Efficient and Scalable Neural-Symbolic Search for Complex Query Answering over Incomplete Knowledge Graphs

Anonymous ACL submission

Abstract

Complex Query Answering (CQA) is a crucial first-order logical reasoning task on Knowledge Graphs (KGs). While neural-symbolic methods leveraging neural link predictors achieve strong performance, they face significant complexity bottlenecks: quadratic data complexity scaling with the number of entities, and NP-hard query complexity for cyclic queries. Consequently, these approaches struggle to scale effectively to large knowledge graphs and complex queries. To address these limitations, we propose an efficient and scalable symbolic search framework comprising two key components: (1) constraint strategies that drastically reduce the variable search domain, lowering data complexity; and (2) a local search algorithm that approximately solves the NP-hard query complexity for cyclic queries. Experiments on various CQA benchmarks demonstrate that our framework achieves 97% accuracy with a 10× speedup using only 10% of the search space for tree-form queries. Furthermore, it demonstrates robust performance on complex cyclic queries and large-scale KGs, effectively alleviating efficiency and scalability challenges.

1 Introduction

Complex Query Answering (CQA) (Wang et al., 2022; Ren et al., 2023) is proposed to answer complex logical queries from incomplete Knowledge Graphs (KGs). Given the observed KG, this task reasons the unobserved answers to first-order logic queries (Ren and Leskovec, 2020) by leveraging the generalization ability of Machine Learning (ML). An illustrative example for this task is provided in Figure 1. Traditional graph traversal algorithms lack generalization, often overlooking implicit knowledge that can be predicted from the observed knowledge (Hamilton et al., 2018). Meanwhile, large language models face challenges with hallucination (Rawte et al., 2023), struggling with precisely memorizing and reasoning factual knowledge, especially domain knowledge (Zheng et al., 2024). To address the above challenges, CQA is the crucial knowledge representation and reasoning framework, effectively combining faithfulness and generalization. Consequently, it enables various practical applications, including product recommendations and query-answering systems (Bai et al., 2024; Ren et al., 2023).

Table 1: The complexity and performance (MRR) of different neural symbolic search methods, where $|\mathcal{E}|$ is the size of entities within KG and n is query size. The symbol * indicates tree decomposition (approximation) for cycles, and κ represents our reduced factor.

	Complexity		Performance	
Query category	Tree	Cyclic	Tree	Cyclic
QTO (Bai et al., 2023b)	$\mathcal{O}(n\|\mathcal{E}\|^2)$	$\mathcal{O}(n\|\mathcal{E}\|^2)$*	37.1	36.8
FIT (Yin et al., 2024)	$\mathcal{O}(n\|\mathcal{E}\|^2)$	$\mathcal{O}(\|\mathcal{E}\|^n)$	37.1	42.0
NLISA (our)	$\mathcal{O}(n(\frac{\|\mathcal{E}\|}{\kappa})^2)$	$\mathcal{O}(n(\frac{\|\mathcal{E}\|}{\kappa})^2)$	37.3	40.5

With great advancement, numerous ML models have been proposed to extend CQA (Ren et al., 2023). Neural-symbolic search methods (Amayuelas et al., 2021; Yin et al., 2024) have emerged as a promising paradigm, typically supporting whole existential first-order queries while achieving state-of-the-art (SoTA) performance. However, their complexity poses significant scalability issues.

We provide the detailed complexity and performance of main search methods in Table 1. For simple tree-form queries, the complexity of QTO and FIT is $\mathcal{O}(n|\mathcal{E}|^2)$. From the perspective of data complexity, this complexity grows quadratically with the size of the entities, making it challenging to scale to large-scale KGs (Ren et al., 2022). For cyclic queries, the complexity of FIT is even $\mathcal{O}(|\mathcal{E}|^n)$. From the perspective of query complexity, this complexity is NP-hard and increases exponentially with query size. This complexity bottleneck limits the efficiency in handling complex cyclic queries (Yin et al., 2024). As shown in Table 1, we propose an efficient framework, NLISA, to improve both scalability and efficiency.

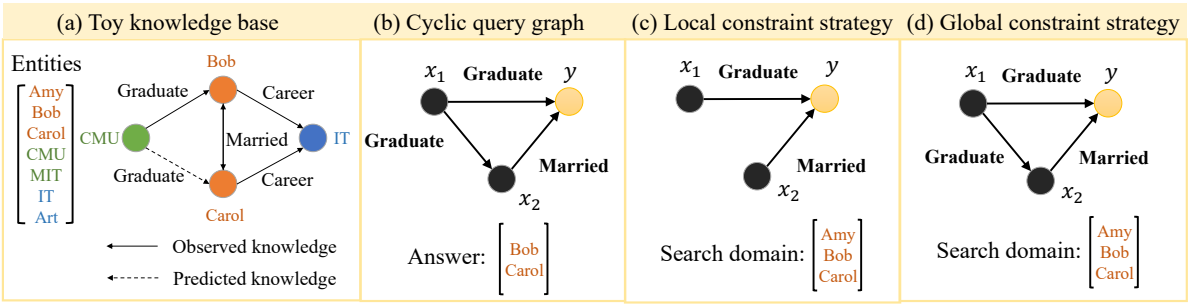

Figure 1: **(a)** One toy knowledge graph recording a simple social network. **(b)** A simple cyclic query, with formula $\exists x_1, x_2, \text{Graduate}(x_1, x_2) \wedge \text{Graduate}(x_1, y) \wedge \text{Married}(x_2, y)$. This can be interpreted as "Find a person who graduated from the same institution (s) as his/her spouse". Since the answers $\{\text{Bob}, \text{Carol}\}$ must be inferred with predicted edges. To predict the potential answers, CQA models must consider all entities within the given KG. **(c)** and **(d)** present the local constraints strategy and the global constraints strategy for free variable y, respectively. We then utilize the search domain instead of considering all entities, which accelerates the search process.

Our first technical contribution is the proposed Neural Logical Indices (NLI) to narrow the search domain for variables involved in logical queries, thereby reducing the data complexity. This technique is motivated by well-studied Constraint Satisfaction Problems (CSPs) (Chen et al., 2011; Gottlob et al., 2000), and the sparsity nature observed in KGs (Pujara et al., 2017). Under mild assumptions, we can prove that correct answers will be fully retained by a very high probability. Furthermore, we provide two strategies to flexibly compute NLI, including the **local constraints** involving neighborhood constraints and **global constraints** strategy using global constraints. Further details are presented in Section 4.

The second technical contribution is an approximate search algorithm to address the **NP-hard** cyclic query in quadratic complexity $\mathcal{O}(n|\mathcal{E}|^2)$. Our approximate solution autoregressively searches the most likely assignments for variables within cyclic and employs local search to avoid exponential growth of the search space. Notably, this algorithm supports parallel execution, significantly enhancing efficiency. Detailed introduction is shown in Section 5.

To validate the effectiveness of our framework, we conduct comprehensive experiments on classic CQA benchmarks. Our experimental results reveal three key insights. (1) The search space for variables can be significantly reduced by 90%, achieving 97% accuracy with a 10× speedup. (2) For cyclic queries, our framework achieves 50× speedup while maintaining 95% accuracy. (3) We demonstrate that our framework can be extended to KG with 400,000 entities, highlighting the scalability of our method.

2 Related work

CQA over incomplete KGs gradually grows by extending the scope of logical queries, ranging from conjunctive queries (Hamilton et al., 2018) to real existential first-order queries (Yin et al., 2024). There are primarily two lines of research in CQA. One is **query embedding methods**, which model the query using representations including vector, box, and beta distribution of low-dimensional space (Hamilton et al., 2018; Ren et al., 2020; Ren and Leskovec, 2020). In this approach, logical operations are transformed into set operations within an operator tree (Wang et al., 2021; Ren et al., 2023), modeled by neural networks in alignment with their semantics. Although the representational capabilities have been thoroughly explored (Zhang et al., 2021; Choudhary et al., 2021), current query embedding methods still face limitations in both performance and expressiveness (Yin et al., 2024). The second line is **neural-symbolic search** methods (Arakelyan et al., 2020; Bai et al., 2023b; Yin et al., 2024), which use knowledge graph embedding methods (Bordes et al., 2013) as the backbone to predict missing facts and model logical operations using fuzzy logic (Klir and Yuan, 1995). In particular, QTO (Bai et al., 2023b) and FIT (Yin et al., 2024) store the $\mathcal{O}(|\mathcal{R}||\mathcal{E}|^2)$ completed matrices and then execute the symbolic search based on fuzzy logic. The search of QTO is a precise for tree-form queries, then FIT (Yin et al., 2024) extends it to general existential first-order queries by enumeration to handle cyclic queries, where the complexity is $|\mathcal{E}|^n$ (Yin et al., 2024). NLISA, our proposed framework, reduces both space complexity and time complexity, improving the scalability of symbolic search methods.

3 Background

3.1 Knowledge graph and logical queries

We represent KG as a first-order logic knowledge base, and subsequently define the logical query.

