Learning Joint Structural and Temporal Contextualized Knowledge Embeddings for Temporal Knowledge Graph Completion

Yifu Gao[1], **Yongquan He**[2], **Zhigang Kan**[1], **Yi Han**[3], **Linbo Qiao**[1], **Dongsheng Li**[1]*
[1] National Key Laboratory of Parallel and Distributed Computing,
National University of Defense Technology, Changsha, China
[2] Meituan, Beijing, China
[3] College of Meteorology and Oceanography, National University of Defense Technology
{gaoyifu, kanzhigang13, hanyi12, qiao.linbo, dsli}@nudt.edu.cn
heyongquan@meituan.com

Abstract

Temporal knowledge graph completion that predicts missing links for incomplete temporal knowledge graphs (TKG) is gaining increasing attention. Most existing works have achieved good results by incorporating time information into static knowledge graph embedding methods. However, they ignore the contextual nature of the TKG structure, i.e., query-specific subgraph contains both structural and temporal neighboring facts. This paper presents the SToKE, a novel method that employs the pre-trained language model (PLM) to learn joint **S**tructural and **T**emporal C**o**ntextualized **K**nowledge **E**mbeddings. Specifically, we first construct an event evolution tree (EET) for each query to enable PLMs to handle the TKG, which can be seen as a structured event sequence recording query-relevant structural and temporal contexts. We then propose a novel temporal embedding and structural matrix to learn the time information and structural dependencies of facts in EET. Finally, we formulate TKG completion as a mask prediction problem by masking the missing entity of the query to fine-tune pre-trained language models. Experimental results on three widely used datasets show the superiority of our model.

1 Introduction

Knowledge graphs have facilitated many real-world applications, including question answering, dialogue systems and speech recognition (Ji et al., 2022). The rapidly growing facts on the knowledge graph often show dynamic relations or interactions of entities along the timeline, which creates the need for introducing the concept of temporal knowledge graph (TKG). Such TKGs often suffer from incompleteness due to their own dynamic features. Therefore, the temporal knowledge graph completion (TKGC) task that predicts missing links across these TKGs is gaining increasing attention from researchers (Boschee et al., 2015).

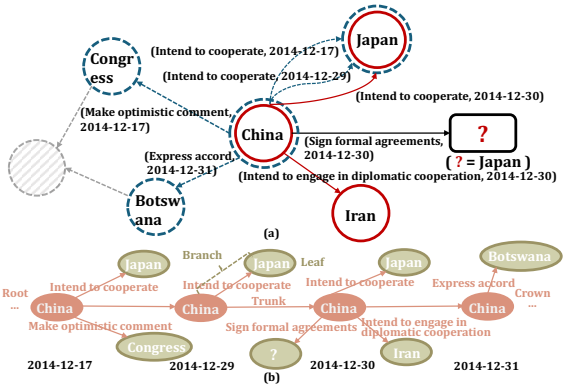

Figure 1: Part (a) shows an example of a query-specific subgraph, which consists of the facts where the query's subject *China* participates at the same or different timestamps. Part (b) shows the event evolutionary tree converted from the subgraph in Part (a).

Recent works on the TKGC task have primarily focused on static knowledge graph embedding (KGE) methods. They extend KGE models by encoding time information into conventional score functions in different ways (Jiang et al., 2016; García-Durán et al., 2018; Xu et al., 2020b; Lacroix et al., 2020; Messner et al., 2022), called temporal knowledge graph embedding (TKGE). Although TKGE methods outperform static KGE on this task, they mostly fail to consider the rich contextual information related to the query in the TKG structure.

As a subgraph of TKG shown in Figure 1(a), when facing the query (*China, Sign formal agreements, ?, 2014-12-30*), its relevant contexts include both concurrent facts within the same timestamp (solid red) and temporal adjacency facts at different timestamps (dashed blue). The structural contexts (*China, Intend to cooperate, Japan, 2014-12-30*) and (*China, Engage in diplomatic cooperation, Iran, 2014-12-30*) suggests that *China* may sign formal agreements with either *Japan* or *Iran*. Moreover, the temporal contexts illustrate that *China* has been expressing intent to cooperate with *Japan* for some time, making *Japan* a more reasonable

*Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2023, pages 417–430
July 9-14, 2023

answer to the query. Hence, learning knowledge embeddings that could effectively capture query-relevant structural and temporal contextual information facilitates the inference of missing facts. However, existing methods either only encode temporal adjacent facts residing in the query's local neighborhood (Jung et al., 2021), or only focus on concurrent events within the same timestamp and integrate entity representations of different timestamps with additional recurrence or attention mechanisms (Wu et al., 2020; Zhang et al., 2021). They fail to encode two query-relevant contexts simultaneously only using graph neural network variants, which requires us to solve the above problem by introducing a new architecture. Recent advances in pre-trained language models (PLMs) are able to learn word representations and graph representations through distinct structured contexts (Devlin et al., 2019; Zhang et al., 2020a). Intuitively, PLMs can capture contextual meanings of entity and relation representations along joint structural and temporal dimensions.

Inspired by this, we propose the SToKE, a novel BERT-based model to learn knowledge embeddings with joint structural and temporal contexts. Another challenging problem is to draw connections of query-specific substructure to natural language sentences. To handle this issue, we construct an event evolution tree (EET) for each individual query as shown in Figure 1(b). The EET represents the temporal evolution in the order from "Root" to "Crown", where each layer corresponds to different timestamps. Facts occurring at the same timestamp constitute the "Branches" of this layer, and query's subject entity *China* acts as a "Trunk" connecting different structural and temporal adjacent facts. EET is essentially a structured event sequence that helps PLMs better understand contextual information in the query-specific subgraph. Besides, we add a novel temporal embedding in the initial embeddings to ensure the sequential order of facts appearing in the EET. Considering the knowledge noise problem caused by introducing query-relevant contexts with different dimensions, we present a structural matrix to restrict the structure dependencies between facts. Overall, this paper makes the following contributions:

1) We propose SToKE, a novel model for the TKGC task that learns knowledge embeddings in terms of joint structural and temporal contexts via the pre-trained language model.

