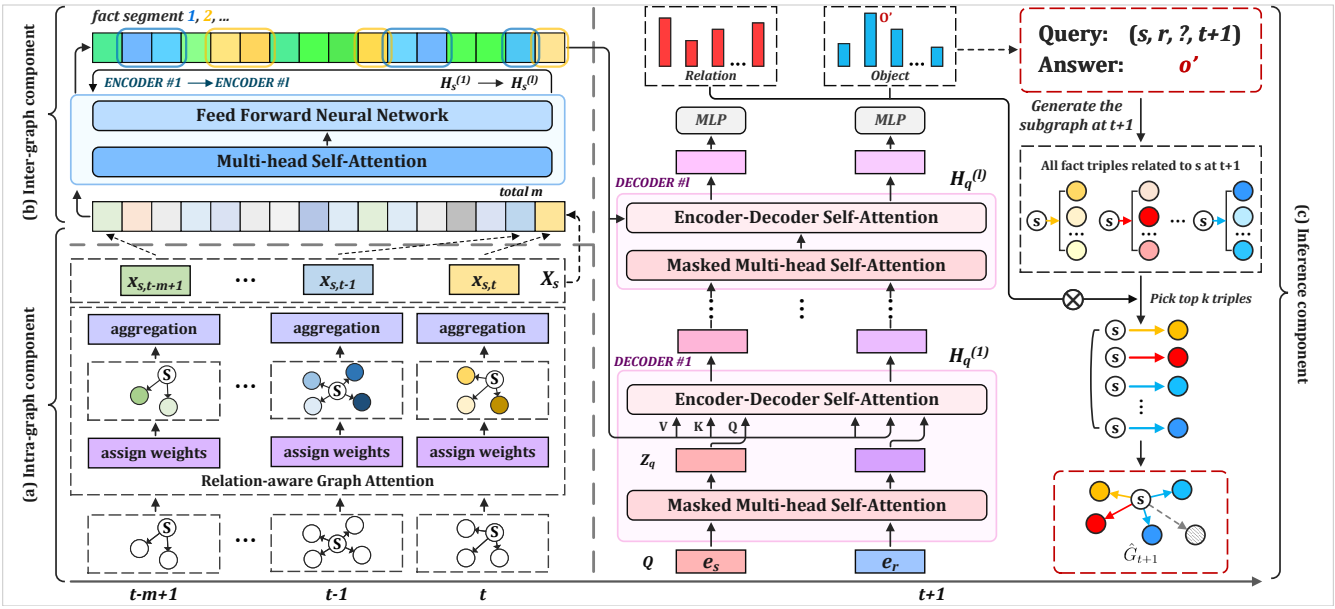


Figure 2: The framework of our rGalT model consists of three modules. The intra-graph component to capture query-relevant facts within the same timestamp. The inter-graph component to discover multiple fact fragments from the graph snapshot sequence. And the inference component to identify and decode precursor information that is used to infer missing facts and generate subgraphs.

of knowledge graph snapshots ordered by timestamp, i.e., $\mathcal{G} = \{G_1, G_2, \ldots, G_t, \ldots\}$. Each event (fact) in the snapshot G_t can be represented as a quadruple (s, r, o, t) or (s_t, r_t, o_t), corresponding to subject entity $s \in \mathcal{E}$, relation type $r \in \mathcal{R}$, object entity $o \in \mathcal{E}$ and timestamp $t \in \mathcal{T}$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}$ represent the sets of entities, relationships and timestamps, respectively.

The purpose of the extrapolation reasoning task over TKG is to infer the missing object entity o given the query $(s, r, ?, t + \Delta t)$, or missing subject entity s of the query $(?, r, o, t + \Delta t)$. It is worth noting that the missing entity s or o on $G_{t+\Delta t}$ is inferred using the historical facts of $\{G_1, G_2, \ldots, G_t\}$, while the event information in the time period Δt is unknown. Specifically, following [Jin *et al.*, 2020], we only use the ground truth of the training set when making inferences on the validation and test sets.

3.2 Model Overview

As shown in Figure 2, our model consists of three parts: Intra-graph Component, Inter-graph Component, and Inference Component.

At the intra-graph component, we use the relation-aware graph attention layer to compute an attention score for each query-relevant entity and then perform weighted aggregation of these neighboring entities. As for the inter-graph component, we employ the transformer encoder to interact with facts at different timestamps, with the aim of capturing multiple fact fragments from the graph sequence. Finally, at the inference component, we adopt the masked self-attention layer and encoder-decoder attention layer of the transformer decoder to identify fact segments with high relevance to the given query, called fact precursors. Then we decode in parallel the relation and object entity associated with the given query to predict what happens next.

