Pre-trained Language Model with Prompts for Temporal Knowledge Graph Completion

Wenjie Xu[1], **Ben Liu**[1], **Miao Peng**[1], **Xu Jia**[1], **Min Peng**[1*]
[1]School of Computer Science, Wuhan University, China
{vingerxu,liuben123,pengmiao,jia_xu,pengm}@whu.edu.cn

Abstract

Temporal Knowledge graph completion (TKGC) is a crucial task that involves reasoning at known timestamps to complete the missing part of facts and has attracted more and more attention in recent years. Most existing methods focus on learning representations based on graph neural networks while inaccurately extracting information from timestamps and insufficiently utilizing the implied information in relations. To address these problems, we propose a novel TKGC model, namely **P**re-trained Language Model with **P**rompts for **T**KGC (PPT). We convert a series of sampled quadruples into pre-trained language model inputs and convert intervals between timestamps into different prompts to make coherent sentences with implicit semantic information. We train our model with a masking strategy to convert TKGC task into a masked token prediction task, which can leverage the semantic information in pre-trained language models. Experiments on three benchmark datasets and extensive analysis demonstrate that our model has great competitiveness compared to other models with four metrics. Our model can effectively incorporate information from temporal knowledge graphs into the language models. The code of PPT is available at https://github.com/JaySaligia/PPT.

1 Introduction

In recent years, temporal knowledge graphs(TKGs) have attracted much attention. TKGs describe each fact in quadruple (*subject, relation, object, timestamp*). Compared to static knowledge graphs, TKGs need to consider the impact of timestamps on events. For example, (*Donald Trump, PresidentOf, America, 2018*) holds while (*Donald Trump, PresidentOf, America, 2022*) does not. There are missing entities or relations in the TKGs, therefore, temporal knowledge graph completion (TKGC) is one of the most important tasks of temporal knowledge graphs. TKGC task can be divided into two categories: interpolation setting and extrapolation setting(Jin et al., 2020). Interpolation setting aims to predict missing facts in the known timestamps while extrapolation setting attempts to infer future facts in the unknown ones. The latter is much more challenging, and in this work, we focus on the extrapolation setting. Some TKGC methods are developed from static knowledge graph completion (KGC). Such as adding time-aware score functions to KGC models(Jiang et al., 2016; Dasgupta et al., 2018), adding time-aware relational encoders to graph neural networks (Jin et al., 2020; He et al., 2021), adding a new time dimension to the tensor decomposition(Lacroix et al., 2020; Shao et al., 2022), etc. In addition to those KGC-based models, reinforcement learning(Sun et al., 2021), time-aware neural network modeling(Zhu et al., 2021), and other methods are also applied to TKGC. However, the methods mentioned above have some drawbacks, as follows: (1) **Insufficient temporal information extraction from timestamps**. Most

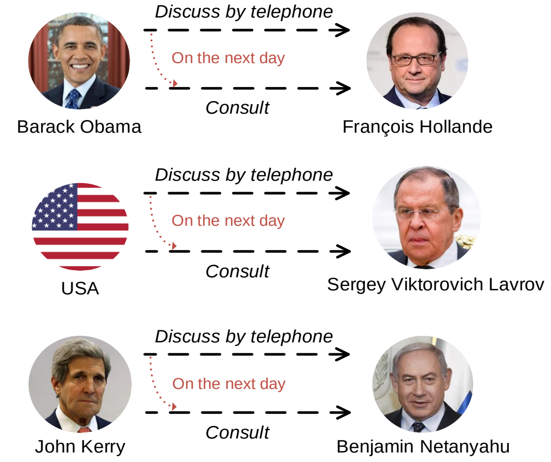

Figure 1: An example of the time-related semantic information between relations in three pairs of entities.

[*]Corresponding author

Findings of the Association for Computational Linguistics: ACL 2023, pages 7790–7803
July 9-14, 2023 ©2023 Association for Computational Linguistics

existing TKGC methods model timestamps explicitly or implicitly. Explicit modeling utilizes low-dimensional vectors to represent timestamps. However, real-life timestamps are infinite, and explicit modeling cannot learn all timestamp representations and predict events with unseen timestamps. Implicit modeling does not represent timestamps directly but takes timestamps to connect multiple knowledge graphs by determining the sequential relationship of these knowledge graphs. This approach often requires modeling the knowledge graph one by one, requires a lot of computation, and timestamps are used only to determine before and after things happen. All the above methods do not give full play to the temporal information of timestamps. (2) **Insufficient information mining of associations in relations in TKGC**. Existing methods often focus on the structural information of the triples or quadruples when modeling KGs without enough consideration of the implied information in relations. This problem is particularly evident in TKGs because some relations contain information with potential temporal hints. As shown in Figure 1, between three different pairs of subject and object entities, after establishing relation *Discuss by telephone*, one day apart, they all establish relation *Consult*. If relation *Discuss by telephone* is established between the same pair of entities, there is a high probability that they will establish relation *Consult* within a short period. Among the entity pairs in ICEWS14, there are 10,887 types of relation pairs, out of which 2,652 exhibit obvious temporal correlations, where one relation in the pair high probably occurred before the other, and they have a stable time interval between them.

To address these problems, we propose a novel temporal knowledge graph completion method based on pre-trained language models (PLMs) and prompts. TKGs contain timestamps, and events occurring at different occurrence times have sequential relationships with each other, which are well-suited as inputs to sequence models. Inspired by the successful application of pre-trained language models in static knowledge graph representation(Yao et al., 2019; Kim et al., 2020; Petroni et al., 2019; Lv et al., 2022), we apply PLMs to temporal knowledge graph completion to get implicit semantic information. However, simply splicing entities and relations in the input of PLMs generates incoherent sentences, resulting in the inability to use PLMs(Lv et al., 2022) fully. Therefore, We sample the quadruples in TKGs and construct prompts for each type of timestamps, which we call **time-prompts**. Then we train PLMs with a masking strategy. In this way, TKGC can be converted into a masked token prediction task.