Definition 3.1 (Knowledge graph) *Let $\mathcal{E}$ and $\mathcal{R}$ be the finite set of entities and relations, a knowledge graph is a collection of factual triples $\mathcal{G} = \{(s_i, r_i, o_i)\}$, where s_i is entity subject, o_i is entity object, and r_i is a relation predicate.*

Definition 3.2 (EFO1 query) *The existential first-order logic query is defined as:*

$$\psi(y; x_1, \cdots, x_m) = \exists x_1, \cdots x_m, \\ (c_1^1 \wedge \cdots \wedge c_{n_1}^1) \vee \cdots \vee (c_1^k \wedge \cdots \wedge c_{n_k}^k), \quad (1)$$

where c_j^i is atomic formula $r(h, t)$ or its negation $\neg r(h, t)$, $r \in \mathcal{R}$ is a predicate, h and t are entities belonging to $\mathcal{E}$ or a variable ranging from $\mathcal{E}$.

Definition 3.3 (Answer set) *Given a query $\psi(y)$, its answer set is defined by*

$$\mathcal{A}[\psi(y)] = \{s \in \mathcal{E} | \psi|_{y=s} = \text{TRUE}\}, \quad (2)$$

where $\psi|_{y=s}$ represents substituting the occurrence of variable y within ψ to s simultaneously.

With the Disjunctive Normal Form (DNF), we can primarily consider the conjunctive query ϕ.

3.2 Neural-symbolic search

To address the incompleteness, Knowledge Graph Embedding (KGE) models (Sun et al., 2018) were proposed to predict the truth of the unseen facts. With fuzzy logic (Mendel, 1995) (see the Appendix F to generalize logical operations, we then introduce the truth value function $T(\cdot)$ to infer formulas as follows:

Definition 3.4 (Truth value function) *Let ϕ and ψ be existential formulas, $\top_P$ and $\bot_P$ are product t-norms[1] and t-conorms, and $r \in \mathcal{R}$, $s, o \in \mathcal{E}$, with $P_r(s, o)$ representing the truth value of atomic formula $r(s, o)$. The truth value function T, whose range is $[0, 1]$, is defined as follows:*

(i) $T(r(s, o)) = P_r(s, o)$;
(ii) $T(\neg\phi) = 1 - T(\phi)$;
(iii) $T(\phi \wedge \psi) = T(\phi) \top_P T(\psi)$, *where* $a \top_P b = a \times b$;
(iv) $T(\phi \vee \psi) = T(\phi) \bot_P T(\psi)$, *where* $a \bot_P b = 1 - (1 - a) \top_P (1 - b)$;
(v) $T(\exists x, \phi(x)) = \max_{s \in \mathcal{E}} T(\phi|_{x=s})$.

[1]We default to product t-norm unless otherwise stated.

Then we can rewrite the logical query answering as the following combinatorial optimization:

$$T(\phi|_{y=s}) = max \quad T(\exists x_i, c_1|_{y=s} \wedge \cdots \wedge c_n|_{y=s}) \\ \text{s.t.} \quad x_i \in \mathcal{E}, 1 \leq i \leq m. \quad (3)$$

Neural-symbolic search methods solve this problem by continually pruning the query graph.

Definition 3.5 (Query graph) *The query graph of ϕ is defined as a set of quadruples, $G_\phi = \{(h_i, r_i, t_i, \text{NEG}_i)\}$. The h and t are nodes, where we refer to entity as constant node and to variable as variable node. The edge is defined by a quadruple, along with two attributes: r, which denotes the relation, and NEG_i, which is the boolean variable indicating whether the atom is positive.*

Definition 3.6 (Cyclic and tree-form query) *Cyclic queries have variable nodes that form cycles within the query graph, whereas tree-form queries are characterized by a query graph structured as a tree.*

Definition 3.7 (Fuzzy vector) *Given the domain $\mathcal{D}$ of variable and a membership function $\mu : \mathcal{D} \to [0, 1]$, the fuzzy vector of $\mathcal{D}$ is defined with $D_i = \mu(s, \mathcal{D})$, where $s \in \mathcal{D}$.*

Proposition 3.8 (RemoveConstNode) *The constant node in G_ϕ can be removed in $\mathcal{O}(|\mathcal{E}|)$.*

Proposition 3.9 (RemoveLeafNode) *A leaf node is a variable node that connects to only one other variable node. The leaf node in G_ϕ can be removed in $\mathcal{O}(|\mathcal{E}|^2)$.*

These search methods assign variables to fuzzy vectors and utilize proposition 3.8 and proposition 3.9 to simplify constraints while updating fuzzy vectors. During execution, all constant nodes are removed first, followed by removal of leaf nodes gradually. Then acyclic queries can be effectively solved in $\mathcal{O}(n|\mathcal{E}|^2)$, where n is query size. However, for cyclic queries, FIT enumerates one remaining variable, leading to exponential $\mathcal{O}(|\mathcal{E}|^n)$.

4 Neural logical indices

As shown in Figure 1, it is unnecessary to use whole entity set $\mathcal{E}$ for symbolic search methods (Bai et al., 2023b; Yin et al., 2024). For the logical query $\exists x_1, x_2, \text{Graduate}(x_1, x_2) \wedge \text{Graduate}(x_1, y) \wedge \text{Married}(x_2, y)$, it is evident that y must be both the tail of the "Graduate" relation

and "Married" relation . Searching the variable y within $\{\text{Amy}, \text{Bob}, \text{Carol}\}$ is equivalent to searching the entire space $\mathcal{E}$. We denote the reduced search space as domain $\mathcal{D}$ and utilize the CUTDOMAIN function to define the process to obtain the reduced domain, thus effectively simplifying optimization in Equation 3 as follows:

$$T(\phi|_{y=s}) = \max \quad T(\exists x_i, c_1|_{y=s} \wedge \cdots \wedge c_n|_{y=s}) \\ \text{s.t. } x_i \in \mathcal{D}_i, 1 \leq i \leq m, \tag{4}$$

where $\mathcal{D}_i$ is the reduced domain for the variable x_i.

4.1 Motivation and definition

Complex query answering can be formally reduced to Constraint Satisfaction Problems (CSPs), where each atomic formula corresponds to a constraint (Yin et al., 2024). Domain pruning is a classical technique in CSPs that utilizes consistency with surrounding constraints (Chen et al., 2011). In particular, connectivity in KGs is generally sparse. As shown in Appendix C, the size distribution of domains follows a power law, characterized by a high frequency of small domains. Therefore, we can develop ML methods to compute the pruned domain in advance to accelerate symbolic search.

Definition 4.1 (Neural Logical Indices (NLI)) *Given the variable set $\mathcal{X} = \{x_1, \cdots, x_m, y\}$ in a query ϕ, neural logical indices are defined as a mapping from a variable in $\mathcal{X}$ to a subset of the entity set $\mathcal{E}$: $\mathcal{D}(x) : \mathcal{X} \to 2^{\mathcal{E}}$.*

We denote $\mathcal{D}_x$ and $\mathcal{D}_y$ as the domains of x and y, respectively. To compute the $\mathcal{D}_x$ for each variable, we conceptually define the constraints as the subgraph pattern G_x^S from the query graph G_ϕ. Then we propose the strategy to apply neural embedding models $h(\cdot)$ to find the entities that satisfy the constraints. Let $h(G_x^S)$ denote the ranking of the entities, we select the top k entities from this ranking and determine $\mathcal{D}(x) = \textbf{Topk}(h(G_x^S), k)$, which serves as our reduced search domain $\mathcal{D}_x$ accelerating symbolic search algorithms.

4.2 Local constraints strategy

The local constraints strategy only involves the constraints within the following neighbor subgraph:

Definition 4.2 (Neighbor subgraph) *Let G_ϕ be the query graph. For variable x in G_ϕ, the edges of its neighbor subgraph are $\mathcal{N}_e(x, G_\phi) = \{(h_i, r_i, t_i, \text{NEG}_i) \in G_\phi | h_i = x \text{ or } t_i = x\}$, formed from the relation connected with x_i. The $\mathcal{N}_n(x, G_\phi)$ is the corresponding node set.*

We only consider relations within the neighbor subgraph for ensuring generality. To utilize the relations to prune the search domain, we propose a new task, predicting the tail sets given a relation r in KG $\mathcal{G}$. [2] This task is similar to predicting missing facts of KG. To improve the generalization and training efficiency, we propose a simple and effective Hypernet (Ha et al., 2016) to adapt the KG embedding models (Trouillon et al., 2016; Chen et al., 2021) to address this new task. Then the likelihood of relation tail (r, o) can be computed by $g_s(\texttt{HN}(r), \texttt{HN}(o))$, where HN is the hyper-network and g_s is the adapted scoring function. The details of this task can refer to Appendix E.

Then we can employ fuzzy logic to combine the constraints to compute the final ranking that sorts the candidate entities. For a possible entity e_o in the Figure 1, we employ the T-norm to calculate the scores as follows:

$$g_s(\texttt{HN}(\text{Married}), \texttt{HN}(e_o)) \top g_s(\texttt{HN}(\text{Graduate}), \texttt{HN}(e_o)).$$

4.3 Global constraints strategy

The global constraints extend the constraints to include the entire query graph, which means that $G_y^S = G_\phi$ for the free variable y. Although problem formulation has become more complex, we can leverage the power of query embedding to address the problem directly. For further details, see Appendix E. Following this, the two-stage coarse-to-fine ranking process is implemented, similar to the coarse-to-fine ranking used in information retrieval (Liu et al., 2019).

