Improving Complex Reasoning over Knowledge Graph with Logic-Aware Curriculum Tuning

Anonymous ACL submission

Abstract

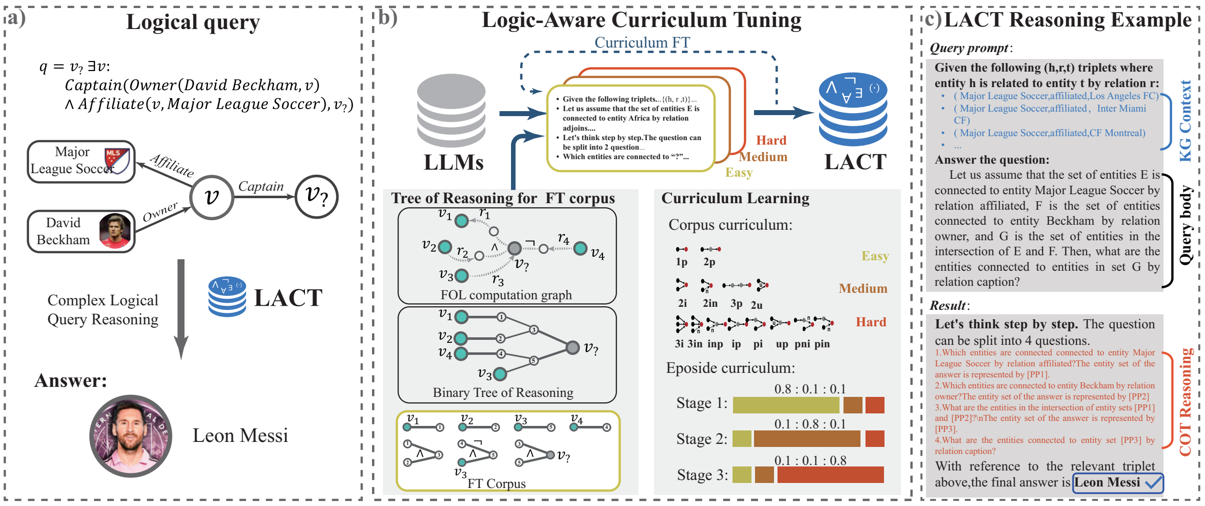
Answering complex logical queries over incomplete knowledge graphs (KGs) is challenging. Most previous works have focused on learning entity/relation embeddings and simulating first-order logic operators with various neural networks. However, they are bottlenecked by the inability to share world knowledge to improve logical reasoning, thus resulting in suboptimal performance. In this paper, we propose a complex logical reasoning schema over knowledge graphs upon large language models (LLMs), containing a curriculum-based logical-aware instruction tuning framework, named LACT. Specifically, we augment the arbitrary first-order logical queries via binary tree decomposition, to stimulate the reasoning capability of LLMs. To address the difficulty gap among different types of complex queries, we design a simple and flexible logic-aware curriculum learning framework. Experiments across widely used datasets demonstrate that LACT has substantial improvements (brings an average +5.5% MRR score) over advanced methods, achieving the new state-of-the-art.

1 Introduction

Large-scale knowledge graphs (KGs) such as FreeBase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and WikiData (Vrandečić and Krötzsch, 2014) stores structural knowledge in a collection of fact triplets and are widely adopted by many domains. Unfortunately, KGs are often incomplete, leaving many missing triplets undiscovered. Thus, complex logical reasoning over such KGs (Hamilton et al., 2018; Ren and Leskovec, 2020) is challenging and has attracted much attention in the recent years. A complex logical query can be represented with First-Order Logic (FOL) that includes logical operators such as conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), and existential quantifier ($\exists$), etc. A more direct approach involves the representation of computation graphs as Directed Acyclic Graphs (DAGs), which can be resolved through the systematic traversal of Knowledge Graphs (KG). This process entails the allocation of suitable entities to intermediate variables based on their structural attributes (Dalvi and Suciu, 2007).

Inspired by the success of knowledge graph embedding (KGE) (Bordes et al., 2013; Bai et al., 2021), a line of research proposes to answer complex logical queries by learning query embedding and simulating logical operators with well-designed neural networks (Chen et al., 2022; Zhu et al., 2022; Zhang et al., 2021; Arakelyan et al., 2020) Current research based on embeddings primarily focuses on the creation of diverse latent space geometries, such as vectors (Hamilton et al., 2018), boxes (Ren et al., 2019), hyperboloids (Choudhary et al., 2021), and probabilistic distributions (Ren et al., 2019), to effectively capture the semantic position and logical coverage of knowledge graph entities.

However, these approaches are limited in their performance due to the following. (1) **Limited information**: The information contained in a knowledge graph is usually incomplete and limited. When only the information from the knowledge graph can be used, it is difficult to answer some complex reasoning that lacks relevant information. (2) **High complexity of logical queries**: The intricacies of world knowledge determine the complexity of reasoning in practical applications, which determines that it is difficult to model the relationship of world knowledge through simple space geometries figures that may lose potentially complex relationship information (Choudhary et al., 2021), thus limiting the effect of complex logical reasoning. (3) **Generalizability**: KGE method for a particular KG can not generalize to other KGs which limits the applicability of these approaches in real-world scenarios where KGs can vary widely in terms of their structure and content.



- Given the following triplets...{(h, r ,t)}...
- Let us assume that the set of entities E is connected to entity Africa by relation adjoins....
- Let's think step by step.The question can be split into 2 question...
- Which entities are connected to "?"...

Query prompt:

Given the following (h,r,t) triplets where entity h is related to entity t by relation r:

- (Major League Soccer,affiliated,Los Angeles FC)
- (Major League Soccer,affiliated， Inter Miami CF)
- (Major League Soccer,affiliated,CF Montreal)
- ...

Answer the question:

Let us assume that the set of entities E is connected to entity Major League Soccer by relation affiliated, F is the set of entities connected to entity Beckham by relation owner, and G is the set of entities in the intersection of E and F. Then, what are the entities connected to entities in set G by relation caption?

Result:

Let's think step by step. The question can be split into 4 questions.
1.Which entities are connected connected to entity Major League Soccer by relation affiliated?The entity set of the answer is represented by [PP1].
2.Which entities are connected to entity Beckham by relation owner?The entity set of the answer is represented by [PP2]
3.What are the entities in the intersection of entity sets [PP1] and [PP2]?\nThe entity set of the answer is represented by [PP3].
4.What are the entities connected to entity set [PP3] by relation caption?
With reference to the relevant triplet above,the final answer is **Leon Messi** ✓

Figure 1: Schematic illustration: a) Answering logical query over KG with LACT. b) The framework of Logic-Aware Curriculum Tuning over LLama. We leverage binary tree decomposition strategy (Seen in Section 4.2) to construct logic-rich FT corpus and Curriculum learning strategy (Seen in Section 4.3) to fine-tune a base LLM. c) Performing reasoning using well-designed prompts.

