KaeDe: Progressive Generation of Logical Forms via Knowledge-Aware Question Decomposition for Improved KBQA

Ranran Bu[1], Jian Cao[1,*], Jianqi Gao[1], Shiyou Qian[1], Hongming Cai[1]
[1]Shanghai Jiao Tong University
{buranran,cao-jian,193139,qshiyou,hmcai}@sjtu.edu.cn

Abstract

Knowledge base question answering (KBQA) refers to the task of answering natural language questions using large-scale structured knowledge bases (KBs). Existing semantic parsing-based (SP-based) methods achieve superior performance by directly converting questions into structured logical form (LF) queries using fine-tuned large language models (LLMs). However, these methods face the key challenge of difficulty in directly generating LFs for complex graph structures, which often leads to non-executable LFs that negatively impact overall KBQA performance. To address this challenge, we propose **KaeDe**, a novel generate-then-retrieve method for KBQA. This approach integrates **K**nowledge-**a**ware qu**e**stion **De**composition and subsequent progressive LF generation within the generation phase, followed by an unsupervised retrieval phase. Specifically, the original question is decomposed into simplified, topic entity-centric sub-questions and explanations within the KB context. Path-level LFs are derived from these intermediate expressions and then combined into a comprehensive graph-level LF. Finally, the LF is refined through unsupervised entity and relation retrieval. Experimental results demonstrate that our method achieves state-of-the-art (SOTA) performance on WebQuestionSP (WebQSP) and ComplexWebQuestions (CWQ) benchmarks, particularly with fewer model parameters. The code is available at https://github.com/pvfeldt/KaeDe.

1 Introduction

Knowledge base question answering (KBQA) refers to the task of retrieving answers to natural language questions leveraging factural information from large-scale knowledge bases (KBs). In KBs such as Freebase (Bollacker et al., 2008) and Wikidata (Vrandečić and Krötzsch, 2014), entities are connected through relations, forming structured knowledge graphs (KGs).

Building on graph structures, most KBQA methods, including both information retrieval-based (IR-based) (Shi et al., 2021; Zhang et al., 2022; He et al., 2021; Chen et al., 2019) and semantic parsing-based (SP-based) (Chen et al., 2021; Ye et al., 2022; Gu and Su, 2022) approaches, emphasize graph-based retrieval to obtain answers. Specifically, SP-based approaches generate concise structured expressions, such as logical forms (LFs), which can be transformed into SPARQL queries for further execution in KBs. These LFs represent the retrieval subgraphs, which are composed of reasoning paths originating from each topic entity mentioned in the question. As a result, errors in the subgraph description lead to non-execution, which signifies the failure to derive any answer after execution, posing a significant challenge for SP-based methods.

Most existing SP-based methods adopt retrieve-then-generate strategies, where relevant information is first retrieved from KBs and then used to construct the LFs (Shu et al., 2022; Zhang et al., 2023b; Tian et al., 2024; Xiong et al., 2024; Feng and He, 2025). However, the retrieval complexity introduces additional computational overhead and distracting noise, with the performance of subsequent generation being heavily dependent on this prior process.

Apart from these traditional methods, novel generate-then-retrieve strategies (Luo et al., 2024a; Wang and Qin, 2024) for direct LF generation have been proposed and demonstrated to be effective, leveraging the remarkable reasoning capabilities of large language models (LLMs) (Wei et al., 2022b; Huang and Chang, 2023). In this context, high-quality LFs are directly generated using instruction-tuned (Wei et al., 2022a) LLMs, which are equipped with knowledge from specific domains, and refined with minimal retrieval, primarily for entity linking or additional relation align-

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 10958–10973

Model	Overall		Hop=3		Hop=4	
	Hits@1	F1	Hits@1	F1	Hits@1	F1
Llama 2 7B	81.37	76.82	81.58	76.56	72.22	68.81
Llama 2 13B	85.27	81.55	85.15	79.49	73.70	69.20

Table 1: Limitations of direct LF generation for complex questions from CWQ dataset.

ment. Yet, **(1) these LLMs often fail to focus on all the details in complex questions**. To statistically illustrate the problem, we reproduce experiments on direct LF generation followed by retrieval with a beam size of 8, using Llama 2 (Touvron et al., 2023) 7B and 13B models on the ComplexWebQuestions (CWQ) dataset (Talmor and Berant, 2018), as shown in Table 1. A significant drop is observed in more complex questions involving multiple hops, based on the same models. Furthermore, the reduction in model parameters leads to a 3.90% decline in Hits@1 and a 4.73% decrease in F1 scores, indicating the performance reliance on the model sizes.

To alleviate the performance decline in complex questions, prior research in question answering (QA) suggests simplifying the process by decomposing the questions into more manageable sub-questions and addressing each one separately (Zhang et al., 2024; Yixing et al., 2024). Specifically, the original questions are initially decomposed using tree-based (Zhang et al., 2023a; Huang et al., 2023), rule-based (Hu et al., 2021) and other strategies (Zhang et al., 2019), followed by IR-based or SP-based processes applied individually to each sub-question. However, **(2) these decomposition processes often prioritize the original question itself, with less emphasis on grounding in the knowledge from KBs**, which may introduce potential inaccuracies in the subsequent procedures due to the semantic gap between the original question and the specific knowledge. Under the condition of following SP-based processes, errors in the sub-questions may lead to schema errors during the generation of intermediate LFs and, further, result in the non-executability of the final LF.