The contributions of our work can be summarized as follows:

- To the best of our knowledge, we are the first to convert the temporal knowledge graph completion task into the pre-trained language model masked token prediction task.
- We construct prompts for each type of interval between timestamps to better extract semantic information from timestamps.
- We apply our experiments on a series of datasets of ICEWS and achieve satisfactory results compared to graph neural network learning methods.

2 Related Work

2.1 Static KG representation

Static KG representation learning can roughly be divided into distance-based models, semantic matching models, graph neural network models, and PLM-based models.

Distance-based models represent the relation of two entities into a translation vector, such as TransE(Bordes et al., 2013), RotatE(Sun et al., 2019), TransH(Wang et al., 2014). Semantic matching models measure the plausibility of facts using a triangular norm, such as RESCAL(Nickel et al., 2012), Distmult(Yang et al., 2015), ConvE(Dettmers et al., 2018), ComplEx(Trouillon et al., 2016). Graph neural network models use feed-forward or convolutional layers or extend Laplacian matrix to learn the representation of entities and relations, such as GCN(Kipf and Welling, 2017), GAT(Velickovic et al., 2018), R-GCN(Schlichtkrull et al., 2018), SAGE(Hamilton et al., 2017).

PLM-based models have also been considered for static KG representation in recent years due to the ability to capture context information. KG-BERT(Yao et al., 2019) first introduces PLMs into static KG representation. Among PLM-based models, prompt-learning has attracted much attention in recent years and has been shown to be effective on many NLP tasks. LAMA(Petroni et al., 2019) first introduces prompt-based knowledge to PLM.

Other prompt-based models based on LAMA are dedicated to improving the presentation of KGs by automatic prompt generation or by adding soft prompts(Shin et al., 2020; Zhong et al., 2021; Liu et al., 2021). PKGC(Lv et al., 2022) proposes a new prompt-learning method to accommodate the open-world assumption based on KG-BERT.

2.2 Temporal KG representation

Temporal KG representation requires consideration of how the facts are modeled in time series. Some temporal KG representation models are extended from static models. TTransE(Jiang et al., 2016) incorporates temporal information into the scoring function based on TransE(Bordes et al., 2013), and HyTE(Dasgupta et al., 2018) extends TransH(Wang et al., 2014) similarly. TNT-ComplEx(Lacroix et al., 2020) extends ComplEx(Trouillon et al., 2016) inspired by the CP decomposition of order-4 tensor.

These expanded approaches consider timestamps as an additional dimension but lack consideration from a temporal perspective. Some models attempt to combine message-passing and temporal information to solve the problem. RE-NET(Jin et al., 2020) applies R-GCN(Schlichtkrull et al., 2018) for message passing for each snapshot and then uses temporal aggregation across multiple snapshots. HIP Network(He et al., 2021) utilizes structural information passing and temporal information passing to model snapshots. RE-GCN(Li et al., 2021) uniformly encodes the evolutional representations representation of entities and relations corresponding to different timestamps to apply to the extrapolational TKGC task.

Besides, some models use other strategies to model TKG. CyGNet(Zhu et al., 2021) is divided into a copy mode and a generative mode to predict missing entities using neural networks with a time dictionary. TITer(Sun et al., 2021) introduces reinforcement learning in TKG representation learning.

3 Preliminary

Temporal Knowledge Graph $\mathcal{G}$ is a set of networks of entities and relations that contain timestamps. It can be defined as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}\}$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{T}$ is the set of timestamps. $\mathcal{Q} = \{(s, r, o, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$ is the quadruple set, where s and o are the subject entity (head entity) and object entity (tail entity), r is the relation between them at timestamp t. $\mathcal{G}_t = \{(s, r, o) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ is called the TKG snapshot at t, and it can be taken as a static KG filtering the triple set from $\mathcal{G}$ at t.

Temporal Knowledge Graph completion (TKGC) is the task of predicting the evolution of future KGs given KGs of a known period. Given a quadruple $(\mathrm{s}, \mathrm{r}, ?, t_n)$ or $(?, \mathrm{r}, \mathrm{o}, t_n)$, we have a set of known facts from TKG snapshots $G_{(t_i<t_n)}$ to predict the missing object entity or subject entity in the quadruple. The probability of prediction of missing the entity o in quadruple $(\mathrm{s}, \mathrm{r}, ?, t_n)$ can be formalized as follows:

$$p(o|\mathcal{G}_{<t_n}, \mathrm{s}, \mathrm{r}, t_n). \tag{1}$$

4 Methodology

In this paper, we propose PPT, a novel PLM-based model with prompts to solve TKGC task. The framework of our model is illustrated in Figure 2. We sample quadruples and convert them into pre-trained language model inputs. The prediction of **[MASK]** token is the completed result.

4.1 Prompts

We design different prompts for entities (ent-prompts), relations (rel-prompts), and timestamps (time-prompts) to convert quadruples into a form suitable for input to PLMs. We add a soft prompt **[EVE]** before the beginning of each fact tuple due to introducing soft prompts in the input sentences can improve the expressiveness of the sentences(Han et al., 2021).

Ent-prompts. We convert each entity into a special token **[ENT-i]** according to its index. We use a special token instead of the name of an entity because, in the prediction task, we need to predict the whole entity but not a part of it. To maintain the semantic information from entities, we do average pooling of embedding for all words in each entity as the initial embedding of its token.

Rel-prompts. For each relation, we convert it into its original phrase. It is worth noting that to maintain the coherence of sentences, we supplemented each relation with the preposition it was missing. For example, we supplement the relation *Make a visit* to *Make a visit to*.

