# Query Structure Modeling for Inductive Logical Reasoning Over Knowledge Graphs

**Siyuan Wang**[1], **Zhongyu Wei**[1,2]*, **Meng Han**[3],
**Zhihao Fan**[1], **Haijun Shan**[4], **Qi Zhang**[5], **Xuanjing Huang**[5]
[1]School of Data Science, Fudan University, China
[2]Research Institute of Intelligent and Complex Systems, Fudan University, China
[3]Huawei Poisson Lab, China [4]CEC GienTech Technology Co., Ltd, China
[5]School of Computer Science, Fudan University, China
{wangsy18,zywei,fanzh18,qz,xjhuang}@fudan.edu.cn
hanmeng12@huawei.com; haijun.shan@gientech.com

## Abstract

Logical reasoning over incomplete knowledge graphs to answer complex logical queries is a challenging task. With the emergence of new entities and relations in constantly evolving KGs, inductive logical reasoning over KGs has become a crucial problem. However, previous PLMs-based methods struggle to model the logical structures of complex queries, which limits their ability to generalize within the same structure. In this paper, we propose a structure-modeled textual encoding framework for inductive logical reasoning over KGs. It encodes linearized query structures and entities using pre-trained language models to find answers. For structure modeling of complex queries, we design stepwise instructions that implicitly prompt PLMs on the execution order of geometric operations in each query. We further separately model different geometric operations (i.e., projection, intersection, and union) on the representation space using a pre-trained encoder with additional attention and maxout layers to enhance structured modeling. We conduct experiments on two inductive logical reasoning datasets and three transductive datasets. The results demonstrate the effectiveness of our method on logical reasoning over KGs in both inductive and transductive settings. [1]

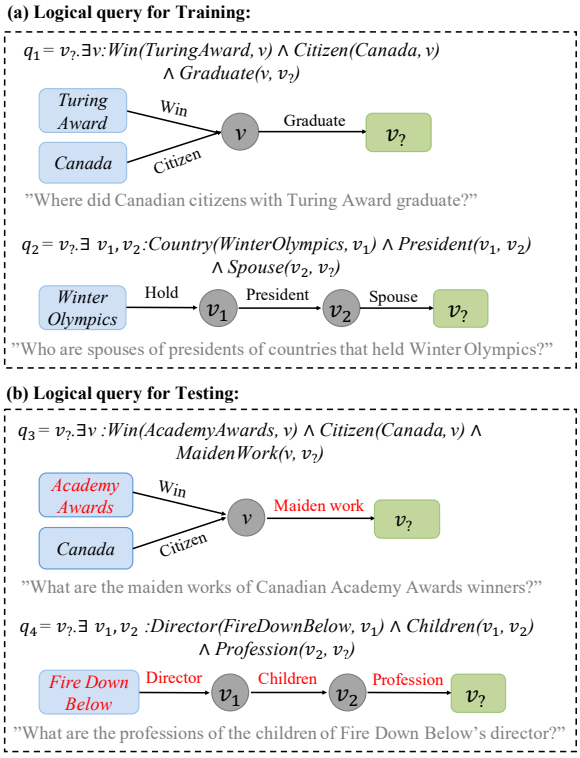


Figure 1: Examples of inductive logical reasoning over KGs: testing queries contain unseen entities and relations (in red) during training. Each query is associated with an intrinsic logical structure and its natural language interpretation.

## 1 Introduction

Logical reasoning over knowledge graphs (KGs) aims to answer complex logical queries given large-scale KGs (Guu et al., 2015; Hamilton et al., 2018). Recent years have witnessed increasing attention on logical reasoning over widely used KGs such as Freebase (Bollacker et al., 2008), Yago (Suchanek et al., 2007), NELL (Carlson et al., 2010) and Wikidata (Vrandečić and Krötzsch, 2014). With missing relations in the KG, it is challenging to deduce correct answers to complex queries by traversing the graph. Previous work primarily focuses on transductive logical reasoning where the training and testing are done on the same KG with the same group of entities. They typically rely on geometric embedding-based methods to map both entities

*Corresponding author
[1]Codes are publicly available at https://github.com/WangsyGit/InductiveLR.



*Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*
*Volume 1: Long Papers*, pages 4706–4718
July 9-14, 2023 

and queries into a joint low-dimensional vector space (Hamilton et al., 2018; Ren et al., 2020; Ren and Leskovec, 2020). The goal is to push the embeddings of answer entities and queries to be close to each other, allowing answers to be predicted through embedding similarity even when the involved relation is absent. In contrast, the inductive setting of logical reasoning has been rarely studied which requires generalizing to unseen entities and relations or even new KGs. As real-world KGs are usually dynamic with emerging unseen entities and relations, it's significant to explore the inductive setting for complex query answering.

Existing research on inductive logical reasoning mainly follows two directions. The first inherits embedding-based methods and incorporates type as additional information to improve inductive capability (Hu et al., 2022), which can not generalize to unseen types of entities and relations. The second direction leverages pre-trained language models (PLMs) to encode textual information of entities/relations for generalization to unseen elements (Wang et al., 2021b; Daza et al., 2021; Wang et al., 2021a). PLMs-based approaches provide more flexible solutions and generate better results. However, they only explore link prediction tasks of one-step reasoning, and simply linearize the triplet or subgraph into text sequence without modeling explicit reasoning structure (Yao et al., 2019; Zha et al., 2022). An example is shown in Figure 1. Two findings stand out. (1) The query $q_1$ and $q_2$ appear to be similar in format (both as a conjunction of three terms) but actually have different logical structures. PLMs-based methods that encode flattened queries can not model this structure information for correct logical reasoning. (2) Although queries $q_1$ and $q_3$ (also $q_2$ and $q_4$) contain different elements, they share the same logical structure. Motivated by these, we argue that structure modeling of different complex queries can further boost the generalization ability of logical reasoners.

