# Graph Hawkes Transformer for Extrapolated Reasoning on Temporal Knowledge Graphs

**Haohai Sun**[1*] **Shangyi Geng**[1*] **Jialun Zhong**[1] **Han Hu**[2] **Kun He**[1†]
[1] School of Computer Science and Technology,
Huazhong University of Science and Technology
[2] Microsoft Research Asia
{haohais, shangyigeng, zhongjl}@hust.edu.cn
hanhu@microsoft.com brooklet60@hust.edu.cn

## Abstract

Temporal Knowledge Graph (TKG) reasoning has attracted increasing attention due to its enormous potential value, and the critical issue is how to model the complex temporal structure information effectively. Recent studies use the method of encoding graph snapshots into hidden vector space and then performing heuristic deductions, which perform well on the task of entity prediction. However, these approaches cannot predict when an event will occur, and have the following limitations: 1) there are many facts not related to the query that can confuse the model; 2) there exists information forgetting caused by long-term evolutionary processes. To this end, we propose a Graph Hawkes Transformer (GHT) for both TKG entity prediction and time prediction tasks in the future time. In GHT, there are two variants of Transformer, which capture the instantaneous structural information and temporal evolution information, respectively, and a new relational continuous-time encoding function to facilitate feature evolution with the Hawkes process. Extensive experiments on four public datasets demonstrate its superior performance, especially on long-term evolutionary tasks.

## 1 Introduction

Knowledge Graph (KG), a multi-relational directed graph database that stores human knowledge and facts, is widely used in downstream applications such as recommendation systems (Guo et al., 2020; Wang et al., 2018), web search (Paulheim, 2017) and question answering (Saxena et al., 2021). Conventionally, KGs store each fact in the form of a triplet ($subject$, $predicate$, $object$). However, many facts may change over time and may contain event-based interaction data. To encode the temporal information, Temporal Knowledge Graph (TKG) is proposed so that each temporal fact is stored as a quadruple ($subject$, $predicate$, $object$, $timestamp$). Typically, we represent TKG as a sequence of static KG snapshots associated with timestamps.

There are two types of reasoning for TKG. Interpolated reasoning is designed to complete the missing facts on known historical snapshots, while extrapolated reasoning predicts future events or facts. This work focuses on extrapolated reasoning, a problematic but meaningful task, which can be used for crisis warning, user behavior prediction, supply chain management, etc. While most existing works (Han et al., 2021a; Sun et al., 2021) perform link predictions on snapshots at the next timestamp, we consider predicting events that may occur over a long period in the future, which are more appropriate in the real-world setting.

The task requires the assistance of known historical facts. A key feature required by the model is to retrieve key information from complex temporal structured data that can help answer the query and make correct judgments. Recent methods of RE-NET (Jin et al., 2020) and RE-GCN (Li et al., 2021b) employ R-GCN (Schlichtkrull et al., 2018) to capture structural information from historical snapshots and then use Recurrent Neural Networks (RNNs) to model the latent vector sequences. While the two methods work well on entity prediction task, they also meet some limitations. (1) Query information cannot be fully utilized by R-GCN. (2) RNN-based models assume that sequences are equidistant, which is inconsistent with real-life event sequences. (3) Step-by-step inference methods accumulate errors during training. (4) They cannot predict the timestamp that an event will occur in the future.

There are some early methods (Trivedi et al., 2017, 2019) that could do both tasks simultaneously by using a temporal point process to represent the evolution of facts, with the key concept of conditional intensity function denoting the con-

[*] Equal Contribution.
[†] Corresponding author.

*Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7481–7493
December 7-11, 2022 ©2022 Association for Computational Linguistics

ditional probability of an event occurring over a period. Nevertheless, these methods could not perform well on the entity prediction task as they did not capture the structural information from the graph snapshots. On the other hand, Transformer (Vaswani et al., 2017) has recently been widely used in various fields (Lin et al., 2021) owing to its powerful modeling capabilities. However, to our knowledge, there exists no work that uses Transformer or its variants to solve the TKG evolution problem.

In this work, we propose a new model termed **G**raph **H**awkes **T**ransformer (GHT), which introduces Transformer into the TKG evolution modeling and further integrates the neural temporal point process. Specifically, we design two Transformer variants to capture the structural and temporal information in TKGs by building conditional intensity function. One variant is used to aggregate multi-relational graphs, capturing structural information for each timestamp and generating feature vectors. The model can learn which interactions are more critical in query with the attention mechanism. The other variant captures temporal information based on a sequence of feature vectors and simultaneously outputs a hidden state for each timestamp in the future. Finally, the model uses the hidden state to calculate the conditional intensity and then gets the candidate entity score or the time probability density of the next event.

Our main contributions are as follows:

- We propose a new Transformer-based model for TKG extrapolated reasoning, which captures both structural and temporal information. Not only can it simultaneously predict events of multiple timestamps in the future, but it can also predict when an event will occur. To our knowledge, this is the first Transformer-based temporal point process model for the TKG evolutionary representation learning.

- We design a new relational continuous-time encoding function that can handle unseen timestamps and provide personalized responses to different queries. It can be used to construct conditional intensity in the Hawkes process.

- State-of-the-art performance has been achieved on four popular benchmarks, indicating the effectiveness of our model, especially on long-term evolution tasks.

## 2 Related Work

### 2.1 Neural Hawkes Process

The Hawkes process is a self-exciting temporal point process applied to model sequential discrete events occurring in continuous time (Hawkes, 1971). It assumes that past events can temporarily excite future events, characterized via an intensity function. The intensity function $\lambda_k^*(t)$ represents the expected number of events happened in interval $(t, t+dt]$ defined as follows, where $*$ is a shorthand for conditioning on history $\mathcal{H}_t$:

$$\lambda_k^*(t) = \lim_{\Delta t \to 0} \frac{P(event\ of\ type\ k\ in[t, t+\Delta t)|\mathcal{H}_t)}{\Delta t}. \tag{1}$$

The traditional Hawkes process can only capture simple patterns of the events. In contrast, the neural Hawkes process (Shchur et al., 2021) that introduces neural networks to parameterize the intensity function exhibits high model capacity in complex real-life scenarios. Researchers have recently started adapting neural networks, especially RNN models, to the temporal point process to build more flexible and efficient models. RMTPP (Zhou et al., 2013) embeds the sequence data into RNN and models the conditional intensity function considering the historical non-linear dependence. Mei and Eisner (2016) develop a neural Hawkes process based on LSTM to model the asynchronous event sequence. Zhang et al. (2020) apply the attention mechanism to the neural Hawkes process. Moreover, Zuo et al. (2020) propose a Transformer Hawkes process, which utilizes Transformer to capture the complicated short-term and long-term temporal dependencies. However, these works do not model structured information.

