Chain-of-Box Empowered Language Models for Logical Reasoning over Knowledge Graphs

Anonymous ACL submission

Abstract

Complex logical reasoning over large-scale knowledge graphs (KGs) is a fundamental yet challenging task. Current approaches mainly focus on embedding logical queries as well as KG entities into the same vector space and retrieving answers based on similarity matching. However, the incompleteness issue of KGs severely hinders the effectiveness of previous studies. To tackle the challenging knowledge deficiency problem, we propose to leverage the latent knowledge of language models and design a unified framework to integrate knowledge graph reasoning and natural language reasoning by utilizing box embeddings of reasoning footprint as the chain-of-box and fusing it into the language models to empower the capability of logical reasoning. Extensive experiments on two standard benchmark datasets demonstrate that COB-LM significantly improves over state-of-the-art methods.

1 Introduction

Knowledge graphs (KGs) have emerged as the prevailing manner for organizing and managing knowledge, and have been extensively embraced as foundational components in practical applications, including search engines (Reinanda et al., 2020), professional networking platforms (Shinavier et al., 2019), and query answering (Saxena et al., 2020).

Apart from the conventional KG completion (Yang et al., 2014; Sun et al., 2018), in recent years, there has been a notable surge in the prominence of logical reasoning over KGs, which entails answering logical queries based on a given KG and involves a diverse class of logical queries, particularly Existential Positive First-Order (EPFO) queries (Dalvi and Suciu, 2007). These queries consist of operators such as existential quantification ($\exists$), conjunction ($\wedge$), and disjunction ($\vee$). Taking a logical query "$V_?.\exists V : Invest(V_?, V) \wedge Founder(OpenAI, V)$" as an example, as shown in Figure 1(a), this query involves multi-hop reasoning in a KG with logical operators (e.g., $\wedge$, $\vee$) and refers to the question "*What companies have the OpenAI founders invested in?*".

To answer the logical queries, the embedding-based logical reasoning methods primarily focus on query encoding by designing various geometric embedding structures (Ren et al., 2020; Choudhary et al., 2021b; Zhang et al., 2021) or resorting to probabilistic distributions (Ren and Leskovec, 2020; Yang et al., 2022). As illustrated in Figure 1(b), an embedding-based approach (Ren et al., 2020) encodes the 2-hop query via two projections, ultimately determining the closest neighboring entities of the last projected region as the final answer. However, the logical query reasoning relies on the given background knowledge, the *knowledge* in KGs is typically significantly incomplete (West et al., 2014), thereby rendering the set of answers inherently deficient.

Exploring additional knowledge to augment KGs poses a formidable challenge. Despite the extensive development of KG completion (Yang et al., 2014; Sun et al., 2018) aimed at addressing the issue of incompleteness, its ineffectiveness and inaccuracy in practice remain obstacles to achieving a comprehensive KG for precise logical query reasoning. Rather than undertaking the challenging task of building a complete KG, in this paper, we explore strategies for the direct use of additional knowledge for logical reasoning. Thanks to the emergent abilities of large language models (LLMs) (Bubeck et al., 2023; Webb et al., 2023), LLMs can serve as latent background knowledge and query reasoning can be conceptualized as a natural language reasoning task, where the frozen LLM functions as the retrieval module to locate relevant answers, as shown in Figure 1(c). Yet, the challenge that LLMs face in addressing intricate logical reasoning problems (Liu et al., 2023) impedes the capability to answer logical queries. Moreover, the reliability

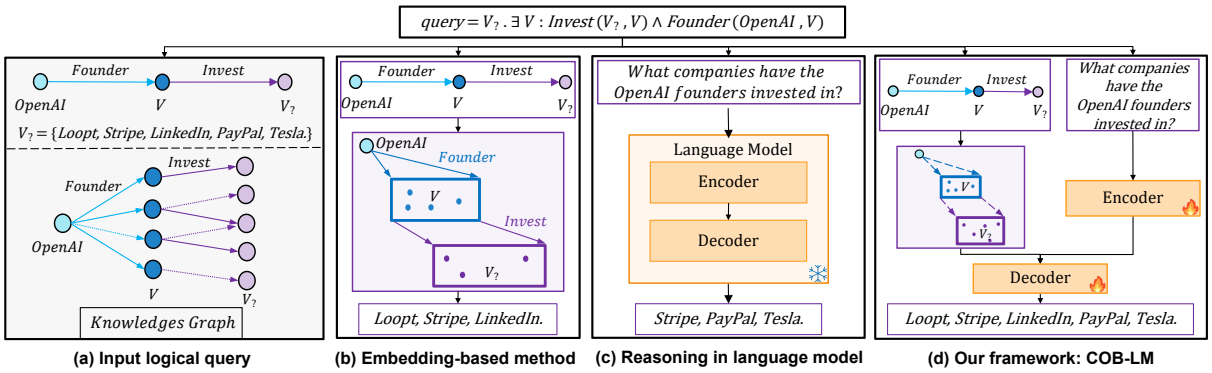

Figure 1: (a) Above is a logical query with multi-hop logic and its corresponding answers. The KG has missing relations that prevent the logical query from being correctly answered. Solid lines represent relations, and dotted lines represent missing relations. (b) The embedding-based method follows logical order to execute the query in the embedding space. (c) Reasoning in language models requires the design of prompts. (d) COB-LM incorporates the advantages of both reasoning strategies to enable the capability of logical reasoning.

of latent knowledge within LLMs is not assured, and issues such as hallucination and the absence of domain-specific knowledge hinder the applicability of LLMs (Ji et al., 2023; Raunak et al., 2021).