2) To enable BERT-like PLMs to handle the query-specific subgraph, we propose an event evolution tree that transforms structural and temporal contexts into a structured event sequence. The temporal order and structural dependencies of facts in the EET are limited by adopting the temporal embedding and structural matrix.

3) We formulate the TKGC task as a mask prediction task by masking the missing entity of the query to fine-tune pre-trained BERT models.

4) Our SToKE outperforms the state-of-the-art methods over three widely used datasets, which demonstrates our model's superiority.

2 Related Work

2.1 Knowledge Graph Embedding

KGE aims to map the entities and relations into continuous vector space and score the plausibility of a triple, which can be roughly divided into distance-based models, semantic matching models and neural network models. Distance-based models measure the plausibility of a triple as the distance between the relation-translated subject and object entity embeddings, such as TransE (Bordes et al., 2013) and its various extensions (Wang et al., 2014; Lin et al., 2015). Especially, RotatE (Sun et al., 2019) treats each relation as a rotation from the subject to the object. Semantic matching models measure the plausibility of a fact by mining the underlying semantics between entity and relation embeddings, e.g., DisMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016) and SimplE (Kazemi and Poole, 2018). They employ a bilinear score function that represents relations as linear transformations acting on entity embeddings. Besides, some models incorporate neural networks to encode the semantics of knowledge graphs. ConvE (Dettmers et al., 2018) and ConvKB (Nguyen et al., 2018) apply convolutional layers to model interactions between entities in the scoring function. Several methods (Schlichtkrull et al., 2018; Nathani et al., 2019; Zhang et al., 2020b) adopt variants of graph neural network to contextualize entity embedding with the corresponding neighborhood structure. However, above methods are not applicable to TKGs due to their ignorance of time information.

2.2 Temporal Knowledge Graph Completion

Recently, some attempts have extended the static KG embedding methods by incorporating time

information to improve the performance. Some TKGE models are extended from distance-based models (Jiang et al., 2016; Dasgupta et al., 2018; Xu et al., 2020a,b; Chen et al., 2022; Messner et al., 2022). They encode time information into a translation-based score function in different ways. ChronoR (Sadeghian et al., 2021) builds on RotatE, representing time-relation pairs with rotation and scaling in the embedding space. Some models are temporal extensions of semantic matching methods. TA-DistMult (García-Durán et al., 2018) adopts recurrent neural networks (Cho et al., 2014) to learn time-aware representations of relations. DE-SimplE (Goel et al., 2020) utilizes diachronic entity embeddings to represent entities at different time steps. Furthermore, TNTComplEx (Lacroix et al., 2020), TeLM (Xu et al., 2021) and Time-LowFER (Dikeoulias et al., 2022) perform a tensor decomposition involving the timestamp embedding with distinct embedding representations and product operators. While successfully extending to TKGs, these models ignore the rich contextual information in the graph structure.

Another line of models focuses on the TKG structure information based on variants of graph neural networks (GNN). TeMP (Wu et al., 2020) and ST-ConvKB (Zhang et al., 2021) regard TKG as a sequence of KGs corresponding to different timestamps and use GNN variants and sequential models to generate dynamic entity representations. T-GAP (Jung et al., 2021) attentively aggregates query-relevant information from each entity's local temporal adjacent facts with its temporal GNN. Moreover, SPA (Wang et al., 2022c) utilizes neural architecture search to design data-specific GNN architectures for different datasets. Other similar works (Jin et al., 2020; Li et al., 2021; Gao et al., 2022) are designed for the extrapolation problem rather than TKGC, and EvoExplore (Zhang et al., 2022) focuses on local and global structure evolutions using the temporal point process. However, above methods cannot encode structural and temporal contexts simultaneously using only GNN variants. To the best of our knowledge, we are the first to handle two query-relevant contexts simultaneously with PLMs.

2.3 Language Model and Knowledge Graph

Joint pre-trained language models (PLMs) and knowledge graph approaches can be broadly classified into two categories: one that introduces knowledge from KGs to enhance PLMs on NLP downstream tasks (Zhang et al., 2019; Peters et al., 2019; He et al., 2020). However, they freeze the knowledge embedding during training PLMs, which are not real models for learning knowledge representations. To solve the above problem, K-BERT (Liu et al., 2020) and CoLAKE (Sun et al., 2020) combine entities and relations in the form of tokens with text, which are jointly fed into the PLMs. The other is to learn the knowledge embeddings from natural language texts through PLMs, called text-based methods, which are orthogonal to our work. Recent methods learn to generate entity embeddings with PLMs from entity text descriptions (Zhang et al., 2020c; Yao et al., 2019; Wang et al., 2021a). Specially, KEPLER (Wang et al., 2021b) encodes textual entity descriptions with jointly optimizing KE and language modeling objectives. LMKE (Wang et al., 2022b) and SimKGC (Wang et al., 2022a) introduce efficient contrastive learning to improve the performance of text-based methods. But none of these models consider the temporal aspect of knowledge graphs. ECOLA (Han et al., 2022) uses PLMs to enhance temporal knowledge graph embeddings with temporally relevant texts. Nevertheless, not all facts have summary texts in the practical application. Besides, above approaches focus on the combination of structured knowledge and unstructured text, which is fundamentally different from our method focusing on the contextual information of the TKG subgraph.

3 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). Each event (fact) in the $\mathcal{G} = \{G_0, G_1, \ldots, G_T\}$ 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 temporal knowledge graph completion task is to infer the missing object entity o given the query $(s, r, ?, t)$, or missing subject entity s of the query $(?, r, o, t)$, $t \in \{0, \ldots, T\}$.

4 Method

As shown in Figure 2, our model consists of two stages. The first step is to construct an event evolution tree (EET) for each query. EET can be viewed

as a structured event sequence incorporating structural and temporal contextual information related to the predicted fact. Second, we present our SToKE model, which learns joint structural and temporal contextualized knowledge representations based on the constructed EET.