Time-prompts. We convert the time interval between two timestamps into a phrase that can describe the period. We construct a dictionary called *interval-dictionary*, which maps each period to a prompt. As shown in Figure 3, we convert each

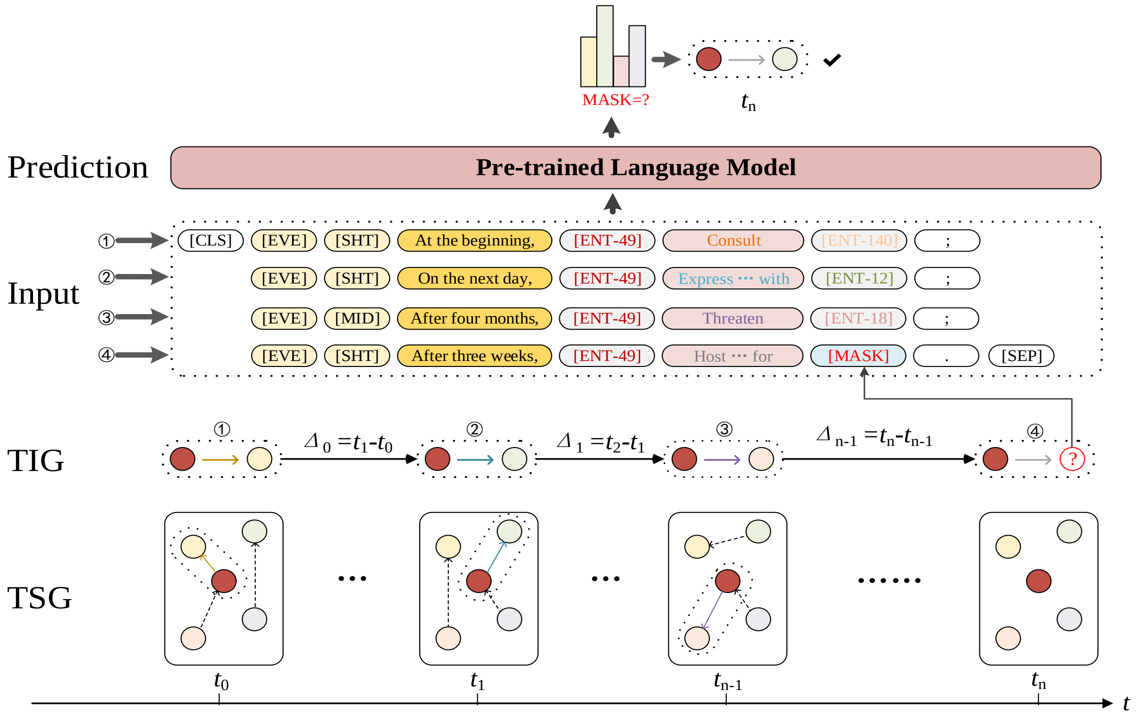
MASK=?
t_n
Prediction
Pre-trained Language Model
Input
① [CLS] [EVE] [SHT] At the beginning, [ENT-49] Consult [ENT-140] ;
② [EVE] [SHT] On the next day, [ENT-49] Express ··· with [ENT-12] ;
③ [EVE] [MID] After four months, [ENT-49] Threaten [ENT-18] ;
④ [EVE] [SHT] After three weeks, [ENT-49] Host ··· for [MASK] . [SEP]
TIG
$\Delta_0 = t_1 - t_0$
$\Delta_1 = t_2 - t_1$
$\Delta_{n-1} = t_n - t_{n-1}$
TSG
t_0
t_1
t_{n-1}
t_n
t

Figure 2: Illusion of PPT for TKGC. Quadruples are sampled and normalized to convert into PLM inputs with prompts. We calculate the time interval of adjacent quadruples in TSG to get TIG. We use the prompts to convert TIG into the input of PLM and then make the prediction for the mask. This way, The TKGC task is converted into a pre-trained language model masked token prediction task.

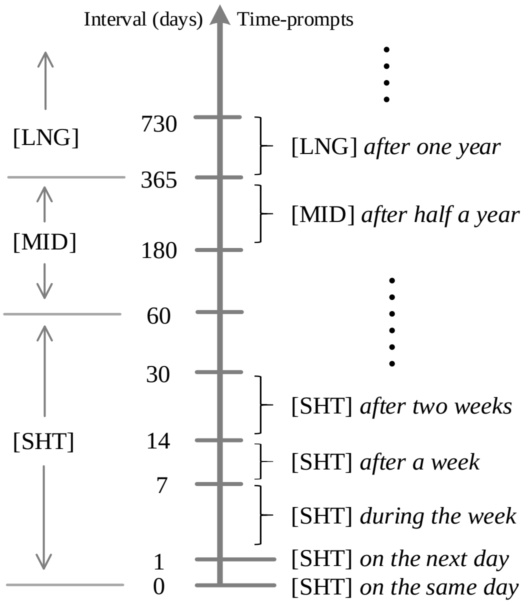

Figure 3: Illusion of *interval-dictionary*. The left side of the vertical axis indicates the interval between two timestamps, and the right side indicates the time-prompts corresponding to the timestamp interval. [SHT] for short intervals ($\Delta_t \leq 60$), [MID] for medium intervals ($60 < \Delta_t \leq 365$), [LNG] for long intervals ($\Delta_t > 365$).

timestamp interval into a prompt. Each prompt contains two parts. The front part is a soft prompt indicating the length of time, such as **[SHT]** for a short time (less than 60 days), **[MID]** for a medium time (from 60 days to 365 days), and **[LNG]** for a long time (above 365 days); the back part is a statement describing the interval. During our analysis, we observed that news reports frequently use distinctive time descriptors to indicate time intervals, which inspired us to develop these prompts.