### 2.2 Temporal Knowledge Graph Reasoning

Embedding-based methods (Bordes et al., 2013; Trouillon et al., 2016) achieve excellent results on static KGs, and have been extended to temporal KGs (Leblay and Chekol, 2018; García-Durán et al., 2018; Goel et al., 2020; Lacroix et al., 2020). However, these methods cannot handle extrapolated reasoning since the timestamps in the test dataset do not exist in the training dataset. For extrapolation, Know-Evolve (Trivedi et al., 2017) and GHNN (Han et al., 2020) use temporal point processes to estimate conditional probability. However, they fail to model the structural information in historical graphs, leading to low performance.

Recently, TITer (Sun et al., 2021) uses reinforcement learning to search answer in the history. xERTE (Han et al., 2021a) finds answers on subgraphs through subgraph sampling and attention flow. Both methods restrict the answer domain to N-order neighbors and do not capture evolutionary representations. CyGNet (Zhu et al., 2021) uses a generate-copy mechanism to let the model remember recurring historical events. RE-NET (Jin et al., 2020) and RE-GCN (Li et al., 2021b) use RGCN (Schlichtkrull et al., 2018) to capture structural information and then use RNN to perform representational evolution. Additionally, TANGO (Han et al., 2021b) uses a neural ordinary differential equation to model this task in the continuous-time domain. CluSTeR (Li et al., 2021a) uses a two-stage approach for clue searching and candidate ranking. These methods achieve good performance on the entity prediction task but cannot predict time.

## 3 The Proposed Model

Our method first obtains concurrent structural information from subgraphs and generates feature vectors for each source node. Then it captures the temporal evolution information from the vector sequences and finally outputs the hidden state vectors to participate in the construction of the conditional strength function to complete the prediction task. Compared with previous evolutionary representation learning methods, *e.g.*, RE-NET (Jin et al., 2020) and RE-GCN (Li et al., 2021b), we expect the model to make up for the four limitations introduced in Section 1. Figure 1 illustrates the overall architecture of our model.

### 3.1 Notations and Task Definition

Let $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively. Let $\mathcal{F}_t$ denote the set of facts at time $t$. A TKG can be represented as a sequence of static KG snapshots, *i.e.*, a known TKG from time 1 to time $t$ can be described as $\mathcal{G}_{(1,t)} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_t\}$, where $\mathcal{G}_t = (\mathcal{E}, \mathcal{R}, \mathcal{F}_t)$ is the KG snapshot at time $t$, a directed multi-relational graph. Each fact in $\mathcal{F}_t$ is described in the form of a quadruple $(e_s, r, e_o, t)$, where $e_s, e_o \in \mathcal{E}$ and $r \in \mathcal{R}$. The quadruple can be seen as an edge from $e_s$ to $e_o$ of type $r$ at graph $\mathcal{G}_t$.

Based on the historical TKG $\mathcal{G}_{(1,t)}$, the model needs to predict the facts in time $t_q$. We further consider another task of predicting the time of an event that will occur in the future. For a concrete fact, the two tasks can be defined as follows.

**Task 1. Entity Prediction.** Given a query $(e_s, r_q, ?, t_q)$, the model needs to predict the missing entity [1].

**Task 2. Time Prediction.** Given a query $(e_s, r_q, e_o, ?)$, the model needs to predict the timestamp $t_q$ that this event will occur next time.

In the framework of the neural Hawkes process (Shchur et al., 2021; Han et al., 2020), $\mathcal{H}_t^q = (e_s, r_q, \mathcal{G}_{(1,t)})$ denotes the historical information related to the query until time $t$. Let $\lambda_e(t|\mathcal{H}_t^q)$ be the conditional intensity function of a candidate object entity $e$, abbreviated as $\lambda_t^e$, we can accomplish both tasks with the conditional intensity function:

$$e_o = \underset{e \in \mathcal{E}}{argmax}\{\lambda_{t_q}^e\}, \tag{2}$$

$$p(t'|e_o, \mathcal{H}_t^q) = \lambda_{t'}^{e_o} exp\left(-\int_t^{t'} \tau \lambda_\tau^{e_o} d\tau\right), \tag{3}$$

$$t_q = \int_t^{+\infty} \tau p(\tau|e_o, \mathcal{H}_t^q) d\tau. \tag{4}$$

Since all candidate entities share the same survival term (Han et al., 2020), we can directly compare the intensity value to get the answer entity, as shown in Eq. 2. Eq. 3 is the corresponding conditional time density function, according to which we can estimate the time through the integral formula of Eq. 4. Then, we introduce how to model the intensity function.

### 3.2 Relational Graph Transformer

Capturing the structural information on historical snapshots is the first key to answer the query. Previous works (Jin et al., 2020; Li et al., 2021b) use R-GCN (Schlichtkrull et al., 2018) for information aggregation to store interaction information between nodes in the form of hidden vectors.

However, in a historical snapshot, many events relate to an entity simultaneously, yet only a tiny fraction of the relational information is helpful to answer the query. R-GCN cannot handle this issue as it treats every message equally important. Therefore, we design a Relational Graph Transformer (RGT) to let our model know which concurrent events are more critical in a snapshot. There are two forms of graph representation: global receptive field (Ying et al., 2021) and local receptive field (Velickovic et al., 2018; Dwivedi and Bresson,

[1] Without loss of generality, we can transform $(?, r_q, e_o, t_q)$ into $(e_o, r_q^{-1}, ?, t_q)$ by changing subject prediction to object prediction, where $r_q^{-1}$ is the reciprocal relation of $r_q$.