4.1 Event Evolution Tree Construction

The event evolution tree is used to transform structural and temporal contexts related to the query into a structured knowledge sequence. Specifically, for the query $(s, r, ?, t)$, we construct an EET $C = \{c_{t-m}, ..., c_{t-1}, c_t, c_{t+1}, ..., c_{t+m}\}$, where $c_i = \{s\{(r_{i1}, o_{i1}), ..., (r_{in}, o_{in})\}\}$ denotes the fact set at timestamp i that the subject entity s participates in. We constrain the time interval between the timestamp of the query and each fact to be no more than m. Note that in the TKGC setting, we assume there is missing data at some time point t but the other snapshot information is available during training (Wu et al., 2020). Hence, we integrate more temporal information from both past and future timestamps. Each fact set c_i contains at most n facts occurring at timestamp i which are composed of the subject entity s of the query, the 1-hop neighbor entity o_i and the relation r_i linking two entities. Each fact can be viewed as a "Branch" of the ith layer, and subject entities s of distinct layers constitute the "Trunk". We design two strategies to select n facts in each set c_i:

1) Active Facts First(AFF) (Wu et al., 2020): Prefer facts that appear more often in time window m among neighbor facts;

2) Repetitive Facts First(RFF) (Zhu et al., 2021): Prefer facts that have the same relation as query among neighbor facts, and the AFF strategy is followed if the number of facts does not reach n.

As shown in Figure 2(a), the EET is constructed from the query-specific subgraph in Figure 1(a) (m=1, n=2). The query's subject entity *China* is shared by each event in this EET structure, connecting distinct events in both structural and temporal dimensions. Section 4.2.2 shows the reason for China's ability to bridge different dimensional facts from a fine-grained perspective. Similarly, for the query $(?, r, o, t)$, the fact set c_i is defined by $\{o\{(\dot{r}_{i1}, s_{i1}), ..., (\dot{r}_{in}, s_{in})\}\}$, where $\dot{r}$ denotes the inverse relation of r.

4.2 The Proposed SToKE Model

As shown in Figure 2, the overall framework of SToKE has three main components. The first part is an embedding layer to learn initial knowledge representations with three distinct embeddings. The second part is a mask transformer encoder to model the interactions among facts in different dimensional contexts and learn contextualized knowledge representations. The third part is a prediction layer to infer the missing entity based on the hidden representation of the [*MASK*] token.

4.2.1 Learning Initial Knowledge Embeddings

The critical challenge is how to preserve the temporal information of the EET in the knowledge representations. We add a novel temporal embedding to focus on the relative position and absolute displacement of facts in the temporal dimension.

Token Embedding Similar to BERT, we use two special tokens, i.e., [*CLS*] and [*SEP*], as the beginning and end of the text, respectively. We employ the [*MASK*] token to mask the missing entity of the given query and align the different lengths of text with [*PAD*] token. As shown in Figure 2(a), we treat each fact component in the EET as individual tokens and unroll the tree structure in absolute position order to obtain a new input text, $S = \{e_0, e_1, \ldots, e_{l-1}\}$, where e represents entity, relation or special tokens, and l is the max sequence length. However, some tokens consisting of multiple words, such as *Intend_to_cooperate*, will lose original semantics as part of the event while being tokenized with the BERT vocabulary. Hence, we create two new lookup tables for entities and relations, denoted by $\boldsymbol{T}_{ent}$ and $\boldsymbol{T}_{rel}$. We add four special tokens and consider the inverse relation of r for query $(?, r, o, t)$, so the entity lookup table $\boldsymbol{T}_{ent} \in \mathbb{R}^{(|\mathcal{E}|+4)\times d}$, and the relation lookup table $\boldsymbol{T}_{rel} \in \mathbb{R}^{2|\mathcal{R}|\times d}$, where d is the hidden size, $|\mathcal{E}|$ and $|\mathcal{R}|$ are the total number of elements in the entity set and relation set, respectively.

Temporal Embedding To exploit the order information of tokens in the sequence, BERT adds an absolute position embedding. However, only considering the absolute position index will disrupt the temporal order of the facts occurring in the EET. Taking the input text in Figure 2(b) as an example, although *Intend_to_cooperate* and *Japan* are inserted after *Sign_formal_agreements* and [*MASK*], (*China*, *Intend_to_cooperate*, *Japan*) and (*China*, *Sign_formal_agreements*, [*MASK*]) are both events occurring at the same time and should have consistent temporal location information. To solve above issue, we present a temporal position embedding