Recently, large language models (LLMs) (Achiam et al., 2023; Touvron et al., 2023a; Zeng et al., 2022) show outperforming capabilities to a wide range of tasks (Zhao et al., 2023; Ouyang et al., 2022). (Choudhary and Reddy, 2023) construct prompt templates and apply LLMs as text-generators to answer complex queries. However, it suffers from hallucination problem (Zhang et al., 2023c) Besides, a KG preserves intricate structural information such as subgraph structure, relational patterns, relative entities/relations. Due to the inherent hallucination problem of LLMs, overlooking these information results in a significant loss. Finally, the overhead of using the current closed-source LLM in work (Choudhary and Reddy, 2023) and the source overhead caused by multiple inferences also have to be considered.

In this paper, we propose **L**ogic-**A**ware **C**urriculum **T**uning (**LACT**), a novel fine-tune framework for answering complex logical query, which stimulates the ability of LLMs to perform complex reasoning on knowledge graphs. We propose a strategy to incorporate the knowledge contained in the KGs into our training corpus to activate the corresponding knowledge of the LLMs and supplement the missing relevant knowledge of the LLMs during the fine-tuning process. At the same time, we have proven that data argument by binary tree decomposition can arouse the corresponding capabilities of LLMs and effectively improve their reasoning performance. At last, we show that curriculum learning(Bengio et al., 2009) can effectively smooth the difficulty differences between different types of queries and greatly improve the results of difficult queries. In summary, our contribution is three-folded:

- We propose a logic-aware curriculum fine-tuning (LACT) paradigm for complex logical reasoning over KGs.
- LACT achieves state-of-the-art performance beyond embedding-based and PLM-based methods, using just a 7B llama2 model.
- Through extensive experimentation, we found that fine-tuning corpus constructed with rigorous logical context over KGs and curriculum learning can significantly enhance LLM logical reasoning ability.

2 Related Works

2.1 Logical Reasoning over Knowledge Graph

Given a FOL query over a KG, complex logical reasoning aims to answer correct entities, which contains not only multi-hop paths but logical operators(Guu et al., 2015; Hamilton et al., 2018). Most of current approaches concentrated on learning meaningful query embeddings (Chen et al., 2022; Zhu et al., 2022; Zhang et al., 2021; Arakelyan et al., 2020; Wang et al., 2023b). Neuralizing logical operators through a specific embedding space, thereby embedding FOL queries into a vector space(Hamilton et al., 2018; Ren et al., 2019; Choudhary et al., 2021), or probabilistic distribution(Ren et al., 2019), and predict answers by locating nearest neighbours to answer set representation. Additionally, approaches such as CQD(Arakelyan et al., 2020) have focused on improving the performance of complex reasoning tasks through the answer composition of simple intermediate queries, and QTO (Bai et al., 2023) proposes query computation tree optimization that can efficiently find the exact optimal solutions. However, embedding-based methods usually lack interpretability as there is no explicit mapping between the embedding and the set of entities, and this limits their generalization ability to more complicated query structures.

2.2 LLMs for KG Reasoning

In recent years, substantial advancements have been witnessed in the domain of LLMs (Achiam et al., 2023; Touvron et al., 2023b; Peng et al., 2023; Zhong et al., 2023). Among these, instruction tuning (IT) (Ouyang et al., 2022) and the alignment of the model (Wang et al., 2023c) with human preferences stand out.

Within the realm of LLM, the integration of LLMs with Knowledge Graphs (KG) (Pan et al., 2024; Wang et al., 2023a; Luo et al., 2024) constitutes a prominent and consequential research avenue.

Leveraging its potent generative capabilities, LLMs prove invaluable in addressing Knowledge Graph-related tasks, including but not limited to Knowledge Graph Completion (KGC) (Zhu et al., 2023; Zhang et al., 2023b), entity alignment (Zhang et al., 2023a), Knowledge Graph Question Answering (KGQA) (Luo et al., 2024), and others (Luo et al., 2023). Consequently, the synergy between Knowledge Graphs for LLMs (KG4LLM) and LLMs for Knowledge Graphs (LLM4KG) emerges as an essential focal point, bearing significance in advancing the collective capabilities of both entities.

We focus on applying LLMs in the Complex Logical Reasoning task, which has not been carefully studied yet. (Choudhary and Reddy, 2023) made the initial attempt by prompt engineer but it lacks in-depth research and simply uses LLMs as text generators.

3 Preliminary

3.1 Knowledge Graph

In our work, a knowledge graph is $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ where $\mathcal{E}, \mathcal{R}$ are the set of entity and relation respectively. With regard to generality, KG can be organized as a set of triplets $\{\mathcal{T} = (v_s, r, v_t) | v_s, v_t \in \mathcal{E}, r \in \mathcal{R}\}$, where v_s/v_t denotes the head/tail entity.

3.2 Complex logical query

Complex logical query is used for retrieving and manipulating data stored in knowledge graphs, which is grounded in a subset of FOL. The process of answering a complex logical query involves trying to match a suitable results using the composition of queries:

$$q[v_?] = \exists v : q_1 \wedge q_2 \wedge \cdots \wedge q_n, \tag{1}$$

or,

$$q[v_?] = \exists v : q'_1 \vee q'_2 \vee \cdots \vee q'_n, \tag{2}$$

where q denotes a FOL query. Note that Eq. (1) is conjunctive normal form (CNF) and Eq. (2) is disjunctive normal form (DNF). The two can be equivalently converted to each other via De Morgan's law. Following previous works (Ren et al., 2019), we focus on modeling the operations: projection $r(\cdot)$, conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). Additionally, note that existential positive first-order (EPFO) queries only includes projection, conjunction ($\wedge$), and disjunction ($\vee$).

4 Methodology

4.1 Instruction Tuning on LLMs

In this section, we introduce how to incorporate the KG information in the text-based prompt.