4.2 Construction for Graphs

Unlike sampling one fact tuple as input to a pre-trained language model in some static knowledge graph models(Yao et al., 2019; Lv et al., 2022), we sample multiple fact tuples simultaneously because we need to model the temporal relationship between facts. We take the head/tail entity for each quadruple in the training dataset and randomly sample each quadruple from the entire training dataset while fixing the head/tail entity. The sampled quadruples are then arranged in chronological order. We demonstrate different sampling strategies in A.1. The sampled list is called *Temporal Specialization Graph* (TSG). TSG can be described as a time-ordered sequence $TSG =$

$[q_0, q_1 \ldots, q_n], q_i = (s_i, r_i, o_i, t_i) \in \mathcal{Q}, t_i \leq t_{i+1}$. We have a total of three types of TSG, which are TSG^s_{obj}, TSG^r_{sub} and TSG^o_{rel}:

$$\begin{aligned}
TSG^s_{obj}(n) =& [q^s_0, q^s_1 \ldots, q^s_n], \\
q^s_i =& (obj, r_i, o_i, t_i) \in \mathcal{Q}, t_i \leq t_{i+1}, \\
TSG^r_{rel}(n) =& [q^r_0, q^r_1 \ldots, q^r_n], \\
q^r_i =& (s_i, rel, o_i, t_i) \in \mathcal{Q}, t_i \leq t_{i+1}, \\
TSG^o_{sub}(n) =& [q^o_0, q^o_1 \ldots, q^o_n], \\
q^o_i =& (s_i, r_i, sub, t_i) \in \mathcal{Q}, t_i \leq t_{i+1},
\end{aligned} \tag{2}$$

where we fix object entity obj to sample TSG^s_{obj}, fix subject entity sub to get sample TSG^r_{sub}, and fix relation rel to sample TSG^o_{rel}. We set a minimum sampling quadruple number K_{min} and a maximum sampling quadruple number K_{max}.

The timestamps in TSGs are independent and cannot reflect the time relationship between events. We convert each TSG to a *Time Interval Graph* (TIG) by calculating the time interval of adjacent quadruples. We take the earliest time in TSG as the initial time τ_0 and calculate the time interval between the timestamp in (s_i, r_i, o_i, t_i) and the timestamp in $(s_{i-1}, r_{i-1}, o_{i-1}, t_{i-1})$ as the new timestamp τ_i:

$$\begin{aligned}
& TIG_{*,*=\{s,r,o\}} = [p^*_0, p^*_1, \cdots, p^*_n], \\
& p^*_i = (q^*_i(s, r, o), \tau_i), \\
& \begin{cases} \tau_o = 0 \\ \tau_i = t_i - t_{i-1} \end{cases},
\end{aligned} \tag{3}$$

where $q^*_i(s, r, o)$ means keeping the fact triple (s_i, r_i, o_i) of q^*_i.

4.3 Training

The algorithm of our training strategy can be summarized in Algorithm 1. We do not train each quadruple separately in the training set for each epoch because we believe that independent quadruples cannot provide temporal information in TKGs. We sample each entity multiple times by fixing it at the object entity position and the subject entity position, thus generating TSGs of entities. Similarly, we fix the relations in the quadruples and, for each relation generate the TSGs of the relations. Then we convert all the TSGs to TIGs. For each quadruple in a TIG, we convert the entities, relation, and time interval into PLM inputs with prompts described in Section.4.1. We use a pre-trained language model with the masking strategy (also known as a masked language model, MLM)(Devlin et al., 2019) to train our model. Masked language models aim to predict masked parts based on their surrounding context. When training, we mask 30% of tokens in an input sequence.

Algorithm 1: Training for PPT

```
Input: TKG $\mathcal{G}$ with training data, maximum number of epochs
       $max\_epochs$, maximum number of sampling TSG of one
       entity or one relation B, minimum sampling sequence length
       $\mathrm{K}_{min}$, maximum sampling sequence length $\mathrm{K}_{max}$.
repeat
    $epoch \leftarrow 1$;
    $\mathcal{S} = \{\}$;
    for $b \leftarrow 1$ to B do
        foreach $ent \in \mathcal{E}$ do
            // sample TSG for entities
            $k = random(\mathrm{K}_{min}, \mathrm{K}_{max})$;
            Sample a $TSG_{ent}$ with length = k;
            Convert $TSG_{ent}$ into $TIG_{ent}$;
            add $TIG_{ent}$ to $\mathcal{S}$;
        end
        foreach $rel \in \mathcal{R}$ do
            // sample TIG for relations
            $k = random(\mathrm{K}_{min}, \mathrm{K}_{max})$;
            Sample a $TSG_{rel}$ with length = k;
            Convert $TSG_{rel}$ into $TIG_{rel}$;
            add $TIG_{rel}$ to $\mathcal{S}$;
        end
    end
    foreach $TIG \in \mathcal{S}$ do
        // convert TIG into input with prompts
        $seq = \mathbf{Prompt}(TIG)$;
        // train in PLM with masking strategy
        $\mathbf{MASK\_TRAIN}(PLM(seq))$;
    end
    $epoch \leftarrow epoch + 1$;
until $epoch = max\_epochs$;
;
```

4.4 Objective optimization discussion

The distribution of all facts in Eq 1 can be considered as the joint distribution of facts on all timestamps:

$$\begin{aligned}
p(\mathcal{G}_{<t_n}) =& p(\mathcal{G}_{t_0}, \mathcal{G}_{t_1}, \cdots, \mathcal{G}_{t_{n-1}}) \\
=& \prod_t \prod_{(s_t, r_t, o_t) \in \mathcal{G}_t} p\left(s_t, r_t, o_t \mid G_{<t_n}\right).
\end{aligned} \tag{4}$$