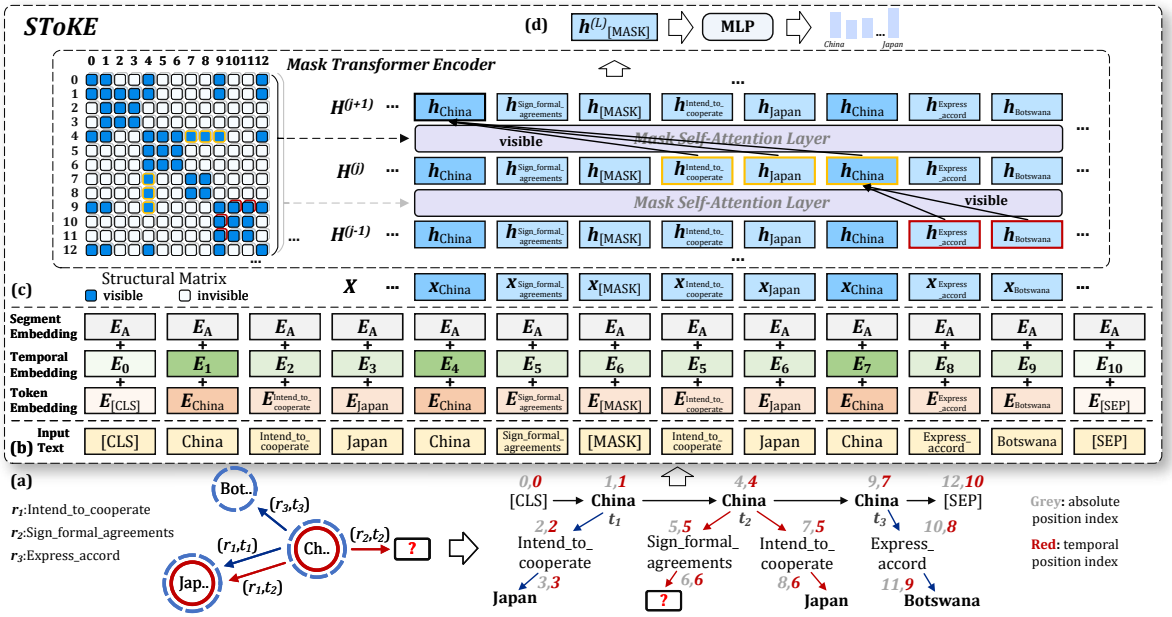

Figure 2: The framework of our proposed SToKE model. Suppose (a) is our constructed EET from the subgraph in Figure 1. Our SToKE consists of three components: Part (b) is to convert the EET into an embedding representation with the temporal position and displacement information. Part (c) is to learn contextualized knowledge representations with the structural matrix. And part (d) is to predict the missing entity with the [*MASK*] token.

to replace the origin absolute position embedding of BERT. Specifically, *Intend_to_cooperate* and *Japan* have the same temporal position indexes 5 and 6 as *Sign_formal_agreements* and [*MASK*], rather than absolute position indexes 7 and 8. From the view of temporal order, the above two events are equivalent to occurring at the same timestamp.

However, the temporal position embedding considers the relative temporal position of tokens in the text, ignoring the absolute temporal gap between timestamps of the query and each event. Concretely, we add a temporal displacement embedding as follows:

$$\boldsymbol{E}_j = \boldsymbol{W}_{\Delta t}(\boldsymbol{e}_j + \boldsymbol{\tau}_{|\Delta t_j|}), \tag{1}$$

$$\boldsymbol{W}_{\Delta t} = \begin{cases} \boldsymbol{W}_{past} & \Delta t_j < 0 \\ \boldsymbol{W}_{now} & \Delta t_j = 0 \\ \boldsymbol{W}_{future} & \Delta t_j > 0 \end{cases},$$

where $\Delta t_j = t_j - t_{query}$, $\boldsymbol{e}_j \in \mathbb{R}^d$ is the temporal position embedding, and j is the temporal position index. We learn the discrete embedding of temporal displacement $\boldsymbol{\tau}_{|\Delta t_j|} \in \mathbb{R}^d$, and consider the sign of the displacement by applying different weights $\boldsymbol{W}_{\Delta t} \in \mathbb{R}^{d \times d}$. Then, we follow the segment embeddings of BERT, and treat each input text as one sentence with the same segment tag. At last, we sum the above three embeddings to get the final embeddings $\boldsymbol{X} \in \mathbb{R}^{l \times d}$, and feed it to the mask transformer encoder.

4.2.2 Contextualized Knowledge Embeddings

In order to learn contextualized embeddings of the missing entity, we propose the mask transformer encoder, which differs from the BERT block by introducing the structural matrix $\boldsymbol{M}$ to restrict the self-attention area. Specifically, the matrix $\boldsymbol{M}$ controls the dependencies among structural and temporal contextual facts, making our model feasible and efficient. As shown in the left part of Figure 2(c), the row and column of the matrix are absolute position indexes, where blue dots mean visible and white dots mean invisible. Specifically, three tokens *China*, *Intend_to_cooperate* and *Japan* interact with each other because they are components of the same fact. Therefore the points in the matrix where their corresponding indexes (1, 2 and 3) cross are blue. In contrast, *Japan* should not be affected by *Express_accord* since they are the object and relation of two facts, respectively. The points where their corresponding indexes (3 and 10) intersect are white. Hence, the structural matrix $\boldsymbol{M} \in \mathbb{R}^{l \times l}$ is defined as shown below:

$$\boldsymbol{M}[a,b] = \begin{cases} 0, & \text{if } e_a \ominus e_b \\ -\infty, & \text{others.} \end{cases}, \tag{2}$$

here, $e_a \ominus e_b$ indicates that e_a and e_b are in the same "Branch" or "Trunk" mentioned in Section 4.1, which means e_a and e_b are visible to each other, a and b are the absolute position indexes. In particular, the [*CLS*] and [*SEP*] tokens are also treated as a part of "Trunk".

Moreover, the mask transformer encoder is stacked by L mask self-attention layers, which maps the input embeddings $\boldsymbol{X}$ to contextual representations $\boldsymbol{H}^{(L)}$, where $\boldsymbol{X}, \boldsymbol{H}^{(L)} \in \mathbb{R}^{l \times d}$. At each layer, we also 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 embeddings $\boldsymbol{X}$ into queries, keys, and values of the i-th scaled dot-product attention head, where $d' = \frac{d}{K}$, and $i = 1, 2, \ldots, K$. The specific function is shown below:

$$\boldsymbol{H}^i = f(\frac{\left(\boldsymbol{X}\boldsymbol{W}_q^i\right)\left(\boldsymbol{X}\boldsymbol{W}_k^i\right)^T}{\sqrt{d'}} + \boldsymbol{M})(\boldsymbol{X}\boldsymbol{W}_v^i), \quad (3)$$

where $\boldsymbol{H}^i \in \mathbb{R}^{l \times d'}$ is the output representation of corresponding attention head, f is the softmax function, and we cascade them to get the output $\boldsymbol{H} \in \mathbb{R}^{l \times d}$. When $\boldsymbol{M}[a, b] = -\infty$, the softmax function makes the attention weight to zero, preventing token e_a from computing the attention score of token e_b.

As the example illustrated in Figure 2(c), $\boldsymbol{h}_{[\text{MASK}]}^{(j+1)}$ is affected by $\boldsymbol{h}_{\text{China}}^{(j+1)}$, and obtain the information of $\boldsymbol{h}_{\text{Japan}}^{(j)}$ indirectly through $\boldsymbol{h}_{\text{China}}^{(j+1)}$. Meanwhile, $\boldsymbol{h}_{\text{Botswana}}^{(j-1)}$ can pass information to $\boldsymbol{h}_{[\text{MASK}]}^{(j+1)}$ through $\boldsymbol{h}_{\text{China}}^{(j)}$ and $\boldsymbol{h}_{\text{China}}^{(j+1)}$. Similarly, *Intend_to_cooperate* and *Express_accord* tokens also have an effect on [*MASK*]. Thus, the [*MASK*] token incorporates the information of structural neighboring fact (*China*, *Intend_to_cooperate*, *Japan*) and temporal adjacency fact (*China*, *Express_accord*, *Botswana*). The *China* token acts as a "bridge" between facts of different dimensions.