The incompleteness of KGs and the limitations of LLMs motivate us to explore the fusion of structural knowledge in KGs with latent knowledge in LLMs for logical query reasoning. However, the materialization is non-trivial. First, the conventional way to integrate a LLM and KGs (Pan et al., 2023) is to fine-tune the LLM using KGs, which converts KGs into sequences and inevitably loses the structural knowledge, and still suffers from the inability of logical reasoning. Second, structural knowledge in KGs facilitates explicit logical reasoning, in contrast to the implicit reasoning employed in LLMs. There is no straightforward method for the integration. Third, the emergent ability of LLMs heavily depends on the scale of language models (LMs) (Magister et al., 2022). Nonetheless, obtaining access to professional computing GPUs to fine-tune LLMs can be challenging for individuals seeking to engage in logical reasoning over domain-specific KGs. Therefore, utilizing small-scale LMs to facilitate logical query reasoning in KGs deserves more exploration.

To leverage the latent knowledge in LMs, in this paper, we propose a novel framework for logical query reasoning over KGs with chain-of-box empowered language models, named **COB-LM**, which integrates knowledge graph reasoning and natural language reasoning, aiming to leverage the two different knowledge, as shown in Figure 1(d). Specifically, **COB-LM** consists of three modules, i.e., chain-of-box construction, LM prompting, and answer generation. First, to obtain the structural knowledge in KGs, we utilize a logical decomposition mechanism to decompose the logical query into a series of sub-queries and map the sub-queries into an embedding space through the box operators (Ren et al., 2020). A sequence of box embeddings in the logical order can be obtained in the reasoning process and serves as the chain-of-box. Second, to leverage the latent knowledge in LMs, we convert complex logical queries to natural language prompts and use them to obtain the hidden representation from the LM encoder. Third, to enable the language model for effective logical query answering, we propose to fuse the chain-of-box into the hidden representation before passing it to the LM decoder to generate answers. The fusion enables LMs to be aware of the reasoning footprint and combine the structural knowledge in KGs, empowering LMs with the ability to reason logically over KGs. After the fine-tuning of language models, **COB-LM** retains the semantic reasoning capability and also gains the capability of logical reasoning.

Overall, our contributions in this work include:

(1) We propose to leverage the external knowledge from LMs to alleviate the impact of KG incompleteness on logical reasoning over KGs.

(2) We design a novel framework to fuse the structural knowledge in KGs with the latent knowledge in LMs by injecting the representation of the chain-of-box into language models.

(3) We perform extensive experiments on two KG datasets. The experimental results show the superior performance of **COB-LM** over the state-of-the-art with significant improvement in complex logical query reasoning over KGs.

2 Related Work

Logical Reasoning over Knowledge Graphs In recent years, there has been a rise in the number of embedding-based methods for logical query answering in knowledge graphs. The basic idea is to embed logical queries and entities into a joint vector space and utilize the embedding similarity for answer prediction. Some models embed queries/entities into points in the vector space (Guu et al., 2015; Hamilton et al., 2018). Furthermore, some efforts extend the single-point embedding into region embedding, such as box (Ren et al., 2020), hyperboloid (Choudhary et al., 2021b) , and cone (Zhang et al., 2021), while other methods represent the query/entity in terms of probability distributions in the vector space, such as Gaussian distribution (Choudhary et al., 2021a), Beta distribution (Ren and Leskovec, 2020), and Gamma distribution (Yang et al., 2022). In addition, query decomposition methods have also achieved significant performance. In recent studies, CQD (Arakelyan et al., 2020) uses a neural link predictor trained on 1-hop queries and QTO (Bai et al., 2022) proposes a neural search method based on query computation graphs. While these methods demonstrate superior performance, their effectiveness is hampered by incomplete KGs. This inherent limitation significantly undermines their capacity for logical reasoning. The latest work (Choudhary and Reddy, 2023) relies on the direct utilization of a frozen large language model through a step-by-step questioning approach, yet still suffers from the inherent issues of LMs.

Reasoning in Language Models Recent research has shown that multi-step reasoning ability can be triggered in language models by chain-of-thought (CoT) prompting (Wei et al., 2022). The improved performance in zero-shot reasoning tasks demonstrates the effectiveness of prompt questions (Kojima et al., 2022; Yang et al., 2023). Furthermore, studies have shown that by explicitly decomposing the question into multiple sub-questions, the language models can be guided to pay more attention to and reason about these sub-problems (Zhou et al., 2022; Khot et al., 2022; Choudhary and Reddy, 2023). However, the success of the prompt-based method significantly depends on carefully composed prompts and the size of the language model. While language models with over 100 billion parameters exhibit enhanced reasoning capabilities, those with fewer parameters not only struggle to leverage the CoT prompts for reasoning but may even compromise the accuracy of their original responses (Magister et al., 2022). In order to transfer reasoning ability to smaller language models, another research interest is to elicit CoT reasoning by fine-tuning language models. Datasets containing chains of thought processes are constructed by manual collection (Lu et al., 2022) or generation by LLMs (Ho et al., 2022) for fine-tuning LMs. Unlike the existing works, our work focuses on logical reasoning over KGs. Since the direct use of LMs cannot solve the logical reasoning task, we propose to exploit the latent knowledge in LMs to alleviate the incompleteness issue of KGs and facilitate logical query reasoning over KGs.

3 Methodology

In this section, we propose the **COB-LM** framework. We will first describe the problem definition of logical reasoning over knowledge graphs, then overview the procedure of the framework, and finally detail the sub-modules and technical design.

3.1 Preliminaries

A knowledge graph consists of a set of entities E, a set of relations R, and a set of triples T in which each triple (e_i, r, e_j) includes two entities $e_i, e_j \in E$, and a relation $r \in R$, which denotes the type of relation between e_i and e_j. Each relation type is a binary function that indicates whether the relation exists between a pair of entities, i.e., $e_1 \xrightarrow{r} e_2 \iff r(e_1, e_2) = True$.

We consider the Existential Positive First-Order (EPFO) queries, including existential quantification ($\exists$), conjunction ($\wedge$) and disjunction ($\vee$). And we define valid EPFO queries as its disjunctive normal form (DNF) (Davey and Priestley, 2002):

$$q[V_?] = V_?.\exists V_1, \ldots, V_k : (c_1) \vee (c_2) \vee \ldots \vee (c_n), \tag{1}$$

where $V_?$ is the target of the query, and $V_1, \ldots, V_k$ denote the existentially quantified bound variables. Each $c_i = (a_{i1} \wedge \ldots \wedge a_{in}), n < k$, and a_{ij} represents an atomic formula, i.e., $r(e, V)$ or $r(V', V)$, where $e \in E, r \in R, V \in \{V_1, \ldots, V_k, V_?\}, V' \in \{V_1, \ldots, V_k\}, V' \neq V$.