Therefore, we propose **KaeDe**, a novel generate-then-retrieve approach for KBQA. The method integrates ***K***nowledge-***a***ware qu***e***stion ***De***composition and subsequent progressive LF generation within the generation phase, followed by an unsupervised retrieval phase. Specifically, to address **Issue (1)**, we break down the complex LF generation into simpler tasks through question decomposition. In this approach, sub-questions and explanations, starting from each potential topic entity, are generated by a fine-tuned LLM within the KB context to tackle **Issue (2)**. Guided by these intermediate expressions, path-level LFs are generated and assembled into a complete graph-level LF using the same LLM. Finally, the graph-level LF is refined during the retrieval phase and executed within the KBs to obtain answers.

Our contributions can be summarized as follows:

- We propose KaeDe framework, which splits LF generation into knowledge-aware question decomposition and progressive LF generation. This approach effectively reduces the complexity of generating an LF at once by breaking it into simpler tasks.
- We innovatively decompose the original question into simpler sub-questions and explanations grounded in the KB context. These intermediate expressions narrow the semantic gap between the original question and knowledge from the KBs, serving as effective guidance for the subsequent procedures.
- We conduct experiments on WebQuestionSP (WebQSP) and CWQ datasets. KaeDe outperforms strong baseline methods, achieving state-of-the-art (SOTA) performance, particularly with fewer model parameters.

2 Related Works

2.1 Semantic Parsing-Based Methods

SP-based methods interpret questions into structured expressions including LFs and S-expressions. Traditional SP-based methods employ a retrieve-then-generate strategy (Chen et al., 2022; Hu et al., 2022b; Xie et al., 2022; Das et al., 2021; Yih et al., 2016a; Patidar et al., 2024; Nie et al., 2024; Li et al., 2024; Faldu et al., 2024), where relevant information is first retrieved from KBs and then organized to construct the final LF. Early RnG-KBQA method (Ye et al., 2022) first enumerates and ranks retrieval candidates, then performs LF generation using a T5-based (Raffel et al., 2020) sequence-to-sequence (seq2seq) model. To further optimize performance, TIARA (Shu et al., 2022) and FC-KBQA (Zhang et al., 2023b) divide the retrieval into multiple grains, handling entities, relations, and logical skeletons separately. Recently,

the LF generator has been replaced by training-free in-context learning using closed-source LLMs (e.g., GPT). ARG-KBQA (Tian et al., 2024) completes the generation task using few-shot learning, based on reference reasoning paths retrieved from KBs, while Interactive-KBQA (Xiong et al., 2024) follows a step-wise generation approach, retrieving relevant information for intermediate steps and gradually forming the final query. However, these retrieve-then-generate methods suffer from significant computational overhead during retrieval. To address this, novel generate-then-retrieve approaches (Luo et al., 2024a; Wang and Qin, 2024) have been proposed, adapting open-source LLMs to specific knowledge via fine-tuning. TFS-KBQA (Wang and Qin, 2024) and ChatKBQA (Luo et al., 2024a) directly generate LFs based on input questions, followed by minimal entity or relation retrieval. While remarkable performance has been achieved, particularly by ChatKBQA, LLMs still fail to focus on all the details in complex questions. To alleviate this problem, there is significant reliance on larger model parameters for complex questions. This results in a substantial increase in both training and inference costs compared to smaller models. The solution for handling complex questions with smaller models is the key focus of our study.

2.2 Question Decomposition Methods

Question decomposition methods aim to reduce the complexity of the original question by breaking it down into simpler sub-questions and addressing them individually using IR-based or SP-based approaches to derive the final answers (Zhang et al., 2024, 2023a, 2019). Earlier methods such as SimpQA (Talmor and Berant, 2018) and DecompRC (Min et al., 2019) split original questions into two sub-questions using pointer network. To further generalize the decomposition for various types of questions, EDG (Hu et al., 2021) constructs entity-centric graph structures based on rules to represent complex questions. Since rule-based methods suffer from limited coverage and a lack of global awareness, the recent QDT (Huang et al., 2023) optimizes graph structure construction by preserving the original question's contents and employing a concise, linearized representation to enable flexible, neural-based generation. However, for these methods, question decomposition serves as the initial step, focusing more on the original expression and lacking knowledge alignment with the KB, which may lead to failures in the subsequent processes. Thus, the recent CoQ (Yixing et al., 2024) adopts a joint framework that integrates answer prediction and LF generation, where the prediction is based on chain-of-thought-prompted (CoT-prompted) step-wise question decomposition. The method is closely grounded in KBs due to the retrieval process in LF generation, but the retrieve-then-generate sequence introduces additional computational overhead and noise. Hence, combining question decomposition with knowledge from KBs under the generate-then-retrieve strategy is the primary objective of our study to reduce question complexity.