4.2.3 Predicting the Missing Entity

As described in Section 4.1, we create two training instances for each fact, one by replacing the missing object entity of query $(s, r, ?, t)$ with a special token [MASK], and the other by replacing the subject entity of query $(?, r, o, t)$ with the [MASK] token. Then we treat the TKGC task as a mask prediction problem, and input the final contextual embedding $\boldsymbol{h}_{[\text{MASK}]}^{(L)}$ into the multi-layer perceptron (MLP) decoder to predict the occurrence probabilities of all entities:

$$p\left(\text{e}\right) = \text{Softmax}((\boldsymbol{h}_{[\text{MASK}]}^{(L)}\boldsymbol{w}_1 + \boldsymbol{b}_1)\boldsymbol{w}_2 + \boldsymbol{b}_2), \quad (4)$$

where $\boldsymbol{w}_1 \in \mathbb{R}^{d \times d}$, $\boldsymbol{b}_1 \in \mathbb{R}^d$ are the parameters of the first linear layer, and $\boldsymbol{w}_2 \in \mathbb{R}^{d \times (|\mathcal{E}|+4)}$, $\boldsymbol{b}_2 \in \mathbb{R}^{(|\mathcal{E}|+4)}$ are the parameters of second.

Datasets	$\mid \mathcal{E} \mid$	$\mid \mathcal{R} \mid$	$\boldsymbol{N}_{train}$	$\boldsymbol{N}_{valid}$	$\boldsymbol{N}_{test}$	$\mid \mathcal{T} \mid$
ICEWS14	7,128	230	72,826	8,941	8,963	365
ICEWS05-15	10,488	251	368,962	46,275	46,092	4,017
GDELT	500	20	2,735,685	341,961	341,961	366

Table 1: Dataset Statistics of three datasets. ($\boldsymbol{N}_{train}$, $\boldsymbol{N}_{valid}$ and $\boldsymbol{N}_{test}$ are the number of facts in training, validation and test sets.)

4.3 Training Objective

We regard the mask prediction problem as a multi-class classification task and use the cross-entropy function to calculate the loss during the training process:

$$\mathcal{L} = -\sum_{(s,r,o,t)} \log p\left(o_t\right) + \log p\left(s_t\right), \quad (5)$$

$(s, r, o, t) \in \mathcal{G}$ represents the known facts in the training set, $p(*)$ represents the probability scores of corresponding entities obtained from Eq.(4).

5 Experiments

5.1 Dataset and Metrics

We evaluate our model on three typical datasets commonly used in previous studies, namely ICEWS14, ICEWS05-15 (García-Durán et al., 2018) and GDELT (Trivedi et al., 2017). Dataset statistics are described in Table 1. For ICEWS (Boschee et al., 2015), a well-established event-based datasets, we use two subsets corresponding to facts from 2014/1/1 to 2014/12/31 and facts from 2005/1/1 to 2015/12/31, i.e., ICEWS14 and ICEWS05-15. For GDELT (Leetaru and Schrodt, 2013), we use the subset provided by Trivedi et al. (2017) corresponding to facts from 2015/4/1 to 2016/3/31. Finally, we split all datasets into train, validation and test set with the same partitioning by Goel et al. (2020). And we report the link prediction performance on two evaluation metrics under the time-wise filtered setting (Goel et al., 2020): MRR and Hits@k (1, 3, 10). More details about metrics and implementation are summarized in Appendix A and B, respectively.

Method	ICEWS14				ICEWS05-15				GDELT			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE (2013)	.280	.094	–	.637	.294	.090	–	.663	–	–	–	–
DisMult (2015)	.441	.325	.498	.668	.457	.338	.515	.691	.210	.133	.224	.365
ComplEx (2016)	.442	.440	.430	.664	.464	.347	.524	.696	.213	.133	.225	.366
SimplE (2018)	.458	.341	.516	.687	.478	.359	.539	.708	.206	.124	.220	.366
TTransE (2016)	.255	.074	–	.601	.271	.084	–	.616	.115	.000	.160	.318
TeRo (2020)	.562	.468	.621	.732	.586	.469	.668	.795	.245	.154	.264	.420
ChronoR (2021)	.625	.547	.669	.773	.675	.596	.723	.820	–	–	–	–
RotateQVS (2022)	.591	.507	.642	.754	.633	.529	.709	.813	.270	.175	.293	.458
BoxTE (2022)	.613	.528	.664	.763	.667	.582	.719	.820	.352	.269	.377	.511
TA-DisMult (2018)	.477	.363	–	.686	.474	.346	–	.728	.206	.124	.219	.365
DE-SimplE (2020)	.526	.418	.592	.725	.513	.392	.578	.748	.230	.141	.248	.403
TNTComplEx (2020)	.620	.520	.660	.760	.670	.590	.710	.810	.223	.142	.237	.379
TeLM (2021)	.625	.545	.673	.774	.678	.599	.728	.823	–	–	–	–
Time-LowFER (2022)	.623	.549	.671	.757	.638	.555	.690	.791	–	–	–	–
TeMP (2020)	.601	.478	.681	**.828**	.691	.566	.782	**.917**	.275	.191	.297	.437
T-GAP (2021)	.610	.509	.677	.790	.670	.568	.743	.845	–	–	–	–
SPA (concurrent work)	.658	.544	.737	.857	.713	.580	.820	.933	.360	.282	.384	.510
SToKE	**.659**	**.574**	**.693**	.803	**.712**	**.605**	**.790**	.885	**.371**	**.290**	**.399**	**.525**

Table 2: Performance comparison on three benchmark datasets. The best and second best results are marked in **bold** and underlined, respectively. Results are taken from the responding literature, and "–" means not reported.