A complex query can be decomposed into a series of sub-queries by following basic rules:

$$A_{q_1 \wedge q_2} = A_{q_1} \cap A_{q_2}, \tag{2}$$

$$A_{q_1 \vee q_2} = A_{q_1} \cup A_{q_2}, \tag{3}$$

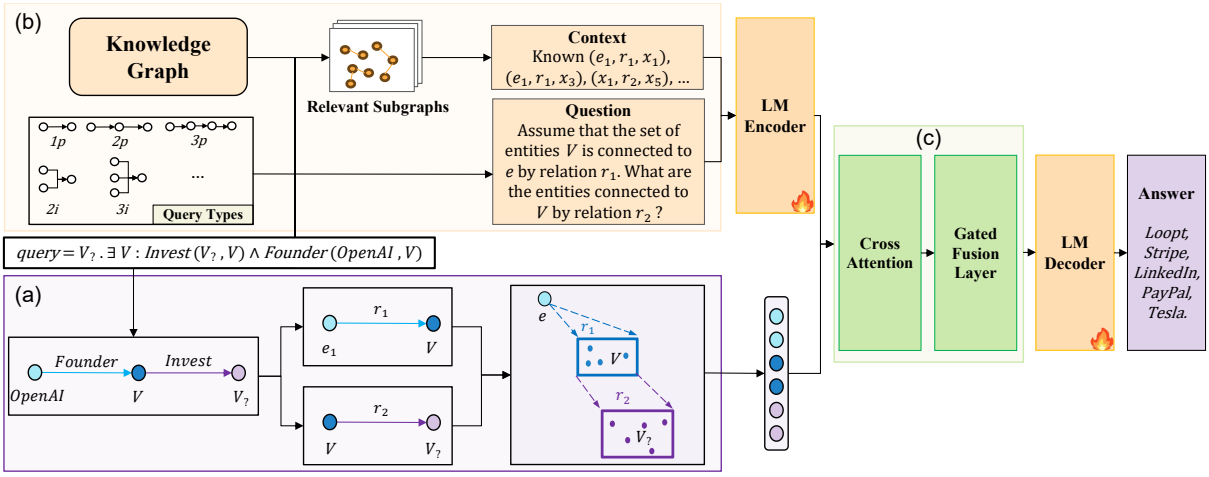

Figure 2: Overview of COB-LM. (a) Knowledge graph reasoning decomposes the query and represents reasoning sub-queries by mapping them to box embeddings, forming the chain of boxes; (b) prompts are constructed by retrieving subgraphs as context and converting the logical query into a natural language question, then language model reasoning is triggered by prompts. (c) Integrating the language model encoding and the chain of boxes via cross-attention and gated fusion layer for obtaining answers.

where A_{q_i} is the answer to the logical query q_i, defined as a set of entities $A_{q_i} = \{e|e \in E, q_i[e] \text{ holds true}\}$, $i = 1, 2$.

The logical reasoning process can further be modeled as a sequence of reasoning tasks, which is of executing logical operators along the order after the logical decomposition. For example, a $2p$ query can be decomposed as follows:

$$e_1 \xrightarrow{r_1} \xrightarrow{r_2} A \quad \Rightarrow \quad \mathrm{e}_1 \xrightarrow{r_1} A_1, A_1 \xrightarrow{r_2} A.$$

We follow the same decomposition mechanism as in previous work (Ren et al., 2020; Ren and Leskovec, 2020; Choudhary and Reddy, 2023) to decompose complex queries as a series of sub-queries. In the example shown above, $e_1 \xrightarrow{r_1} A_1$ is a sub-query, and $A_1 \xrightarrow{r_2} A$ is another sub-query.

3.2 The Proposed Method: COB-LM

The overall architecture of the proposed model **COB-LM** is shown in Figure 2. **COB-LM** consists of three modules, i.e., chain-of-box Construction, LM prompting, and answer generation. First, we utilize a geometric representation approach, i.e., box embeddings, to encode the KGs and preserve the structural knowledge by the geometric embeddings of sub-queries in the logical order as the chain of boxes. In parallel, we convert the logical queries into prompts using logical decomposition and encode them through a language model encoder. Finally, we propose to employ cross-attention and gated fusion mechanisms to fuse the chain of boxes into the language model encoding and pass them to the language model decoder to generate answers.

3.2.1 Chain-of-Box Construction

Due to the incompleteness issue of KGs, it is difficult to reason and answer complex multi-hop queries by directly traversing the KG. Instead, we propose to leverage the geometric representation learning methods, which transform queries and entities into geometric regions in the embedding space, and make logical operators as operations on geometric regions, i.e., relational projections and intersections. Updating the embedding regions according to the logical order promotes the final inclusion of answer entities in the mapping region of the logical query. To achieve effective logical reasoning and modeling, we define geometric regions as boxes (hyper-rectangles) (Ren et al., 2020). In the embedding space, the box embedding of a query q is defined as $\mathbf{q} = (\mathrm{Cen}(\mathbf{q}), \mathrm{Off}(\mathbf{q}))$, where $\mathrm{Cen}(\mathbf{q})$ denotes the center vector of the box and $\mathrm{Off}(\mathbf{q})$ denotes the offset vector of the box, then the region in the box is defined as follows:

$$\mathrm{Box}_{\mathbf{q}} \equiv \{\mathbf{e} : \mathrm{Cen}(\mathbf{q}) - \mathrm{Off}(\mathbf{q}) \preceq \mathbf{e} \preceq \mathrm{Cen}(\mathbf{q}) + \mathrm{Off}(\mathbf{q})\}. \tag{4}$$

For a relation $r \in R$, it is associated with relation embedding $\mathbf{r} = (\mathrm{Cen}(\mathbf{r}), \mathrm{Off}(\mathbf{r}))$. And for any entity $e \in E$ within the knowledge graph, its box embedding is defined as $\mathbf{e} = (\mathrm{Cen}(\mathbf{e}), 0)$, namely a zero-sized box. If it satisfies $\mathbf{e} \in \mathrm{Box}_{\mathbf{q}}$, it is considered as an answer entity to query q.