It is not realistic to focus on all quadruples in the TKG. When predicting the missing subject entities, we fix the object entities because relations in the neighborhood are of most interest to entities. Further, we simulate the original quadruple distribution by sampling, thus Eq 4 can be approximated as:

$$
\begin{aligned}
p(\mathcal{G}_{<t_n}) &\approx \prod_{t} \prod_{(\mathrm{s},r_t,o_t)\in\mathcal{G}_t} p\left(\mathrm{s}, r_t, o_t \mid G_{<t_n}\right) \\
&\approx \prod_{k=1}^{K} p\left(\mathrm{s}, r_k, o_k \mid G_{<t_n}\right) \\
&\approx \prod_{k=1}^{K} p\left(TSG_{\mathrm{s}}^{s}[k] \mid G_{<t_n}\right) \\
&\approx \prod_{k=1}^{K} p\left(TIG_{\mathrm{s}}^{s}[k] \mid G_{<t_n}\right),
\end{aligned} \tag{5}
$$

where K is the number of sampling.

We calculate the generation probability of the quadruples by the pre-trained language model's ability to predict unknown words. We use seq_k to present the converted inputs with prompts of $TIG_{\mathrm{s}}^{s}[k]$:

$$
seq = \mathbf{Prompt}(TIG_{\mathrm{s}}^{s}[k]). \tag{6}
$$

For example, as illustrated in Figure 2, here are two quadruples in TSG:$(49, 62, 12, 2)$ in timestamp t_1 and $(49, 38, 18, 130)$ in timestamp t_{n-1}, the time interval between them is 128 days, $\Delta_1 = t_{n-1} - t_1$. Then the quadruple $(49, 38, 18, 128)$ in TIG can be converted into an input sentence with prompts: **[EVE] [MID]** *After four months*, **[ENT-49]** *Threaten* **[ENT-18]**.

The formalization of prediction can be defined as follows:

$$
\begin{aligned}
&\prod_{k=1}^{K} p\left(TIG_{\mathrm{s}}^{s}[k] \mid G_{<t_n}\right) \\
&= \prod_{k=1}^{K} p(PLM(seq_k)),
\end{aligned} \tag{7}
$$

where $PLM(\cdot)$ means inputting a sequence into the pre-trained language model.

Combining Eq 1 and Eq 7, we convert the TKGC task into an MLM prediction task:

$$
\begin{aligned}
&p(o|\mathcal{G}_{<t_n}, \mathrm{s}, \mathrm{r}, t_n) \\
&\approx \prod_{k=1}^{K} p(PLM(seq_k)) \\
&\cdot p(PLM(\mathbf{Prompt}(\mathrm{s}, \mathrm{r}, t_n))),
\end{aligned} \tag{8}
$$

where $\mathbf{Prompt}(\cdot)$ means converting entities, relations, and timestamps into input sequences for PLM.

By Eq 8, the original knowledge-completion task can be equated to the pre-trained language model masked token prediction task.

5 Experiments

5.1 Experimental Setup

Datasets. Intergrated Crisis Early Warning System (ICEWS)(Boschee et al., 2015) is a repository that contains coded interactions between socio-political actors with timestamps. We utilize three TKG datasets based on ICEWS named ICEWS05-15((García-Durán et al., 2018); from 2005 to 2015), ICEWS14((García-Durán et al., 2018); from 1/1/2014 to 12/31/2014) and ICEWS18((Boschee et al., 2015); from 1/1/2018 to 10/31/2018) to perform evaluation. Statistics of these datasets are listed in Table 1.

Evaluation Protocals. Following prior work(Li et al., 2021), we split each dataset into a training set, validation set, and testing set in chronological order following extrapolation setting. Thus, we guarantee that *timestamps of train* < *timestamps of valid* < *timestamps of test*. Some methods(Jin et al., 2020; Zhu et al., 2021; Wu et al., 2020) apply *filter schema* to evaluate the results by removing all the valid facts that appear in the training, validation, or test sets from the ranking list. Since TKGs are evolving in time, the same event can occur at different times(Li et al., 2021). Therefore, we apply *raw schema* to evaluate our experiments by removing nothing. We report the result of Mean Reciprocal Ranks(MRR) and Hits@1/3/10 (the proportion of correct test cases that are ranked within the top 1/3/10) of our approach and baselines following *raw schema*.

Baselines. We compare our model with two categories of models: static KGC models and TKGC models. We select DistMult(Yang et al., 2015), ComplEx(Trouillon et al., 2016), R-GCN(Schlichtkrull et al., 2018), ConvE(Dettmers et al., 2018), ConvTransE(Shang et al., 2019), RotatE(Sun et al., 2019) as static models. We select HyTE(Dasgupta et al., 2018), TTransE(Jiang et al., 2016), TA-DistMult(García-Durán et al., 2018), RGCRN(Seo et al., 2018), CyGNet(Zhu et al., 2021), RE-NET(Jin et al., 2020), RE-GCN(Li et al., 2021) as baselines of TKGC.

Hyperparameters. We use bert-base-cased[1] as our pre-trained model. Bert-base-cased has been pre-trained on a large corpus of English data in a self-supervised fashion. Bert-base-cased has a parameter size of 110M with 12 layers and 16 attention heads, and its hidden embedding size is

[1] https://huggingface.co/bert-base-cased

Dataset	$\mathcal{E}$	$\mathcal{R}$	#Granularity	#Train	#Valid	#Test
ICEWS05-15	10094	251	24 (hours)	368868	46302	46159
ICEWS14	6869	230	24 (hours)	74845	8514	7371
ICEWS18	23033	256	24 (hours)	373018	45995	49545

Table 1: Statistics of the datasets we use.

dataset	seq_len	min_sample	max_sample
ICEWS05-15	256	2	16
ICEWS14	256	2	12
ICEWS18	256	2	16

Table 2: Parameters for datasets.