3.3 Intra-graph Component

In this section, we propose a relation-aware graph attention layer (RGAL) to calculate the query-dependent score for each historical fact that the subject entity of the given query participates in. Unlike RGCN [Schlichtkrull *et al.*, 2018], RGAL assigns different correlation weights to neighboring entities within the same timestamp by considering the entity and relation factors of the given query.

In addition, considering that the fact (event) meanings vary with the relationship, we incorporate the relation embedding of each query-relevant fact into the attention function as shown below:

$$u_{so} = \boldsymbol{a}^T \sigma\left(\boldsymbol{W}\left[\boldsymbol{e}_o \| \boldsymbol{e}_r \| \boldsymbol{e}_s \| \boldsymbol{e}_{r_q}\right]\right), \tag{1}$$

where u_{so} is the attention score of the fact (s, r, o) in graph snapshot G relative to the query $(s, r_q, ?, t+1)$; $\boldsymbol{e}_s, \boldsymbol{e}_o \in \mathbb{R}^d$ are embeddings corresponding to subject entity and object entity, and $\boldsymbol{e}_r, \boldsymbol{e}_{r_q} \in \mathbb{R}^d$ are the relation embeddings; $\|$ is the concatenation operation and $\boldsymbol{W} \in \mathbb{R}^{d \times 4d}$ is a weight matrix applied to each node in the graph; $\boldsymbol{a} \in \mathbb{R}^d$ is used as a feed-forward layer to parameterize the attention function and σ is a tanh non-linearity function to compute attention weights. We compute the normalized attention score using the softmax function and obtain the final output $\boldsymbol{x}_s$ as follows:

$$\alpha_{so} = \frac{\exp\left(u_{so}\right)}{\sum_{w \in \mathcal{N}_s} \exp\left(u_{sw}\right)}, \quad \boldsymbol{x}_s = \sum_{o \in \mathcal{N}_s} \alpha_{so} \boldsymbol{e}_o, \tag{2}$$

where $\mathcal{N}_s$ is the set of immediate neighbors of entity node s, and α_{so} denotes the correlation fraction of o with respect to s in snapshot G. The output representation $\boldsymbol{x}_s \in \mathbb{R}^d$ aggregates factual information about graph snapshot G associated with the given query.

3.4 Inter-graph Component

We find that there exists a tighter connection called the fact segment within the historical fact sequence. To capture various fact segments, we use the encoder part of transformer to dynamically interact with graph aggregation information at different timestamps. Specifically, for the entity s of given query, the encoder maps an input sequence of aggregated representations $\{\boldsymbol{x}_{s,t-m+1}, \boldsymbol{x}_{s,t-m+2}, \ldots, \boldsymbol{x}_{s,t}\}$, $\boldsymbol{x}_s \in \mathbb{R}^d$, to a sequence of continuous representations $\{\boldsymbol{h}_{s,t-m+1}, \boldsymbol{h}_{s,t-m+2}, \ldots, \boldsymbol{h}_{s,t}\}$, $\boldsymbol{h}_s \in \mathbb{R}^d$, where $\boldsymbol{x}_s$ for each timestamp is obtained by (2), m represents the previous time steps related to the entity s. We denote the input and output sequences as $\boldsymbol{X}_s, \boldsymbol{H}_s \in \mathbb{R}^{m \times d}$ respectively.

As shown in Figure 2(b), each layer of the transformer encoder is identical and divided into two sub-layers, a multi-head self-attention layer and a feed-forward network. For the first sub-layer, we set up the self-attention mechanism of K heads to capture richer position information. And we use three independent linear transformation matrices $\boldsymbol{W}_q^i, \boldsymbol{W}_k^i, \boldsymbol{W}_v^i \in \mathbb{R}^{d \times d'}$ to transform the input representations $\boldsymbol{X}_s$ into queries, keys, and values of the i-th scaled dot-product attention head ($i = 1, 2, \ldots, K$). The specific model details are shown below:

$$\boldsymbol{Z}_s^i = \text{Softmax}(\frac{\left(\boldsymbol{X}_s \boldsymbol{W}_q^i\right)\left(\boldsymbol{X}_s \boldsymbol{W}_k^i\right)^T}{\sqrt{d'}})(\boldsymbol{X}_s \boldsymbol{W}_v^i), \tag{3}$$

where $\boldsymbol{Z}_s^i \in \mathbb{R}^{m \times d'}$ is the output representation of corresponding attention head, $d' = \frac{d}{K}$. When we get the output sequence of all attention heads in parallel, we cascade them to get the final output $\boldsymbol{Z}_s \in \mathbb{R}^{m \times d}$:

$$\boldsymbol{Z}_s = \text{Concat}(\boldsymbol{Z}_s^1, \boldsymbol{Z}_s^2, \ldots, \boldsymbol{Z}_s^K). \tag{4}$$