In the vector space, the initial box embedding of atomic q[1] and relation r are modeled as a new

[1]We use q to denote the query and decomposed sub-queries with a slight notation abuse.

region $\mathbf{q} + \mathbf{r}$ after the projection operator. And for the intersection operator, an attention operation is performed on the box centers and a sigmoid function is applied to update the offset vectors to obtain $\mathbf{q}_\cap = (\text{Cen}(\mathbf{q}_\cap), \text{Off}(\mathbf{q}_\cap))$.

$$\text{Cen}(\mathbf{q}_\cap) = \sum_i \mathbf{a}_i \odot \text{Cen}(\mathbf{q}_i), \tag{5}$$

$$a_i = \frac{\exp(\text{MLP}(\mathbf{q}_i))}{\sum_j \exp(\text{MLP}(\mathbf{q}_j))}, \tag{6}$$

$$\begin{aligned}\text{Off}(\mathbf{q}_\cap) =& \text{Min}(\text{Off}(\mathbf{q}_1), \ldots, \text{Off}(\mathbf{q}_n)) \\ & \odot \sigma(\text{DeepSets}(\mathbf{q}_1, \ldots, \mathbf{q}_n)),\end{aligned} \tag{7}$$

where a_i is the attention weight over the box center, $\odot$ represent the element-wise multiplication, $\text{MLP}(\cdot)$ is the Multi-Layer Perceptron, and $\text{DeepSets}(\cdot)$ is the permutation-invariant function (Zaheer et al., 2017; Hamilton et al., 2018).

Given a query embedding $\mathbf{q} = (\text{Cen}(\mathbf{q}), \text{Off}(\mathbf{q}))$ and an entity embedding $\mathbf{e}$, their distance is defined as (Ren et al., 2020):

$$d(\mathbf{e}, \mathbf{q}) = d_{\text{out}}(\mathbf{e}, \mathbf{q}) + \alpha\, d_{\text{in}}(\mathbf{e}, \mathbf{q}), \tag{8}$$

where $d_{\text{out}}(\mathbf{e}, \mathbf{q})$ denote the outer distance between the entity and the nearest box corner, while $d_{\text{in}}(\mathbf{e}, \mathbf{q})$ corresponds to the inner distance between the center of the box and its corner, and α is a fixed scalar coefficient and $d_{\text{out}}(\mathbf{e}, \mathbf{q})$ and $d_{\text{in}}(\mathbf{e}, \mathbf{q})$ are defined as follows:

$$\begin{aligned}d_{\text{out}}(\mathbf{e}, \mathbf{q}) =& \max(\mathbf{e} - \mathbf{q}_{\text{max}}, 0) \\ & + \max(\mathbf{q}_{\text{min}} - \mathbf{e}, 0),\end{aligned} \tag{9}$$

$$\begin{aligned}d_{\text{in}}(\mathbf{e}, \mathbf{q}) =& ||\min(\mathbf{q}_{max} - \max(\mathbf{q}_{\text{min}}, \mathbf{e})) \\ & - \text{Cen}(\mathbf{q})||,\end{aligned} \tag{10}$$

$$\mathbf{q}_{\text{max}} = \text{Cen}(\mathbf{q}) + \text{Off}(\mathbf{q}), \tag{11}$$

$$\mathbf{q}_{\text{min}} = \text{Cen}(\mathbf{q}) - \text{Off}(\mathbf{q}), \tag{12}$$

In order to make the answer embedding as close as possible to the query embedding, the loss of logical reasoning over KGs is defined as follows (Sun et al., 2019):

$$\begin{aligned}\mathcal{L} =& -\log\sigma(\gamma - d(\mathbf{e}, \mathbf{q})) \\ & - \sum_{k=1}^{K} \frac{1}{K} \log\sigma(d(\mathbf{e}_k^{'}, \mathbf{q}) - \gamma),\end{aligned} \tag{13}$$

where $\mathbf{q}$ is the query embedding, $\mathbf{e}$ is the positive answer embedding, and $\mathbf{e}'$ is the embedding of a negative answer, γ is a fixed margin.

By training the box embedding model, we characterize the reasoning process over knowledge graphs as a chain of boxes (COB). Specifically, for a EPFO query Q_{query} which can be decomposed as a set of sub-queries $\{q_1, ..., q_n\}$. We regard the projection area $\mathbf{q}_i$ of each step as the representation of a sub-query q_i and construct a list of regions $\{\mathbf{q}_1, ..., \mathbf{q}_n\}$ in logical order to preserve the entire reasoning process, as shown in Figure 2(a). To be more specific, we concatenate the a list of regions $\{\mathbf{q}_1, ..., \mathbf{q}_n\}$ in the logical order and obtain the representation of the chain of boxes as follow:

$$\mathbf{H}_C = \mathbf{q}_1 \oplus ... \oplus \mathbf{q}_n, \tag{14}$$

where $\oplus$ denotes the concatenation operation.

3.2.2 LM Prompting

In order to utilize the latent knowledge and reasoning ability of language models, we convert the logically decomposed query in Section 3.1 into a natural language question, and use it as the context part of the structural knowledge in knowledge graphs, consequently constructing the prompts for language models. Specifically, we perform neighborhood retrieval for the logical query (Choudhary and Reddy, 2023). Assuming that the original multi-hop query is decomposed into k sub-queries, let E_i represent the entities and R_i represent the relations in a sub-query q_i. The neighborhood retrieval process is defined as:

$$\mathcal{N}(q_i) = \left\{(e, r, e') | e \in E_i, r \in R_i, e' \in E_i\right\}, \tag{15}$$

$$E_i = \{e, e' | (e, r, e') \in \mathcal{N}(q_{i-1})\}, \tag{16}$$

$$R_i = \{r | (e, r, e') \in \mathcal{N}(q_{i-1})\}, \tag{17}$$

$$\mathcal{N}(q_1) = \left\{(e, r, e') | e \in E_1, r \in R_1, e' \in E_1\right\}. \tag{18}$$

As shown in Figure 2(b), we concatenate the neighborhoods retrieved by each sub-query as the context. When the length of the context exceeds the input limit, i.e., the input token limit of the language model, we stop adding contexts. The logical decomposition mechanism used above and the decomposed sub-queries align with the order in Section 3.2.1.