In this paper, we propose to model query structure for inductive logical reasoning over KGs. Specifically, we transform the query structure into a sequence using textual names of involved entities, relations, and logical operators. For complex query structures composed of multiple geometric operations over entities and relations, we introduce two measures to enable logical structure modeling during text encoding. First, we design stepwise instructions for different query types to indicate which operation in the query structure should be conducted at each step and feed them as the structural prompt to PLMs. Besides, we extend the pre-trained encoder with an additional attention layer and a maxout layer to respectively model different geometric operations including projection, intersection, and union on the representation space, to implicitly inject structured modeling into PLMs. Our proposed method is a generic inductive framework, which can be plugged into different PLMs for better performance.

We conduct experiments on two datasets for inductive logical reasoning over KGs, FB15k-237-V2 and NELL-V3 (Teru et al., 2020) as well as three transductive datasets, FB15k (Bordes et al., 2013), FB15k-237 (Toutanova and Chen, 2015), and NELL995 (Xiong et al., 2017). The results demonstrate that our method achieves strong inductive performance on unseen entities and relations, even across different KGs, without sacrificing logical reasoning capability and generalizability to new query structures.

## 2 Methodology

In this work, we study the task of complex logical reasoning over KGs. The input is a first-order logic query $q$ which can include any set of existential quantification ($\exists$), conjunction ($\wedge$), and disjunction ($\vee$) operators (such as the query in Figure 1). Our goal is to predict a set of entities $\mathcal{A} = \{a_1, a_2, a_3...\}$ that answer the query $q$ based on an incomplete KG $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ which consists of a set of triplets $(h, r, t)$ but lacks several involved relations. Here $h, t \in \mathcal{E}$ are the head and tail entities and $r \in \mathcal{R}$ is the relation between them. We mainly focus on the inductive setting of KG logical reasoning, where the evaluated queries contain entities/relations that are completely unseen during the training period.

Figure 2 shows the overall architecture of our model. We propose to encode the text sequences of query structures and predict the answer entities based on representation similarity for inductive logical reasoning over KGs. In this section, we first list different types of query structures studied in logical reasoning over KGs(§ 2.1). Then according to various query types, we introduce our structure linearization and structural prompts for textual encoding(§ 2.2). The geometric operation modeling and query answer prediction modules are described in (§ 2.3) and (§ 2.4). Finally, we provide the de-

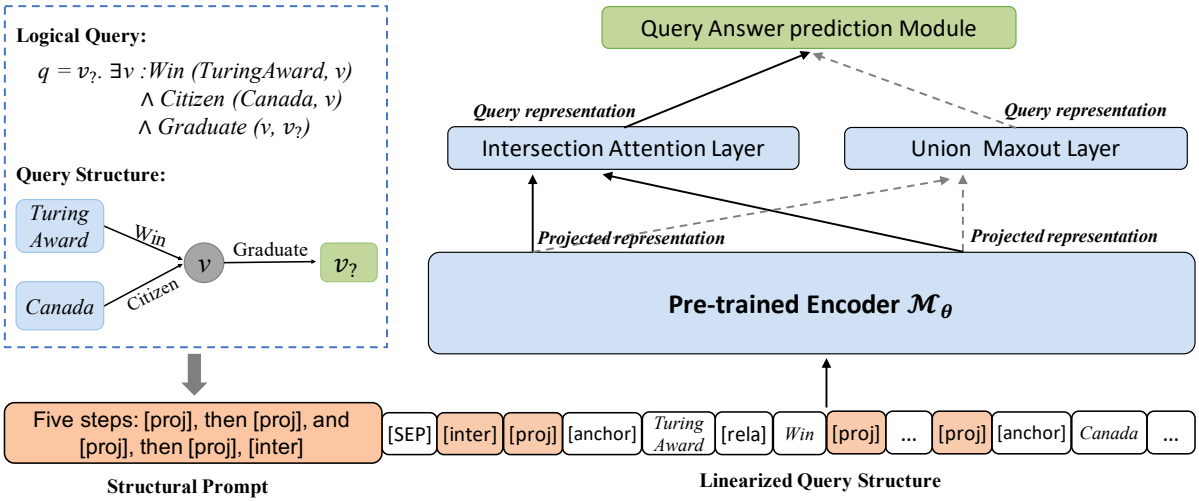


Figure 2: The overall architecture of our method for inductive KG logical reasoning with structure knowledge modeling. [inter] and [proj] are respectively abbreviations of [intersection] and [projection].

tails about training and inference (§ 2.5).