5.2 Baselines

We compare our model with representative static KG embeddings: TransE (Bordes et al., 2013), DisMult (Yang et al., 2015), SimplE (Kazemi and Poole, 2018) and ComplEx (Trouillon et al., 2016), existing advanced TKGE approaches, including distance-based extensions: TTransE (Jiang et al., 2016), TeRo (Xu et al., 2020b), ChronoR (Sadeghian et al., 2021), RotateQVS (Chen et al., 2022) and BoxTE (Messner et al., 2022), extensions of semantic matching methods: TA-DisMult (García-Durán et al., 2018), DE-SimplE (Goel et al., 2020), TNTComplEx (Lacroix et al., 2020), TeLM (Xu et al., 2021) and Time-LowFER (Dikeoulias et al., 2022), and GNN-based methods: TeMP (Wu et al., 2020), T-GAP (Jung et al., 2021) and concurrent work SPA (Wang et al., 2022c). Among the above baselines, GNN-based methods focus on query-relevant contextual information.

5.3 Main Results

Table 2 reports the link prediction results of all methods on three benchmark datasets. We can observe that SToKE consistently outperforms the baselines on all datasets. Especially on the GDELT dataset, our model achieves improvements of 5.4% in MRR and 7.8% in Hits@1 over the best baseline. The possible reason is that the GDELT dataset is substantially denser (the training set contains about 2.7 million facts for 500 entities, 20 relations), and thus involves richer structural and temporal contextual information.

In general, most temporal models perform much better than the static KGE methods since they consider temporal information in distinct ways. Specifically, the temporal extensions of static methods outperform the original counterpart for TKGC task, e.g., TeRo and TransE, DE-SimplE and SimplE, suggesting that it is feasible to incorporate temporal information into the embeddings or scoring functions. Among temporal extensions of semantic matching methods, TeLM achieves the best performance on ICEWS14 and ICEWS05-15 due to its more expressive multi-vector embedding for modeling entities, relations and timestamps of TKGE. For distance-based TKGE approaches, ChronoR and BoxTE outperform other methods because they incorporate temporal information into translation model variants, i.e., rotational or spatio-translational score functions. Overall, both types of TKGE methods achieve good results on this task.

GNN-based methods perform strongly on three datasets, especially achieving the best results other than our model on Hits@3 and Hits@10 metrics, most notably because they additionally consider query-relevant contextual information to infer the missing facts. However, query-relevant facts may introduce knowledge noise effects, causing GNN approaches to be slightly worse than some TKGE models on the Hits@1 metric. There is no doubt that our model achieves better results because we consider both structural and temporal facts on inference. We construct the input text by filtering out some relevant facts through heuristic strategies,

Query: (A, Criticize_or_denounce, ?, 2014-1-29)						Answer: B				
Layer	2014-1-27					2014-1-29				
12	A	Criticize_or_denounce	B	Engage_in_negotiation	B	A	Criticize_or_denounce	***[MASK]***	Host_a_visit	C
	–	–	–	–	–	.691	.138	.171	–	–
11	A	Criticize_or_denounce	B	Engage_in_negotiation	B	***A***	Criticize_or_denounce	[MASK]	Host_a_visit	C
	.256	–	–	–	–	.097	.065	.083	.021	.034
10	***A***	Criticize_or_denounce	B	Engage_in_negotiation	B	A	Criticize_or_denounce	[MASK]	Host_a_visit	C
	.068	.066	.282	.057	.199	.030	–	–	–	–

Table 3: A partial input text related to the query (*A*, *Criticize_or_denounce*, *?*, *2014-1-29*), where A, B and C represent *South_Korea*, *North_Korea* and *Japan*, respectively. We only list the attention scores with bolded token as the query value in each layer.

	ICEWS14			
Models	MRR	Hits@1	Hits@3	Hits@10
SToKE	.659	.574	.693	.803
- *SC*	.647	.556	.688	.792
- *TC*	.543	.428	.593	.755
+*AFF*	.641	.550	.678	.796
- *TE*	.641	.557	.675	.791
- *SM*	.634	.547	.669	.778
+ *BERT-large*	.662	.576	.700	.809

Table 4: Performance of different variants on ICEWS14.

which reduces the noise effect to some extent and allows our model to achieve the best results on the Hits@1 metric. Meanwhile, heuristic strategies may also ignore some inactive entities as query answers (Wu et al., 2020), which makes our model slightly less effective than TeMP on Hits@10. Furthermore, the performance of our model still shows strong competitiveness compared to the concurrent work. SPA designs specific GNN architectures for different datasets, while we use a unified BERT architecture to explore TKG's topological and temporal properties simultaneously.

5.4 Ablation Study

As shown in Table 4, we conduct ablation experiments on ICEWS14, and discuss the effects of different variants as follows:

EET Variants. The third block shows the results corresponding to the different construction methods of EET in the first step. - *SC (Structural Contexts)* indicates that we construct the EET only considering temporal adjacency facts associated with the query, ignoring the interactions between facts within the same timestamp, i.e., making the hyperparameter n to 1. Similarly, - *TC (Temporal Contexts)* means we only incorporate concurrent facts with the query into the EET, making m to 0. The results demonstrate that both structural and temporal contexts in the query-specific subgraph contribute to the prediction, which exactly validates our motivation. +*AFF* indicates that we replace the RFF strategy with the AFF strategy mentioned in Section 4.1, which implies that the RFF heuristic has the advantage of exploring neighboring entities related to the query.

Model Variants. We use an absolute position embedding and a fully connected matrix instead of the temporal embedding (TE) and the structural matrix (SM), and denote them by - *TE* and - *SM*, respectively. It can be seen that the performance of two model variants decreases a lot, proving the validity of two model components. Besides, we use a larger pre-trained BERT-large model to replace the original BERT-base. The more parameters fully explore the query-related contexts, resulting in a performance improvement, which indicates that our model can fine-tune other pre-trained BERT models for the link prediction task.