Inspired by the chain of thought (Wei et al., 2022), we use templates that convert the sub-queries obtained from the logical decomposition into step-by-step natural language questions in the logical order. The templates are provided in Appendix A.

Finally, as shown in Figure 2(b), we use the context and the question as the prompt $\mathcal{P}$ for feeding into the language model encoder, the encoder works by:

$$\mathbf{H}_{LM} = \Phi_E(\mathcal{P}), \tag{19}$$

where $\Phi_E(\cdot)$ denotes the LM encoder, and $\mathbf{H}_{LM}$ refers to the output of the encoder.

3.2.3 Answer Generation

With the encoding $\mathbf{H}_{LM}$ obtained from a language model encoder and the representation of the chain of boxes $\mathbf{H}_C$, the next step is to effectively integrate the chain of boxes into the language model decoder for answer generation. Specifically, we first propose to fuse the representation of the chain of boxes $\mathbf{H}_C$ into $\mathbf{H}_{LM}$ and leverage a single-head attention network which works by:

$$\mathbf{H}_{att} = \text{Softmax}\left(\frac{\mathbf{H}_{LM}\mathbf{W}_C\mathbf{H}_C^\top}{\sqrt{d}}\right)\mathbf{H}_C, \tag{20}$$

where d is the dimension of $\mathbf{H}_{LM}$ and $\mathbf{W}_C$ is a trainable projection matrix to project $\mathbf{H}_C$ to the same dimension as $\mathbf{H}_{LM}$.

Then we apply a gated fusion mechanism (Zhang et al., 2023, 2019; Wu et al., 2021; Li et al., 2022) to fuse the attention output $\mathbf{H}_{att}$ with LM encoding $\mathbf{H}_{LM}$. The final output $\mathbf{H}$ is defined as:

$$\mathbf{H} = (1-\lambda)\cdot\mathbf{H}_{LM} + \lambda\cdot\mathbf{H}_{att}, \tag{21}$$

$$\lambda = \text{Sigmoid}(\mathbf{W}_{LM}\mathbf{H}_{LM} + \mathbf{W}_{att}\mathbf{H}_{att}), \tag{22}$$

where $\mathbf{W}_{LM}$ and $\mathbf{W}_{att}$ are trainable weights.

Finally, we input the fused representation $\mathbf{H}$ into the language model decoder to predict the answers. The decoder works by:

$$\mathcal{A} = \Phi_D(\mathbf{H}), \tag{23}$$

The loss function for fine-tuning language models is defined as:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N}\log p(\mathcal{A}_i|\mathbf{H}_i;\theta), \tag{24}$$

where N is the number of training examples, A_i is the answer for the i-th example, $\mathbf{H}_i$ is the fused representation for the i-th example and θ are the model parameters.

3.3 Optimization

We implement the above process as an end-to-end framework, as shown in Figure 2. By fine-tuning the language model, we empower the language model with the capability to reason over KGs with the representation of the chain of boxes. The algorithmic process is given in Appendix B.

4 Experiments

4.1 Datasets

We adopt two widely used knowledge graphs FB15k-237 (Toutanova and Chen, 2015) and NELL995 (Xiong et al., 2017) for evaluation. Both datasets are English language datasets covering domains such as movies, music, sports, etc. The query datasets are provided by BetaE (Ren and Leskovec, 2020). More details about datasets can be found in Appendix C.

4.2 Baselines and Evaluation Merics

To validate the effectiveness of our method, we compared **COB-LM** with GQE (Hamilton et al., 2018), Query2Box (Ren et al., 2020), BetaE (Ren and Leskovec, 2020), CQD (Arakelyan et al., 2020), and FuzzQE (Chen et al., 2022) on both datasets. And following previous studies (Chen et al., 2022; Hamilton et al., 2018), we adopt the evaluation protocol and use Mean Reciprocal Rank (MRR) as the main evaluation metric for each answer corresponding to a logical query.

For query q, we denote the rank of the answer e by $\text{Rank}(e)$, then calculate Mean Reciprocal Rank (MRR) as follows:

$$\text{Metrics}(q) = \frac{1}{|\mathcal{A}|}\sum_{e\in\mathcal{A}}\frac{1}{(\text{Rank}(e))}, \tag{25}$$

where $|\mathcal{A}|$ is the cardinality of set $\mathcal{A}$.

4.3 Model Setup

We employ the FLAN-T5-small model with an encoder-decoder architecture as our base language model (Wei et al., 2021), which has been primarily used for natural language processing tasks. The language model is publicly available in the Huggingface library (Wolf et al., 2020), and allows for fine-tuning and deployment on consumer-grade GPUs. Implementation details can be found in Appendix C.

Table 1: MRR results (%) on answering EPFO queries. Performance comparisons are between ours and the baselines on FB15k-237 and NELL995. The best results are in bold.