The second sub-layer is a fully connected feed-forward network, which consists of two linear transformations with a ReLU activation in between:

$$\boldsymbol{H}_s = \text{ReLU}(\boldsymbol{Z}_s \boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{5}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d \times d_f}, \boldsymbol{b}_1 \in \mathbb{R}^{d_f}$ are the parameters of the first linear layer, and $\boldsymbol{W}_2 \in \mathbb{R}^{d_f \times d}, \boldsymbol{b}_2 \in \mathbb{R}^d$ are the parameters of second. Although the model network is the same for each layer of the transformer encoder, different parameters are used from layer to layer. We use the output of the feed-forward network at the last layer l, $\boldsymbol{H}_s^{(l)}$, as input to the next module, where l is the number of layers in the encoder. $\boldsymbol{H}_s^{(l)}$ is a sequence of hidden states containing information about different fact segments. Similar to the transformer [Vaswani *et al.*, 2017], we employ a residual connection around each sub-layer, followed by layer normalization, and use the same positional encoding to learn the sequential temporal information at different time steps.

3.5 Inference Component

In order to find the fact precursors related to the given query $(s, r, ?, t+1)$, we use the first two sub-layers of decoder, masked multi-head self-attention, and encoder-decoder self-attention, to model the sequence of hidden states $\boldsymbol{H}_s^{(l)}$ from the previous module. Then we decode the precursor information in parallel by MLP to get the probabilities of relations and object entities, respectively.

First, we take the embeddings of the entity $\boldsymbol{e}_s \in \mathbb{R}^d$ and the relation $\boldsymbol{e}_r \in \mathbb{R}^d$ as the input sequence $\boldsymbol{Q} \in \mathbb{R}^{n \times d}$ of the masked multi-head self-attention layer, which differs from Eq. 3 by adding the mask matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$. The specific function is shown below:

$$\boldsymbol{Z}_q^i = \text{Softmax}(\frac{\left(\boldsymbol{Q}\boldsymbol{W}_q^i\right)\left(\boldsymbol{Q}\boldsymbol{W}_k^i\right)^T}{\sqrt{d'}} + \boldsymbol{M})(\boldsymbol{Q}\boldsymbol{W}_v^i), \tag{6}$$

where $\boldsymbol{Z}_q \in \mathbb{R}^{n \times d}$ is the final output of this layer after cascading the outputs of each head using Eq. 4, the meanings of the parameters are consistent with those of the self-attention mechanism in the above section. Primarily, $\boldsymbol{M}$ matrix is represented as follows:

$$\boldsymbol{M}[a, b] = \begin{cases} 0, & \text{if } a \geq b \\ -\infty, & \text{others.} \end{cases}, \tag{7}$$

here, when $\boldsymbol{M}[a, b] = -\infty$, the softmax function makes the attention weights to zero, preventing the query's subject entity from computing weight information for the relation in the query.

For the multi-head self-attention layer, we treat the output $\boldsymbol{Z_q}$ of the first masked self-attention layer as queries, and the encoder's outputs $\boldsymbol{H}_s^{(l)}$ are keys and values. We obtain the output of each head $\boldsymbol{H}_q^i \in \mathbb{R}^{n \times d'}$ by the following formula:

$$\boldsymbol{H}_q^i = \text{Softmax}(\frac{(\boldsymbol{Z}_q \boldsymbol{W}_q^i)(\boldsymbol{H}_s^{(l)} \boldsymbol{W}_k^i)^T}{\sqrt{d'}})(\boldsymbol{H}_s^{(l)} \boldsymbol{W}_v^i), \tag{8}$$

and we also get the concatenation form $\boldsymbol{H}_q \in \mathbb{R}^{n \times d}$ of all heads by Eq. 4. This process matches the encoder's output $\boldsymbol{H}_s^{(l)}$ to the decoder's input $\boldsymbol{Z}_q$, allowing the subject entity and relation of the predicted fact to decide which fact segment is relevant to themselves. Hence, $\boldsymbol{h}_{q,r}, \boldsymbol{h}_{q,o} \in \mathbb{R}^d$ are decoupled hidden states contained in the $\boldsymbol{H}_q$, corresponding to the outputs of subject entity s and relation r at this sub-layer. Given a query $(s, r, ?, t+1)$, we feed the above representation at the l layer into a multi-layer perceptron (MLP) decoder parameterized by $\boldsymbol{w}_o$ to predict the occurrence probabilities of all entities at timestamp $t+1$, i.e.,