5 Efficient search framework

In this section, we introduce the Neural Logical Indices for Search Approximately (NLISA) framework. Our framework first computes the NLI and uses them to improve the efficiency of the node removal operation in previous methods (Bai et al., 2023a; Yin et al., 2024). Additionally, we propose a parallel local optimization operation with quadratic complexity to replace the enumeration for handling cyclic queries. We provide a visualization in Figure 2 that presents the step-by-step process of NLISA inferring a cyclic query.

5.1 Search with neural logical indices

With NLI to reduce the search domain, we can reduce the complexity of two node removal opera-

[2] Predicting the head of relation r_+ can be modeled as the tail of reverse relation r_-.

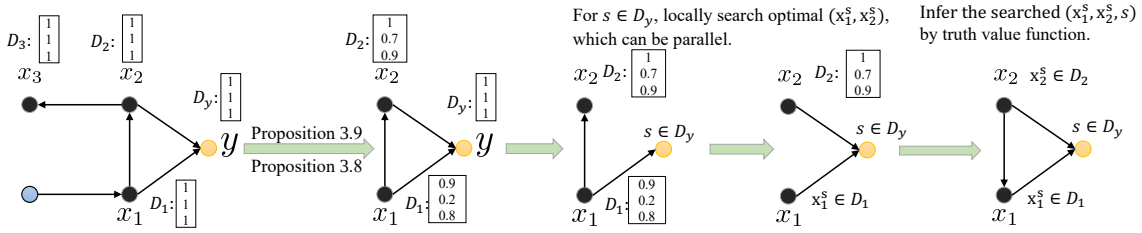

Figure 2: The execution steps of NLISA for inferring a cyclic query.

tions. The reduced complexity is $\mathcal{O}(|\mathcal{D}_x| + |\mathcal{D}_y|)$ and $\mathcal{O}(|\mathcal{D}_x|^2 + |\mathcal{D}_y| \times |\mathcal{D}_y|)$ for the constant node and leaf node, respectively. Notably, our improvement even exhibits quadratic growth about $|\mathcal{E}|/|\mathcal{D}|$ for leaf variable nodes. The implementations of REMOVECONSTNODE and REMOVELEAFNODE functions can refer to Appendix H.

5.2 Approximate search for cyclic queries

For cyclic queries, after removing the constant and leaf nodes, at least one loop will remain. FIT performs enumeration to break this loop (Yin et al., 2024), which is exponential in the query size. Instead, we perform an approximate local search to construct variable assignments for each candidate of free variable $s \in \mathcal{D}_y$. To avoid exploring the entire space, we restrict each update to the neighboring subgraph of the corresponding variable.

Without loss of generality, let remaining nodes $\mathcal{X}_r = \{x_1, \cdots, x_m\}$, ordered by their shortest-path distance to y (from near to far). For each $s \in \mathcal{D}_y$, we autoregressively build an assignment $\{x_1^s, \cdots, x_m^s\}$ over $\mathcal{X}_r$. At step i, we define the partially instantiated query $\phi^{(i)} = \phi(s; x_1^s \cdots, x_{i-1}^s, x_i, \cdots, x_m)$ containing $i-1$ assigned variables and denote the current query graph as $G_{\phi^{(i)}}$. We update x_i by maximizing a local objective composed of edge constraints and existing domain memberships of x_i:

$$x_i^s = \arg\max_{x_i} [\top_{e_i \in \mathcal{N}_e^{(i)}} T(e_i)] \top [\top_{x \in \mathcal{N}_n^{(i)}} \mu(x, C_{x_i})],$$

where C_{x_i} is the fuzzy vector of x_i, $\mathcal{N}^{(i)}$ denote the neighbor subgraph of x_i within $G_{\phi^{(i)}}$, $\mathcal{N}_e^{(i)}$ and $\mathcal{N}_n^{(i)}$ refers the edges and nodes of $\mathcal{N}^{(i)}$. We terminate after $|\mathcal{X}_r|$ times local update, where all variables in the cycle have been assigned. The truth value of $\phi|_{y=s}$ is aggregated over $\phi^{(|\mathcal{X}_r|+1)}$ by t-norms. Thus the complexity of the local search at each step is $\mathcal{O}(|\mathcal{D}_x|^2 + |\mathcal{D}_x| \times |\mathcal{D}_y|)$, which is still quadratic and can be processed in parallel. Realization of the above approximate search induces a function LOCALOPTIMIZE. We present an example of LOCALOPTIMIZE in Figure 2.

5.3 Complexity analysis

The space complexity of our method is $\mathcal{O}((|\mathcal{E}| + |\mathcal{R}| + d_h) \times d)$, where d is the embedding dimension of the KGE models and d_h is the hidden dimension of hyper-network. This complexity is on the same order as standard KGE models, which scales linearly with the sizes of entities and relations.

Inferring queries with n atom formulas, the time complexity is $\mathcal{O}((|\mathcal{D}_x||\mathcal{D}_y| + |\mathcal{D}_x|^2 + |\mathcal{D}_x| + |\mathcal{D}_y|)nd)$, where $|\mathcal{D}_x|$ and $|\mathcal{D}_y|$ are the size of search domain of the existential and free variables, respectively. Our algorithm **exhibits quadratic complexity for any EFO1 query** and can flexibly reduce the complexity.

5.4 Theoretical guarantee

Under the reasonable assumption, we provide the theoretical guarantee of our reduced domain, where detailed discussion can refer to Appendix C.

Assumption 5.1 *Let $\mathcal{E}$ be the entity set of given KG. For the domain of any variable $D_x \subseteq \mathcal{E}$, assume $|D_x|$ follows a Zipf distribution with $\alpha > 1$:*

$$P(|D_x| = r) = \frac{1}{Z_\alpha} r^{-\alpha}, Z_\alpha = \sum_{k=1}^{|\mathcal{E}|} k^{-\alpha}.$$

Statistical tests were conducted on three KGs to validate the hypothesis. For simplicity, we assume our strategy perfectly select the top τ entities.

Theorem 5.2 *When selecting the top-τ entities $D_\tau = \{o_1, \ldots, o_\tau\}$, the successful preservation probability of one variable satisfies:*

$$P_{\mathcal{A}} = P(D_x \subset D_\tau) = \sum_{r=1}^{\tau} \frac{1}{Z_\alpha} r^{-\alpha} \geq 1 - \frac{1}{Z_\alpha} \frac{\tau^{1-\alpha}}{\alpha - 1}.$$

This lower bound $1 - \Theta(\tau^{1-\alpha})$ even ensures $P_{\mathcal{A}} \approx 1$ when $\tau = |\mathcal{E}|/\kappa$ ($\kappa \geq 10$).

Theorem 5.3 *For a complex query with m variables $x_1, \ldots, x_m$. These variables have independent and identically distributed domain sizes and pruning strategies, the probability for retaining all answers is given by:*

$$P\left(\wedge_{j=1}^{m} (D_x^j \subseteq D_m^j)\right) \geq \left(1 - \frac{1}{Z_\alpha} \frac{\tau^{1-\alpha}}{\alpha - 1}\right)^m.$$

6 Experimental settings

6.1 Benchmarks

The BetaE benchmark (Ren and Leskovec, 2020) is the standard CQA benchmark, which primarily contains tree-form queries. The benchmark is comprised of three KGs: FB15K (Bordes et al., 2013), FB15K-237 (Toutanova et al., 2015), and NELL995 (Xiong et al., 2022). Specifically, the BetaE benchmark contains 14 distinct query types, with 5 types involving negation operations.

The real EFO1 benchmark (Yin et al., 2024) proposes 10 new query types beyond the tree-form queries, with the same KGs as BetaE. In particular, the real EFO1 benchmark introduces new patterns, including multi-graph, and cyclic graphs. The visualization of these query types of BetaE and Real EFO1 benchmark is presented in Appendix B.

The Smore benchmark (Ren et al., 2022) considers the same tree-form queries as BetaE benchmark, but the queries are sampled from a much larger-scale KG. Since only the FB400K dataset with 400,000 entities has been released, we select this dataset as the large-scale benchmark.

6.2 Evaluation protocol

To evaluate the answering effectiveness over incomplete (KGs), we adopt the evaluation scheme from Ren and Leskovec (2020), distinguishing the answers to each query into easy and hard sets. "Hard answers" are defined as non-trivial answers that require predicting at least one missing link in the test and validation splits. Based on the rank r for each hard answer against non-answers, we compute the Mean Reciprocal Rank (MRR) and HIT@k (Hamilton et al., 2018) based on the obtained ranks, where the detailed evaluation can refers to Appendix I.

To evaluate efficiency, we report the averaged Queries Per Second (QPS). To ensure fair comparison, we also select the top 100 queries for each query types, which reduces the bias caused by distributions of query types.