Dataset	Models	Avg	1p	2p	3p	2i	3i	ip	pi	2u	up
FB15k-237	GQE	16.3	35.0	7.2	5.3	23.3	34.6	16.5	10.7	8.2	5.7
	Q2B	20.1	40.6	9.4	6.8	29.5	42.3	21.2	12.6	11.3	7.6
	BetaE	20.9	39.0	10.9	10.0	28.8	42.5	22.4	12.6	12.4	9.7
	CQD	21.7	46.3	9.9	5.9	31.7	41.3	21.8	15.8	14.2	8.6
	FuzzQE	24.2	42.2	13.3	10.2	33.0	47.3	26.2	18.9	15.6	10.8
	COB-LM (Ours)	**42.0**	**62.5**	**35.3**	**31.8**	**49.8**	**64.6**	**36.6**	**45.6**	**31.4**	**20.1**
NELL995	GQE	18.66	32.8	11.9	9.6	27.5	35.2	18.4	14.4	8.5	8.8
	Q2B	22.9	42.2	14	11.2	33.3	44.5	22.4	16.8	11.3	10.3
	BetaE	24.6	53	13	11.4	37.6	47.5	24.1	14.3	12.2	8.5
	CQD	28.4	60.0	16.5	10.4	40.4	49.6	27.6	20.8	16.8	12.6
	FuzzQE	29.3	58.1	19.3	15.7	39.8	50.8	28.1	21.8	17.3	13.7
	COB-LM (Ours)	**52.3**	**73.4**	**52.0**	**49.1**	**57.1**	**63.5**	**48.1**	**51.9**	**46.9**	**28.7**

4.4 Main Results

The results of all evaluated logical reasoning methods on two different KGs are shown in Table 1. We can see that **COB-LM** consistently outperforms the previous state-of-the-art baselines for all types of EPFO logical queries on both benchmark datasets, with average performance gains of 17.8% and 22.9% on FB15k-237 and NELL995, respectively. We also observe that this improvement is broad and stable, and also shows promising generalization on more complex logical query types, e.g., ip, pi, $2u$, up. The results validate the effectiveness of **COB-LM**. Therefore, we think that these results highlight the advantages of our designed chain of boxes to fuse the geometric representations of knowledge graphs in language model reasoning. This can be attributed to the fact that **COB-LM** utilizes both the structured knowledge of the knowledge graph and the latent knowledge of the language model.

4.5 Ablation Study

4.5.1 Impact of Chain-of-Box

In order to illustrate the impact of chain-of-box in our framework, we conduct comparative experiments on two datasets. Specifically, we compare **COB-LM** with two variants including **LM** and **LM-FT**. For **LM**, we only leverage a language model and convert logical queries into natural language prompts. As a result, new natural language prompts datasets are constructed. We then evaluate the original language model and the language model **LM-FT** fine-tuned using the natural language prompts datasets separately. The evaluation results are shown in Table 2.

Table 2: Performance comparisons between language model reasoning and COB-LM using MRR scores, where LM refers to the unfine-tuned language model, LM-FT refers to the fine-tuned language model.

Models	**1p**	**2p**	**3p**	**2i**	**3i**	**ip**	**pi**	**2u**	**up**
FB15k-237									
LM	37.9	16.5	15.8	14.0	9.0	16.4	12.8	17.9	8.7
LM-FT	61.3	28.5	25.7	46.9	61.4	30.8	38.2	27.6	18.5
COB-LM	**62.5**	**35.3**	**31.8**	**49.8**	**64.6**	**36.6**	**45.6**	**31.4**	**20.1**
NELL995									
LM	26.3	21.6	19.8	12.7	8.7	20.6	16.4	21.9	10.1
LM-FT	51.7	38.1	36.9	54.9	60.4	36.8	43.4	39.1	25.2
COB-LM	**73.4**	**52.0**	**49.1**	**57.1**	**63.5**	**48.1**	**51.9**	**46.9**	**28.7**

The original and unfine-tuned language model **LM** suffers from high-performance degradation, although the evaluation datasets have been converted into natural language form. Observing the answers **LM** incorrectly predicts, we find that it always outputs nonsensical texts. This observation illustrates its failure neither to demonstrate logical reasoning ability nor even to understand the prompts at all. The results of the fine-tuned language model **LM-FT** show a significant improvement in performance and no more nonsensical texts. The results prove that effective logical reasoning can be performed using the latent knowledge of the language model, and show the importance of fine-tuning the language model to understand the form of logical reasoning task.

Going further, after fusing the representation of the chain of boxes, the reasoning performance of the language model obtains a remarkable improvement on all the 9 types of logical queries. We argue that this is because the chain of boxes and

Table 3: Performance comparisons between Query2Box and COB-LM using MRR scores.

Models	1p	2p	3p	2i	3i	ip	pi	2u	up
FB15k-237									
Q2B	40.6	9.4	6.8	29.5	42.3	21.2	12.6	11.3	7.6
COB-LM	**62.5**	**35.3**	**31.8**	**49.8**	**64.6**	**36.6**	**45.6**	**31.4**	**20.1**
NELL995									
Q2B	42.2	14	11.2	33.3	44.5	22.4	16.8	11.3	10.3
COB-LM	**73.4**	**52.0**	**49.1**	**57.1**	**63.5**	**48.1**	**51.9**	**46.9**	**28.7**

Table 4: MRR scores of COB-LM using Flan-T5-small and UnifiedQA-small as the base language model.

Models	1p	2p	3p	2i	3i	ip	pi	2u	up
FB15k-237									
UnifiedQA	61.1	27.9	24.4	49.8	63.8	30.5	44.4	26.1	15.6
FLAN-T5	**62.5**	**35.3**	**31.8**	**49.8**	**64.6**	**36.6**	**45.6**	**31.4**	**20.1**
NELL995									
UnifiedQA	71.1	33.2	34.5	51.0	62.1	31.8	39.7	29.2	22.6
FLAN-T5	**73.4**	**52.0**	**49.1**	**57.1**	**63.5**	**48.1**	**51.9**	**46.9**	**28.7**

language model reasoning essentially depict two forms of the same reasoning process. More specifically, the representations of the chain of boxes over knowledge graphs have the advantage of the structured and deterministic nature, which is a lack of language model reasoning, and can therefore be used to empower the logical reasoning ability of language models. Overall, the comparative experiments highlight the potential of using the chain of boxes to empower language models, and our results suggest that fusing structural knowledge could be a promising way to empower language models for complex reasoning tasks.