$$p\left(\mathbf{o} \mid s, r, G_{t-m+1:t}\right) = \text{Softmax}([\boldsymbol{e}_s : \boldsymbol{h}_{q,o}^{(l)} : \boldsymbol{e}_r]^T \cdot \boldsymbol{w}_o), \tag{9}$$

where $\boldsymbol{w}_o \in \mathbb{R}^{3d \times |\mathcal{E}|}$, $\boldsymbol{e}_s$, $\boldsymbol{e}_r$ are learnable embedding vectors. $\boldsymbol{h}_{q,o}^{(l)}$ is the object entity representation of precursor information. Similarly, we define probabilities of all relations associated with the subject entity s as follows:

$$p\left(\mathbf{r} \mid s, G_{t-m+1:t}\right) = \text{Softmax}([\boldsymbol{e}_s : \boldsymbol{h}_{q,r}^{(l)}]^T \cdot \boldsymbol{w}_r), \tag{10}$$

where $\boldsymbol{w}_r \in \mathbb{R}^{2d \times |\mathcal{R}|}$, $\boldsymbol{h}_{q,r}^{(l)}$ is the relation representation of precursor information.

With the above three modules, we can infer the missing entity o of the given query $(s, r, ?, t+1)$, provided that the graph snapshot sequence $\{G_1, G_2, \ldots, G_t\}$ is known. While the extrapolation reasoning task is to infer the missing entity at the future moment $t + \Delta t$, even if the ground truth within Δt is unknown. Therefore we propose a subgraph generation algorithm, which can generate new graphs sequentially within Δt to handle the multi-step reasoning problem. The exact procedure of the algorithm is described in Appendix A.

Method	ICEWS18				ICEWS14				ICEWS05-15				GDELT				YAGO		
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@3	Hits@10
DistMult	13.86	5.61	15.22	31.26	20.32	6.13	27.59	46.61	19.91	5.63	27.22	47.33	8.61	3.91	8.27	17.04	44.05	49.70	59.94
R-GCN	15.05	8.13	16.49	29.00	28.03	19.42	31.95	44.83	27.13	18.83	30.41	43.16	12.17	7.40	12.37	20.63	20.25	24.01	37.30
ConvE	22.81	13.63	25.83	41.43	30.30	21.30	34.42	47.89	31.40	21.56	35.70	50.96	18.37	11.29	19.36	32.13	41.22	47.03	59.90
RotatE	14.53	6.47	15.78	31.86	25.71	16.41	29.01	45.16	19.01	10.42	21.35	36.92	3.62	0.52	2.26	8.37	42.08	46.77	59.39
HyTE	7.41	3.10	7.33	16.01	16.78	2.13	24.84	43.94	16.05	6.53	20.20	34.72	6.69	0.01	7.57	19.06	14.42	39.73	46.98
TTransE	8.44	1.85	8.95	22.38	12.86	3.14	15.72	33.65	16.53	5.51	20.77	39.26	5.53	0.46	4.97	15.37	26.10	36.28	47.73
TA-DistMult	16.42	8.60	18.13	32.51	26.22	16.83	29.72	45.23	27.51	17.57	31.46	47.32	10.34	4.44	10.44	21.63	44.98	50.64	61.11
Know-Evolve	7.41	3.30	7.87	14.76	–	–	–	–	–	–	–	–	15.88	11.70	15.69	22.28	5.23	5.63	10.23
RGCRN	23.46	14.24	26.62	41.96	33.31	24.08	36.55	51.54	35.93	26.23	40.02	54.63	18.63	11.53	19.80	32.42	43.71	48.53	56.98
CyGNet	24.98	15.54	28.58	43.54	34.68	25.35	38.88	53.16	35.46	25.44	40.20	54.47	18.05	11.13	19.11	31.50	46.72	52.48	61.52
RE-NET	26.67	16.98	30.23	45.66	36.27	26.65	40.69	54.21	36.65	26.14	41.64	56.79	19.40	11.91	20.47	33.68	46.81	52.71	61.93
RE-GCN	27.45	17.70	31.06	46.42	37.44	27.43	41.66	57.02	37.96	26.89	43.34	58.90	19.01	11.81	20.31	32.98	**58.20**	**65.49**	**75.74**
rGalT	**27.88**	**18.01**	**31.59**	**47.02**	**38.33**	**28.57**	**42.86**	**58.13**	**38.89**	**27.58**	**44.19**	**59.10**	**19.56**	**12.11**	**20.89**	**34.15**	51.45	57.76	68.31

Table 1: Performance comparison on TKG reasoning under the raw setting.