5.5 Case Study

In order to show how SToKE learns contextual information related to the query, we provide an example in Table 3 from the test set of ICEWS14. When facing query (*A*, *Criticize_or_denounce*, *?*, *2014-1-29*), the model tries to find the answer from its relevant contextual facts. Due to the limitations of the structural matrix, the [*MASK*] token puts more attention on the *A* token at the 12th layer. Through attention scores at the 11th layer, it can be observed that *A* does not assign a higher score to *C* at the same timestamp but focuses on its adjacent timestamped *A* token. Moreover, the *A* token at *2014-1-27* aggregates the hidden information of *B* and passes it to the [*MASK*] token through the *A* at *2014-1-29*, which illustrates the reason why *A* as a "Trunk" can "bridge" different contexts mentioned in Section 4.1. We argue that the model assigns

scores in this way because the concurrent fact *(A, Host_a_vist, C)* holds an opposite meaning to the query, while the relation *Engage_in_negotiation* of temporal adjacent fact is typically accompanied by *Criticize_or_denounce*. The example also verifies that our model can explore both structural and temporal contexts related to the query.

6 Conclusion

In this paper, we propose a novel model for the temporal knowledge graph completion task named SToKE, which learns contextualized knowledge representations in terms of joint structural and temporal dimensions. Unlike other GNN-based methods, our model uses a unified BERT architecture to simultaneously explore contextual information of the TKG substructure, i.e., query-relevant structural and temporal neighboring facts. To enable BERT to handle the TKG, we construct an event evolution tree (EET) for each individual query, and introduce temporal embedding and structural matrix to ensure the temporal order and structural dependencies among facts in EET. Through masking the missing entity of query to fine-tune the pre-trained BERT, our model outperforms other methods on three widely used datasets.

Limitations

Our model simultaneously encodes structural and temporal contexts of the TKG substructure, and uses heuristic strategies to select a portion of query-relevant facts as input texts for PLMs. We can achieve stunning results with these selected facts. However, this work only considers the query-relevant one-hop neighbor facts to achieve a good performance improvement, but ignores the benefits of multi-hop neighbor facts. We leave it for future work to verify the effectiveness of multi-hop paths.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 62025208 and 61932001.

References

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

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

Kai Chen, Ye Wang, Yitong Li, and Aiping Li. 2022. Rotateqvs: Representing temporal information as rotations in quaternion vector space for temporal knowledge graph completion. In *ACL*, pages 5843–5857.

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

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

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818.

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

Ioannis Dikeoulias, Saadullah Amin, and Günter Neumann. 2022. Temporal knowledge graph reasoning with low-rank and model-agnostic representations. In *RepL4NLP@ACL*, pages 111–120.

Yifu Gao, Linhui Feng, Zhigang Kan, Yi Han, Linbo Qiao, and Dongsheng Li. 2022. Modeling precursors for temporal knowledge graph reasoning via auto-encoder structure. In *IJCAI*, pages 2044–2051.

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

Rishab Goel, Seyed Mehran Kazemi, Marcus A. Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *AAAI*, pages 3988–3995.

Zhen Han, Ruotong Liao, Beiyan Liu, Yao Zhang, Zifeng Ding, Heinz Köppl, Hinrich Schütze, and Volker Tresp. 2022. Enhanced temporal knowledge embeddings with contextualized language representations. *CoRR*, abs/2203.09590.

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. 2020. BERT-MK: Integrating graph contextualized knowledge into pre-trained language models. In *EMNLP (Findings)*, pages 2281–2290.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications.

IEEE Transactions on Neural Networks and Learning Systems, 33(2):494–514.

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

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

Jaehun Jung, Jinhong Jung, and U Kang. 2021. Learning to walk across time for interpretable temporal knowledge graph completion. In *KDD*, pages 786–795.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*, pages 4289–4300.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. In *ICLR*.

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

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

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908.

Johannes Messner, Ralph Abboud, and İsmail İlkan Ceylan. 2022. Temporal knowledge graph completion using box embeddings. In *AAAI*, pages 7779–7787.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *ACL*, pages 4710–4723.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT*, pages 327–333.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP*, pages 43–54.

Ali Sadeghian, Mohammadreza Armandpour, Anthony Colas, and Daisy Zhe Wang. 2021. Chronor: Rotation based temporal knowledge graph embedding. In *AAAI*, pages 6471–6479.

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

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. In *COLING*, pages 3660–3670.

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

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

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

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021a. Structure-augmented text representation learning for efficient knowledge graph completion. In *WWW*, pages 1737–1748.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022a. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. In *ACL*, pages 4281–4294.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Xintao Wang, Qianyu He, Jiaqing Liang, and Yanghua Xiao. 2022b. Language models as knowledge embeddings. In *IJCAI*, pages 2291–2297.

Zhen Wang, Haotong Du, Quanming Yao, and Xuelong Li. 2022c. Search to pass messages for temporal knowledge graph completion. In *EMNLP (Findings)*, pages 6189–6201.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119.

Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L. Hamilton. 2020. Temp: Temporal message passing for temporal knowledge graph completion. In *EMNLP*, pages 5730–5746.

Chengjin Xu, Yung-Yu Chen, Mojtaba Nayyeri, and Jens Lehmann. 2021. Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings. In *NAACL-HLT*, pages 2569–2578.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Jens Lehmann, and Hamed Shariat Yazdi. 2020a. Temporal knowledge graph embedding model based on additive time series decomposition. In *ISWC*, pages 654–671.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020b. Tero: A time-aware knowledge graph embedding via temporal rotation. In *COLING*, pages 1583–1593.

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

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193.

Jiasheng Zhang, Shuang Liang, Zhiyi Deng, and Jie Shao. 2021. Spatial-temporal attention network for temporal knowledge graph completion. In *DASFAA*, pages 207–223.

Jiasheng Zhang, Shuang Liang, Yongpan Sheng, and Jie Shao. 2022. Temporal knowledge graph representation learning with local and global evolutions. *Knowledge-Based Systems*, 251:109234.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020a. Graph-bert: Only attention is needed for learning graph representations. *CoRR*, abs/2001.05140.

Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhi-Ping Shi, Hui Xiong, and Qing He. 2020b. Relational graph neural network with hierarchical attention for knowledge graph completion. In *AAAI*, pages 9612–9619.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL*, pages 1441–1451.