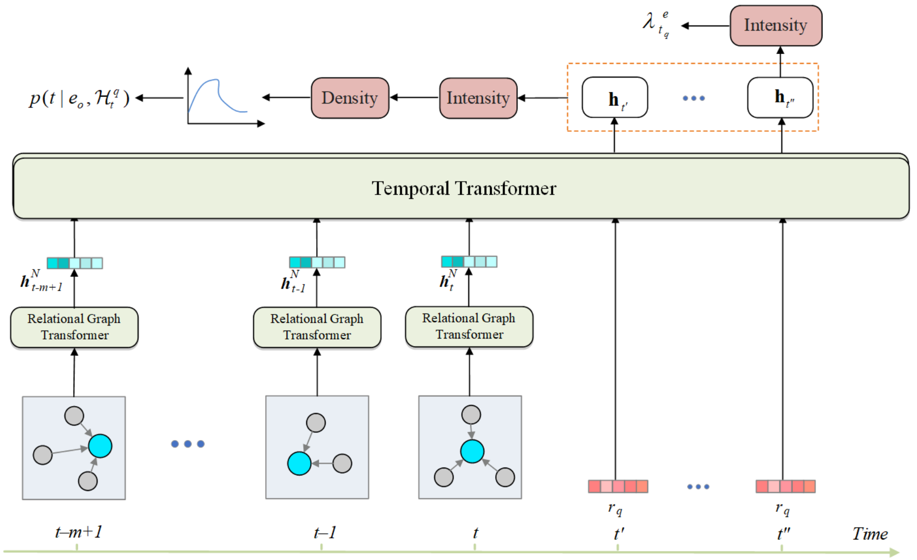


Figure 1: An illustration of the proposed GHT model. The Relational Graph Transformer encodes the graph structure for each historical snapshot. The Temporal Transformer updates the hidden state to any future timestamp based on the historical state sequence. The conditional intensity and probability density are calculated based on the hidden vector sequences.

2020; Hu et al., 2020). The global form cannot cope with graphs with too many nodes. Thus, to maximize the preservation of structural information of the graph, RGT operates in local graph neighborhoods.

Let $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d}$ and $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}| \times d}$ be the initial embedding matrices of the entities and relations, respectively, where $d$ represents the dimension of the embedding[2]. $\mathbf{e}_i = \mathbf{E}[i]$ is the embedding of entity $e_i$, and $\mathbf{r}_i = \mathbf{R}[i]$ is the embedding of relation $r_i$. RGT aggregates structural information from each incoming edge for each entity in the snapshots sequence.

Specifically, for $\mathcal{G}_t^q$, we initialize the hidden states of the nodes as the initial embeddings of the corresponding entities, and each node will update its hidden state via the message-passing framework. *E.g.*, for an entity $e_i$, we concatenate the hidden state of the source node and the corresponding relation embedding as the message for each incoming edge $e_i$. Then we pack them together as the key matrix $K \in \mathbb{R}^{|M| \times 2d}$ and value matrix $V \in \mathbb{R}^{|M| \times 2d}$, where $|M|$ is the number of messages. For each message, we use $\mathbf{r}_q$ as the query, and the query matrix is $Q \in \mathbb{R}^{|M| \times d}$. Finally, the hidden state of an entity at layer $l \in [1, N]$ is defined as follows:

$$\mathbf{h}_t^l = LN(FFN(MHA(Q, K, V) + \mathbf{h}_t^{l-1}), \qquad (5)$$

where $LN$ is the layer normalization, $FFN$ is the feed-forward blocks, and $MHA$ is Multi-Head Attention (Vaswani et al., 2017). Finally, we output the hidden state sequence of the query entity $H_e = \{\mathbf{h}_{t-m+1}^N, ..., \mathbf{h}_t^N\}$. We do not introduce query entity information $\mathbf{e}_q$ in the query matrix $Q$, because as a TKG develops, new entities could appear on the graph, and the model has not learned their initial embeddings. In contrast, the semantics of relations are stable and context-independent , so all layers share the relational embedding matrix.

### 3.3 Temporal Transformer

We design the Temporal Transformer (TT) to model the temporal evolution of entity representations in the continuous-time domain. Transformer learns location information through a position encoding function. Although researchers have designed a variety of position encoding methods (Devlin et al., 2019; Shaw et al., 2018; Dai et al., 2019; Ke et al., 2021), most of them do not apply to our scenario due to the following two reasons: (1) These position encoding methods are performed in the discrete domain and are not suitable for the Hawkes process. (2) Different from discrete tokens, our input is a continuous vector after the graph information aggregation, and the distribution shift between the training set and the test set will lead them (Vaswani et al., 2017; Devlin et al., 2019) to have poor generalization performance.

[2] For brevity, we define all definable parameter's dimension as $d$.

Moreover, the temporal distribution of attention could be different for different query event types. For example, diarrhea is more likely to be caused by eating spoiled food the day before, while long-term eating habits may cause obesity. Thus, we redesign a relational continuous-time encoding function to assist the attention calculation. We need to ensure the inductive ability of the function because future timestamps cannot be seen during the model training. Based on the calculation principle of Transformer attention, we need to ensure that (Xu et al., 2020), $\forall c \in \mathbb{R}$:

$$\langle TE(t_k + c), TE(t_q + c)\rangle = \langle TE(t_k), TE(t_q)\rangle,$$

where $TE$ denotes the time encoding function, $\langle \cdot, \cdot \rangle$ indicates inner product, $t_k$ and $t_q$ represent the time of key and time of query in the attention calculation, respectively. We can use absolute time encoding to represent relative time information.