Method	ICEWS05-15			ICEWS18			GDELT		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
xERTE	39.3	29.6	58.5	27.9	19.1	45.9	18.1	11.2	31.7
CluSTeR	**44.6**	**34.9**	**63.0**	**32.3**	**20.6**	**55.9**	18.3	11.6	31.9
rGalT w.GT	41.8	31.3	61.6	30.3	19.4	48.5	**20.3**	**13.1**	**35.3**

Table 2: Performance comparison on TKG reasoning with ground truth under the raw setting.

3.6 Training Objective

Giving a query $(s, r, ?, t')$ to infer missing object entity o can be considered as a multi-class classification task, where each class corresponds to each entity. Similarly, relation prediction of a given subject entity s can also be viewed as a multi-class classification task where categories correspond to each relation. Here the loss function can be defined as follows:

$$\mathcal{L} = -\sum_{(s,r,o,t')} \log p\left(o_{t'}|s_{t'}, r_{t'}\right) + \lambda \log p\left(r_{t'}|s_{t'}\right), \qquad (11)$$

where $(s, r, o, t') \in G_{1:t}$ represents the known facts in the training set, $p(*)$ are the probability scores obtained from Eq.9 and Ep.10, and λ is a hyperparameter that regulates the importance of the two classification tasks.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics. There are five typical datasets commonly used in previous studies [Li *et al.*, 2021a; Li *et al.*, 2021b], namely, ICEWS14, ICEWS05-15, ICEWS18 [Boschee *et al.*, 2015], GDELT [Leetaru and Schrodt, 2013] and YAGO [Mahdisoltani *et al.*, 2015]. More details about datasets are shown in Table 4 of Appendix B. We split all datasets into three sets, with a proportion of train(80%), valid(10%) and test(10%) by timestamps following [Li *et al.*, 2021b]. Mean Reciprocal Rank (MRR) and Hits@{1,3,10} are employed as the metrics in our experiments. We compare the experimental results under the raw setting [Li *et al.*, 2021a; Li *et al.*, 2021b] rather than the filtered setting used in [Jin *et al.*, 2020; Zhu *et al.*, 2021; He *et al.*, 2021]. A detailed explanation is provided in Appendix B.

Baselines. Our model is compared with two categories of models, i.e., static KG reasoning models and TKG reasoning models. For static KG models, we select DisMult [Yang *et al.*, 2015], R-GCN [Schlichtkrull *et al.*, 2018], ConvE [Dettmers *et al.*, 2018], RotatE [Sun *et al.*, 2019]. For temporal models under the interpolation setting, we choose HyTE [Dasgupta *et al.*, 2018], TTransE [Jiang *et al.*, 2016] and TA-DistMult [García-Durán *et al.*, 2018]. Then, we compared most of the methods under the extrapolation setting, including Know-Evolve [Trivedi *et al.*, 2017], RGCRN [Jin *et al.*, 2020], CyGNet [Zhu *et al.*, 2021], RE-NET [Jin *et al.*, 2020], RE-GCN [Li *et al.*, 2021b], xERTE [Han *et al.*, 2021] and CluSTeR [Li *et al.*, 2021a]. Note that xERTE and CluSTeR use previous all ground truth when inferencing in the validation and test sets, while other methods are only given a training set as ground truth to conduct the multi-step inference.

4.2 Results on TKG

Without loss of fairness, we will compare the experimental results separately, as shown in Tables 1 and 2. Table 1 illustrates that rGalT consistently outperforms the baselines on the three ICEWS datasets and GDELT. Especially on the ICEWS14 dataset, rGalT achieves the improvements of 4.2% in Hits@1, 2.9% in Hits@3 over the best baseline.

Specifically, rGalT performs much better than all static methods (the first block) and temporal models under the interpolation setting (the second block) since our model considers temporal factors and learns historical evolution patterns. For those temporal models under the extrapolation setting (the third block), Know-Evolve performs the least well because it ignores the mutual interactions of concurrent events. The performance of RGCRN is worse than other models except for Know-Evolve because it models all history from several latest timestamps, while others consider a more extended history sequence. CyGNet and RE-NET achieve relatively good results for the inference task, on the grounds that CyGNet can learn repetitive facts of a given query, and RE-NET propagates the local and global structural information related to the query sequentially over the temporal pattern. There is no doubt that rGalT achieves better results because it can find more useful historical facts related to the prediction by modeling fact precursors. Note that rGalT outperforms RE-GCN on all datasets except YAGO. Considering evolutionary representations of all historical facts introduces noise to affect performance, which may cause RE-GCN to be less effective than rGalT on most datasets. Besides, the time interval (1 year) of YAGO is much larger than the other datasets resulting in more structural dependencies of the KG at each timestamp, which is friendly to some models with static properties, such as RE-GCN. Overall, rGalT is more capable of handling datasets with obvious dynamic features.