4.5.2 Impact of Language Model

To illustrate the impact of the language model in **COB-LM**, we remove the language model and perform a comparative analysis. When only the method in Section 3.2.1 is employed, the module is similar to the Query2Box (Ren et al., 2020). It is worth noting that Query2Box method uses only the structural knowledge from KGs to reason using iterative embedding of boxes, after obtaining the query projection in the last step, it regards the entities closest to the final box as the answer.

The comparative results are summarized in Table 3. It is evident that with the introduction of the language model, the performance has improved remarkably, and the relative improvement is notably higher for complex multi-hop queries, e.g., the improvement is greater on logical query types $2p$ and $3p$ than on $1p$. This illustrates the excellent ability of the language model with chain-of-box to handle complex logical queries and the effectiveness of using the language model in **COB-LM**. The incorporation of the reasoning ability of language models for complex logical reasoning is not only a novel solution but will continue to benefit from the development of pre-trained language models.

4.5.3 Impact of Base Models

To evaluate the adaptability of **COB-LM**, we changed the base language model from FLAN-T5-small to UnifiedQA-small (Khashabi et al., 2020), which is another language model with fewer parameters than FLAN-T5-small. The results in Table 4 imply that when the number of parameters decreases, there is a slight decrease in the overall reasoning performance. However, the smaller UnifiedQA-small model still shows superior reasoning ability in the **COB-LM** framework, proving the adaptability of our approach to different base language models.

We hypothesize that integrating very large language models as the base language model for **COB-LM** could potentially result in performance enhancements. However, it is important to note that fine-tuning the very large language models is not applicable for logical reasoning tasks on domain-specific KGs due to the inaccessibility of high-performance GPU resources. Therefore, in our work, we adopt only small language models that can be easily used for downstream development and deployment.

5 Conclusion

In this paper, we proposed a novel framework **COB-LM** for logical query reasoning over KGs. In order to overcome the knowledge deficiency in KGs, the proposed **COB-LM** makes use of the latent knowledge of language models and integrates reasoning over knowledge graphs and language model reasoning. Specifically, we proposed to obtain the chain-of-box to represent the footprint of knowledge graph reasoning and fused it in the language models to enable the capability of logical reasoning. Extensive experimental results demonstrate the rationality and effectiveness of our proposed method on logical reasoning over KGs.

Limitations

Our approach requires more computational resources than using knowledge graph embedding methods or language model methods alone. And training for **COB-LM** could have an increased impact on the environment. In addition, our work focuses only on logical reasoning performance. While fine-tuning leads to performance improvements on the current logical reasoning tasks, it may also lead to performance degradation on other tasks (Kotha et al., 2023).

References

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2020. Complex query answering with neural link predictors. *arXiv preprint arXiv:2011.03459*.

Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. 2022. Answering complex logical queries on knowledge graphs via query tree optimization. *arXiv preprint arXiv:2212.09567*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Xuelu Chen, Ziniu Hu, and Yizhou Sun. 2022. Fuzzy logic based logical query answering on knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3939–3948.

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

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

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*, 16:523–544.

Brian A Davey and Hilary A Priestley. 2002. *Introduction to lattices and order*. Cambridge university press.

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

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

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with numpy. *Nature*, 585(7825):357–362.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. 2023. Understanding catastrophic forgetting in language models via implicit inference. *arXiv preprint arXiv:2309.10105*.

Bei Li, Chuanhao Lv, Zefan Zhou, Tao Zhou, Tong Xiao, Anxiang Ma, and JingBo Zhu. 2022. On vision features in multimodal machine translation. *arXiv preprint arXiv:2203.09173*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. *arXiv preprint arXiv:2104.06683*.

Ridho Reinanda, Edgar Meij, Maarten de Rijke, et al. 2020. Knowledge graphs: An information retrieval perspective. *Foundations and Trends® in Information Retrieval*, 14(4):289–444.

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

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

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

Joshua Shinavier, Kim Branson, Wei Zhang, Shima Dastgheib, Yuqing Gao, Bogdan Arsintescu, Fatma Özcan, and Edgar Meij. 2019. Panel: Knowledge graph industry applications. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 676–676.

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*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

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

Taylor Webb, Keith J Holyoak, and Hongjing Lu. 2023. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

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. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhiyong Wu, Lingpeng Kong, Wei Bi, Xiang Li, and Ben Kao. 2021. Good for misconceived reasons: An empirical revisiting on the need for visual context in multimodal machine translation. *arXiv preprint arXiv:2105.14462*.

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

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. 2022. Gammae: Gamma embeddings for logical queries on knowledge graphs. *arXiv preprint arXiv:2210.15578*.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. *Advances in neural information processing systems*, 30.

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.

Zhuosheng Zhang, Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, Zuchao Li, and Hai Zhao. 2019. Neural machine translation with universal visual representation. In *International Conference on Learning Representations*.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Prompt Templates

The prompt templates for the language model are provided in Tables 5.

B Fine-tune Procedure

We adopt the encoder-decoder architecture of the T5 language model (Raffel et al., 2020), which provides flexibility for fine-tuning. The complete procedure is shown in Algorithm 1.

Algorithm 1 Fine-tune COB-LM

Input: Text question prompt $\mathcal{P}$, the query query q
Output: Generated reasoning answer $\mathcal{A}$

1: Encode the prompt input and embed the query to respectively obtain $\mathbf{H}_{\text{LM}}$ and $\mathbf{H}_{\text{C}}$
2: Construct the interaction between the prompt and chain of box representations using cross attention to get $\mathbf{H}_{\text{att}}$
3: Fuse $\mathbf{H}_{\text{LM}}$ with $\mathbf{H}_{\text{att}}$ using a gated fusion mechanism, resulting in $\mathbf{H}$
4: Decode the fused representation $\mathbf{H}$ to obtain the target prediction $\mathcal{A}$
5: **return** $\mathcal{A}$

C Implementation Details

C.1 Datasets

We use two standard KG benchmark datasets:

• FB15k-237 (Toutanova and Chen, 2015) : It is a challenging benchmark dataset based on the large-scale knowledge graph project Freebase, containing 14,505 entities, 237 relations, and 310,079 triples. The sizes of the training, validation, and test sets for FB15k-237 are 164657, 25101, and 27812, respectively

• NELL995 (Xiong et al., 2017): This is a dataset created using a machine learning system, namely Never-Ending Language Learning (NELL) system, which contains 63,631 entities, 200 relations, and 142,804 triplets. The sizes of the training, validation, and test sets for NELL995 are 118780, 20927, and 21034, respectively

The datasets are provided by BetaE (Ren and Leskovec, 2020), which contain 9 types of EPFO queries. Compared to the earlier datasets provided by Query2Box, it improves the answer constraints of the queries and regenerate data of the 9 query types , making the data more realistic and challenging. In particular, training set involves 5 distinct query types: $1p$, $2p$, $3p$, $2i$, $3i$. For evaluation, we use a total of nine query structures, including the original 5 query types and 4 new query types that are never seen during training: ip, pi, $2u$, up, to assess the model's generalization ability (naming conventions: p for projection, i for intersection, u for union).

Both datasets are publicly available, licensed "as-is". They do not contain any sensitive or personal information about individuals or entities.

C.2 Baselines

We study the performance of the COB-LM framework by evaluating against five baselines.

• GQE (Hamilton et al., 2018) : GQE encodes the query as a single vector in a low-dimensional space.

• Query2Box (Ren et al., 2020) : Query2Box extends the embedding to a box region of vector space for answering existential positive first-order logic queries

• BetaE (Ren and Leskovec, 2020) : BetaE uses beta distributions to capture uncertainty in query and answer set.

• CQD (Arakelyan et al., 2020), a method that decomposes the logical queries to multiple atomic

queries, employs a pre-trained link predictor to solve, and independently evaluates atom predicates.

• FuzzQE (Chen et al., 2022), which proposes a logical query embedding framework that satisfies the laws of logic and uses learning-free logical operators.

For both BetaE and CQD, we benchmark against their respective model variants with enhanced general performance, namely $\text{BetaE}_{\text{DNF}}$ and CQD-BEAM.

C.3 Chain-of-Box Construction

We set the embedding dimension to $d = 400$ and configure the parameters $\gamma = 24$ and $\alpha = 0.2$ for the loss.

The batch size is 512 and the training epoch is 300. For each of these queries in the batch, we select one answer entity and 128 negative entities. We use the Adam optimizer (Kingma and Ba, 2014) to minimize the loss with a learning rate of 0.0001.

C.4 Language Model

FLAN-T5-small is a language model with 80 million parameters. We fine-tune the model up to 3 epochs. The learning rate is 0.00005 and the batch size is 16. We implemented COB-LM in Pytorch (Paszke et al., 2019). We also leveraged NLTK (Bird et al., 2009) and NumPy (Harris et al., 2020), and run on 4 NVIDIA GeForce RTX 3090 GPUs with 40 GB VRAM. The total GPU hours consumed for the fine-tuning process was about 30 hours. Results are reported from a single run. We will make the code publicly available upon acceptance to facilitate reproduction and further research.

Table 5: Prompt Templates.

Type	Logical Query	Template for Full Prompts
Context	$\mathcal{N}_k(q_\tau[Q_\tau])$	Known $(h_1, r_1, t_1), (h_2, r_2, t_2), (h_3, r_3, t_3),$ $(h_4, r_4, t_4), (h_5, r_5, t_5), (h_6, r_6, t_6)$
1p	$\exists X.r_1(X, e_1)$	Which entities are connected to e_1 by relation r_1?
2p	$\exists X.r_1(X, \exists Y.r_2(Y, e_1)$	Assume that the set of entities E is connected to entity e_1 by relation r_1. Then, what are the entities connected to E by relation r_2?
3p	$\exists X.r_1(X, \exists Y.r_2(Y, \exists Z.r_3(Z, e_1)$	Assume that the set of entities E is connected to entity e_1 by relation r_1 and the set of entities F is connected to entities in E by relation r_2. Then, what are the entities connected to F by relation r_3?
2i	$\exists X.[r_1(X, e_1) \wedge r_2(X, e_2)]$	Assume that the set of entities E is connected to entity e_1 by relation r_1 and the set of entities F is connected to entity e_2 by relation r_2. Then, what are the entities in the intersection of set E and F, i.e., entities present in both F and G?
3i	$\exists X.[r_1(X, e_1) \wedge r_2(X, e_2) \wedge r_3(X, e_3)]$	Assume that the set of entities E is connected to entity e_1 by relation r_1, the set of entities F is connected to entity e_2 by relation r_2 and the set of entities G is connected to entity e_3 by relation r_3. Then, what are the entities in the intersection of set E, F and G, i.e., entities present in all E, F and G?
ip	$\exists X.r_3(X, \exists Y.[r_1(Y, e_1) \wedge r_2(Y, e_2)]$	Assume that the set of entities E is connected to entity e_1 by relation r_1, F is the set of entities connected to entity e_2 by relation r_2, 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 r_3?
pi	$\exists X.[r_1(X, \exists Y.r_2(Y, e_2)) \wedge r_3(X, e_3)]$	Assume that the set of entities E is connected to entity e_1 by relation r_1, F is the set of entities connected to entities in E by relation r_2, and G is the set of entities connected to entity e_2 by relation r_3. Then, what are the entities in the intersection of set F and G, i.e., entities present in both F and G?
2u	$\exists X.[r_1(X, e_1) \vee r_2(X, e_2)]$	Assume that the set of entities E is connected to entity e_1 by relation r_1 and F is the set of entities connected to entity e_2 by relation r_2. Then, what are the entities in the union of set F and G, i.e., entities present in either F or G?
up	$\exists X.r_3(X, \exists Y.[r_1(Y, e_1) \vee r_2(Y, e_2)]$	Assume that the set of entities E is connected to entity e_1 by relation r_1 and F is the set of entities connected to entity e_2 by relation r_2. G is the set of entities in the union of E and F. Then, what are the entities connected to entities in G by relation r_3?