We design a learnable sinusoid function that meets the above condition, and use the query relation $\mathbf{r}_q$ to control the amplitude of the function:

$$TE_r(t) = [\alpha_1^r cos(w_1 t), \alpha_1^r sin(w_1 t), ..., \\ \alpha_d^r cos(w_d t), \alpha_d^r sin(w_d t)], \quad (6)$$

where $[\alpha_1^r, ..., \alpha_d^r]$ is the linear projection of the relation embedding $\mathbf{r}$, and $[w_1, ..., w_d]$ is a $d$-dimensional learnable vector. We use the time information to calculate a bias term for the attention matrix $A$. Let $T_q = [TE_{r_q}(t_{q_1}); ...; TE_{r_q}(t_{q_s})]$ denote the queries' time encoding matrix, and $T_k = [TE_{r_q}(t_{k_1}); ...; TE_{r_q}(t_{k_m})]$ denotes the keys' time encoding matrix, we have the attention matrix $A$ as follows:

$$A = \frac{(H_q W_Q)(H_e W_K)^T + (T_q)(T_k)^T}{\sqrt{2d}}, \quad (7)$$

where $W_Q, W_K$ are weight matrices. $H_q$ is the embedding matrix of the query relation and $H_e$ is the packed matrix of the query entity embedding sequence obtained from RGT.

### 3.4 Conditional Intensity

After the encoding described in Section 3.3, we can generate a hidden representation at any time in the future. Then, we can use it to construct a continuous-time conditional intensity function $\lambda_{t_q}^{e_i}$ for all candidate entities:

$$\lambda_{t_q}^{e_i} = f(\langle [\mathbf{e}_s, \mathbf{r}_q, \mathbf{h}_{t_q}] W_\lambda, \ \mathbf{e}_i \rangle), \quad (8)$$

where $W_\lambda \in \mathbb{R}^{3d \times d}$ is the projection matrix, $\mathbf{h}_{t_q}$ is the hidden state at time $t_q$ obtained from TT, $f(x) = \beta \cdot log(1 + exp(\frac{x}{\beta}))$ is softplus function with parameter $\beta$ that guarantees a positive intensity, and $\mathbf{e}_s$, $\mathbf{r}_q$, and $\mathbf{e}_i$ are the embeddings of the query entity, query relation and candidate entity, respectively.

Finally, we can predict the entity or time based on Eq. 2, Eq. 3 and Eq. 4. For the integral operation, we use the trapezoidal rule (Stoer and Bulirsch, 2013) for approximation:

$$t_q = \sum_{j=1}^{L} \frac{t_j - t_{j-1}}{2} \big(t_j p(t_j | e_o, \mathcal{H}_t^q) + \\ t_{j-1} p(t_{j-1} | e_o, \mathcal{H}_t^q)\big), \quad (9)$$

where $t_j \in [t, +\infty)$, and $L$ is the number of samples. Other estimation methods, such as Simpson's rule (Stoer and Bulirsch, 2013) and Monte Carlo integration (Stoer and Bulirsch, 2013) can also be used.

### 3.5 Training

We view the entity prediction task as a multi-class classification task and the time prediction task as a regression task. Then, we use the cross-entropy loss for the entity prediction task and the mean square error (MSE) loss for the time prediction task. Let $\mathcal{D}_{train}$ denote the training set, $L_e$ be the loss of the entity prediction and $L_t$ be the loss of time prediction. Then,

$$L_e = -\sum_{i=1}^{|\mathcal{D}_{train}|} \sum_{c=0}^{|\mathcal{E}|-1} y_c log(p(e_{o_i} = c | \mathcal{H}_{t_i}^q)), \quad (10)$$

$$L_t = \sum_{i=1}^{|\mathcal{D}_{train}|} (t_i - \hat{t}_i)^2. \quad (11)$$

Here $\hat{t}_i$ is the estimated time. We jointly train the two tasks, and the final loss $L = L_e + \mu L_t$, where $\mu$ is a hyperparameter.

## 4 Experiments

### 4.1 Experimental Setup

**Datasets** We evaluate our model on four public TKG datasets, ICEWS14, ICEWS18 ICEWS05-15 and GDELT. Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015) is an international event dataset. Its three subsets, ICEWS14, ICEWS18 and ICEWS05-15, are usually used to evaluate the performance of TKG reasoning models, which contains events occurring in 2014, 2018, and 2005 to 2015 respectively. Global Database of Events, Language and Tone (GDELT) is a large comprehensive event dataset that records data every 15 minutes. Following the previous work (Jin et al., 2020), we split the dataset into train/valid/test by timestamps, and *train time* < *valid time* < *test time*. Statistics of these datasets are shown in Appendix A.1.

**Evaluation Metrics** The MRR and Hits@$k$ ($k \in \{1, 3, 10\}$) are standard metrics for the entity prediction task. MRR is the average reciprocal of the correct query answer rank. Hits@$k$ indicates the proportion of correct answers among the top $k$ candidates. As mentioned in (Han et al., 2020; Sun et al., 2021), the static filtered setting (Jin et al., 2020; Zhu et al., 2021), which removes entities from candidates according to the triples without considering the time, is unsuitable for TKG reasoning. Thus, we adopt the time-aware filtered setting (Han et al., 2020; Sun et al., 2021; Han et al., 2021a).

Moreover, most previous works only evaluate the performance of their models for the entity prediction at the next timestamp. Such evaluation cannot adequately reflect the model's performance for future predictions. Therefore, we evaluate the model's short-term and long-term evolution by setting different forecasting time window size of $\Delta t$.

For the time prediction task, we use the mean absolute error (MAE) between the predicted time and ground truth time as the metric.

**Baselines** For the entity prediction task, we compare our model with three types of KG reasoning models: (1) static KG reasoning models, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2014) and ComplEx (Trouillon et al., 2016). Ignoring the time information, we can get a static knowledge graph, and then apply these model. (2) TKG interpolated reasoning models, including TTransE (Leblay and Chekol, 2018), TA-DistMult (García-Durán et al., 2018), DE-SimplE (Goel et al., 2020), and TNTComplEx (Lacroix et al., 2020); (3) TKG extrapolated reasoning models, including RE-NET (Jin et al., 2020), CyGNet (Zhu et al., 2021), RE-GCN (Li et al., 2021b) and TITer (Sun et al., 2021). It is worth noting that RE-NET and RE-GCN are the most relevant model, which also learn evolutionary representation.

For the time prediction task, we compare our model with the previous temporal point process model for TKG, GHNN (Han et al., 2020). As for Know-evolve (Trivedi et al., 2017), it has a problematic formulation that has already been discussed in (Jin et al., 2020; Han et al., 2020). Thus, we do not choose it as a baseline.