As shown in Table 2, the effects of rGalT w.GT and CluSTeR on these three datasets exceed xERTE, probably because

Models	ICEWS14 MRR	ICEWS14 Hits@3	ICEWS14 Hits@10	YAGO MRR	YAGO Hits@3	YAGO Hits@10
rGalT	38.33	42.86	58.13	51.45	57.76	68.31
- *RGAL*	29.87	33.91	48.57	40.10	45.70	57.07
+*RGCN*	36.38	40.46	55.10	48.83	55.52	64.74
+*GRU*	36.83	41.02	55.13	49.75	55.98	66.48
+*average*	33.69	38.13	52.43	46.57	52.94	61.57
- *multi-step*	35.01	39.43	53.34	47.46	53.77	62.58

Table 3: Ablation Studies on ICEWS14 and YAGO.

the inference subgraph of xERTE focuses more on direct causes leading to the predicted event, while the precursor information and the clues contain more query-related factual information. Despite the CluSTeR achieving better results on two ICEWS datasets, it does not follow the assumptions of the extrapolation reasoning task and was unable to make inferences only using the ground truth of the training set. In addition, rGalT's best performance on the GDELT dataset verifies our motivation that rGalT model can discover relevant historical facts at farther time steps apart because frequent changes of historical facts in the GDELT dataset (15-minute time intervals) may cause related facts to be far away in the time window.

4.3 Ablation Studies

To facilitate generalized conclusions, we conduct ablation experiments on ICEWS14 and YAGO datasets with different time intervals, 24 hours and one year, respectively. We discuss the effects of each component as follows:

Impact of Intra-graph Component. As shown in Table 3, we remove the relation-aware graph attention layer, denoted as - *RGAL*. It can be observed that - *RGAL* has a more significant impact on the final results, suggesting that introducing RGAL makes the transformer capable of capturing query-relevant fact information inside the graph. +*RGCN* indicates that we replace RGAL method with a relational graph convolutional network (RGCN). The result performance implies our method has the advantage of exploring neighboring entities related to the query.

Impact of Inter-graph Component. +*GRU* indicates that the GRU method is used instead of the transformer encoder to learn the temporal patterns among different graphs. It can be observed that the performance of +*GRU* decreases a lot in two datasets, demonstrating the transformer encoder's importance in our model. There are several facts helpful for future prediction occurring at more distant time steps. Compared to the traditional auto-regressive model, the transformer will allow each fact within a time window to focus on other timestamped facts and form narrative fact segments with highly correlated facts to avoid forgetting parts of the information during the time shift.

Impact of Inference Component. +*average* in Table 3 denotes a variant of rGalT directly using the average representations of transformer encoder outputs as the input of MLP. It can observe that the performance of +*average* decreases rapidly, which sufficiently indicates the necessity of self-attention mechanism. Attention methods help rGalT capture

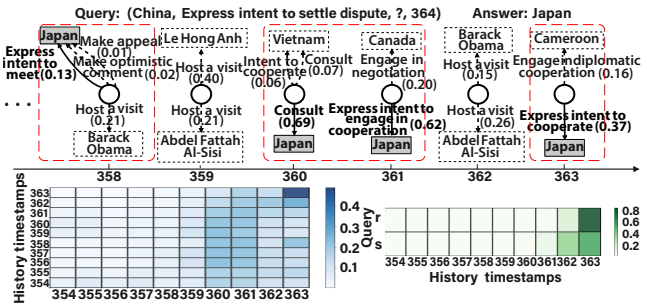


Figure 3: An example to illustrate how rGalT finds fact precursors. Each black circle represents the query entity *China*.

query-related precursor information from various fact segments. - *multi-step* represents rGalT keeps its last history in the training set and does not update history during inference. It can be seen that the performance is worse than the initial rGalT, which proves the effectiveness of our subgraph generation algorithm.

4.4 Case Study

rGalT provides an explanation for model predictions through the discovery of precursor information. Precursor information can be viewed as query-relevant narrative fact segments that reveal the evolutionary process behind the occurrence of an event. We study the query selected from the ICEWS14 test set *(China, Express intent to settle dispute, ? , 364)*, and Figure 3 shows how rGalT finds the fact precursors of this query for prediction. Here, the previous m time steps refer to the timestamp from 354 to 363.