## 2.1 Query Structure Types

Following (Ren et al., 2020), we consider 9 types of complex query structures which are composed of different sets of geometric operations (including projection, intersection and union) over entities and relations. These include six single-operation query structures and three mixed-operation ones. Specifically, three query types only focus on projection, including one-relation projection (1p), two-relation projection (2p), and three-relation projection (3p). Two query types focus on the intersection of two triplets (2i) and three triplets (3i), and another one focuses on the union of two triplets (2u). The three mixed-operation query structures are respectively the combinations of intersection&projection (ip), projection&intersection (pi), and union&projection (up). The different query structures are illustrated as the following formula:

$$
\begin{aligned}
&[1p]\ q = v_? : r_1(e_1, v_?) \\
&[2p]\ q = v_?.\exists v : r_1(e_1, v) \wedge r_2(v, v_?) \\
&[3p]\ q = v_?.\exists v_1, v_2 : r_1(e_1, v_1) \wedge r_2(v_1, v_2) \\
&\qquad\qquad \wedge r_3(v_2, v_?) \\
&[2i]\ q = v_? : r_1(e_1, v_?) \wedge r_2(e_2, v_?) \\
&[3i]\ q = v_? : r_1(e_1, v_?) \wedge r_2(e_2, v_?) \wedge r_3(e_3, v_?) \\
&[pi]\ q = v_?.\exists v : r_1(e_1, v) \wedge r_2(v, v_?) \wedge r_3(e_2, v_?) \\
&[ip]\ q = v_?.\exists v : r_1(e_1, v) \wedge r_2(e_2, v) \wedge r_3(v, v_?) \\
&[2u]\ q = v_? : r_1(e_1, v_?) \vee r_2(e_2, v_?) \\
&[up]\ q = v_?.\exists v : (r_1(e_1, v) \vee r_2(e_2, v)) \wedge r_3(v, v_?)
\end{aligned} \tag{1}
$$

where $e_i$ and $v_i$ are the anchor entities and existentially quantified bound variables entities, and $v_?$ are the target answer entities to the query. As these complex queries contain rich logical structural information, we need to model the structure knowledge during textual encoding for better inductive generalization within the same logical structure.

## 2.2 Query Structure Encoding

In order to use PLMs for better generalization to unseen entities/relations, we first need to linearize the query structures into text sequences. We also design instructions for each query type as a structural prompt to implicitly indicate the order of geometric operations execution to PLMs. We concatenate the linearized query and structural prompt as the input, and encode them to obtain the query representation for matching with the answer entities.

**Query Structure Linearization** Given a query structure $q$, we customize its linearization method according to the query type. For each triplet $r_i(e_i, v)$ in the query, we formulate it as "`[anchor]` $t(e_i)$ `[relation]` $t(r_i)$" and ignores the intermediate variable entities, where $t(e_i)$ and $t(r_i)$ are the textual names of anchor entity $e_i$ and relation $r_i$. Then we add the names of logical operations before the involved subqueries. For example, the query structure of type [2p] can be linearized into sequence "`[projection]` `[anchor]` $t(e_1)$ `[relation]` $t(r_1)$ `[projection]` `[relation]` $t(r_2)$" and the query structure of type [2i] can be mapped into "`[intersection]` `[projection]` `[anchor]`

$t(e_1)$ [relation] $t(r_1)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_2)$".

For query types [ip] and [up] that are composed of intersection/union and projection, the last relation projection is conducted over the intersection/union of previous triplets. Directly flattening the query is unable to keep such structural information. Therefore, we propose to split the intersection/union and repeatedly connect each triplet with the last relation projection which moves the intersection/union operation to the final step. This transformation is equivalent to the original query structure. For example, the following two query structures are equivalent and are both of type [up].

$$\begin{aligned}&(r_1(e_1, v) \vee r_2(e_2, v)) \wedge r_3(v, v_?)\\&(r_1(e_1, v) \wedge r_3(v, v_?)) \vee (r_2(e_2, v) \wedge r_3(v, v_?))\end{aligned} \tag{2}$$

Based on this transformation, we linearize the query structure of type [up] into the text sequence as "[union] [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [relation] $t(r_3)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_2)$ [projection] [relation] $t(r_3)$". The details of structure linearization templates for each query type are listed in Appendix A.

**Structural Prompt** Besides feeding the linearized query structure into PLMs, we also introduce stepwise instructions that indicate the order of geometric operations execution to prompt the pre-trained encoder with implicit structural information of the query. Specifically, each prompt consists of two parts: the number of total execution steps and the operation order in the query, which is formulated as "total steps: operation order". For query types [ip] and [up], we define the total steps and operation order according to the transformed query structure in Eq 2. The detailed structural prompts of each query type are presented in Table 1.

Then the input $t$ can be obtained by concatenating the structural prompt $s$ and the linearized query structure $t(q)$, which is formulated as "[CLS] [qtype] s [SEP] t(q)". We feed it to the pre-trained encoder and obtain the output hidden states $H = (h_1, h_2, ..., h_{|t|})$.

### 2.3 Geometric Operation Modeling

To further enhance the structure modeling during textual encoding, we propose to separately model different geometric operations in logical queries to explore their spatial characteristics. As Transformer-based encoders are widely used to implicitly learn the translation function for simple link prediction and question answering (Yao et al., 2019; Wang et al., 2021a), we directly utilize it for modeling multi-hop relation projection. For each path of relation projection $r_1(e_1, v_1) \wedge ... \wedge r_j(v_j, v_?)$, we extract the hidden states corresponding to sequence "[anchor] $t(e_1)$ [relation] $t(r_1)$ ... [projection] [relation] $t(r_j)$" from $H = (h_1, h_2, ..., h_{|t|})$. We then take the average as the representation of target entity $v_?$, which also can be viewed as the representation $h_q$ of the query that only involves relation projection operation.