## 4.2 Implementation Details

**Baseline details** During the inference, static reasoning models, interpolated reasoning models, and CyGNet do not encode the historical information. Therefore, they perform the same in the short-term as well as the long-term evolution task, so we directly use the results reported in previous works (Sun et al., 2021; Han et al., 2021a). For TITer[3], RE-NET[4], RE-GCN[5], and GHNN[6], using their released source code with default hyperparameters, we rerun these models on four benchmarks. To ensure fairness, for RE-GCN, we do not use the module that encodes the node type information.

**GHT details** We implement our model in PyTorch. We set the dimension of all embeddings and hidden states to 100. The history length is limited to 6. The layer number of the Relational Graph Transformer is set to 2, and the attention head number is set to 1. For Temporal Transformer, the layer number is 2, and the attention head number is 2. We apply dropout with dropout rate of 50% to each layer and use label smoothing to suppress overfitting. The hyperparameter $\mu$ used for loss function is set to 0.2. We use Adam to optimize the parameters, with a learning rate of 0.003 and a weight decay of 0.0001. The batch size is set to 256. All experiments were done on Tesla P100.

## 4.3 Results and Discussion

### 4.3.1 Entity Prediction

Table 1 and Figure 2 report the experimental results on the entity prediction task. We report results for $\Delta t = 10$ in details, other cases can be found in Appendix A.2. To explore the effect of evolution time on our model's performance, we report MRR over four datasets as the evolution timespan grows, where $dt$ indicates that the model predicts events after $dt$ interval. Due to layout limitations, the experimental results for GDELT are in Appendix A.2.

[3] https://github.com/JHL-HUST/TITer

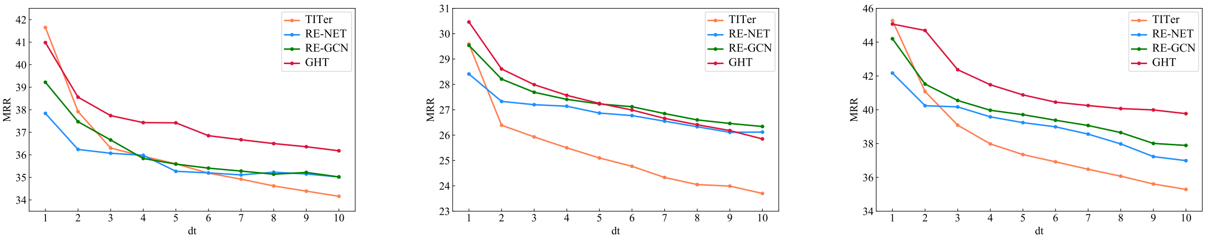
[4] https://github.com/INK-USC/RE-Net

[5] https://github.com/Lee-zix/RE-GCN

[6] https://github.com/Jeff20100601/GHNN_clean

| Model | ICEWS14 | | | | ICEWS18 | | | | ICEWS0515 | | | | GDELT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 |
| TransE | 22.48 | 13.36 | 25.63 | 41.23 | 12.24 | 5.84 | 12.81 | 25.10 | 22.55 | 13.05 | 25.61 | 42.05 | - | - | - | - |
| Distmult | 27.67 | 18.16 | 31.15 | 46.96 | 10.17 | 4.52 | 10.33 | 21.25 | 28.73 | 19.33 | 32.19 | 47.54 | 8.61 | 3.91 | 8.27 | 17.04 |
| ComplEx | 30.84 | 21.51 | 34.48 | 49.59 | 21.01 | 11.87 | 23.47 | 39.97 | 31.69 | 21.44 | 35.74 | 52.04 | 9.84 | 5.17 | 9.58 | 18.23 |
| TTransE | 13.43 | 3.11 | 17.32 | 34.55 | 8.31 | 1.92 | 8.56 | 21.89 | 15.71 | 5.00 | 19.72 | 38.02 | 5.50 | 0.49 | 4.99 | 15.18 |
| TA-DistMult | 26.47 | 17.09 | 30.22 | 45.41 | 16.75 | 8.61 | 18.41 | 33.59 | 24.31 | 14.58 | 27.92 | 44.21 | 11.17 | 5.09 | 11.58 | 22.65 |
| DE-SimplE | 32.67 | 24.43 | 35.69 | 49.11 | 19.30 | 11.53 | 21.86 | 34.80 | 35.02 | 25.91 | 38.99 | 52.75 | - | - | - | - |
| TNTComplEx | 32.12 | 23.35 | 36.03 | 49.13 | 21.23 | 13.28 | 24.02 | 36.91 | 27.54 | 19.52 | 30.80 | 42.86 | - | - | - | - |
| CyGNet | 32.73 | 23.69 | 36.31 | 50.67 | 24.93 | 15.90 | 28.28 | 42.61 | 34.97 | 25.67 | 39.09 | 52.94 | 18.05 | 11.13 | 19.11 | 31.50 |
| TITer | 36.06 | 27.51 | 40.16 | 52.05 | 25.34 | 18.09 | 28.17 | 38.95 | 38.11 | 26.83 | 44.43 | 59.44 | 15.75 | 10.94 | 15.74 | 25.37 |
| RE-NET | 35.71 | 26.79 | 39.57 | 52.81 | 26.88 | 17.70 | 30.35 | 45.04 | 39.11 | 29.04 | 44.10 | 58.90 | - | - | - | - |
| RE-GCN | 36.08 | 27.30 | 39.89 | 53.01 | 27.34 | **18.44** | 30.48 | 44.85 | 39.89 | 30.10 | 44.63 | 56.98 | 19.07 | 12.16 | 20.20 | 32.39 |
| GHT | **37.40** | **27.77** | **41.66** | **56.19** | **27.40** | 18.08 | **30.76** | **45.76** | **41.5** | **30.79** | **46.85** | **62.73** | **20.04** | **12.68** | **21.37** | **34.42** |
| | ± 0.13 | ± 0.10 | ± 0.09 | ± 0.07 | ±0.12 | ±0.09 | ±0.12 | ±0.14 | ±0.15 | ±0.17 | ±0.19 | ±0.15 | ±0.18 | ±0.19 | ±0.10 | ±0.14 |

Table 1: Entity prediction results (forecasting time window size is 10 units of time$\Delta t = 10$). Evaluation metrics are time-aware filtered MRR and Hits@1/3/10. All results are multiplied by 100. The last row of the table illustrates the bias of GHT during experiments.