768. Without loss of generality, we also list other pre-trained models in A.3. The input sequence length, min sampling number, and max sampling number of each dataset are listed in Table 2. When training, we mask 30% tokens randomly, and we choose AdamW as our optimizer. The learning rate is set as 5e-5. We make a detailed analysis of the parameters in A.2.

5.2 Results

We report the results of PPT and baselines in Table 3.

It can be observed that PPT outperforms all static models much better. Compared with ConvTransE, which has the best results among static models, we achieve 28.3%, 21.97%, and 14.69% improvement with MRR metric in the three datasets, respectively. We believe temporal information matters in TKGC tasks, while static models do not utilize temporal information.

As can be seen that PPT performs better than HyTE, TTransE, and TA-DistMult. These models are under the interpolation setting. For instance, we achieve 41.22%, 46.53%, and 62.18% improvements with MRR metric in the three datasets compared to TA-DistMult. We believe that HyTE and TA-DistMult only focus on independent graphs and do not establish the temporal correlation between graphs. TTransE embeds timestamps into the scoring function while not taking full advantage of them.

With MRR, Hits@1, and Hits@3 metrics on ICEWS05-15 and ICEWS14, PPT achieves the best results compared to other TKGC models. For instance, PPT improves 6.5% over the second-best result with Hit@1 metric. On ICEWS18, PPT has a slight gap with the best model RE-GCN. We believe this is because ICEWS18 has more entities than other datasets. GNN-based models using the message-passing mechanism have better learning ability for such graphs with many nodes. Furthermore, RE-GCN adds additional edges to assist learning for the static parts of the graph.

Besides the masking strategy for our model, we also attempt other forms of application for pre-trained language models, which are illustrated in A.3.

5.3 Ablation study

To investigate the contribution of time-prompts in our model, we conduct ablation studies for our model by testing all datasets under the same parameter settings of different variants. The experiment results are shown in Table 4.

PPT w/o prompts denotes PPT without time-prompts. In this variant, we set all timestamps as 0. To ensure that the sequence length does not affect the experiments, we replaced all the time-prompts with *on the same day*. *PPT w/o prompts* gets worse results than raw PPT with all metrics on three datasets except with Hits@10 on ICEWS14. ICEWS14 has a smaller number of entities and data size than the other two datasets, so it is possible to achieve better results in some metrics after removing the timestamps.

PPT rand prompts denotes PPT with random timestamps set. We replace raw timestamps in quadruples with other timestamps randomly. Random timestamps should not affect the results if our model does not learn the timestamp information correctly. As shown in Table 4, the raw model shows better results than this variant on all metrics.

These experiments demonstrate that applying time-prompts in our model can benefit the learning of temporal information between events.

	ICEWS05-15				ICEWS14				ICEWS18			
Method	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
DistMult	19.91	5.63	27.22	47.33	20.32	6.13	27.59	46.61	13.86	5.61	15.22	31.26
ComplEx	20.26	6.66	26.43	47.31	22.61	9.88	28.93	47.57	15.45	8.04	17.19	30.73
R-GCN	27.13	18.83	30.41	43.16	28.03	19.42	31.95	44.83	15.05	8.13	16.49	29.00
ConvE	31.40	21.56	35.70	50.96	30.30	21.30	34.42	47.89	22.81	13.63	25.83	41.43
ConvTransE	30.28	20.79	33.80	49.95	31.50	22.46	34.98	50.03	23.22	14.26	26.13	41.34
RotatE	19.01	10.42	21.35	36.92	25.71	16.41	29.01	45.16	14.53	6.47	15.78	31.86
HyTE	16.05	6.53	20.20	34.72	16.78	2.13	24.84	43.94	7.41	3.10	7.33	16.01
TTransE	16.53	5.51	20.77	39.26	12.86	3.14	15.72	33.65	8.44	1.85	8.95	22.38
TA-DistMult	27.51	17.57	31.46	47.32	26.22	16.83	29.72	45.23	16.42	8.60	18.13	32.51
RGCRN	35.93	26.23	40.02	54.63	33.31	24.08	36.55	51.54	23.46	14.24	26.62	41.96
CyGNet	35.46	25.44	40.20	54.47	35.45	26.05	39.91	53.20	26.46	16.62	30.57	45.58
RE-NET	36.86	26.24	41.85	57.60	35.77	25.99	40.10	54.87	26.17	16.43	29.89	44.37
RE-GCN	38.27	27.43	43.06	**59.93**	37.78	27.17	42.50	**58.84**	**27.51**	**17.82**	**31.17**	**46.55**
PPT	**38.85**	**28.57**	**43.35**	58.63	**38.42**	**28.94**	**42.5**	57.01	26.63	16.94	30.64	45.43

Table 3: Results on three datasets. The best results are boldfaced, and the second best ones are underlined. The results of baselines are from RE-GCN(Li et al., 2021).

Method	ICEWS05-15				ICEWS14				ICEWS18			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
PPT	**38.85**	**28.57**	**43.35**	**58.63**	**38.42**	**28.94**	**42.5**	57.01	**26.63**	**16.94**	**30.64**	**45.43**
PPT w/o prompts	38.44	28.09	43.09	58.46	38.24	28.52	42.4	**57.31**	25.44	15.68	29.26	44.88
PPT rand prompts	37.43	27.05	42.16	57.49	36.84	26.89	41.41	55.73	24.22	14.31	28.09	44.32

Table 4: Ablation experiments results of PPT. The best results are boldfaced and the second best ones are underlined.