For the intersection and union of multiple subqueries, we adopt an attention layer (Bahdanau et al., 2014) and a maxout layer (Goodfellow et al., 2013) on top of the pre-trained encoder to respectively model these two operations in the geometric representation space. Specifically, we feed the representations of target entities in all subqueries to these two additional layers to achieve the intersection and union operations. The output can be taken as the query representation $h_q$ that contains intersection or union.

As presented in (Ren and Leskovec, 2020), the complete set of first-order logic operations encompasses existential quantification ($\exists$), conjunction ($\wedge$), disjunction ($\vee$) and negation($\neg$). Our approach covers the first three operations by modeling relation projection, intersection, and union respectively. The negation operation is not individually modeled as pre-trained encoders are capable of capturing se-

| Query Type | Structural Prompt |
|---|---|
| [1p] | One step: [proj] |
| [2p] | Two steps: [proj], then [proj] |
| [3p] | Three steps: [proj], then [proj], then [proj] |
| [2i] | Three steps: [proj], and [proj], [inter] |
| [3i] | Four steps: [proj], and [proj], and [proj], [inter] |
| [pi] | Four steps: [proj], then [proj], and [proj], [inter] |
| [ip] | Five steps: [proj], then [proj], and [proj], then [proj], [inter] |
| [2u] | Three steps: [proj], and [proj], [union] |
| [up] | Five steps: [proj], then [proj], and [proj], then [proj], [union] |

Table 1: Structural prompts of sifferent query types. "[proj]" and "[inter]" are the abbreviations of "[projection]" and "[intersection]".

mantic exclusion for negation. We can add negative terms such as "not" before the corresponding relations within the input and feed it into pre-trained encoders to naturally address this task.

### 2.4 Query Answer Prediction

To answer the complex query $q$, we adopt the Siamese dual encoder (Gillick et al., 2018) to respectively encode the query $q$ and the candidate entity $c_i$ to match the answer. We formulate the entity input as the sequence "`[CLS] [target]` $t(c_i)$" and feed it into the pre-trained encoder to obtain the candidate entity representation $h_{c_i}$ by taking the average of the hidden states. Then we compute the similarity $d_i$ between the query representation $h_q$ and entity representation $h_{c_i}$, and encourage the query representation to be similar to the positive answer entities while dissimilar to negative entities. The entities whose representations are similar enough to the query will be predicted as the answers. We can pre-compute the representations of all candidate entities and utilize them to predict answers for different queries more efficiently.

The above matching scheme can handle the inductive setting when the candidate entity set is not closed and new entities may arise. To improve the answer prediction accuracy in the transductive setting where the candidate entity set is closed, we also employ a classification layer on top of the query representation $h_q$. Given the fixed candidate entity set $(c_1, c_2, ..., c_N)$, the classification layer with a softmax function will output an $N$-dimensional plausibility distribution $(s_{c_1}, s_{c_2}, ..., s_{c_N})$ for each candidate entity $c_i$ to be the answer.

### 2.5 Training & Inference

We simultaneously optimize a matching objective and a classification objective to train our inductive model for answering complex logical queries. For the former, we adopt contrastive learning (Chen et al., 2020) which needs to separate the positive and negative answers for the query. We take the given ground-truth answer $c_+$ as positive and implement in-batch negative sampling to collect the negatives. We measure the similarity between the query and entities using dot product, and follow (He et al., 2020) to utilize InfoNCE as the contrastive loss. The loss function is formulated as Eq. 3 where $\tau$ is the temperature hyper-parameter and $N$ is the total number of candidate entities including positives and negatives. For the classification objective, we take all entities in each KG as candidate entities and calculate the cross-entropy loss as Eq. 4.

$$\mathcal{L}_M = -\log \frac{\exp(h_q \cdot h_{c_+}/\tau)}{\sum_{i=1}^{N}(\exp(h_q \cdot h_{c_i}/\tau)} \quad (3)$$

$$\mathcal{L}_C = -\log \frac{s_{c_+}}{\sum_{i=1}^{N} \exp(s_{c_i})} \quad (4)$$

These two losses are combined in a weighted manner as $\mathcal{L} = \mathcal{L}_M + \lambda \mathcal{L}_C$ and $\lambda$ is the weighted hyper-parameter.

During inference, we perform differently for inductive and transductive logical query answering. For the inductive reasoning, we utilize the matching scheme and rank the representation similarities between the query and all candidate entities for query answer prediction. For the transductive inference, we only adopt the classification scheme and find the most plausible answer according to classification scores.

## 3 Experiments

### 3.1 Experiment Setup

We conduct experiments on complex logical reasoning over KGs in both inductive and transductive setting. For the inductive setting, we adopt two datasets, FB15k-237-V2 and NELL-V3, that have disjoint sets of entities for training and evaluation, as introduced by (Teru et al., 2020). To further challenge our model, we also illustrate the cross-KG inductive generalization performance by respectively taking FB15k and NELL995 for training/inference and inference/training that contain completely different entities and relations. In the transductive setting, we evaluate our model on the generated queries from three datasets: FB15k (Bordes et al., 2013), FB15k-237 (Toutanova and Chen, 2015), and NELL995 (Xiong et al., 2017), as proposed by (Ren et al., 2020). All these datasets cover nine types of query structures. We follow the setting of (Ren et al., 2020) to illustrate the generalization within the same structure to unseen entities and relations, and also the generalization to more complicated unseen structures composed of different structures. Specifically, we train our model on the first five types of query structures (1p, 2p, 3p, 2i, 3i) and evaluate it on all nine query types (1p, 2p, 3p, 2i, 3i, pi, ip, 2u, up), including both seen and unseen query structures during training. The data split statistics of logical queries in these datasets are provided in Appendix B.