When applying LLMs to complex logical reasoning, we denote an LLM as $\mathcal{M}$ which is a text decoder to generate the corresponding output. If we start from the above definition, this task can be modelled as a text generation task. However, triplet generation is different from vanilla text generation because the entities and the relation in the triplet prompt have complex semantic information defined by the given KG. In fact, we want the generated answers to be entities that exist in KG itself. Without these knowledge, the predicted answers are unreliable and unstable. Thus, incorporating the KG information into the prompt to provide more auxiliary information is the key to engage LLMs in complex logical reasoning.

In particular, when we fine-tune $\mathcal{M}$, we can treat the training corpus as a set of question-answer pairs $(\mathcal{S}, \mathcal{A})$. For the task of complex logical reasoning over knowledge graph, the input textual sequence $\mathcal{S}$ consists of the description of question $\mathcal{D}$, knowledge graph neighbourhood information(i.e. related triplets) $\mathcal{X}$ and logical query. In our work, we used a simple but effective method called *greedy* *depth traversal algorithm* to search for neighbourhood information (Detailed algorithm can be found in Appendix A). The logical query contains the textual information about the query q_τ with the query structure τ and specific query content Q_τ that needs to be processed, which can be denoted as $f_1(q_\tau)$. Likewise,we can denote the output answer A as $f_2(V_\tau)$, where f_2 indicates textualization of V_τ here. In summary, the fine-tune training corpus $\mathcal{C}$ can be expressed in the following form:

$$\mathcal{C} = (\mathcal{S}, \mathcal{A}) = (\mathcal{D} \oplus \mathcal{X} \oplus f_1(q_\tau), f_2(V_\tau)). \quad (3)$$

The model $\mathcal{M}$(parameterized by θ) is fine-tuned by the next token prediction task. We fine-tune $\mathcal{M}$ to obtain the final model by maximizing the log-likelihood of the next token. The training objective can be formulated as

$$\mathcal{L} = -\frac{1}{|\mathcal{C}|} \sum_{i=1}^{|\mathcal{C}|} \log P_{\mathcal{M}} \left(c_i \mid c_{<i} \right), \quad (4)$$

where $c_i(i = 1, 2, ..., |\mathcal{C}|)$ represents the textual tokens of the training corpus $\mathcal{C}$. For our task, the training objective can be transferred as

$$\mathcal{L} = -\frac{1}{|\mathcal{C}|} \sum_{i=1}^{|\mathcal{C}|} \log P_{\mathcal{M}} \left(\mathcal{A} \mid \mathcal{S} \right). \quad (5)$$

4.2 Data Augmentation by Binary Tree Decomposition

This section introduces how to build fine-tuning corpora that make LLMs (Large Language Models) logic-aware based on instruction tuning.

Chain of thought (COT) enables models to decompose multi-step problems into intermediate steps, subsequently improving the reasoning abilities of LLMs (Wei et al., 2022). However, pure-prompt based reasoning needs more incontext memory to perform complex logical reasoning. Considering complex logical queries, which query structure can be transferred into the form of a DAG and its hierarchical structure becomes a natural fit for decomposition into a series of sub-problems. So we propose a method for data augment based on *Binary Tree Decomposition Mechanism* to stimulate LLMs with the potential to decompose a complex query into a chain of simple queries. **Binary Tree Decomposition Mechanism.** The Binary Tree Decomposition Mechanism is divided into the following three steps:

Query Computation Tree. For a complex FOL query, like the example shown in Figure 1, its computation graph that is a directed acyclic graph can be converted into a tree where the root node is $v_?$. *query computation tree*. The answer variable and the constant entities in the query correspond to root and leaf nodes in the query computation tree. Each edge in the query computation tree points from the child node to the parent node. It can be recursively deduced that the subtree rooted at any non-leaf node in the tree corresponds to a subquery. In Appendix B, we provide a systematical procedure for transforming an FOL query to its query computation tree.

Binary Tree Decomposition. For the one-to-many intersection/union structures in the tree, we separate each parent node into two child nodes. Note that the union branches merging step may create one-to-many structures that consist of both intersection and union edges, take Figure 1 for an example. This can be taken care of by first separating $v_?$ into an intersection structure (v'_3 and v'_5 in the example), and then separating the child node into an intersection structure (v'_1 and v'_2 for v'_3, v'_4, v_3 for v'_5), where v' denotes an intermediate entity retrieved by Neighborhood Retrieval Algorithm (Seen in Appendix A).

Reverse Level Traversal. Finally, we decompose the binary computation tree into independent branches. Since the root node of the calculation tree is the answer entity, we perform a hierarchical traversal of all non-leaf nodes of the binary tree in reverse. As shown in Figure 1, the complex FOL query is decomposed into a sequence:

$$[(v_1, r, v'_1), (v_2, r, v'_2), (v_3, r, v'_5), (v_4, r, v'_4),$$
$$(v'_1, r, v'_3), \wedge, (v'_2, r, v'_3), \neg, (v'_4, r, v'_5), \wedge, (v_3, r, v'_5),$$
$$(v'_3, r, v_?), \wedge, (v'_5, r, v_?)].$$

Data Augmentation. Now we can turn any loopless FOL query into a series of separate subqueries. We use a defined template to integrate the decomposition process into the answers to the training corpus. So, the training corpus $\mathcal{C}$ can be transferred into the following form:

$$\mathcal{C} = (\mathcal{S}, \mathcal{A}) = (\mathcal{D} \oplus \mathcal{X} \oplus f_1(q_\tau), f_2(V_{\tau,\mathcal{D}ecomposed})), \quad (6)$$

where $V_{\tau,\mathcal{D}ecomposed}$ indicates the answer corresponding to the logical query with the decomposition reasoning path.

4.3 Fine-tuning Enhanced by Curriculum Learning

As mentioned in previous sections, though decomposing into chain responses, complex queries still vary greatly in difficulty and complexity due to differences between query structure.

Naturally, we believe that these different types of samples should not be simply lumped together. Intuitively, we incorporate curriculum learning into our training. To be specific, In view of the particularity of complex reasoning data, when we decompose it into logical chains, naturally, we can use the number of decomposed sub-logical queries as a natural difficulty discriminator to select different types of queries, e.g., a 1p query would be defined as difficulty-1, while a 2p query, which can be decomposed into two projection queries and an intersection query, would be defined as difficulty-3. The detailed difficulty discriminating process will be shown in Appendix (Table S1).

Finally, we divided samples into three parts: easy samples, medium samples and difficult samples according to the difficulty level. Correspondingly, our training process is also divided into three stages. After we did some exploratory experiments, we did not simply train three data sets in the order of easy-medium-difficult. On the contrary, we decided to first use 80% easy samples, 10% medium samples, and 10% difficult samples for the first stage of training and the subsequent two-stage training process is a Leto, and experimental results in the next few sections also proved that this is effective.