(a) ICEWS14

(b) ICEWS18

(c) ICEWS05-15

Figure 2: MRR on ICEWS datasets with regard to various $dt$, which denote the models' performance at future $dt$ moments.

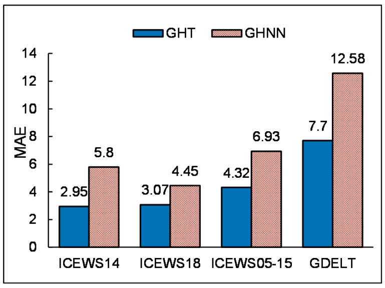


Figure 3: Time prediction results. Lower is better.

The extrapolated reasoning methods can deal with unseen timestamps. GHT outperforms all these methods on metrics MRR, Hits@3, and Hits@10. As for Hits@1, GHT also achieves the state-of-the-art (SOTA) performance. We further analyze and compare these models through the curve diagrams in Figure 2. As can be seen, although TITer outperforms the other models when $dt = 1$, it decays faster on long-term future predictions. We think this is because it searches the answer in local subgraphs, focusing more on the explicit cues. As the evolution timespan grows, the explicit cues will gradually decrease, and the implicit cues will become more important. Moreover, retrieving answers from neighbor subgraphs limits the candidate answer space, which is more pronounced on long-term evolution tasks. Our model outperforms RE-NET and RE-GCN in predicting long-term events because they use a heuristic evolutionary method to predict long-term events through gradual evolution, which forgets previous knowledge during the evolution. By contrast, we introduce an attention mechanism when encoding structural information, which helps the model capture more helpful information.

Overall, our model has excellent performance, providing more accurate answers when performing entity prediction tasks, especially over long intervals.

#### 4.3.2 Time Prediction

Figure 3 shows the results of the time prediction task. The result indicates the superiority of our model utilizing Transformer for conditional intensity function construction. GHNN aggregates

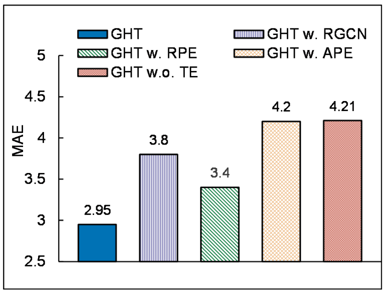


Figure 4: Time prediction ablation study on ICEWS14.

| Model | MRR | H@1 | H@3 | H@10 |
|---|---|---|---|---|
| GHT | **37.40** | **27.77** | **41.66** | **56.19** |
| GHT w.o. TE | 36.21 | 26.52 | 40.31 | 55.69 |
| GHT w. APE | 36.42 | 26.91 | 40.46 | 55.20 |
| GHT w. RPE | 36.49 | 26.88 | 40.73 | 55.24 |
| GHT w. RGCN | 33.86 | 24.25 | 37.87 | 52.75 |

Table 2: Entity prediction ablation study on ICEWS14.

neighbor information through simple mean pooling and only focuses on the most relevant first-order neighbors. In comparison, our RGT has a better structural information extraction ability. Moreover, GHNN uses continuous-time LSTM (Mei and Eisner, 2016) to estimate the intensity function, which is not as good as Transformer in capturing complex long-term and short-term dependencies(Zuo et al., 2020). Therefore, GHT outpeforms GHNN on the time prediction task.

During the experiment, our model also shows good efficiency. The discussion on efficiency analysis can be found in Appendix A.3.

## 4.4 Ablation Study

In this subsection, we study the effect of each module separately, and the results are reported in Table 2 and Figure 4. We do all ablation experiments on the ICEWS14 dataset. Specially, we try various position encoding functions to show the effectiveness of the relational time encoding function, and use other relational GNNs instead of RGT to verify the performance of RGT. We report the entity prediction results in Table 2, and the time prediction results in Figure 4. We also study the hyperparameter's sensitivity of GHT, which can be seen in Appendix A.4

### 4.4.1 Different Graph Aggregator

R-GCN, the most common relational graph aggregator, is used in RE-NET and RE-GCN. We replace RGT with R-GCN and denote the model as *GHT w. RGCN*. Results in Table 2 and Figure 4 demonstrate that GHT significantly outperforms *GHT w. RGCN* on both entity and time prediction tasks. It indicates that RGT is more suitable for TKG extrapolated reasoning than R-GCN. Compared with R-GCN, RGT can extract more useful information for query answering from complex structural information.

### 4.4.2 Different Position Encoding Functions

To verify whether our proposed relational continuous-time encoding function is effective, we compare GHT with several variants, which use different position encoding methods: (1) Do not use the relational continuous-time encoding function, denoted as *GHT w.o. TE*; (2) Replace it with absolute position encoding (Devlin et al., 2019), denoted as *GHT w. APE*; (3) Replace it with relative position encoding (Ke et al., 2021), denoted as *GHT w. RPE*. In Table 2 and Figure 4, we observe that GHT outperforms all other variants on both entity and time prediction tasks. Note that *GHT w. APE* and *GHT w. RPE* perform almost the same as not using the position encoding function. This indicates that they did not learn the location information. By contrast, relational continuous-time encoding can effectively capture temporal information and help the model achieve good performance.

## 5 Conclusion

We propose a new TKG reasoning model, called Graph Hawkes Transformer (GHT), a neural temporal point process model based on Transformer. We first analyze four limitations of the previous state-of-the-art methods, RE-GCN (Li et al., 2021b) and RE-NET (Jin et al., 2020). Then we design two Transformer blocks, which are used to capture the structural and temporal information, respectively. Based on Hawkes process, the model can learn a conditional intensity function to solve the above issues with the attention mechanism and the proposed relational continuous-time encoding function. Moreover, most previous works only evaluate their methods for entity prediction at the next future timestamp, while we evaluate the models comprehensively by setting different forecasting time window sizes. Experimental results on four popular datasets demonstrate the superior performance of our model on both entity prediction and time prediction tasks. Notably, our model performs much better under long-term evolution scenarios.