Zhiyuan Zhang, Xiaoqian Liu, Yi Zhang, Qi Su, Xu Sun, and Bin He. 2020c. Pretrain-kge: Learning knowledge representation from pretrained language models. In *EMNLP (Findings)*, pages 259–266.

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

A Metrics

We use Mean Reciprocal Rank (MRR) and the proportion of correct quadruples ranked in top 1, 3 and 10 (Hits@1, Hits@3, and Hits@10) to evaluate the model performance. There are two filtered setting, static filtered (Bordes et al., 2013) and time-wise filtered (Goel et al., 2020). The static filtered setting is not suitable for TKGs. For example, given a test query $(s, r, ?, t)$ with the answer o, assume that there are two other quadruples (s, r, o', t') and (s, r, o'', t), where $t' < t$. The static filtered setting ignores time information and removes both o' and o'' from the candidates. However, 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. A more appropriate time-wise filtered setting is only to remove o'' from the candidates. Specifically, for each test quadruple (s, r, o, t), we create two queries: $(s, r, ?, t)$ and $(?, r, o, t)$. For the first query, the model ranks all entities in $o \cup C$ with their scores from Eq.(4), where $C = \{o' : o' \in \mathcal{E}, (s, r, o', t) \notin \mathcal{G}\}$. We follow a similar approach for the second query.

B Detailed Experimental Settings

We choose pre-trained BERT-base model (Devlin et al., 2019) with L=12 layers, K=12 self-attention heads and d=768 hidden dimension of embeddings as the initialization. We select the optimal hyper-parameters by grid searching according to MRR on the validation set, and set the following hyper-parameters in fine-tuning with Adam: batch size: 128, learning rate: 2e-5 and dropout rate: 0.1. We use the RFF strategy to construct the event evelution tree. In EET, the length of time window m and the number of concurrent facts n are set to 10 and 2 for ICEWS14, 14 and 2 for ICEWS05-15, 5 and 2 for GDELT dataset. The length of input text l is set to 107, 147 and 57 for ICEWS and GDELT respectively, where $l = (m*2+1)*(n*2+1)+2$.

Specifically, our SToKE consists of a embedding layer, a mask transformer encoder and a MLP. The parameters of embedding layer consists of token embedding $(|\mathcal{E}| + 4, d)$, segment embedding $(2, d)$, temporal position and displacement embedding $(l + |\mathcal{T}|, d)$ and three matrices $\boldsymbol{W}_{\Delta t}$ $3d^2$. The mask transformer encoder contains 12 mask self attention layers, where the parameters of each layer consist of three matrices $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v$, a concatenated linear layer, where the parameter quantity of

GPU Resource	NVIDIA RTX A6000	
Hyperparameter	Search Bound	Best Setup
length of time window m	choice{5, 10, 12, 14}	10(ICEWS14), 14(ICEWS05-15), 5(GDELT)
number of concurrent facts n	choice{1, 2, 4}	2
length of input text l	choice{57, 107, 147}	107(ICEWS14), 147(ICEWS05-15), 57(GDELT)
number of epochs	choice{30, 50}	30 (ICEWS05-15, GDELT), 50 (ICEWS14)
batch size	choice{32, 64, 128}	128
learning rate	choice{2e-5, 5e-4}	2e-5
droupout rate	choice{0.1, 0.2}	0.1
number of model parameters	–	98.6M(ICEWS14), 106.6M(ICEWS05-15), 88.4M(GDELT)
gpu hours	–	12h(ICEWS14), 40h(ICEWS05-15), 60h(GDELT)

Table 5: Additional implementation details of SToKE.

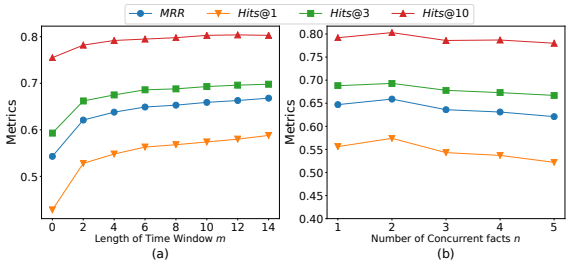

Figure 3: Parameter sensitivity on our SToKE.

each part is d^2, and the FFN is $8d^2$. The number of MLP parameters is $d^2 + d(|\mathcal{E}| + 4)$. Ignoring Layernorm and Bias parameters, the amount of parameters is $148d^2 + d(2|\mathcal{E}| + |\mathcal{T}| + l + 10)$. Finally, we implement our model in PyTorch with NVIDIA RTX A6000. As shown in Table 5, we report the hyperparameter search bounds and best configurations along with the gpu resource.

C Sensitivity Analysis

We report the performance changes on the ICEWS14 dataset by varying the hyper-parameters, including the length of time window m and the number of concurrent facts n. Figure 3(a) shows the performance with various time window lengths. It can be observed that our model performs better with the longer time window m used. However, MRR is relatively stable at around 10, and considering the computation cost, we set m to 10 on this dataset. As shown in Figure 3(b), the model performance rises and falls as the number of neighboring facts within the same timestamp increases. The probable reason is that too many concurrent facts bring knowledge noise, which affects the model's judgment of the missing fact.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☒ A2. Did you discuss any potential risks of your work?
There are no potential risks in our work. We use datasets from open sources, and do not create new datasets or use data from a particular source.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and 1 Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

4.1 Dataset and Metrics, 4.2 Baselines, Appendix B and Appendix C

☑ B1. Did you cite the creators of artifacts you used?
4.1 Dataset and Metrics, 4.2 Baselines, Appendix B and Appendix C

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We use datasets from open sources, and do not create new datasets or use data from a particular source.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use datasets from open sources, and do not create new datasets or use data from a particular source.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We use public datasets and do not use any related content, so it is not necessary to take this step in our work.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
5.1 Dataset and Metrics

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
5.1 Dataset and Metrics

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

4.3 Main Results, 4.4 Ablation Study and 4.5 Case Study

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix B

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4.3 Main Results

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.