6.3 Baselines

We consider representative and strong CQA model as baselines from four categories: query embedding methods (BetaE (Ren and Leskovec, 2020) and ConE (Zhang et al., 2021)), graph neural network method (LMPNN (Wang et al., 2023b) and GNN-QE (Zhu et al., 2022)), and symbolic search method (CQD (Arakelyan et al., 2020), QTO (Bai et al., 2023b), and FIT (Yin et al., 2024)). To ensure fairness, we use the same checkpoint for CQA, QTO, FIT, and our method. Since FIT is equivalent to QTO in tree-form queries (Yin et al., 2024), we don't distinguish them in this case.

Table 2: The results of efficiency (QPS) and effectiveness (MRR) on the tree-form queries under $\kappa \approx 10$. We employ BetaE benchmark (Ren and Leskovec, 2020) for FB15K-237, FB15K, and NELL, as well as utilize Smore Benchmark (Ren et al., 2022) for FB400K. Best in column are shown in bold.

KG	Entities	Metric	BetaE	CQD-CO	QTO/FIT	NLISA (L)	NLISA (G)
FB15K-237	14,505	QPS ↑	100	14	31	**115**	86
		MRR ↑	16.1	15.2	**23.1**	21.2	22.8
FB15K	14,951	QPS ↑	97	10.1	20	**109**	79
		MRR ↑	32.4	32.5	61.1	56.4	**61.9**
NELL	63,361	QPS ↑	23	7	3	**35**	20
		MRR ↑	18.8	19.9	27.2	27.0	**27.3**
FB400K	409,829	QPS ↑	5.4	1.6	OOM	**8.2**	
		MRR ↑	50.5	43.3	OOM	**58.9**	

6.4 Implementation details

Both NLISA (L) and NLISA (G) utilize the local constraints strategy for existential variables. NLISA (L) apply the local strategy for free variables, while NLISA (G) solves the global constraints by LMPNN (Wang et al., 2023b). For more details of the implementation, see Appendix A.

7 Experimental results

We show that NLISA achieves substantial speedup with negligible performance loss by reducing the search domain (large κ). First, we present the overall results across various KGs with $\kappa \approx 10$, highlighting NLISA's scalability to the larger FB400K dataset. Second, we provide accuracy and efficiency results across various query types, particularly illustrating that NLISA significantly accelerates cyclic queries. Lastly, we conduct an ablation study on domain sizes to further explore the effect of κ on NLISA.

7.1 Overall results on various KGs

Setup. Taking less than 10% entities, we set $|\mathcal{D}_x| = |\mathcal{D}_y|$ as 2000, 2000, 6000, and 8000 for FB15K-237, FB15K, and NELL, FB400K respectively. The results are presented in Table 2.

Analysis. From a methodological perspective, our NLISA consistently yields strong results in both QPS and MRR compared to baseline methods. The efficient query embedding method (BetaE) exhibits low MRR, while existing symbolic search methods (QTO and FIT) are limited in QPS. NLISA (L) demonstrates the best efficiency,

Table 3: MRR results (%) for various query types under $\kappa \approx 10$. We include tree-form queries without negation from the BetaE benchmark (Ren and Leskovec, 2020), where A_P denotes the average for these query types. Additionally, we incorporate all queries from the Real EFO1 benchmark (Yin et al., 2024), with A_R representing the average for these query types. The best results are highlighted in red, while the second-best results are shown in blue.

Method	1p	2p	3p	2i	3i	ip	pi	2u	up	A_P	pni	2il	3il	2m	2nm	3mp	3pm	im	3c	3cm	A_R
FB15K-237																					
CQD-CO	**46.7**	9.6	6.2	31.2	40.6	16.0	23.6	14.5	8.2	21.9	7.7	29.6	46.1	6.0	1.7	6.8	3.3	12.3	25.9	23.8	16.3
ConE	41.8	12.8	11.0	32.6	47.3	25.5	14.0	14.5	10.8	23.4	10.8	27.6	43.9	9.6	7.0	9.3	7.3	14.0	28.2	24.9	18.3
LMPNN	45.9	13.1	10.3	34.8	48.9	17.6	22.7	13.5	10.3	24.1	10.7	28.7	42.1	9.4	4.2	9.8	7.2	15.4	25.3	22.2	17.5
GNN-QE	42.8	**14.7**	11.8	**38.3**	**54.1**	18.9	**31.1**	16.2	**13.4**	26.8											
QTO	**46.7**	**14.6**	**12.8**	**37.5**	**51.6**	**21.9**	30.1	**18.0**	**13.1**	**27.4**	12.1	28.9	47.9	8.5	10.7	11.4	6.5	17.9	38.3	35.4	21.8
FIT	**46.7**	**14.6**	**12.8**	**37.5**	**51.6**	**21.9**	30.1	**18.0**	**13.1**	**27.4**	**14.9**	**34.2**	**51.4**	**9.9**	**12.7**	**11.9**	**7.7**	**19.6**	**39.4**	**37.3**	**23.9**
NLISA (L)	46.6	13.8	12.2	33.1	45.8	20.2	27.1	16.9	11.4	25.2	12.5	34.0	51.8	9.2	**9.8**	10.3	7.4	17.8	34.9	34.5	22.2
NLISA (G)	46.6	14.1	12.3	37.2	51.3	**21.3**	**30.2**	**17.6**	12.2	**27.0**	**14.2**	**34.3**	**52.7**	**9.5**	9.2	**10.7**	**7.4**	**18.1**	**36.0**	**35.5**	**22.8**
FB15K																					
CQD-CO	89.2	25.6	13.6	77.4	78.3	44.2	33.2	41.7	22.1	46.9	7.7	29.6	46.1	6.0	1.7	6.8	3.3	12.3	25.9	23.8	16.3
ConE	75.3	33.8	29.2	64.4	73.7	50.9	35.7	55.7	31.4	49.8	37.0	40.1	57.3	33.3	11.5	23.9	27.6	38.7	35.0	36.3	34.1
LMPNN	85.0	39.3	28.6	68.2	76.5	43.0	46.7	36.7	31.4	50.6	38.7	43.2	57.8	40.3	7.9	24.0	30.5	48.4	32.2	30.9	35.4
GNN-QE	88.5	**69.3**	**58.7**	**79.7**	**83.5**	70.4	69.9	**74.1**	**61.0**	**72.8**											
QTO	**89.4**	**65.6**	**56.9**	**79.1**	**83.5**	**71.8**	**73.1**	**73.9**	59.0	**72.5**	48.2	49.5	68.2	64.6	19.4	48.5	53.7	73.9	53.3	54.9	53.4
FIT	**89.4**	**65.6**	**56.9**	**79.1**	**83.5**	**71.8**	**73.1**	**73.9**	59.0	**72.5**	**57.9**	**70.4**	**77.6**	**73.5**	**39.1**	**57.3**	**64.0**	**79.4**	**63.8**	**65.4**	**64.8**
NLISA (L)	87.2	61	53.1	67.7	75.4	66.2	65	56.3	46.7	64.3	55.9	69.1	76.0	66.0	**38.3**	49.1	55.2	70.5	57.8	59.7	59.8
NLISA (G)	**89.4**	64.7	55.4	76.4	82.3	71.1	**71.1**	71.9	57.5	71.1	**60.7**	**70.4**	**78.0**	**68.8**	36.2	**51.0**	**57.0**	**76.1**	**60.3**	**61.5**	**62.0**
NELL																					
CQD-CO	60.4	17.8	12.8	39.3	46.6	22.1	30.1	17.3	13.2	28.8	7.9	48.7	68.0	31.7	1.5	12.9	13.8	33.9	38.8	35.9	29.3
ConE	53.1	16.1	13.9	40.0	50.8	26.3	17.5	15.3	11.3	27.2	10.3	42.1	65.8	32.4	7.0	12.6	16.8	34.4	40.2	38.2	30.0
LMPNN	60.6	22.1	17.5	40.1	50.3	24.9	28.4	17.2	15.7	30.8	11.6	43.9	62.3	35.6	6.2	15.9	19.3	38.3	39.1	34.4	30.7
GNN-QE	53.3	18.9	14.9	42.4	52.5	18.9	30.8	15.9	12.6	28.9											
QTO	**60.8**	**23.8**	**21.2**	**44.3**	**54.1**	26.6	31.7	**20.3**	**17.6**	**33.4**	12.3	48.5	68.2	38.8	12.3	22.8	19.3	41.1	45.4	43.9	35.3
FIT	**60.8**	**23.8**	**21.2**	**44.3**	**54.1**	26.6	31.7	**20.3**	**17.6**	**33.4**	**14.4**	**53.3**	69.5	**42.1**	**12.5**	**24.0**	22.8	**41.5**	**47.5**	**45.3**	**37.3**
NLISA (L)	60.4	23.5	**20.1**	43.7	**54.1**	**27.0**	**32.8**	**20.3**	17.3	33.2	**14.0**	53.1	**71.7**	23.0	**11.8**	**23.0**	**23.0**	41.1	**45.6**	**44.4**	**36.8**
NLISA (G)	**60.8**	**23.5**	20.0	**44.4**	**54.4**	**27.2**	**33.1**	**20.3**	**17.4**	**33.5**	**14.0**	**53.4**	**72.2**	**40.9**	10.6	22.6	**23.1**	**41.5**	**46.0**	**44.1**	**36.8**

Table 4: MRR results (%) of negation queries from tree-form BetaE benchmark under $\kappa \approx 10.$, where A_N denotes the average for these query types. We highlight the best results in red and the second-best results in blue.