4.4 Reasoning Module

We use the final LACT LLM as the answer generator, as shown in Figure 1(c). We retrieve relevant information and textualize the FOL query, and finally we populate it into the template in Prompt 1 to generate responses.

We use the LLM to do a simple text generation task to get the answer. After fine-tune, LACT LLM can follow the output mode in training stage in Figure 1, so we can extract final answers through simple regular expressions with the template in Prompt 2.

Prompt 1: Query Prompt Template of LACT.

Query Prompt Template

Given the following (h,r,t) triplets where entity h is related to entity t by relation r.
<Related Triplets>

Answer the question:
<question>

Prompt 2: Answer Template of LACT.

Expected Answer Template

Let's think step by step. The question can be split into <k> question.
<k decomposed subqueries>

With reference to the relevant triplet above, the final answer is
<answer entities>.

5 Experiments

5.1 Training Settings

Training datasets We opt for the the most popular datasets: **FB15K**, **FB15K-237**, **NELL995**. Detailed information about dataset is listed in D.1. We used the training set of the above dataset as the original training data.

Training Details We use open-source model LLaMA-2-base, including two different parameter sizes: 7B and 13B, as the base model for fine-tuning. All LLaMA-2-7B and LLaMA-2-13B models are trained by fully fine-tuning.

For the fully fine-tuning setting, we use the AdamW optimizer to train the model with 1 epoch and the batch size is 128. We use 8 NVIDIA A100 GPUS for training with the learning rate of 3e-6.

5.2 Experimental Settings

Baseline Methods For comparing with KGE, we chose the following representative methods as baselines: **GQE** (Hamilton et al., 2018), **Query2Box(Q2B)** (Choudhary and Reddy, 2023), **BetaE** (Ren and Leskovec, 2020), **CQD** (Arakelyan et al., 2020), **ConE** (Zhang et al., 2021), **GNN-QE** (Zhu et al., 2022), **QTO** (Bai et al., 2023). We also compared our method to an LLM-based method, **LARK** (Choudhary and Reddy, 2023).

Evaluation Protocol Following previous works (Ren et al., 2019), we use mean reciprocal rank (MRR) as standard evaluation protocols for evaluating complex reasoning over knowledge graph. In the filtered setting, all easy and hard answers are filtered out during ranking. The detail could be found in Appendix D.3.

5.3 Main Results

The main experiment results of three datasets are shown in Table 1. The baselines were trained on 1p/2p/3p/2i/3i queries, hence other than these 4 types of EPFO queries serve as OOD queries, and we report the average result on these queries in avg_{ood}. We observe that LACT significantly outperforms baseline methods across all datasets. Notably, LACT yields an averag gain of 7.3%, 2.9%, and 6.3% on avg_p, avg_{odd}, and avg_n, compared to

Method	avg_p	avg_{ood}	avg_n	1p	2p	3p	2i	3i	pi	ip	2u	up	2in	3in	inp	pin	pni
								FB15K									
GQE	28.0	20.1	-	54.6	15.3	10.8	39.7	51.4	27.6	19.1	22.1	11.6	-	-	-	-	-
Query2Box	38.0	29.3	-	68.0	21.0	14.2	55.1	66.5	39.4	26.1	35.1	16.7	-	-	-	-	-
BetaE	41.6	34.3	11.8	65.1	25.7	24.7	55.8	66.5	43.9	28.1	40.1	25.2	14.3	14.7	11.5	6.5	12.4
CQD-CO	46.9	35.3	-	89.2	25.3	13.4	74.4	78.3	44.1	33.2	41.8	21.9	-	-	-	-	-
CQD-Beam	58.2	49.8	-	89.2	54.3	28.6	74.4	78.3	58.2	67.7	42.4	30.9	-	-	-	-	-
ConE	49.8	43.4	14.8	73.3	33.8	29.2	64.4	73.7	50.9	35.7	55.7	31.4	17.9	18.7	12.5	9.8	15.1
GNN-QE	72.8	68.9	38.6	88.5	69.3	58.7	79.7	83.5	69.9	70.4	74.1	61.0	44.7	41.7	42.0	30.1	34.3
QTO	74.0	71.8	49.2	89.5	67.4	58.8	80.3	**83.6**	75.2	74.0	**76.7**	**61.3**	61.1	61.2	47.6	**48.9**	27.5
LARK	56.1	43.1	18.4	72.8	50.7	36.2	66.9	60.4	56.1	23.5	52.4	40.6	16.2	5.7	33.7	26.1	10.0
LACT	**82.6**	**71.9**	**56.9**	**93.5**	**73.5**	**59.6**	**92.3**	82.3	**76.8**	**75.9**	74.6	60.4	**81.2**	**61.6**	**52.0**	43.5	**41.7**
								FB15K-237									
GQE	16.3	10.3	-	35.0	7.2	5.3	23.3	34.6	16.5	10.7	8.2	5.7	-	-	-	-	-
Query2Box	20.1	15.7	-	40.6	9.4	6.8	29.5	42.3	21.2	12.6	11.3	7.6	-	-	-	-	-
BetaE	20.9	14.3	5.5	39.0	10.9	10.0	28.8	42.5	22.4	12.6	12.4	9.7	5.1	7.9	7.4	3.5	3.4
CQD-CO	21.8	15.6	-	46.7	9.5	6.3	31.2	40.6	23.6	16.0	14.5	8.2	-	-	-	-	-
CQD-Beam	22.3	15.7	-	46.7	11.6	8.0	31.2	40.6	21.2	18.7	14.6	8.4	-	-	-	-	-
FuzzQE	24.0	17.4	7.8	42.8	12.9	10.3	33.3	46.9	26.9	17.8	14.6	10.3	8.5	11.6	7.8	5.2	5.8
ConE	23.4	16.2	5.9	41.8	12.8	11.0	32.6	47.3	25.5	14.0	14.5	10.8	5.4	8.6	7.8	4.0	3.6
GNN-QE	26.8	19.9	10.2	42.8	14.7	11.8	38.3	54.1	31.1	18.9	16.2	13.4	10.0	16.8	9.3	7.2	7.8
QTO	33.5	27.6	15.5	49.0	21.4	21.2	43.1	56.8	38.1	28.0	22.7	21.4	16.8	26.7	15.1	13.6	5.4
LARK	50.7	41.0	10.6	73.6	40.5	26.8	46.1	43.1	49.9	22.9	**62.8**	28.3	6.5	3.4	23.2	16.5	3.2
LACT	**57.0**	**44.4**	**21.9**	**76.5**	**54.3**	**30.3**	**56.0**	**54.5**	**54.6**	**36.9**	56.5	**29.7**	**17.6**	**33.1**	**27.1**	**19.8**	**11.2**
								NELL995									
GQE	18.6	12.5	-	32.8	11.9	9.6	27.5	35.2	18.4	14.4	8.5	8.8	-	-	-	-	-
Query2Box	22.9	15.2	-	42.2	14.0	11.2	33.3	44.5	22.4	16.8	11.3	10.3	-	-	-	-	-
BetaE	24.6	14.8	5.9	53.0	13.0	11.4	37.6	47.5	24.1	14.3	12.2	8.5	5.1	7.8	10.0	3.1	3.5
CQD-CO	28.8	20.7	-	60.4	17.8	12.7	39.3	46.6	30.1	22.0	17.3	13.2	-	-	-	-	-
CQD-Beam	28.6	19.8	-	60.4	20.6	11.6	39.3	46.6	25.4	23.9	17.5	12.2	-	-	-	-	-
FuzzQE	27.0	18.4	7.8	47.4	17.2	14.6	39.5	49.2	26.2	20.6	15.3	12.6	7.8	9.8	11.1	4.9	5.5
ConE	27.2	17.6	6.4	53.1	16.1	13.9	40.0	50.8	26.3	17.5	15.3	11.3	5.7	8.1	10.8	3.5	3.9
GNN-QE	28.9	19.6	9.7	53.3	18.9	14.9	42.4	52.5	30.8	18.9	15.9	12.6	9.9	14.6	11.4	6.3	6.3
QTO	32.9	24.0	12.9	60.7	24.1	21.6	42.5	50.6	31.3	26.5	20.4	17.9	13.8	17.9	16.9	9.9	5.9
LARK	52.9	26.9	12.4	87.8	45.7	33.5	51.3	48.7	23.1	22.2	20.6	41.1	9.9	5.9	24.5	13.3	7.3
LACT	**60.1**	**32.0**	**17.2**	**91.4**	**53.6**	**40.6**	**62.2**	**54.9**	**31.4**	**34.8**	**27.0**	**34.0**	**16.0**	**21.2**	**21.0**	**16.3**	**11.6**