## 6 Limitations

The inference of GHT relies entirely on the learned entity representation, and new entities that emerge as TKG evolves will get a random initialization, which limits the model's inductive reasoning ability.

When making time predictions, the integral operation is implemented by approximate estimation. The more accurate the estimation, the more calculations are required. The attention computation in Transformer is also of quadratic complexity, which requires much calculation. All of these factors limit the capability of GHT to handle large-scale graphs.

## Acknowledgements

This work is supported by National Natural Science Foundation (62076105) and International Cooperation Foundation of Hubei Province, China (2021EHB011).

## References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems*, pages 2787–2795.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

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

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202*.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3988–3995.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021a. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021b. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Empirical Methods in Natural Language Processing*, pages 8352–8364.

Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. 2020. Graph hawkes neural network for forecasting on temporal knowledge graphs. In *Conference on Automated Knowledge Base Construction*.

Alan G Hawkes. 1971. Point spectra of some mutually exciting point processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 33(3):438–443.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW '20: The Web Conference 2020*, pages 2704–2710.

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

Guolin Ke, Di He, and Tie-Yan Liu. 2021. Rethinking positional encoding in language pre-training. In *9th International Conference on Learning Representations*.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. In *International Conference on Learning Representations*.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference*, pages 1771–1776.

Zixuan Li, Xiaolong Jin, Saiping Guan, Wei Li, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2021a. Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4732–4743.

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021b. Temporal knowledge graph reasoning based on evolutional representation learning. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 408–417.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers.

Hongyuan Mei and Jason Eisner. 2016. The neural hawkes process: A neurally self-modulating multivariate point process. *arXiv preprint arXiv:1612.09328*.

Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.

Apoorv Saxena, Soumen Chakrabarti, and Partha P. Talukdar. 2021. Question answering over temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

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 *European Semantic Web Conference*, pages 593–607.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL-HLT*, pages 464–468.

Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. 2021. Neural temporal point processes: A review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4585–4593.

Josef Stoer and Roland Bulirsch. 2013. *Introduction to Numerical Analysis*, volume 12. Springer Science & Business Media.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Empirical Methods in Natural Language Processing*, pages 8306–8319.

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*, pages 3462–3471.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *7th International Conference on Learning Representations*.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426.

Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform bad for graph representation? *arXiv preprint arXiv:2106.05234*.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. 2020. Self-attentive hawkes process. In *International Conference on Machine Learning*, pages 11183–11193.

Ke Zhou, Hongyuan Zha, and Le Song. 2013. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics*, pages 641–649.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 4732–4740.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. 2020. Transformer hawkes process. In *International Conference on Machine Learning*, pages 11692–11702.

# A Appendix

## A.1 Dataset Statistics

Table 3 details the statistics of ICEWS14, ICEWS18, ICEWS05-15 and GDELT.

## A.2 Supplementary Results

In order to illustrate the performance of our model more comprehensively, we conduct experiments under different $\Delta t$. Table 1 reports the experimental results when $\Delta t = 5$. Since the experiments on the GDELT dataset are very time-consuming, Table 1 only provides the experimental results on the three ICEWS datasets.

Figure 2 presents the results of MRR for the GDELT dataset as $dt$ changes. Since the training of RE-NET and TITer on the GDELT dataset are too time-consuming, we use CyGNet instead for comparison. Because the time span of GDELT is relatively small, the change is not obvious.

## A.3 Efficiency Analysis

We analyzed the computational complexity of GHT for each module. The computational complexity of RGT is $O(NM^2d)$, where $N$ is the layer number, $M$ is the message number(in-degree of the graph), and $d$ is the dimension. For TT, the computational complexity is $O(\hat{N}L^2d)$, and $L$ is the sequence length. Thus, the computational complexity of GHT is $O(NM^2d + \hat{N}L^2d)$.

Figure 6 illustrates the inference time of RE-NET, RE-GCN, and GHT under the setting of $\Delta t = 5$ on ICEWS14. We can see that RE-NET inference is the slowest because it processes the query for each timestamp separately and can only predict the event that occurs at the next timestamp. RE-GCN is also a heuristic model, which only predicts the next timestamp step by step. In contrast, our model can parallelly predict events that occur at multiple different timestamps. Therefore, compared with RE-NET and RE-GCN, the longer the evolution

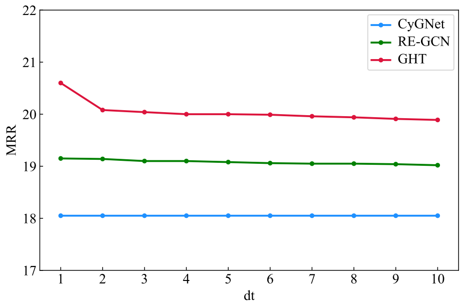


Figure 5: MRR on GDELT dataset with regard to various $dt$.

| Dataset | # entity | # relation | # train | # valid | # test | # timestamp | Time granularity |
|---|---|---|---|---|---|---|---|
| ICEWS14 | 7128 | 230 | 63685 | 13823 | 13222 | 365 | 24 hours |
| ICEWS18 | 23033 | 256 | 373018 | 45995 | 49545 | 304 | 24 hours |
| ICEWS0515 | 10488 | 251 | 322958 | 69224 | 69147 | 4017 | 24 hours |
| GDELT | 7691 | 240 | 1734399 | 238765 | 305241 | 2751 | 15 mins |

Table 3: Statistics of the datasets. The columns include the dataset name, number of entities, number of relationships, number of quadruples in the training set, number of quadruples in the validation set, number of quadruples in the test set, number of timestamps, and the time granularity.