The upper part of Figure 3 shows the subgraphs composed of partial facts related to the query at different timestamps. The weights of edges in the subgraphs are computed by RGAL, representing the correlation between facts and entity *China*. Encoder Self-attention aims to discover fact segments within the historical sequence. The timestamps of fact segments are extracted as follows according to the self-attention matrix in the lower left part of Figure 3: $\langle 356,360,361\rangle$, $\langle 357,360,361\rangle$, $\langle 358,360,361,363\rangle$. With our observations, the facts at $\langle 360,361\rangle$ play a key role in this prediction because they involve multiple fact segments. Encoder-Decoder self-attention aims to capture the correlation between query and fact segments. We can see that the given query has a significant link with the fact segment containing the timestamp $\langle 363\rangle$ from the lower right part. Thus, the facts with high query-dependent scores at timestamp $\langle 358,360,361,363\rangle$ can be considered as precursors to the query (triples with solid lines). Fact precursors suggest that there is a strong willingness to cooperate between *China* and *Japan*, which can be regarded as antecedent events for *(China, Express intent to settle dispute, ? , 364)*.

5 Conclusion

In this paper, we are the first to propose a novel precursor-aware transformer, rGalT, which employs transformer based on a relation-aware graph attention layer to model precursor information over the TKG. We use multiple attention mechanisms to capture diverse fact segments from the graph se-

Algorithm 1 Subgraph generation algorithm

Input:
Known graph sequence $\{G_1, G_2, \ldots, G_t\}$;
Given query $(s, r, ?, t + \Delta t)$ with missing object entity at timestamp $t + \Delta t$.
Output:
$\{\hat{G}_{t+1}, \hat{G}_{t+2}, \ldots, \hat{G}_{t+\Delta t-1}\}$.
1: $t' \leftarrow t + 1$
2: **while** $t' < t + \Delta t$ **do**
3: **for** every $r \in \mathcal{R}$ **do**
4: Get $p(r|s, G_{1:t}, \hat{G}_{t+1:t'-1})$ of relation r with subject entity s by Eq. 10
5: **for** every $o \in \mathcal{E}$ **do**
6: Get $p(o|s, r, G_{1:t}, \hat{G}_{t+1:t'-1})$ of object entity o with subject entity s and relation r by Eq. 9
7: Calculate the probability of triple (s, r, o) by Eq.12.
8: **end for**
9: **end for**
10: Pick top-k triples at t'
$\{(s, r_1, o_1, t'), \ldots, (s, r_k, o_k, t')\}$
$\hat{G}_{t'} \leftarrow \{(s, r_1, o_1, t'), \ldots, (s, r_k, o_k, t')\}$
$t' \leftarrow t' + 1$
11: **end while**
12: **return** $\{\hat{G}_{t+1}, \ldots, \hat{G}_{t+\Delta t-1}\}$

quence and find the fact segment most closely related to the query through joint self-attention methods. Besides, we decode precursor information in parallel to infer missing facts and construct a subgraph generation algorithm to handle the extrapolation task. Experimental results on five datasets demonstrate the effectiveness of our method, and precursor discovery provides interpretability for reasoning results.

Appendices

A Subgraph Generation Algorithm

As shown in Algorithm 1, given all the past facts $G_{1:t}$, we first calculate the conditional probability of each relation relative to the query subject entity(line 3-4), then calculate the conditional probability of each object entity close to the relation and subject entity(line 5-6), and finally obtain the likelihood of the triple (s, r, o) by the following formula:

$$p(s, r, o|G_{1:t}) = p(o|s, r, G_{1:t}) \cdot p(r|s, G_{1:t}) \cdot p(s|G_{1:t}), \tag{12}$$

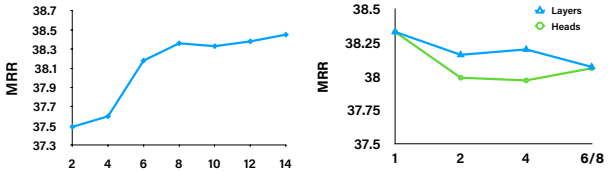
the subgraph we inferred is composed of facts related to s, so $p(s|G_{1:t})$ can be regarded as an inevitable probability. Then we pick the k triples with the highest probability to form the subgraph at time $t + 1$ (line 10), and treat the subgraph as ground truth for inferring future. Finally we estimate all the unknown graph sequence $\{\hat{G}_{t+1}, \ldots, \hat{G}_{t+\Delta t-1}\}$ in the time period Δt, and we can obtain the probability of missing entity o at $t + \Delta t$ by Eq.9.

B Detailed Experimental Settings

We use five TKG datasets in our experiments. Dataset statistics are described in Table 4.