| Model | Avg | 1p | 2p | 3p | 2i | 3i | ip | pi | 2u | up |
|---|---|---|---|---|---|---|---|---|---|---|
| FB15k-237-V2 | | | | | | | | | | |
| *Q2B* | 0.043 | 0.005 | 0.055 | 0.017 | 0.007 | 0.007 | 0.078 | 0.129 | 0.027 | 0.061 |
| *TEMP(GQE)* | 0.163 | 0.146 | 0.221 | 0.141 | 0.139 | 0.144 | 0.157 | 0.220 | 0.097 | 0.201 |
| *BiQE* | 0.158 | 0.286 | 0.151 | 0.107 | 0.187 | 0.240 | 0.125 | 0.155 | 0.085 | 0.091 |
| *SILR* | **0.178** | **0.309** | 0.121 | 0.106 | **0.237** | **0.274** | 0.148 | 0.181 | **0.110** | 0.116 |
| NELL-V3 | | | | | | | | | | |
| *Q2B* | 0.017 | 0.002 | 0.022 | 0.005 | 0.003 | 0.002 | 0.026 | 0.048 | 0.018 | 0.028 |
| *TEMP(GQE)* | 0.062 | 0.096 | 0.057 | 0.060 | 0.072 | 0.081 | 0.047 | 0.062 | 0.040 | 0.040 |
| *BiQE* | 0.089 | 0.178 | 0.081 | 0.081 | 0.082 | 0.092 | 0.067 | 0.081 | 0.066 | 0.069 |
| *SILR* | **0.101** | **0.197** | 0.079 | 0.074 | **0.103** | **0.122** | **0.094** | **0.090** | **0.080** | 0.068 |

Table 2: Inductive H@10 results of different structured queries on FB15k-237-V2 and NELL-V3 datasets. The results of *Q2B* and *TEMP(Q2B)* are taken from (Hu et al., 2022). Avg is the average performance of all query types.

| Model | Avg | 1p | 2p | 3p | 2i | 3i | ip | pi | 2u | up |
|---|---|---|---|---|---|---|---|---|---|---|
| FB15k → NELL995 | | | | | | | | | | |
| *BiQE* | 0.042 | 0.077 | 0.076 | 0.054 | 0.025 | 0.023 | 0.036 | 0.035 | 0.016 | 0.039 |
| *SILR* | **0.069** | **0.166** | **0.101** | **0.096** | **0.033** | **0.040** | 0.020 | 0.027 | **0.063** | **0.073** |
| NELL995 → FB15k | | | | | | | | | | |
| *BiQE* | 0.082 | 0.175 | 0.103 | 0.078 | 0.082 | 0.094 | 0.041 | 0.057 | 0.033 | 0.072 |
| *SILR* | **0.098** | **0.218** | **0.128** | **0.100** | 0.081 | 0.089 | 0.029 | **0.059** | **0.115** | 0.065 |

Table 3: Inductive H@10 results on cross-KG generalization. FB15k → NELL995 and NELL995 → FB15k respectively mean that the model is trained on FB15k and tested on NELL995, and vice versa.

### 3.2 Implementation Details

We take bert-large-cased and bert-base-cased (Devlin et al., 2018) as the pre-trained encoder for encoding the query structure in (FB15k-237-V2, NELL-V3) and (FB15k, FB15k-237, and NELL995), respectively. All models are implemented using Huggingface (Wolf et al., 2019), and trained for 30 epochs on 4 NVIDIA Tesla V100 GPUs with 16 GB of memory. The Adam is taken as the optimizer and the learning rate is 1.5e-4. We use a linear learning rate scheduler with $10\%$ warmup proportion. The weight hyper-parameter to balance losses is set to $\lambda = 0.3$ or $\lambda = 0.4$. For automatic evaluation, we use Hits@$K$ ($K = 3, 10$) as the metrics, which calculate the proportion of correct answer entities ranked among the top-$K$.

### 3.3 Inductive Setting

**Unseen Entities Generalization** To illustrate the inductive performance of complex logical reasoning over KGs, we first make a comparison on FB15k-237-V2 and NELL-V3 datasets for generalizing to unseen entities. We compare our **s**tructure-modeled **i**nductive **l**ogical **r**easoning method (*SILR*), with the baseline embedding-based method *Q2B* (Ren et al., 2020) and the best version of the inductive model *TEMP(GQE)* (Hu et al., 2022). *TEMP(GQE)* enriches embedding method *GQE* (Hamilton et al., 2018) with type information which has achieved the state-of-the-art performance. *BiQE* (Kotnis et al., 2021) is also a textual encoding method with positional embedding for logical query but only in the transductive setting. We reimplement it by replacing the original classification-based prediction with the matching scheme for an inductive comparison.