KG	Method	2in	3in	inp	pin	A_N
FB15K-237	ConE	5.4	8.6	7.8	4.0	5.9
	LMPNN	8.7	12.9	7.7	4.6	8.5
	GNN-QE	10.0	16.8	9.3	7.2	8.5
	FIT/QTO	**14.0**	**20.0**	**10.2**	**9.5**	**13.4**
	NLISA (Local)	12.8	17.3	**9.5**	9.0	12.2
	NLISA (Global)	**13.8**	**20.1**	**9.5**	**9.3**	**13.2**
FB15K	ConE	17.9	18.7	12.5	9.8	14.8
	LMPNN	29.1	29.4	14.9	10.2	20.9
	GNN-QE	**44.7**	**41.7**	**42.0**	30.1	**39.6**
	FIT/QTO	40.2	38.9	34.8	28.1	35.5
	NLISA (local)	44.8	39.9	36.8	**33.0**	38.6
	NLISA (Global)	**48.3**	**44.4**	**38.0**	**34.5**	**41.3**
NELL	ConE	5.7	8.1	10.8	3.5	6.4
	LMPNN	8.5	10.8	12.2	3.9	8.9
	GNN-QE	9.9	14.6	11.4	6.3	10.6
	FIT/QTO	**12.6**	**16.4**	**15.3**	**8.3**	**13.2**
	NLISA (Local)	12.7	**16.5**	**15.2**	**8.3**	**13.2**
	NLISA (Global)	**12.5**	**16.5**	**15.3**	**8.3**	**13.2**

whereas NLISA (G) typically achieves the highest MRR. Regarding KGs, larger entity sets pose greater efficiency challenges for baseline models. Due to their quadratic complexity, QTO and FIT exhibit low QPS for larger graphs and even face OOM issues with FB400K. For embedding methods, training query embedding methods is particularly challenging. Notably, NLISA (L) relies solely on KGE models and easily developed for various sizes of KGs.

7.2 Performance on various query types

Setup. We utilize the same setup to further present the performance NLISA on various query types[3]. The results of positive queries in BetaE and all the queries in Real EFO1 benchmark is provided in Table 3, while the results of negative queries is provided in Table 4.

Analysis. By reducing the search domain with $\kappa \approx 10$ and employing fast approximate search, our method achieves remarkable efficiency without compromising accuracy across a wide spectrum of query types. Specifically, compared to the previous SoTA method FIT, NLISA (G) achieves 99% accuracy on positive tree-form queries and 95% on general EFO1 queries. This shows that the potential performance drop of NLISA is small. In particular, for negation queries and complex EFO1 types like "2il" and "3il", NLISA (G) even outperforms FIT, which demonstrates that reduced search domains even can improve the accuracy by effectively filtering out noise. Regarding cyclic queries, including "3c" and "3cm", our approximate search framework achieves an average performance of nearly 95% compared to FIT, validating the effectiveness of our approximate search algorithms.

Although NLISA (L) produces slightly lower performance compared to NLISA (G), it offers

[3]The query type "pni" is a universal first-order logic query, and is therefore excluded from the evaluation.

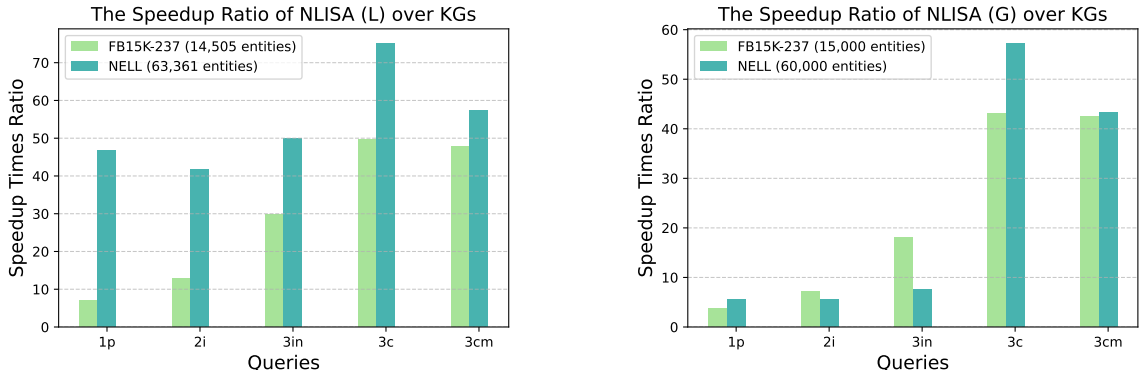

Figure 3: The efficiency results of NLISA for various query types on FB15K-237 and NELL. The included query types are 1p, 2i, 3in, 3c, and 3cm, arranged according to their complexity in symbolic search. Notably, 3c and 3cm are cyclic queries with NP-hard complexity.

significantly higher QPS and greater convenience, making it practical for real-world applications.

7.3 Efficiency results on various query types

To analyze the speedup performance across different query types, we present the speedup ratios compared to FIT in Figure 3. We selected five representative query types, arranged by the complexity of symbolic search. We conduct the ablation study over two methods (NLISA (L), NLISA (G)) and two KGs (FB15K-237, and NELL).

The results indicate that cyclic queries, such as 3c and 3cm, exhibit significantly larger speedup ratios compared to other tree-form queries across various KGs and methods, validating the effectiveness of our approximate search algorithm. Notably, the speedup ratio of NLISA (L) surpasses that of NLISA (G), especially in the NELL dataset with its larger entity set, highlighting the effectiveness of the local constraints strategy.

7.4 Study on domain sizes

We investigate the empirical effect of domain sizes in NLISA. We select NELL to vary domain sizes as NELL is the largest KG of the popular dataset. The results including both MRR and running time results are provided in Table 5.

The results indicate that both the running time and MRR of NLISA (L) and NLISA (G) increase with the growth of domain size. At a domain size of 2000, which represents 3% of the entity set, NLISA (G) achieves an MRR of 99%, while NLISA (L) reaches 95%. This validates the sparse nature of knowledge graphs and underscores the effectiveness of our domain reduction strategy. Additionally, the efficiency advantages of NLISA (L) become evident at lower domain sizes, achieving a 5.5× improvement at a domain size of 1000.

Table 5: The total running times (in seconds) and MRR results for different domain sizes under the NELL dataset, which includes 63,361 entities.

Method	Dataset	1,000	2,000	4,000	6,000	8,000
NLISA (L)	Time	509	640	1008	1429	2351
	MRR	30.3	32.3	33.6	33.7	33.8
NLISA (G)	Time	2821	2823	2856	2856	3944
	MRR	33.6	33.8	33.9	33.9	33.9

8 Conclusion

Complex query answering over knowledge graphs is a crucial multi-hop reasoning task aimed at addressing first-order logical queries over incomplete KGs. Although symbolic search methods exhibit strong performance, they face efficiency challenges that hinders further development and application. Query embedding models offer fast speed but often provide generic performance. In this paper, we integrate their advantage together in a mutually beneficial way and propose an approximate search framework combined with flexible neural logical indices to address these efficiency issues. The neural logical indices can be computed rapidly using embedding methods, significantly reducing the search domain of symbolic methods. In particular, the approximate search framework can handle cyclic queries well with a quadratic complexity. Our approximate search is precise for acyclic queries. Experiments on various benchmarks show that, with a 10% reduced search domain, our method achieves 90% performance, including for cyclic queries, while the QPS assess efficiency is improved to be comparable to that of query embedding methods. Additionally, we demonstrate that our framework can execute on a KG with an order of magnitude more entities than before, highlighting the scalable nature of our approach.

Limitations

While NLISA efficiently answers general logical queries on incomplete knowledge graphs, it has not been evaluated in several practically relevant settings: (i) temporal KGs with time-stamped facts, (ii) uncertain/probabilistic KGs, (iii) attributed KGs where nodes/edges carry rich features, and (iv) inductive scenarios involving unseen entities or relations. Adapting and validating NLISA under these conditions is an important avenue for future research.

Ethical statement

This paper presents work aimed at advancing the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here. However, we recognize that improving the efficiency and scalability of complex query answering over knowledge graphs can have significant implications for various applications, such as information retrieval, natural language understanding, and decision-making systems. By enhancing the ability to process complex queries more efficiently, our framework can support more effective data-driven insights and facilitate better interactions with knowledge-based systems. This could lead to more accurate information dissemination, improved accessibility to knowledge, and ultimately, a more informed society. Moreover, our approach may encourage further exploration of the intersection between symbolic reasoning and embedding techniques, fostering innovations that could benefit numerous sectors, including education, healthcare, and public policy. We are committed to ensuring that our research is conducted responsibly and ethically, and we encourage ongoing dialogue about the societal impacts of advancements in Machine Learning. We hope that our contributions will not only push the boundaries of the field but also promote positive outcomes for society at large.