Table 1: Test **MRR** results (%) on complex query answering across **all query types**. avg_p is the average on EPFO queries; avg_{ood} is the average on out-of-distribution (OOD) queries; avg_n is the average on queries with negation.

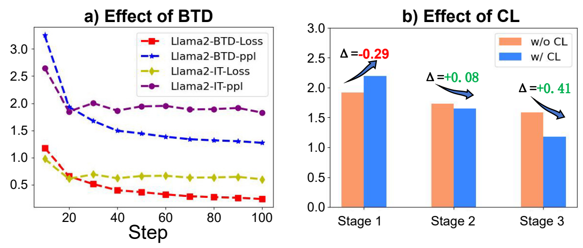


Figure 2: Results of **Ablation Studies.**(a) Comparison **PPL and train loss** results of whether to **use BTD** based on FB15k.(b) Comparison **PPL results** of whether use **CL** based on FB15K high-difficulty queries.

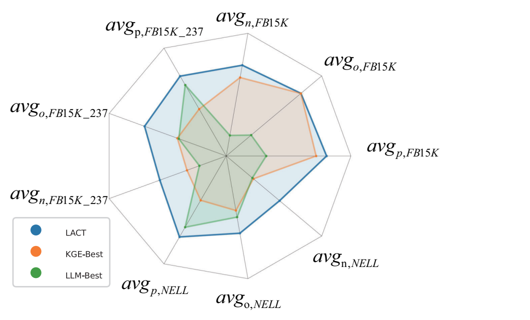


Figure 3: The results of the **main experiment**. We evaluate the performance of **three current state-of-the-art methods** on three datasets.

the previous SOTA method, especially more challenging datasets like FB15K-237 and NELL995. This suggests that our method has better reasoning capability and captures a broad range of relations to effectively utilize this capability for enhancing the performance of complex queries.

5.4 Ablation Studies

To verify the effectiveness of the LACT design, we conduct a two-part ablation study. The first part is designed to verify the effectiveness of logical chain decomposition and the second part is designed to verify the effectiveness of curriculum learning.

Effect of Binary Tree Decomposition (BTD). As shown in Figure 2, logical chain decomposition can stimulate LLM's ability of logical decomposition, thereby greatly improving the performance of difficult queries.

Method	1p	2p	3p	2i	3i	pi	ip	2u	up	2in	3in	inp	pin	pni
Llama2	67.2	42.3	38.3	61.6	44.8	34.1	36.9	44.2	28.4	44.7	38.5	36.9	32.1	30.0
+ IT	94.6	68.8	60.2	84.5	66.7	56.0	60.2	69.5	42.3	66.5	54.4	41.0	38.8	36.9
+BTD	91.5	72.3	65.6	89.7	75.2	60.1	65.4	72.5	49.9	74.3	66.4	48.9	46.5	42.5

Table 2: **Accuracy** results (%) of whether to **use BTD** on *hard* complex query answering across all query types, evaluated on FB15k.

	BTD	CL	1p	2p	3p	2i	3i	pi	ip	2u	up	2in	3in	inp	pin	pni
Llama2 w/IT			94.6	68.8	60.2	84.5	66.7	56.0	60.2	69.5	42.3	66.5	54.4	41.0	38.8	36.9
	✓		94.2	72.3	65.6	89.7	75.2	60.1	65.4	72.5	49.9	74.3	66.4	48.9	46.5	42.5
		✓	94.6	70.9	61.3	84.4	72.7	58.2	63.2	69.5	47.3	68.5	64.0	42.3	40.9	37.1
	✓	✓	94.8	78.7	69.2	94.8	88.1	79.3	80.5	80.7	67.1	90.6	70.4	59.3	53.6	46.7

Table 3: **Accuracy** results (%) of whether to **use CL** on *hard* complex query answering across all query types, evaluated on FB15k.