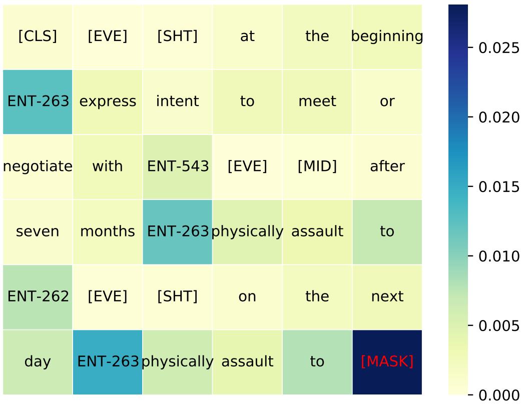

Figure 4: Illustrations of attention patterns of PPT. The quadruple that needs to be completed is $(263, 104, ?, 7536)$, we sample 2 quadruples with earlier timestamps than the test example and fixed object entities. Transparencies of colors reflect the attention scores of other tokens to **[MASK]**.

5.4 Analysis

5.4.1 Attention analysis

To visually show that our model can learn from temporal knowledge graphs, as shown in Figure 4, we visualize attention patterns of PPT. We need to complete the missing tail entity in a test quadruple $(263, 104, ?, 7536)$. As mentioned, we sample data from earlier than timestamp 7536 to form the input sequence and obtain the attention weights from the pre-trained model. In this example, the ground truth is **[ENT-262]**. We observe that in our model, the prediction of **[MASK]** is made by considering all the previous sampling samples together. PPT notes that the same relation *physical assault to* occurred a day earlier and captures the temporal information from token **the**, **next**, and **day**. Therefore, PPT can make correct predictions based on historical events and chronological relationships.

5.4.2 Time-sensitive relation analysis

Using ICEWS05-15 as an example, we analyze the time-sensitive relations present in the dataset. For different relations between the same pairs of entities, there is a clear order of occurrence among some of them. For example, the relation *Obstruct passage, block* is always followed by ones related to assistance, such as *Appeal for aid*, *Appeal for humanitarian aid*, and *Provide humanitarian aid*. Similarly, the relation *Acknowledge or claim responsibility* is always followed by those related to negotiation, such as *Express intent to cooperate militarily*, *Meet at a 'third' location*, and *Demand material cooperation*. We provide more examples in A.5.

To verify the superiority of PPT in handling time-sensitive relations, a new test dataset named *ICEWS05-filter* is constructed from ICEWS05-15. Specifically, we select relations that have a clear chronological order within a predefined time window, resulting in a total of 139 relations. Only the quadruples containing these selected relations are retained to construct the new dataset. As demonstrated in Table 5, PPT achieves better performance when evaluated on the constructed test dataset, indicating its advantage in handling time-sensitive relations.

Dataset	MRR	Hits@1	Hits@3	Hits@10
ICEWS05-15	38.85	28.57	43.35	58.63
ICEWS05-filter	39.4	29.02	43.91	59.31

Table 5: Results of PPT on ICEWS05-15 and ICEWS05-filter.

6 Conclusions

This paper proposes a novel temporal knowledge graph completion model named pre-trained language model with prompts for TKGC (PPT). We use prompts to convert entities, relations, and timestamps into pre-trained model inputs and turn TKGC problem into a masked token prediction problem. This way, we can extract temporal information from timestamps accurately and sufficiently utilize implied information in relations. Our proposed method achieves promising results compared to other temporal graph representation learning methods on three benchmark TKG datasets. For future work, we plan to improve the sampling method in temporal knowledge graphs to get more time-specific inputs. We are also interested in combining GNNs and pre-trained language models in temporal knowledge graph representation learning.

Limitations

This paper proposes a pre-trained language model with prompts for temporal knowledge graph completion. However, there are some limitations in our method: 1) Our prompts in the temporal knowledge graphs, especially the time-prompts, are built manually. It needs to be reconstructed manually for different knowledge graphs. We are exploring a way to build prompts in temporal knowledge graphs automatically. 2) Our model uses a random sampling method, which suffers from the problem of few high-quality training samples and high sample noise. For future work, a more effective way to sample is worth exploring.

Acknowledgements

We would like to thank all the anonymous reviewers for their valuable and insightful comments. This work was supported by National Key Research and Development Program of China (No.2021ZD0113304), General Program of Natural Science Foundation of China (NSFC) (Grant No.62072346), Key R&D Project of Hubei Province (Grant NO.2021BBA099, NO.2021BBA029) and Application Foundation Frontier Project of Wuhan (Grant NO.20200106010121 68). Our work was founded by Joint&Laboratory on Credit Technology.

Ethics Statement

All steps and data described in our paper follow the ACL Ethics Policy[2].

References

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

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

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

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

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

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

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259.

[2] https://www.aclweb.org/portal/content/acl-code-ethics

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

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

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

Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-task learning for knowledge graph completion with pre-trained language models. In *COLING*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

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

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*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *CoRR*.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? A reliable evaluation and a reasonable approach. In *ACL(Findings)*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *WWW*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*.

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*.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *ICONIP*.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI*.

Pengpeng Shao, Dawei Zhang, Guohua Yang, Jianhua Tao, Feihu Che, and Tong Liu. 2022. Tucker decomposition-based temporal knowledge graph completion. *Knowl. Based Syst.*

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *EMNLP*.

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

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

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

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

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.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: learning vs. learning to recall. In *NAACL-HLT*.

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 *AAAI*.