The experimental results are shown in Table 2. We can see that our *SILR* outperforms all other models on both FB15k-237-V2 and NELL-V3 datasets by a considerable margin. This highlights the effectiveness of our method for inductive logical reasoning over unseen entities. Additionally, the im-

| Model | Avg | 1p | 2p | 3p | 2i | 3i | ip | pi | 2u | up |
|---|---|---|---|---|---|---|---|---|---|---|
| FB15k-237 | | | | | | | | | | |
| *GQE* | 0.228 | 0.402 | 0.213 | 0.155 | 0.292 | 0.406 | 0.083 | 0.17 | 0.169 | 0.163 |
| *Q2B* | 0.268 | 0.467 | 0.240 | 0.186 | 0.324 | 0.453 | 0.108 | 0.205 | 0.239 | 0.193 |
| *TEMP(Q2B)* | 0.294 | 0.457 | 0.278 | 0.234 | 0.369 | 0.496 | 0.229 | 0.117 | 0.276 | 0.189 |
| *BiQE* | 0.262 | 0.472 | 0.293 | 0.245 | 0.345 | 0.473 | 0.096 | 0.211 | 0.075 | 0.145 |
| *SILR* | **0.296** | 0.471 | **0.302** | **0.249** | 0.358 | 0.484 | 0.113 | **0.222** | **0.283** | 0.181 |
| FB15k | | | | | | | | | | |
| *GQE* | 0.386 | 0.636 | 0.345 | 0.248 | 0.515 | 0.624 | 0.151 | 0.310 | 0.376 | 0.273 |
| *Q2B* | 0.484 | 0.786 | 0.413 | 0.303 | 0.593 | 0.712 | 0.211 | 0.397 | 0.608 | 0.330 |
| *TEMP(Q2B)* | 0.554 | 0.840 | 0.498 | 0.422 | 0.674 | 0.779 | 0.483 | 0.261 | 0.727 | 0.300 |
| *BiQE* | 0.482 | 0.810 | 0.526 | 0.461 | 0.677 | 0.781 | 0.209 | 0.456 | 0.185 | 0.232 |
| *SILR* | **0.596** | **0.867** | **0.575** | **0.489** | **0.736** | **0.824** | 0.287 | **0.500** | **0.767** | 0.321 |
| NELL995 | | | | | | | | | | |
| *GQE* | 0.247 | 0.418 | 0.228 | 0.205 | 0.316 | 0.447 | 0.081 | 0.186 | 0.199 | 0.139 |
| *Q2B* | 0.306 | 0.555 | 0.266 | 0.233 | 0.343 | 0.480 | 0.132 | 0.212 | 0.369 | 0.163 |
| *TEMP(Q2B)* | 0.373 | 0.625 | 0.343 | 0.342 | 0.410 | 0.552 | 0.209 | 0.141 | 0.477 | 0.262 |
| *BiQE* | 0.309 | 0.632 | 0.310 | 0.332 | 0.370 | 0.525 | 0.091 | 0.164 | 0.173 | 0.184 |
| *SILR* | 0.342 | **0.641** | 0.329 | 0.337 | 0.376 | 0.532 | 0.055 | 0.177 | 0.450 | 0.182 |

Table 4: Transductive H@3 results on FB15k-237, FB15k and NELL995 datasets. The results of *GQE* and *Q2B* are taken from (Ren et al., 2020).

provement over the other positional textual encoding model, *BiQE*, demonstrates that our structure knowledge modeling during textual encoding is capable of enhancing the inductive complex query answering capability.

**Cross-KG Generalization** We further explore a more challenging cross-KG inductive setting, where the model is trained and tested on different datasets and requires generalizing to completely different KGs. Specifically, we take FB15k and NELL995 as the source/target and target/source datasets, respectively. In this scenario, we adopt the few-shot setting, where 500 random samples of the target domain are provided for continual learning to achieve better transferring. As embedding-based methods, even when aware of type information, are unable to embed most entities and relations in new KGs with unseen types, we only compare our *SILR* with the reimplemented *BiQE*. The results in Table 3 show that our *SILR* performs better than *BiQE*, and it can not only generalize to unseen entities but also perform logical reasoning over new KGs with only a few portions observed. This manifests the effectiveness of our method on inductive logical reasoning over KGs even in the real-world challenging cross-KG setting.

### 3.4 Transductive Setting

Although textual encoding methods have the inductive potential, their performance often lags behind embedding-based models due to learning inefficiency and the inability to structure knowledge modeling (Wang et al., 2022). We also compare our *SILR* with transductive logical reasoning methods to illustrate the logical reasoning performance over KGs with structure modeling. The compared models including *GQE*, *Q2B*, *TEMP(Q2B)* and *BiQE* where the first three are embedding-based models while the last one is a positional textual encoding model. Since *BiQE* does not evaluate query types 2u and up, and does not provide results for FB15k, we reimplement it for a fair comparison.

The results are shown in Table 4. Our *SILR* outperforms *BiQE*, particularly in query types involving intersection and union, indicating that our structure knowledge modeling can effectively improve the logical reasoning performance of textual encoding and help generalize to unseen query structures. Although textual encoding methods have the potential for inductive KG reasoning, they still lag behind embedding-based methods for the transductive setting, due to explicit structure modeling and better learning efficiency of embedding-

based methods (Wang et al., 2022). In this work, we mainly focus on improving textual encoding methods for inductive complex reasoning, but our method still achieves comparable transductive performance. This demonstrates the effectiveness of our inductive method with query structure modeling on transductive logical reasoning over KGs.

### 3.5 Further Analysis

**Ablation Study** To dive into the impact of different components in our model on both inductive and transductive logical reasoning over KGs, we conduct an ablation study on the FB15k-237-V2 and FB15k-237 datasets. We respectively take bert-large-cased and bert-base-cased as baseline models for FB15k-237-V2 and FB15k-237. They remove Structural Prompt (*SP*) and Geometric Operation Modeling (*GOM*) from the final model *SILR*, which directly encodes linearized query structures for answer prediction.