Datasets	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	N_{train}	N_{valid}	N_{test}	Time gap
ICEWS14	6,869	230	74,845	8,514	7,371	24 hours
ICEWS05-15	10,094	251	368,868	46,302	46,159	24 hours
ICEWS18	23,033	256	373,018	45,995	49,545	24 hours
GDELT	7,691	240	1,734,399	238,765	305,241	15 mins
YAGO	10,623	10	161,540	19,523	20,206	1 year

Table 4: Dataset Statistics.(N_{train},N_{valid},N_{test} are the numbers of facts in training, validation, and test sets.)



(a) Length of history (b) Number of layers/heads

Figure 4: Parameter sensitivity on our rGalT.

The filtered setting used in [Jin *et al.*, 2020; Zhu *et al.*, 2021; He *et al.*, 2021] is not suitable for extrapolation reasoning over TKG. For example, given a test query $(s, r, ?, t)$ with the answer o in the test set, assume that there is a quadruple (s, r, o', t') in the training set, where $t' < t$. The previous filtered setting ignores time information and considers o' to be correct answer because (s, r, o', t') appears in the training set, and thus removes the quadruple from the corrupted ones. However, o' is incorrect for the given query, as the fact (s, r, o') is temporally valid on t', instead of timestamp t. In this way, the filtered setting wrongly removes quite a few quadruples and thus leads to higher ranking scores. In this paper, we conduct the experiments under the raw setting [Li *et al.*, 2021a; Li *et al.*, 2021b].

We set the size of entity/relation embeddings d to be 200 and the dropout rate to be 0.5. The number of transformer encoder/decoder layers l and heads K are both set to 2 for ICEWS18 and 1 for other datasets; the FFN inner-layer of encoder d_f is set to 800 and the length of decoder input sequence n is set to 2. The optimal length of history m is set to 12 for ICEWS18, and 10 for others. Adam optimizer is adopted for parameter training with the learning rate 0.001. We set λ to 0.1 for the loss function $\mathcal{L}$.

C Sensitivity Analysis

We report the performance change of our rGalT on the ICEWS14 dataset by varying the hyper-parameters.

Length of history in transformer encoder. Figure 4(a) shows the performance with various lengths of past histories. When rGalT uses longer histories, it achieves higher MRR. However, MRR starts to be relatively stable at around 10. Considering the cost, we set m to 10 on this dataset.

Number of layers and heads in the transformer model. We use multi-layer and multi-head transformer to capture the interaction between facts from different potential perspectives. As Figure 4(b) shows, MRR tends to decrease as the number of layers and heads increases, implying that larger l and k lead to smaller subspace dimensions. So we need to choose l and k more carefully according to the different datasets.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 61932001.

References

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[Boschee *et al.*, 2015] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. Icews coded event data. *Harvard Dataverse*, 12, 2015.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Dasgupta *et al.*, 2018] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *EMNLP*, pages 2001–2011, 2018.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.

[Galárraga *et al.*, 2015] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730, 2015.

[García-Durán *et al.*, 2018] Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*, pages 4816–4821, 2018.

[Guo *et al.*, 2018] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Knowledge graph embedding with iterative guidance from soft rules. In *AAAI*, pages 4816–4823, 2018.

[Han *et al.*, 2021] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *ICLR*, 2021.

[He *et al.*, 2021] Yongquan He, Peng Zhang, Luchen Liu, Qi Liang, Wenyuan Zhang, and Chuang Zhang. Hip network: Historical information passing network for extrapolation reasoning on temporal knowledge graph. In *IJCAI*, 2021.

[Jiang *et al.*, 2016] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. Towards time-aware knowledge graph completion. In *COLING*, pages 1715–1724, 2016.

[Jin *et al.*, 2020] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *EMNLP*, pages 6669–6683, 2020.

[Leetaru and Schrodt, 2013] Kalev Leetaru and Philip A Schrodt. Gdelt: Global data on events, location, and tone. In *ISA Annual Convention*, 2013.

[Li *et al.*, 2021a] Zixuan Li, Xiaolong Jin, Saiping Guan, Wei Li, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. In *ACL-IJCNLP*, 2021.

[Li *et al.*, 2021b] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutional representation learning. In *SIGIR*, 2021.

[Mahdisoltani *et al.*, 2015] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR*, 2015.

[Schlichtkrull *et al.*, 2018] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[Sun *et al.*, 2019] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

[Trivedi *et al.*, 2017] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *ICML*, pages 3462–3471, 2017.

[Trivedi *et al.*, 2019] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *ICLR*, 2019.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080, 2016.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Xiong *et al.*, 2017] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573, 2017.

[Yang *et al.*, 2015] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

[Zhang *et al.*, 2019] Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *WWWC*, pages 2366–2377, 2019.

[Zhu *et al.*, 2021] Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhan. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *AAAI*, 2021.