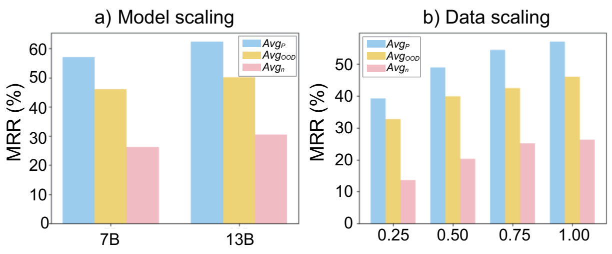


Figure 4: Performance of **scaling** LACT on FB15K-237 with different a) **model** and b) **data scales**.

Metric	pi	ip	2u	up
$\text{Pro}_{decomposed}$	98.7	100.0	97.8	100.0
$\text{Pro}_{true,decomposed}$	98.6	99.9	97.8	99.6

Table 4: In **OOD** queries, the proportion of queries that can be **decomposed** and the proportion of queries that can be **decomposed correctly** on fb15k.

From a training perspective, as shown in Figure 2, although perplexity (PPL) and training loss of decomposed queries before training was slightly higher than that of ordinary queries, we found that as training progresses, the loss and PPL of decomposed queries will quickly decrease to levels much lower than ordinary queries, proving that chain decomposition is effective to reduce the difficulty of learning complex queries.

Effect of Curriculum Learning. Curriculum learning, as illustrated in Table 3, greatly alleviates the gap between difficult training samples and the understanding ability of LLMs.

We can observe from Figure 2 that compared with random shuffle sequence training, difficult training samples under curriculum learning gradually become easier to understand. It is worth mentioning that we found that the gain of curriculum learning on training corpus that has not been decomposed by logical chains is very small, which supports our theory from the side. It is difficult for LLMs to understand the difficulty difference between undecomposed samples, so curriculum learning is also difficult to take effect.

5.5 Transferability Study

Considering the diversity of complex reasoning tasks, we can divide transferability into two levels, task-level and dataset-level transferability.

Task-level transferability. The results in Table 1 show that our method achieves a relative gain of 9.9% on the OOD task, which demonstrates the strong generalization of our fine-tuning framework. Even in the OOD queries, as shown in Table 4, more than 95% of test samples can still follow logical chain reasoning. These phenomena indicate strong generalization ability of LACT.

Dataset-level transferability. In fact, almost all KGE methods, even if some of the optimization methods claim not to require training, require a KGE base model adapted to a specific dataset, which leads to the inherent defect of extremely poor Transferability of the KGE method. However, as previous research has shown, fine-tuning of LLMs is mainly to stimulate the knowledge and capabili-

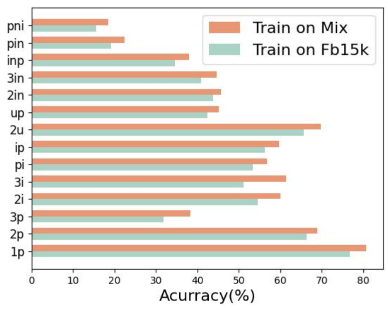


Figure 5: Ablation experimental results of Accuracy (%) **trained on FB15k** and **tested on FB15K-237**, compared to models **trained on all mixed training data**.

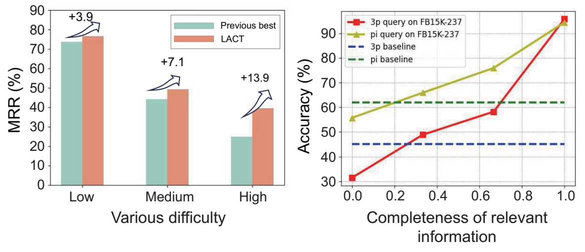
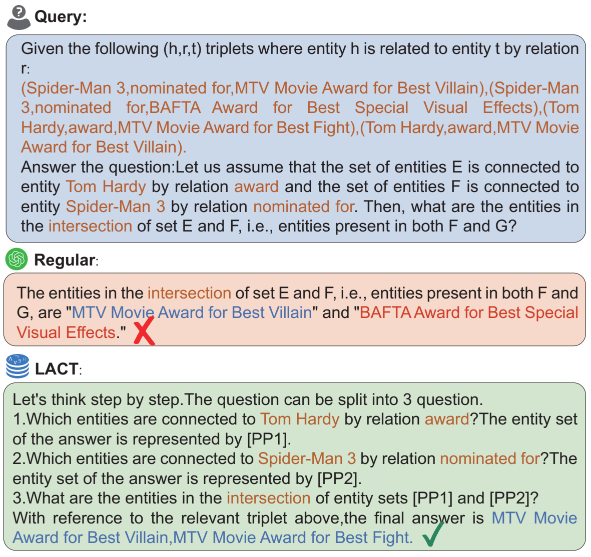


Figure 6: (left) **MRR** performance of LACT and previous SOTA methods at **different difficulties** based on FB15K. (right) The **correlation** between **related information completeness and accuracy** evaluated on FB15K-237, we selected 3p query and pi query with the same inference path length as the task types. We assume that the completeness of all simple queries is 1.



Query:

Given the following (h,r,t) triplets where entity h is related to entity t by relation r:
(Spider-Man 3,nominated for,MTV Movie Award for Best Villain),(Spider-Man 3,nominated for,BAFTA Award for Best Special Visual Effects),(Tom Hardy,award,MTV Movie Award for Best Fight),(Tom Hardy,award,MTV Movie Award for Best Villain).
Answer the question:Let us assume that the set of entities E is connected to entity Tom Hardy by relation award and the set of entities F is connected to entity Spider-Man 3 by relation nominated for. Then, what are the entities in the intersection of set E and F, i.e., entities present in both F and G?

Regular:

The entities in the intersection of set E and F, i.e., entities present in both F and G, are "MTV Movie Award for Best Villain" and "BAFTA Award for Best Special Visual Effects." ✗

LACT:

Let's think step by step.The question can be split into 3 question.
1.Which entities are connected to Tom Hardy by relation award?The entity set of the answer is represented by [PP1].
2.Which entities are connected to Spider-Man 3 by relation nominated for?The entity set of the answer is represented by [PP2].
3.What are the entities in the intersection of entity sets [PP1] and [PP2]?
With reference to the relevant triplet above,the final answer is MTV Movie Award for Best Villain,MTV Movie Award for Best Fight. ✓

Figure 7: **Inference results** of ChatGPT and LACT on 2i query case, respectively.

ties of potentially relevant downstream tasks contained in LLM pre-training. This has also become the theoretical basis for the transferability of fine-tuning methods for LLMs. The results in Figure 5 show that the reasoning ability stimulated by one dataset can still be demonstrated in another dataset, which reflects well in the query performance which only dropped less than 5%.