A Appendix

A.1 Sampling Analysis

We design two sampling strategies, one is the uniform sampling strategy, and the other is the frequency-based sampling strategy. The uniform sampling strategy assigns equal sampling weights to each entity. The frequency-based sampling strategy assigns different weights to each entity based on the different frequencies of each entity appearing in the dataset, where entities with higher occurrences have a higher probability of being sampled. As shown in Table 6, the frequency-based sampling strategy has better results on ICEWS14. We believe this is because if an entity appears frequently, it is more likely to have relations with other entities and should get more attention.

Strategy	MRR	Hits@1	Hits@3	Hits@10
uniform	34.87	25.37	38.77	53.33
frequency-based	38.42	28.94	42.5	57.01

Table 6: Results of different sampling strategies of PPT on ICEWS14.

A.2 Hyperparameter Analysis

To test the effect of different sequence lengths and the maximum number of samples on the effect of the model, we analyze these hyperparameters on ICEWS14. Due to GPU performance limitations, we do not perform experiments on longer sequences.

As shown in Table 7, we get the best results with setting $seq\_len = 256, max\_sample = 12$. We believe that the effect of sequence length is small while the number of samples matters. A larger number of samples can provide more semantic contextual information for the prediction but overly lengthy sampling can cause a decline in effectiveness by not focusing on the most effective information in learning.

seq_len	max_sample	MRR	Hits@1	Hits@3	Hits@10
128	2	35.33	25.71	39.56	53.83
128	4	37.21	27.59	41.08	56.3
128	8	37.67	28.16	41.73	56.22
256	8	37.67	27.78	42.31	56.72
256	12	**38.42**	**28.94**	**42.5**	**57.01**
256	16	37.72	27.74	42.1	56.91

Table 7: Results of different hyperparameters of PPT on ICEWS14. The best results are boldfaced and the second best ones are underlined.

A.3 Variants

In addition to the model we propose in the paper, we also try some variants, all experiments are done with $seq\_len = 256, max\_sample = 12$ on ICEWS14. As demonstrated in Table 8, PPT_CLS does not use the mask training strategy but takes **[CLS]** to do classification with a fully connected layer as the decoder; PPT_LSTM uses a bi-directional LSTM to encode all tokens, maxpool the out embeddings, and use a fully-connected layer as a decoder. These models do not get satisfactory results compared to our raw model.

PPT_CLS only uses sequence embedding to predict the result is not enough because the sequence embedding is suitable for classification task which needs to be focused on the whole input sequence. However, in our task, we need to consider the impact of each token. For PPT_LSTM, we believe that the representation learned by the pre-trained language model is high-level semantic knowledge, especially when additional tokens (entities and relations) are added. Simple neural network models are unable to capture this high-level semantic knowledge and instead cause a decrease in effectiveness.

Variants	MRR	Hits@1	Hits@3	Hits@10
PPT_CLS	32.81	23.62	36.81	51.12
PPT_LSTM	32.6	23.61	36.54	50.06
PPT	38.42	28.94	42.5	57.01

Table 8: Variants of PPT.

A.4 Different PLMs

Besides *bert-base-cased*, we also attempt other pre-trained language models: bert-base-uncased[3] and bert-large-cased[4]. As shown in Table 9. All experiments are done with setting $seq\_len = 128, min\_sample = 2, max\_sample = 8$ on ICEWS14. We find that the experimental results with different PLMs are similar, indicating that our approach does not rely on a specific pre-trained language model and has the ability to generalize.

PLMs	MRR	Hits@1	Hits@3	Hits@10
bert-base-cased	37.67	28.16	41.73	56.22
bert-base-uncased	37.75	28.06	41.74	56.84
bert-large-cased	37.36	27.39	41.39	57.59

Table 9: Experiments on different PLMs.

[3] https://huggingface.co/bert-base-uncased

[4] https://huggingface.co/bert-large-uncased

Pre-relation	Post-relation
Demonstrate for policy change	fight with small arms and light weapons
Demonstrate for policy change	Make optimistic comment
Demonstrate for policy change	Conduct suicide, car, or other non-military bombing
Obstruct passage, block	Appeal for aid
Obstruct passage, block	Appeal for humanitarian aid
Obstruct passage, block	Provide humanitarian aid
Acknowledge or claim responsibility	Express intent to cooperate militarily
Acknowledge or claim responsibility	Meet at a 'third' location
Acknowledge or claim responsibility	Demand material cooperation
Receive inspectors	Expel or deport individuals
Receive inspectors	Express intent to provide material aid
Receive inspectors	Return, release person(s)
Demand release of persons or property	Use unconventional violence
Demand release of persons or property	Demonstrate or rally
Demand release of persons or property	Appeal for military aid
Reject judicial cooperation	Appeal to others to settle dispute
Reject judicial cooperation	Accuse of espionage, treason
Reject judicial cooperation	Retreat or surrender militarily

Table 10: Examples of pre-relations and post-relations

A.5 Pre-relations and post-relations

For one pair of entities, if relation *rel-A* always occurs before relation *rel-B*, *rel-A* is called a pre-relation and *rel-B* is called a post-relation. Table 10 shows some of these relations.

ACL 2023 Responsible NLP Checklist

A For every submission:

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

☒ A2. Did you discuss any potential risks of your work?
Our experiments are reproducible.

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

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

B ☑ Did you use or create scientific artifacts?

See 5.1 Experimental Setup

☑ B1. Did you cite the creators of artifacts you used?
See 5.1 Experimental Setup

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
See 5.1 Experimental Setup

☒ 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)?
The tools we use are consensuses in the field like many other papers do.

☒ 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?
The data we use are consensuses in the field like many other papers do.

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

☑ 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.
See Table 1

C ☑ Did you run computational experiments?

See 5.1 Experimental Setup

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
See 5.1 Experimental Setup

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

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
See 5.1 Experimental Setup and Appendix A.1

☑ 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?
See 5.2 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.)?
See 5.1 Experimental Setup

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

☒

□ 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.