As shown in Table 5, incorporating structural prompting and geometric operation modeling can both improve the baseline model, but still perform worse than our final *SILR*. This indicates that these two measures for modeling structure knowledge during query text encoding can enhance the inductive and transductive performance of logical reasoning over KGs.

| Model | FB15k-237-V2 | | FB15k-237 | |
|---|---|---|---|---|
| | H@3 | H@10 | H@3 | H@10 |
| *Baseline* | 0.072 | 0.151 | 0.258 | 0.397 |
| *w/ SP* | 0.073 | 0.162 | 0.288 | 0.429 |
| *w/ GOM* | 0.071 | 0.167 | 0.281 | 0.420 |
| *SILR* | 0.079 | 0.178 | 0.296 | 0.437 |

Table 5: Ablation study results on FB15k-237-V2 and FB15k-237. *SP* and *GOM* respectively denote Structural Prompt and Geometric Operation Modeling.

**Query Structure Generalization** Embedding-based methods are known to generalize well to unseen query structures due to their explicit spatial structure learning. To analyze the generalizability of our implicit structure-modeled textual encoding method to different logical query structures, we further construct four types of complicated query structures with more relations and geometric operations, including 4p, 5p, 3ip and i2p, based on the query structures in test sets. The detailed illustrations and explanations of these query structures are given in Appendix C. We directly evaluate our method on these more complicated queries in both inductive and transductive datasets FB15k-237-V2 and FB15k-237. The results are listed in Table 6. We can see that compared to seen and unseen query structures in the original datasets in Table 2 and 4, our method can also generalize to these complicated structures with more logical operations and achieves impressive performance. This demonstrates that the design of structural prompt and implicit geometric operation modeling can effectively learn structure knowledge and improve query structure generalizability.

| Model | 4p | 5p | 3ip | i2p |
|---|---|---|---|---|
| Inductive | 0.186 | 0.089 | 0.256 | 0.092 |
| Transductive | 0.192 | 0.188 | 0.367 | 0.199 |

Table 6: Analysis of generalization to more complicated query structures. For inductive, we report the H@10 scores on FB15k-237-V2 and for transductive we report the H@3 results on FB15k-237.

## 4 Related Work

Answering first-order logic queries over incomplete KGs is a challenging task (Guu et al., 2015). Previous research (Lin et al., 2015; Hamilton et al., 2018) mainly studies transductive logical reasoning, where the training and testing are performed on the same KG. Embedding-based methods are widely used to embed both logical queries and entities into a joint low-dimensional vector space and push answer entities and queries to be close enough, enabling answer prediction through embedding similarity, even when the involved relation is absent. Following this paradigm, some further propose extending the embedding of the query/entity from a single point to a region (Ren et al., 2020; Zhang et al., 2021; Choudhary et al., 2021b) or probabilistic distribution (Ren and Leskovec, 2020; Choudhary et al., 2021a) over vector space to map arbitrary first-order logic queries into sets of answer entities. However, these methods are unable to tackle the inductive problem, which requires generalizing to unseen entities and relations. Although (Hu et al., 2022) proposes enriching the entity and relation embedding with type information for inductive logical reasoning, it can only generalize to elements of observed types.

Another line of research focuses on inductive

logical reasoning over KGs using textual encoding methods. With the advance of large-scale pre-trained language models (Devlin et al., 2018; Liu et al., 2019), these methods propose transforming the graph structures into linearized text and utilize PLMs for encoding (Yao et al., 2019; Zha et al., 2022). With the strong generalizability of PLMs, they can easily generalize to unseen entities/relations, but struggle to model structure knowledge during text encoding. Some works (Wang et al., 2021b; Daza et al., 2021; Wang et al., 2021a) propose to follow TransE (Bordes et al., 2013) to apply the translation function between entity and relation representations for geometric structure learning. Nevertheless, these methods usually require descriptions of entities and relations for encoding and assume these descriptions are readily available. Besides, they only focus on simple link prediction tasks without exploring complex structure modeling in logical reasoning, which is essential for generalizing within the same query structure/type. We thus propose to simultaneously encode the linearized query and preserve the logical structure knowledge by structural prompt and separate geometric operation modeling for inductive logical reasoning over KGs.

## 5 Conclusion

In this paper, we present the first flexible inductive method for answering complex logical queries over incomplete KGs which can generalize to any unseen entities and relations. To accomplish this goal, we propose a structure-model textual encoding model that utilizes PLMs to encode linearized query structures to find the answer entities. For structure modeling of complex queries, we design structural prompts to implicitly indicate PLMs the order of geometric operations execution in each query, and separately model three geometric operations on representation space using a pre-trained encoder, an attention layer, and a maxout layer. Experimental results demonstrate the effectiveness of our model on logical reasoning over KGs in both inductive and transductive settings.

## Limitations

This study has potential limitations. First, it only focuses on answering existential positive first-order logic queries but does not support the negation operation. We will later address this limitation by modeling the negation operation. Second, we utilize BERT as the backbone model for inductive generalization due to computing resource limits. We plan to investigate the use of more powerful pre-trained language models with stronger generalizability in future research to improve inductive logical reasoning over KGs.

## Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 6217020551) and Science and Technology Commission of Shanghai Municipality Grant (No.21QA1400600).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan Reddy. 2021a. Probabilistic entity representation model for reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:23440–23451.

Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K Reddy. 2021b. Self-supervised hyperboloid representations from logical queries over knowledge graphs. In *Proceedings of the Web Conference 2021*, pages 1373–1384.

Daniel Daza, Michael Cochez, and Paul Groth. 2021. Inductive entity representations from text via link prediction. In *Proceedings of the Web Conference 2021*, pages 798–808.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. *Advances in neural information processing systems*, 31.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Xiaoli Li, Ru Li, and Jeff Z Pan. 2022. Type-aware embeddings for multi-hop reasoning over knowledge graphs. *arXiv preprint arXiv:2205.00782*.

Bhushan Kotnis, Carolin Lawrence, and Mathias Niepert. 2021. Answering complex queries in knowledge graphs with bidirectional sequence encoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4968–4977.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969*.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33:19716–19726.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

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 *Proceedings of the Web Conference 2021*, pages 1737–1748.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. *arXiv preprint arXiv:2203.02167*.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Hanwen Zha, Zhiyu Chen, and Xifeng Yan. 2022. Inductive relation prediction by bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5923–5931.

Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34:19172–19183.

## A Structure Linearization Templates

In this part, we list the detailed linearization templates of different structural query types in Table 7.

| Query Type | Linearization Template |
|---|---|
| [1p] | [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ |
| [2p] | [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [relation] $t(r_2)$ |
| [3p] | [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [relation] $t(r_2)$ [projection] [relation] $t(r_3)$ |
| [2i] | [intersection] [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_2)$ |
| [3i] | [intersection] [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_2)$ [projection] [anchor] $t(e_3)$ [relation] $t(r_3)$ |
| [pi] | [intersection] [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [relation] $t(r_2)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_3)$ |
| [ip] | [intersection] [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [relation] $t(r_3)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_2)$ [projection] [relation] $t(r_3)$ |
| [2u] | [union] [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_2)$ |
| [up] | [union] [projection] [anchor] $t(e_1)$ [relation] $t(r_1)$ [projection] [relation] $t(r_3)$ [projection] [anchor] $t(e_2)$ [relation] $t(r_2)$ [projection] [relation] $t(r_3)$ |

Table 7: The linearization templates of each query structure.

| Dataset | Training | | Validation | | Test | |
|---|---|---|---|---|---|---|
| | 1p | others | 1p | others | 1p | others |
| FB15k | 273,710 | 273,710 | 59,097 | 8,000 | 67,016 | 8,000 |
| FB15k-237 | 149,689 | 149,689 | 20,101 | 5,000 | 22,812 | 5,000 |
| NELL995 | 107,982 | 107,982 | 16,927 | 4,000 | 17,034 | 4,000 |
| FB15k-237-V2 | 9,964 | 9,964 | 1,738 | 2,000 | 791 | 1,000 |
| NELL-V3 | 12,010 | 12,010 | 2,197 | 2,000 | 1,167 | 1,500 |

Table 8: Statistics of logical queries in different types in each data split. "others" means other eight query types including 2p, 3p, 2i, 3i, pi, ip, 2u and up.

## B Data Statistics

In Table 8, we summarize the statistics of logical queries in our experimented datasets for both inductive and transductive settings.

## C Constructed Query Types

We here introduce our generated more complicated query types involving more relations and logical operations, which are used to illustrate the complicated query structure generalizability.

$$
\begin{aligned}
[4p]\ q &= v_?.\exists v_1, v_2, v_3 : r_1(e_1, v_1) \wedge r_2(v_1, v_2) \\
&\quad \wedge r_3(v_2, v_3) \wedge r_4(v_3, v_?) \\
[5p]\ q &= v_?.\exists v_1, v_2, v_3, v_4 : r_1(e_1, v_1) \wedge r_2(v_1, v_2) \\
&\quad \wedge r_3(v_2, v_3) \wedge r_4(v_3, v_4) \wedge r_5(v_4, v_?) \\
[3ip]\ q &= v_?.\exists v : r_1(e_1, v) \wedge r_2(e_2, v) \wedge r_3(e_3, v) \\
&\quad \wedge r_4(v, v_?) \\
[i2p]\ q &= v_?.\exists v_1, v_2 : r_1(e_1, v_1) \wedge r_2(e_2, v_1) \\
&\quad \wedge r_3(v_1, v_2) \wedge r_4(v_2, v_?)
\end{aligned}
\tag{5}
$$

## ACL 2023 Responsible NLP Checklist

### A For every submission:

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

☐ A2. Did you discuss any potential risks of your work?
*Not applicable. Left blank.*

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

☑ A4. Have you used AI writing assistants when working on this paper?
*Use Grammarly and ChatGPT for spelling checking and polishing.*

### B ☑ Did you use or create scientific artifacts?

*Section 3.1 & 3.2*

☑ B1. Did you cite the creators of artifacts you used?
*Section 3.1 & 3.2*

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
*Not applicable. Left blank.*

☑ 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)?
*Section 3.1 & 3.2*

☐ 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?
*Not applicable. Left blank.*

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
*Section 3.1*

☑ 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.
*Section B*

### C ☑ Did you run computational experiments?

*Section 3*

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*Section 3.2*

*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?
*Section 3.2*

☑ 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?
*Section 3*

☑ 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.)?
*Section 3.2*

**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.*