References

Alfonso Amayuelas, Shuai Zhang, Xi Susie Rao, and Ce Zhang. 2021. Neural Methods for Logical Reasoning over Knowledge Graphs.

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2020. Complex Query Answering with Neural Link Predictors. In *International Conference on Learning Representations*.

Jiaxin Bai, Xin Liu, Weiqi Wang, Chen Luo, and Yangqiu Song. 2023a. Complex Query Answering on Eventuality Knowledge Graph with Implicit Logical Constraints. *arXiv preprint*. ArXiv:2305.19068 [cs].

Jiaxin Bai, Chen Luo, Zheng Li, Qingyu Yin, and Yangqiu Song. 2024. Understanding inter-session intentions via complex logical reasoning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 71–82.

Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. 2023b. Answering Complex Logical Queries on Knowledge Graphs via Query Computation Tree Optimization. In *Proceedings of the 40th International Conference on Machine Learning*, pages 1472–1491. PMLR. ISSN: 2640-3498.

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*, volume 26. Curran Associates, Inc.

Hubie Chen, Víctor Dalmau, and Berit Grußien. 2011. Arc consistency and friends. *J. Log. Comput.*, 23:87–108.

Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. 2021. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*.

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

Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2000. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124(2):243–282.

Cosimo Gregucci, Bo Xiong, Daniel Hernandez, Lorenzo Loconte, Pasquale Minervini, Steffen Staab, and Antonio Vergari. 2024. Is complex query answering really complex? *arXiv preprint arXiv:2410.12537*.

David Ha, Andrew M. Dai, and Quoc V. Le. 2016. Hypernetworks. *ArXiv*, abs/1609.09106.

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.

George Klir and Bo Yuan. 1995. *Fuzzy sets and fuzzy logic*, volume 4. Prentice hall New Jersey.

Hongyu Liu, Shumin Shi, and Heyan Huang. 2019. Coarse-to-fine document ranking for multi-document reading comprehension with answer-completion. *2019 International Conference on Asian Language Processing (IALP)*, pages 407–412.

Jerry M Mendel. 1995. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377.

Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1751–1756.

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

H Ren, W Hu, and J Leskovec. 2020. Query2box: Reasoning Over Knowledge Graphs In Vector Space Using Box Embeddings. In *International Conference on Learning Representations (ICLR)*.

Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. 2022. Smore: Knowledge graph completion and multi-hop reasoning in massive knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1472–1482. ArXiv:2110.14890 [cs].

Hongyu Ren, Mikhail Galkin, Michael Cochez, Zhaocheng Zhu, and Jure Leskovec. 2023. Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases. *arXiv preprint*. ArXiv:2303.14617 [cs].

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.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Zihao Wang, Weizhi Fei, Hang Yin, Yangqiu Song, Ginny Wong, and Simon See. 2023a. Wasserstein-fisher-rao embedding: Logical query embeddings with local comparison and global transport. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13679–13696.

Zihao Wang, Yangqiu Song, Ginny Wong, and Simon See. 2023b. Logical Message Passing Networks with One-hop Inference on Atomic Formulas. In *The Eleventh International Conference on Learning Representations*.

Zihao Wang, Hang Yin, and Yangqiu Song. 2021. Benchmarking the Combinatorial Generalizability of Complex Query Answering on Knowledge Graphs. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1.

Zihao Wang, Hang Yin, and Yangqiu Song. 2022. Logical Queries on Knowledge Graphs: Emerging Interface of Incomplete Relational Data. *Data Engineering*, page 3.

Bo Xiong, Nico Potyka, Trung-Kien Tran, Mojtaba Nayyeri, and Steffen Staab. 2022. Faithful Embeddings for $$$\backslash$mathcal ${$e$}$$\backslash$mathcal ${$l$}$^${$++$}$ $$ Knowledge Bases. In *International Semantic Web Conference*, pages 22–38. Springer.

Hang Yin, Zihao Wang, and Yangqiu Song. 2024. Rethinking existential first order queries and their inference on knowledge graphs. In *The Twelfth International Conference on Learning Representations*.

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.

Tianshi Zheng, Jiaxin Bai, Yicheng Wang, Tianqing Fang, Yue Guo, Yauwai Yim, and Yangqiu Song. 2024. Clr-fact: Evaluating the complex logical reasoning capability of large language models over factual knowledge. *arXiv preprint arXiv:2407.20564*.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. 2022. Neural-Symbolic Models for Logical Queries on Knowledge Graphs. *arXiv preprint arXiv:2205.10128*.

Table 6: Power-law fits for domain-size distributions on three KGs. We report the fitted exponent α and the Kolmogorov–Smirnov (KS) p-value (higher is better).

	FB15K-237	FB15K	NELL
Fitted α	1.93	1.89	1.92
KS p-value	0.84	0.63	0.65

A Implementation

All the results are conducted on the V100 with 32G memory.

Complex query answering For the FB15K, FB15K-237, and NELL datasets, we utilize the checkpoints provided by CQD-CO (Arakelyan et al., 2020). Since FIT is equivalent to QTO in tree-form queries (Yin et al., 2024), we don't distinguish them in this case.

Knowledge graph completion We reproduce the results from previous work (Chen et al., 2021) to train the embedding and hyper-network. For the FB400K dataset, we search the hyper-parameters with the following settings: learning rates of $[1 \times 10^{-1}, 1 \times 10^{-2}]$, embedding dimensions of $[100, 200, 400]$, and λ values of $[0.0005, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 0]$. We utilize the ComplEx model with the N3 regularizer. The embedding initialization is set to 1×10^{-3}, and we employ the Adagrad optimizer.

B Query types within complex logical queries

The datasets in CQA are classified by different query types, as queries of the same type share similar operator trees or query graphs. We present the visualizations of the query types for the BetaE benchmark (Ren and Leskovec, 2020) and the Real EFO1 benchmark (Yin et al., 2024) in Figure 4 and Figure 5, respectively. The visualization for the BetaE benchmark is sourced from Wang et al. (2023a), while the visualization for the Real EFO1 benchmark is derived from Yin et al. (2024).

C Theoretical properties

Empirical evidence of power laws. Let K denote the (answer-set) *domain size* of a free variable under a query instance. We collect K over the BetaE benchmark for three KGs and plot the empirical distributions in Figure 6. The curves are approximately linear in log–log scale, which is consistent with a power-law model. We further fit a discrete power law using the `powerlaw` package and evaluate the fit via the KS statistic; Table 6 shows that the p-values are well above 0.1, indicating that a power-law model is not contradicted by the data.[4]

Modeling assumption. We hence adopt the following working hypothesis for analysis.

Assumption C.1 (Power-law domain sizes) *The random variable K (domain size) follows a discrete power law on $\{1, 2, \ldots, E\}$: $\Pr[K = k] \propto k^{-\alpha}$ with exponent $\alpha > 1$, where $E = |\mathcal{E}|$ is the number of entities.*

Reduced domains and preservation probability. Let $q \in (0, 1]$ be the retained *fraction* of the entity space (e.g., $q = 0.1$ keeps the top 10%). We say a reduction *preserves* an answer if the (true) domain size K does not exceed the retained budget qE and the selection procedure puts all true entities inside the reduced set. The second part is ensured by the following idealization.

Assumption C.2 (Oracle ranking) *Within each variable domain, entities are ranked by their true membership (or a score perfectly aligned with it), so that keeping the top qE entities retains all true elements whenever $K \leq qE$.*

Under Assumptions C.1–C.2, the probability that a single-variable reduction preserves all true answers is

$$P_{\mathrm{A}}(q; E, \alpha) \;=\; \Pr[K \leq qE] \tag{5}$$

$$= \frac{\sum_{k=1}^{\lfloor qE \rfloor} k^{-\alpha}}{\sum_{k=1}^{E} k^{-\alpha}} \;=\; \frac{H_{\lfloor qE \rfloor, \alpha}}{H_{E,\alpha}}, \tag{6}$$

where $H_{n,\alpha} = \sum_{k=1}^{n} k^{-\alpha}$ is the generalized harmonic number.

Multiple variables. Complex conjunctive queries often involve $\tau > 1$ variables (one free and possibly many existential variables). For a product t-norm aggregation and ignoring higher-order correlations in domain sizes, a conservative proxy for the overall preservation probability is the product of single-variable events.

Assumption C.3 (Independence) *For a fixed query pattern, the domain sizes of different variables are independent draws from the distribution in Assumption C.1.*

[4] A large p-value does not *prove* a power law; it indicates that the power-law hypothesis cannot be rejected at common significance levels. We use it as a pragmatic modeling assumption.