5.6 Scalability Study

For verifying the scalability of LACT, we scale LACT to different model sizes and data volumes.

Performance on different model size. We tried scaling model size to see if LACT would have an impact when operating on a larger scale. As Figure 4 shows, the performance of our method improves as the model size increases.

Performance on different data size. We conducted experiments on different ratios of training data to verify the robustness of LACT.

6 Discussion

6.1 When and Where Does LACT Work?

The performance of LACT would be related to the following two aspects: *I. The completeness of relevant information extracted from KG. II. Sophistication of complex reasoning.*

LACT performs consistently better with more complete information. We take the form of a posteriori that set the completeness of relevant triplets to the proportion of triplets in the inference path of complex reasoning in the provided context, and set the completeness of simple queries that can be directly inferred to 1, to obtain the relation between Accuracy and correlation information completeness. As seen in Figure 6, LACT obtains a significant gain when the completeness of relevant information increased, though, with zero relevant information, it remains a certain amount of complex reasoning ability.

LACT performs consistently better on higher difficulties As mentioned before, we simply divide the difficulty of the query by the number of hops in the query. The results in Figure 6 show that our model yields more gain in tasks of higher-level difficulty and complexity, which benefits from our unique and sophisticated fine-tuning framework.

6.2 Case Study

To have a close look, we perform the case studies by analyzing the results of LACT and ChatGPT (GPT-3.5-turbo-0613). As shown in Figure 7, ChatGPT cannot make good use of incomplete knowledge graphs for reasoning in some cases. Conversely, LACT performs reasoning through a complete logical chain, making maximum use of the relevant information provided and deducing the correct answer, which greatly improves the reasoning ability.

7 Conclusion

In this paper, we present a simple and effective fine-tuning framework LACT to boost the complex logical reasoning ability over KGs. LACT is a two-stage method that consists of both Binary Tree Decomposition and Curriculum Learning and can be applied to various size LLMs with different data sizes. We empirically demonstrate the effectiveness and universality of the LACT on a series of widely-used knowledge graph datasets. Further analyses reveal the underlying mechanism of our method, and investigate When and Why Does LACT Work. We hope that our work can inspire more research on combining knowledge graphs and LLMs.

Limitations

Our work has several potential limitations. Although our work revealed the enhanced performance of LACT on complex logical reasoning, we do not focus much on the time cost of our method, which depends on inference speed for LLMs. It is meaningful to explore more efficient inference strategies to accelerate text generation, which is in our future work. Additionally, given the limited computational budget, we only fine-tune our model with the size of 7B and 13B, so there is a need for further research into introducing models of larger scales like 70B.

Ethics Statement

We take ethical considerations very seriously. This paper focuses on improving complex reasoning over incomplete knowledge graphs with logic-aware curriculum tuning. All experiments are conducted on publicly available datasets and models, and the findings and conclusions of this paper are reported accurately and objectively. Thus, we believe that this research will not pose ethical issues.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2020. Complex query answering with neural link predictors. In *International Conference on Learning Representations*.

Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. 2023. Answering complex logical queries on knowledge graphs via query computation tree optimization. In *International Conference on Machine Learning*.

Yushi Bai, Zhitao Ying, Hongyu Ren, and Jure Leskovec. 2021. Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. In *Advances in Neural Information Processing Systems*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase. In *ACM SIGMOD International Conference on Management of Data*.

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

Xuelu Chen, Ziniu Hu, and Yizhou Sun. 2022. Fuzzy logic based logical query answering on knowledge graphs. In *the AAAI Conference on Artificial Intelligence*.

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

Nurendra Choudhary and Chandan K Reddy. 2023. Complex logical reasoning over knowledge graphs using large language models. *arXiv preprint arXiv:2305.01157*.

Nilesh Dalvi and Dan Suciu. 2007. Efficient query evaluation on probabilistic databases. *The VLDB Journal*.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

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

Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. *arXiv preprint arXiv:2309.01538*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Keqin Peng, Liang Ding, Qihuang Zhong, Li Shen, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2023. Towards making the most of ChatGPT for machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2019. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations*.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Advances in Neural Information Processing Systems*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *International Conference on World Wide Web*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata. *Communications of the ACM*.

Haochun Wang, Sendong Zhao, Zewen Qiang, Zijian Li, Nuwa Xi, Yanrui Du, MuZhen Cai, Haoqiang Guo, Yuhan Chen, Haoming Xu, et al. 2023a. Knowledge-tuning large language models with structured medical knowledge bases for reliable response generation in chinese. *arXiv preprint arXiv:2309.04175*.

Siyuan Wang, Zhongyu Wei, Meng Han, Zhihao Fan, Haijun Shan, Qi Zhang, and Xuanjing Huang. 2023b. Query structure modeling for inductive logical reasoning over knowledge graphs. In *Annual Meeting of the Association for Computational Linguistics*.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023c. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Rui Zhang, Yixin Su, Bayu Distiawan Trisedya, Xiaoyan Zhao, Min Yang, Hong Cheng, and Jianzhong Qi. 2023a. Autoalign: Fully automatic and effective knowledge graph alignment enabled by large language models. *IEEE Transactions on Knowledge and Data Engineering*.

Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. 2023b. Making large language models perform better in knowledge graph completion. *arXiv preprint arXiv:2310.06671*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023c. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198*.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *arXiv preprint arXiv:2305.13168*.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. 2022. Neural-symbolic models for logical queries on knowledge graphs. In *International Conference on Machine Learning*.

Appendix

A Neighborhood Retrieval Algorithm

To strike a balance between the completeness of relevant information and the token number limit of LLMs, we search for as many relevant triplets as possible along the possible paths.

Particularly, for the 1p query, we simply find all the triplets containing the entity or the relation.

For another query, as shown in Figure S1 for each leaf node in DAG, we do depth traversal on the graph. For each step in the traversal process, if this step is a projection, we search for all the possible triplets. Otherwise, we perform corresponding operations on intersection and union respectively to filter out the corresponding entities.

We continue this traversal until the obtained entity is empty or reaches the root node. All triplets during the traversal are related to triplets.

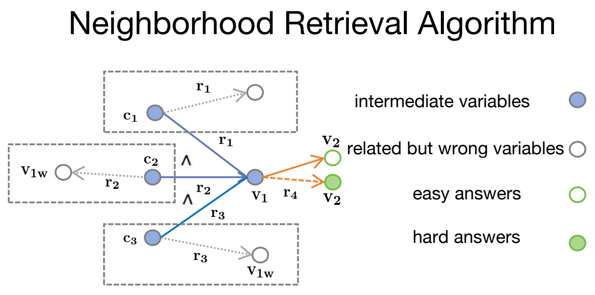


Figure S1: A case on Neighborhood Retrieval Algorithm

B Conversion Between FOL Expression and Query Computation Tree

The conversion from an FOL expression (disjunctive normal form) to its query computation tree involves two steps: dependency graph generation, and union branches duplication.