| Model | ICEWS14 | | | | ICEWS18 | | | | ICEWS0515 | | | | GDELT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 | MRR | H@1 | H@3 | H@10 |
| TransE | 22.48 | 13.36 | 25.63 | 41.23 | 12.24 | 5.84 | 12.81 | 25.10 | 22.55 | 13.05 | 25.61 | 42.05 | - | - | - | - |
| Distmult | 27.67 | 18.16 | 31.15 | 46.96 | 10.17 | 4.52 | 10.33 | 21.25 | 28.73 | 19.33 | 32.19 | 47.54 | 8.61 | 3.91 | 8.27 | 17.04 |
| ComplEx | 30.84 | 21.51 | 34.48 | 49.59 | 21.01 | 11.87 | 23.47 | 39.97 | 31.69 | 21.44 | 35.74 | 52.04 | 9.84 | 5.17 | 9.58 | 18.23 |
| TTransE | 13.43 | 3.11 | 17.32 | 34.55 | 8.31 | 1.92 | 8.56 | 21.89 | 15.71 | 5.00 | 19.72 | 38.02 | 5.50 | 0.49 | 4.99 | 15.18 |
| TA-DistMult | 26.47 | 17.09 | 30.22 | 45.41 | 16.75 | 8.61 | 18.41 | 33.59 | 24.31 | 14.58 | 27.92 | 44.21 | 11.17 | 5.09 | 11.58 | 22.65 |
| DE-SimplE | 32.67 | 24.43 | 35.69 | 49.11 | 19.30 | 11.53 | 21.86 | 34.80 | 35.02 | 25.91 | 38.99 | 52.75 | - | - | - | - |
| TNTComplEx | 32.12 | 23.35 | 36.03 | 49.13 | 21.23 | 13.28 | 24.02 | 36.91 | 27.54 | 19.52 | 30.80 | 42.86 | - | - | - | - |
| CyGNet | 32.73 | 23.69 | 36.31 | 50.67 | 24.93 | 15.90 | 28.28 | 42.61 | 34.97 | 25.67 | 39.09 | 52.94 | 18.05 | 11.13 | 19.11 | 31.50 |
| TITer | 37.49 | **28.69** | 41.88 | 53.93 | 26.51 | **18.98** | 29.59 | 40.59 | 40.15 | 29.55 | 45.75 | 60.49 | 15.75 | 10.94 | 15.74 | 25.37 |
| RE-NET | 36.28 | 27.19 | 40.16 | 53.51 | 27.38 | 17.93 | 31.04 | 45.65 | 40.28 | 30.11 | 45.54 | 60.67 | - | - | - | - |
| RE-GCN | 36.95 | 28.04 | 40.92 | 54.07 | 28.01 | 18.97 | 31.29 | 45.84 | 41.19 | 31.32 | 46.08 | 60.33 | 19.11 | 12.19 | 20.25 | 32.43 |
| GHT | **38.28** | 28.43 | **42.85** | **57.47** | **28.38** | 18.78 | **32.01** | **47.27** | **42.90** | **31.76** | **48.77** | **64.64** | **20.14** | **12.75** | **21.50** | **34.60** |
| | ± 0.25 | ± 0.21 | ± 0.18 | ± 0.09 | ±0.18 | ±0.15 | ±0.17 | ±0.11 | ±0.20 | ±0.19 | ±0.21 | ±0.17 | ±0.18 | ±0.19 | ±0.10 | ±0.14 |

Table 4: Entity prediction results ($\Delta t = 5$).

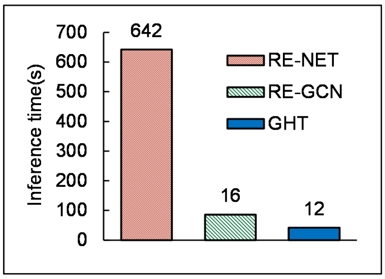
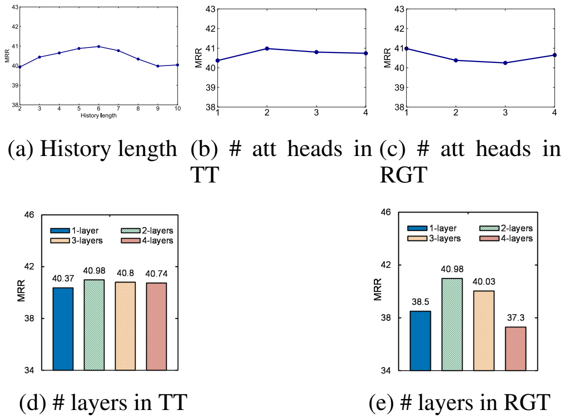


Figure 6: Inference time of RE-NET, RE-GCN and GHT on ICEWS14.



(a) History length (b) # att heads in TT (c) # att heads in RGT

(d) # layers in TT (e) # layers in RGT

Figure 7: Hyperparameter study on ICEWS14.

process is, the less time our model inference takes relatively.

## A.4 Sensitivity Analysis

We also study the hyperparameter's sensitivity of GHT, including the layer number and the attention head number of RGT, the layer number and the attention head number of TT, and the history length. We report the MRR results of entity prediction on ICEWS14 ($dt = 1$) in Figure 7.

Results in Table 1 demonstrate that GHT outperforms all static KG reasoning models and TKG interpolated reasoning models because these baselines fail to utilize the time information. For static methods, it is effortless to understand that it has no ability to model the time. The interpolated reasoning methods learn an embedding for each timestamp. However, in our experimental setting, the timestamps of inference are not seen during training, causing the models to use randomly initialized timestamp embeddings.

**RGT Analysis** The layer number of RGT corresponds to the aggregated neighbor order. In Figure 7e, the model performs best when the layer number is 2, aggregating more structural information than 1-layer. However, the model's performance will deteriorate as the number of layers increases. This may be because, in ICEWS, neighbors above the third order do not provide more information but instead introduce noise. Figure 7c shows that the number of attention heads has a small impact.

**TT Analysis** Figure 7b and 7d show the performance of GHT with different number of TT layers and attention heads. We notice that 2-layer TT and 2-head TT perform better than others. Compared with the number of RGT layers, the model is less sensitive to the number of TT's layers.

**History Length Analysis** GHT needs to model the historical sequence information, and we fix a hyperparameter to limit the maximum sequence length. Figure 7a shows the performance with different history length. If the length is no greater than 6, the longer the history, the better the GHT performance. However, continuing to extend the sequence will make the MRR fall. It shows that the effective information density is inversely proportional to the history length.