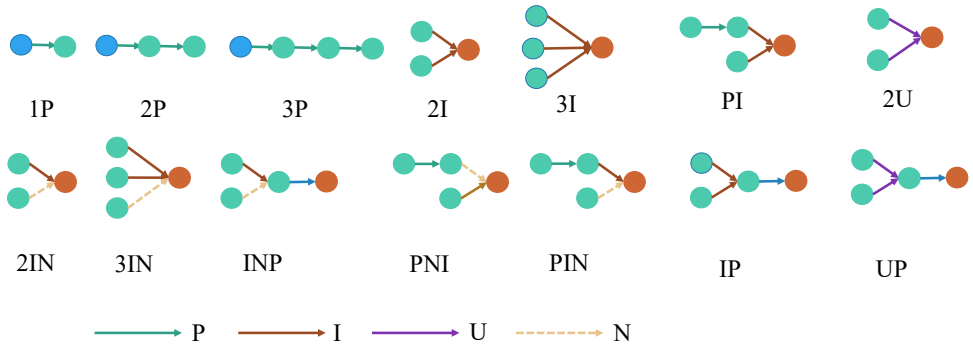

Figure 4: 14 query types proposed in BetaE benchmark (Ren and Leskovec, 2020). These query types are modeled by the operator tree.

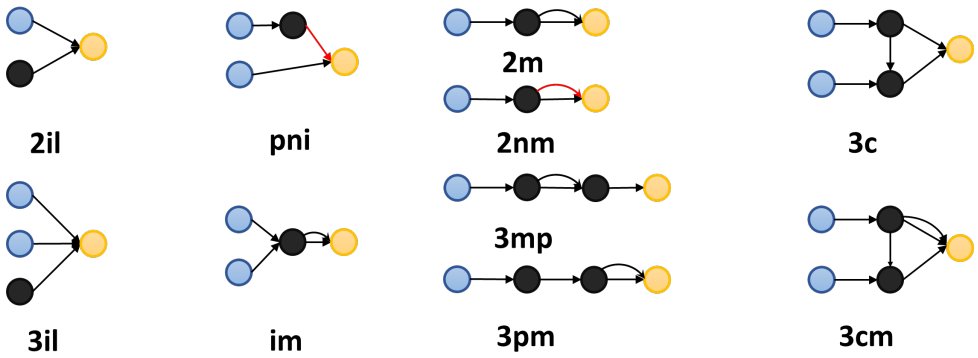

Figure 5: 10 query types proposed in Real EFO1 benchmark (Yin et al., 2024). These query types are modeled by query graph.

Proposition C.4 (Preservation for multi-variable) *Under Assumptions C.1–C.3, retaining a q-fraction per variable preserves all true answers with probability*

$$P_{\mathrm{A}}^{(\tau)}(q; E, \alpha) = \left(\frac{H_{\lfloor qE \rfloor, \alpha}}{H_{E,\alpha}}\right)^{\tau}. \tag{7}$$

By Assumption C.2, the event "all true entities for variable v lie within the top qE" is equivalent to $\{K_v \leq qE\}$. Assumption C.3 then gives $\Pr\left[\bigcap_{v=1}^{\tau}\{K_v \leq qE\}\right] = \prod_{v=1}^{\tau} \Pr[K_v \leq qE]$, and each term equals (6).

Asymptotics and intuition. When E is large and $\alpha > 1$, $H_{E,\alpha} \to \zeta(\alpha)$ and $H_{\lfloor qE \rfloor,\alpha}/H_{E,\alpha} \approx 1 - \frac{\sum_{k>\lfloor qE \rfloor} k^{-\alpha}}{\zeta(\alpha)}$. The tail $\sum_{k>m} k^{-\alpha}$ decays as $\Theta(m^{1-\alpha})$; hence $P_{\mathrm{A}}(q; E, \alpha)$ approaches 1 quickly as q grows, and (7) shows a predictable degradation with the number of variables τ. This formalizes the intuition that (i) power-law domains concentrate most probability mass on small sizes, making modest q sufficient for high preservation, and (ii) more variables compound the risk, calling for slightly larger q or better initialization.

D Results on New benchmark

We provide the further experiments of NLISA on FB15k237+H, and NELL+H, which were proposed to investigate "hard" queries that all the reasoning path to their answers require one incomplete edge. The experiments over Table shows that our results achieve competitive performance, where the MRR of NLISA (G) over 5/14 query types achieve the best across all the baselines.

E Constraints strategies

For local constraints, when the variable involves logical disjunction, we use the T-conorm to handle it. For global constraints, the computation of existential variables should treat them as free variables, while the original free variables are converted into existential variables. This allows us to use query embedding to compute the global constraints for the existential variables.

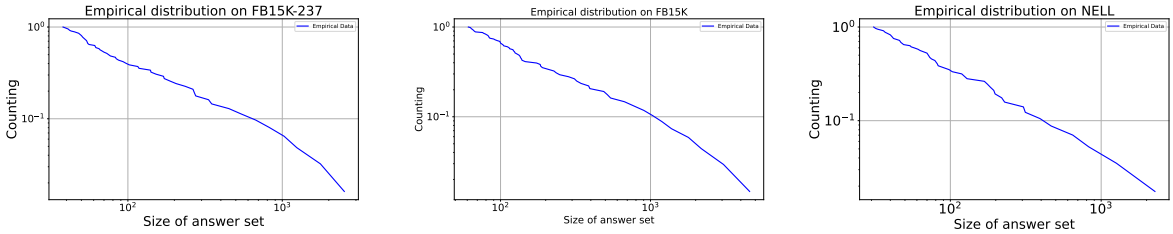

Figure 6: Empirical distributions of answer-set/domain sizes across three knowledge graphs (KGs). Since the distributions resemble power laws, we use logarithmically spaced bins and log–log axes. The near-linearity suggests heavy-tailed behavior.

Table 7: Results on new benchmark (Gregucci et al., 2024). The results of all the baselines are sourced from Gregucci et al. (2024). The best results over all the models are highlight by black.

Model	1p	2p	3p	2i	3i	1p2i	2i1p	2u	2u1p	2in	3in	2pi1pn	2nu1p	2in1p
GNN-QE	42.8	5.2	4.0	6.0	8.8	5.6	9.9	32.5	10.0	6.8	**6.5**	**3.7**	5.0	3.3
ULTRAQ	40.6	4.5	3.5	5.2	7.2	5.3	10.1	29.4	8.3	5.3	5.5	2.6	3.7	2.2
CQD	**46.7**	4.4	2.4	11.3	**12.8**	6.0	11.5	40.1	10.6	3.3	2.6	0.6	4.9	1.2
CQD-Hybrid	**46.7**	4.8	3.1	6.0	8.6	5.5	12.9	**42.2**	12.0	4.7	1.6	1.0	3.2	1.3
ConE	41.8	4.6	3.9	9.1	10.3	3.8	7.9	22.8	6.0	5.1	4.9	2.9	3.3	**3.6**
QTO	**46.7**	4.9	3.7	8.7	10.1	6.1	13.5	30.6	11.2	10.6	3.1	2.0	**5.3**	1.5
CLMPT	45.3	**5.3**	**4.7**	10.2	12.2	5.6	**14.9**	33.6	**14.2**	6.8	2.3	1.6	4.8	2.5
NLISA (G)	46.6	**5.3**	3.6	**11.9**	**13.2**	**6.6**	12.6	34.7	10.1	**13.8**	3.9	1.6	4.4	2.2
GNN-QE	53.6	8.0	6.0	10.7	13.3	16.0	13.5	47.5	9.8	5.5	**6.4**	**5.8**	3.3	4.4
ULTRAQ	38.9	6.1	4.1	7.9	10.2	15.8	9.3	28.1	9.5	4.5	5.9	4.3	2.7	3.6
CQD	60.4	9.6	4.2	18.5	19.6	18.9	22.6	46.3	18.5	4.2	1.5	1.5	4.9	2.6
CQD-Hybrid	60.4	9.0	6.1	12.1	14.4	17.4	21.2	46.4	19.3	5.1	1.2	1.4	4.3	2.4
ConE	53.1	7.9	6.7	21.8	23.6	14.9	11.8	39.9	8.8	4.6	6.0	3.7	2.7	**6.4**
QTO	60.3	9.8	**8.0**	14.6	15.8	17.6	21.1	49.1	18.9	10.2	2.3	3.1	**8.4**	2.4
CLMPT	58.1	10.1	7.8	**22.7**	**25.0**	17.2	**24.4**	50.0	**22.0**	6.5	2.4	4.1	2.3	4.5
NLISA (G)	**60.8**	**10.3**	7.6	19.1	20.1	**18.7**	23.1	**50.1**	18.2	**12.5**	2.6	2.5	5.0	2.5

E.1 Knowledge graph embedding and HyperNet

Knowledge graph embedding first embeds the entities and relations into a continuous space. Given the query $(s, r, ?)$, we first compute its embedding and compare it with the entities within the embedding spce. The estimated embedding of $(s, r, ?)$ is first computed by $f_t(e_s, e_r)$, where $f(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is the transformation function. Then, the likelihood of $r(s, o)$ is computed by the scoring function $f_s(f_t(e_r, e_s), e_o)$, where $f_s(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to [-\infty, +\infty]$ is the scoring function related to the embedding space. The scoring function is usually a distance function or a similar metric in the embedding space.