Dependency Graph Generation. Upon encountering a First-Order Logic (FOL) expression, our primary procedure entails the allocation of distinct nodes to individual variables, while assigning a unique node to the constant entity within each one-hop atom. It is important to acknowledge that multiple nodes may represent the same constant entity, given its occurrence in various one-hop atoms. Subsequently, undirected edges are employed to establish connections between nodes in accordance with the defined one-hop atoms. Specifically, if $e_j^i = r(v', v)$ (or $r(c, v)$), then we connect the nodes of v' (or c) and v by an edge r_i. Similarly, if $e_j^i = \neg r(v', v)$ (or $\neg r(c, v)$), then we connect the nodes of v' (or c) and v by an edge $\neg r_i$. The variable i serves as a distinguishing label for edges emanating from distinct conjunctions. The formulated undirected dependency multigraph must conform to a tree structure, signifying a connected and acyclic graph. Choosing the node $v_?$ as the root, we establish edge directions ensuring that they uniformly point from child nodes to their respective parent nodes, with due consideration to handling inverse relations. Notably, constant entities inherently function as leaf nodes within the tree, given that each entity node exclusively connects to a single variable node.

Union Branches Duplication Then we handle the duplication branches in the query computation tree. On the path τ from root to every leaf node, if exists, we find the first node v_i such that the edges between v_i and its child node v_j are all of the same relations, but in different conjunctions: $r_{t_1}, r_{t_2}, \ldots, r_{t_p}$. We merge these edges into a single edge $r_{t_1,t_2,\ldots t_p}$, since they all correspond to the same one-hop atom but in different conjunctions, they can be merged by the distributive law:

$$(A \wedge B) \vee (A \wedge C) \Leftrightarrow A \wedge (B \vee C) \qquad (7)$$

We assert that there is a subpath from v_i to some v_k within the path τ that only consists of edges $r_{t_1,t_2,\ldots t_p}$, and v_k is connected to different child nodes by relations from conjunctions $t_1, t_2, \ldots t_p$. These edges are annotated as unions, while the remaining one-to-many structures are designated as intersections. Consequently, the multigraph transforms into a simple graph, devoid of multiple edges.

C Difficulty

We divide the difficulty by the number of decomposed subqueries. Query types and their corresponding difficulties are shown in Table S1.

D Experiment Details

D.1 Dataset Details

- **FB15K** is based on Freebase, a large collaborative knowledge graph project that was created by Google. FB15k contains about 15,000 entities, 1,345 relations, and 592,213 triplets (statements that assert a fact about an entity).
- **FB15K-237** is a subset of FB15k, containing 14,541 entities, 237 relations, and 310,116 triplets.

	1p	2p	3p	2i	3i	pi	ip	2u	up	2in	3in	inp	pin	pni
Number of subqueries	1	2	3	3	5	4	4	3	4	3	5	4	4	4
Difficulty	1	1	2	2	3	3	3	2	3	2	3	3	3	3

Table S1: Difficulty of different query types where 1 means easy, 2 means medium, 3 means hard.

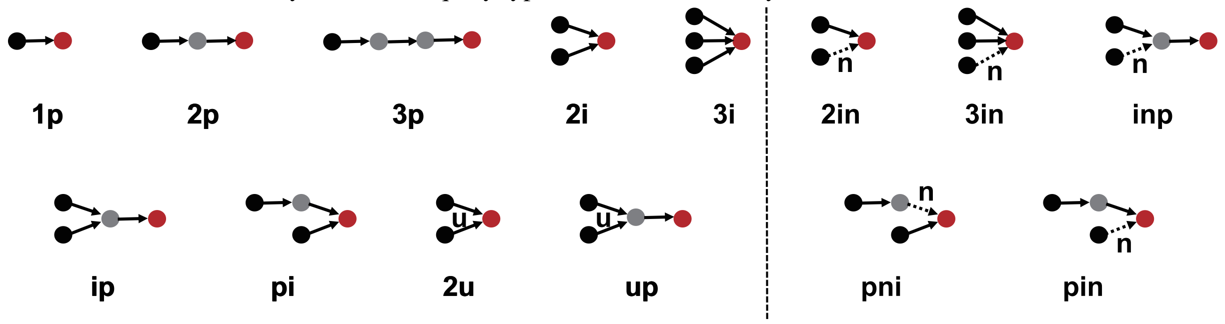


Figure S2: Query structures, illustrated in their query computation graph representations

The relations in FB15k-237 are a subset of the relations in FB15k, and were created to address some of the limitations of FB15k, such as the presence of many irrelevant or ambiguous relations, and to provide a more challenging benchmark for knowledge graph completion models.

• **NELL995** was created using the Never-Ending Language Learning (NELL) system, which is a machine learning system that automatically extracts knowledge from the web by reading text and inferring new facts. NELL995 contains 9,959 entities, 200 relations, and 114,934 triplets. The relations in NELL995 cover a wide range of domains, including geography, sports, and politics.

D.2 Query Structure

For a fair comparison, we use the 14 types of complex queries generated by the same rules in (Ren and Leskovec, 2020). The query structure of each type is shown in Figure S2.

For each complex query, its answers are divided into easy answers and hard answers, based on whether the answer can be derived by existing edges in the graph directly. Specifically, in the valid/test set, the easy answers are the entities that can be inferred by edges in the training/valid graph, while hard answers are those that need to be inferred by predicting missing edges in the valid/test graph. Referring to previous work, we calculate standard evaluation metrics including mean reciprocal rank (MRR), in the filtered setting where all easy and hard answers are filtered out during ranking.

D.3 Evaluation Protocol Detail

For each complex query, its answers are divided into easy answers and hard answers, based on whether the answer can be derived by existing edges in the graph directly. Specifically, in the valid/test set, the easy answers are the entities that can be inferred by edges in the training/valid graph, while hard answers are those that need to be inferred by predicting missing edges in the valid/test graph. Referring to previous work (Ren and Leskovec, 2020), we calculate standard evaluation metrics including mean reciprocal rank (MRR), in the filtered setting where all easy and hard answers are filtered out during ranking.