For relation tail prediction task, we adapt existing KG embedding models using a Hypernet. The Hypernet modifies the embeddings of entities and relations, while we employ the same scoring function as the original embedding models. Given the query $(r, ?)$ and candidate entity t, the score $g_s(\mathsf{HN}(r), \mathsf{HN}(t))$, where HN represents the Hypernet and $g_s = \sigma(f_s)$ with $\sigma()$ being the sigmoid activation function.

E.2 Introduction of Hypernet

Motivation Relation tail prediction is fundamentally similar to knowledge graph completion (KGC) tasks, as both involve inferring missing tails from a knowledge graph. In KGC, the query is presented as $(h, r, ?)$, to predict the tail t. In contrast, relation tail prediction aims to predict the tail t given the relation r. This similarity makes the embedding methods used in KGC equally applicable to relation tail prediction. Hypernet is employed to generate weights dynamically, allowing the model to automatically adjust representations for new tasks. This facilitates the sharing and adaptation of weights across different tasks. By adopting the embeddings from KGC, Hypernet enhances the utilization of

Table 8: MRR results for the trained Hypernet.

	FB15K-237	FB15K	NELL	FB400K
MRR	1.0	0.998	0.99	1.0

existing knowledge and enables quicker adaptation to relation tail prediction. Additionally, Hypernet requires only a few parameters, improving performance while reducing training costs and storage needs.

Training Hypernet The training process is similar to that of KGC, where each triple (h,r,t)(h,r,t) is replaced with (r,t)(r,t). This requires only minor modifications compared to the existing KGC framework.

Performance of Hypernet For the Hypernet used for relation tail prediction, we set the learning rate to 1×10^{-2} and search the hidden dimensions $[100, 200, 400]$ also using the ComplEx model with the N3 regularizer. The embedding initialization remains at 1×10^{-3}, and the optimizer is Adagrad. The performance of our method is quite impressive, as demonstrated in Table 8. The results indicate a strong capability of the Hypernet in addressing relation tail prediction tasks.

F Fuzzy logical inference

G t-norm introduction

Definition G.1 (t-norm) *A t-norm $\top$ is a function: [0,1] x [0,1] $\rightarrow$ [0,1] that satisfies the following properties:*

(i) Communitavity: $a \top b = b \top a$

(ii) Monotonicity: $(a \top b) \leq (c \top b)$ *if* $a \leq c$

(iii) Associativity: $(a \top b) \top c = a \top (b \top c)$

(iv) Neutrality: $a \top 1 = a$

Then the t-conorm $\perp$ is directly defined by $a \perp b = 1 - (1-a) \top (1-b)$, which follows the De Morgan's law.

Finally, we introduce some common t-norms which are of interest:

(i) Godel: $a \top_G b = \min(a, b)$

(ii) Product: $a \top_P b = a \times b$

(iii) Łukasiewicz: $a \top_{LK} b = \max(a + b - 1, 0)$

In the main paper, we mainly focus on Product t-norm.

G.1 Truth values

The symbolic representation in the previous symbolic search method QTO and FIT is constructing the set of truth values matrix for the whole knowledge graph. To convert real number scores computed by knowledge graph embedding modes to truth value that falls into $[0, 1]$, QTO/FIT use the softmax function:

$$P^{\star}_{r,a}(b) = \frac{exp(s(a, r, b))}{\Sigma_{c \in \mathcal{E}} exp(s(a, r, c))}. \tag{8}$$

Next, QTO and FIT scale the results of the softmax function using a factor based on the observed edges in the training graph since softmax outputs a vector that sums to 1: $\mathcal{G}_o$.

$$Q_{a,b} = \begin{cases} \frac{|\{d|(a,r,d) \in \mathcal{G}_o\}|}{\Sigma_{c \in \{d|(a,r,d) \in \mathcal{G}_o\}} P^{\star}_{r,a}(c)}, & \text{if } C^r_a > 0 \\ 1, & \text{if } C^r_a = 0 \end{cases} \tag{9}$$

where $C^r_a = |\{d|(a, r, d) \in \mathcal{G}_o\}|$ and $E_{a,r} = \{b \mid (a, r, b) \in \mathcal{G}_o\}$ represents the set of observed edges. Then the a-th row of r-th matrix is got by clamping the value for each element:

$$P_r(a, b) = min(1, P^{\star}_{r,a}(b) \times Q_{a,b}) \tag{10}$$

They then mark the observed edges and set the truth value for these edges to 1. The scaling and marking operations are performed on a case-by-case basis for each fact, which cannot be parallelized.

We demonstrate that these scaling operations can be parallelized through caching. For cases where $C^r_a > 0$: the truth value is computed by the following normalization:

$$Q_{a,b} = \frac{exp(f_r(a, b))}{\sum_{\{d|(a,r,d) \in \mathcal{G}_o\}} exp(f_r(a, d))}. \tag{11}$$

To simplify the calculations, we use log-scale operations to compute $Q_{a,b}$. If we cache $S^r_a = log(\sum_{\{d|(a,r,d) \in \mathcal{G}_o\}} exp(f_r(a, d)))$ with $|\mathcal{E}| \times |\mathcal{R}|$ size, we can parallel index the cached values to optimize the computation of $Q_{a,b}$. For cases where $C^r_a > 0$, we have:

$$S^r_a = log(\sum_{\{d \in \mathcal{E}\}} exp(f_r(a, d))), \tag{12}$$

which allows the scaling operation to yield the softmax result.

This caching mechanism grows linearly with the size of the knowledge graph. By utilizing this caching strategy, we can perform parallel computations for the scaling operations, facilitating efficient dynamic symbolic representation construction. However, when it comes to listing the marked training facts, our symbolic representation may exhibit slightly lower performance compared to the original construction.

H Edge removing functions

$\textsc{RemoveConstNode}(G_\phi, |\mathcal{D}_x|, |\mathcal{D}_y|)$ removes the constant nodes based on the given domain of variables. We consider the simplest case where $(s, r, e, \textsc{Neg} = \textsc{False})$, with s as the constant node, r as the relation, e as the existential variable, and $\textsc{Neg} = \textsc{False}$ indicating that this edge is positive. We first construct the symbolic representation of this edge: $\mathbf{s} = [P_r(s, e^i)] \in \mathcal{R}^{|\mathcal{D}_e|}, e^i \in \mathcal{D}_e$. We then update this representation into the fuzzy vector of the existential variable e by t-norm: $\mu(e, \mathcal{D}) = \mu(e, \mathcal{D}) \top \mathbf{s}$. Consequently, we can remove this edge while preserving its constraints in the fuzzy vector of e. The above method can be generalized to other cases, such as when the variable is a free variable y and the edge is negative. A similar update approach can be used, and specific details can be found in Yin et al. (2024).

$\textsc{RemoveConstNode}(G_\phi, |\mathcal{D}_x|, |\mathcal{D}_y|)$ removes one leaf node based on the given domain of variables. We consider the most difficult case where $(e_1, r, e_2, \textsc{Neg} = \textsc{False})$, with s as the constant node, r as the relation, e as the existential variable, and $\textsc{Neg} = \textsc{False}$ indicating that this edge is positive. The symbolic representation of this edge results in a matrix: $\mathcal{S} = [P_r(e_1^i, e_2^j)] \in \mathcal{R}^{|\mathcal{D}_{e_1}| \times |\mathcal{D}_{e_2}|}, e_1^i \in \mathcal{D}_{e_1}, e_2^i \in \mathcal{D}_{e_2}$. We then update this representation into the fuzzy vector of the existential variable e by t-norm and max operation: $\mu(e, \mathcal{D}) = \mu(e, \mathcal{D}) \top \max(\mathcal{S}, axis = 0)$. The proof of the effectiveness of the above update can be found in Yin et al. (2024), and this update can be easily extended to other cases, such as when the variable is a free variable y and the edge is negative.

I Evaluation metrics

We present the details of the evaluation metrics in complex query answering, which aims to find non-trivial answers in incomplete knowledge graphs.

Given an underlying KG $\mathcal{G} = \{(s_i, r_i, o_i)\}$ with $\mathcal{E}$ and $\mathcal{R}$, we sample three subgraphs as following:

$$\mathcal{G}_{\text{train}} \subset \mathcal{G}_{\text{valid}} \subset \mathcal{G}_{\text{test}} = \mathcal{G}, \tag{13}$$

where the three KGs have same $\mathcal{E}$ and $\mathcal{R}$.

To find the non-trivial answers, we filter out the answers that can be directly discovered by traversing graphs. We denote $[q]_{\text{train}}$ as the answer set of query q in the train graph $\mathcal{G}_{\text{train}}$, the answer set we focus on is $[q]_{\text{test}} \backslash [q]_{\text{train}}$, and these are easy answers that can only be predicted. The hard answers are $[q]_{\text{test}} \setminus [q]_{\text{valid}}$. Then we would rank the hard answers against all the non-answer sets $\mathcal{E}/[q]_{\text{valid}}(\mathcal{E}/[q]_{\text{test}})$. After getting the rank r, we calculated Mean Reciprocal Rank (MRR): $\frac{1}{r}$ and Hits at K(Hits@K):$1_{r<K}$ as metric to measure the performance of CQA models.