3 Preliminaries

3.1 Knowledge Bases

A KB $\mathcal{K}$ is composed of large-scale resource description framework (RDF) KGs $\mathcal{G}$ where knowledge is stored in the form of subject-predicate-object triplets $\mathcal{G} = \{(s, r, o)|s \in \mathcal{E}, r \in \mathcal{R}, o \in \mathcal{E} \cup \mathcal{I}\}$, with $\mathcal{E}$ standing for the set of entities, $\mathcal{R}$ representing the set of relations and $\mathcal{I}$ denoting the set of literal. Within KBs, each entity $e \in \mathcal{E}$ is stored as an ID starting with "*m.*" or "*g.*". Each relation $r \in \mathcal{R}$ is hierarchically classified with meta descriptions in KBs, typically presented as "$a.b.c$", where a, b and c denote label, domain and range, respectively.

3.2 Logical Forms and SPARQL Queries

LFs are structured expressions that are closer to natural language than SPARQL queries and, as such, are commonly used as intermediate generation targets for SP-based KBQA. We define a `convert`$(\cdot)$ function that transforms a SPARQL query S into the corresponding LF L consisting of three main procedures: entity-centric per-hop projection, operator arrangement, and entity substitution. Through projection, SPARQL lines can be mapped to $JOIN$ clauses for each hop. After forming the path-level LFs for each entity, operators such as AND and $ARGMAX$ are applied to combine these LFs into the final graph-level LF. Finally, the entity IDs are replaced by the corresponding labels in the expression. Details are presented in Appendix A.

4 Methods

In this section, we provide a detailed overview of our proposed KaeDe method. Following the

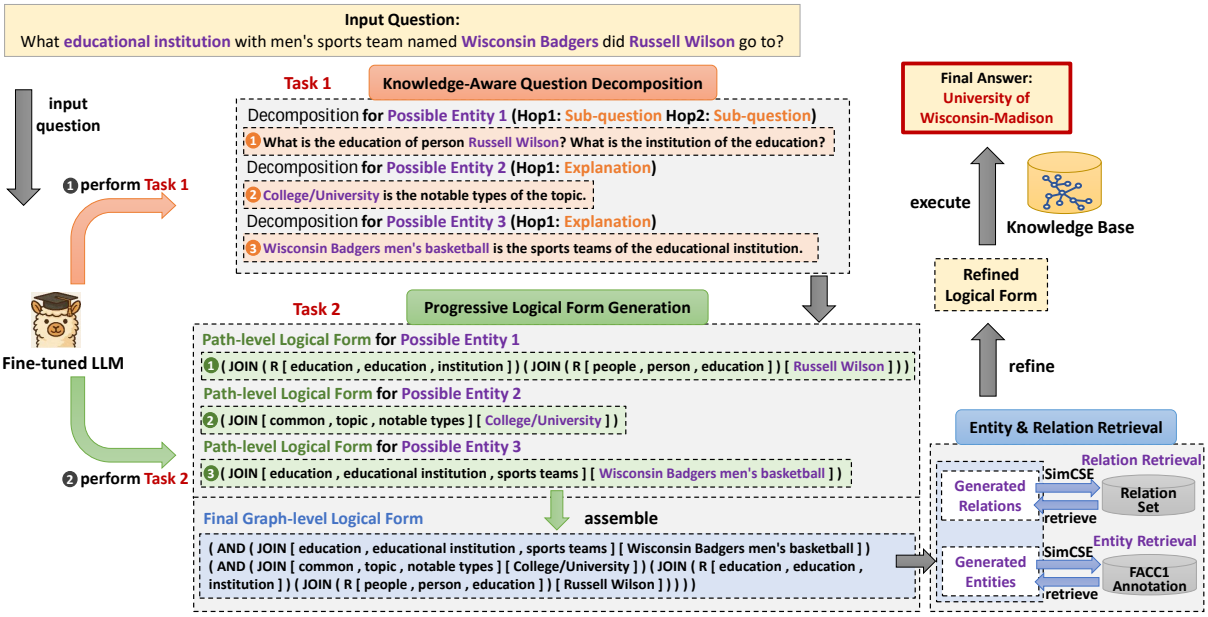

Figure 1: Overview of KaeDe. (1) Knowledge-aware question decomposition and progressive LF generation: Leveraging fine-tuned LLM, the original question is first decomposed into entity-centric simple expressions within the KB context. Following this, path-level LFs are generated through the same LLM based on these expressions and then assembled into the complete graph-level LF. (2) Entity and relation retrieval: The LF undergoes refinement through retrieval and is executed within the KBs for answers.

generate-then-retrieve strategy, KaeDe consists of two main phases: (1) the LLM-based generation phase, which includes Task 1: knowledge-aware question decomposition, and Task 2: progressive LF generation, and (2) the unsupervised retrieval phase, which focuses on entity and relation retrieval.

4.1 Data Preparation

Throughout the generation phase, the same LLM is equipped with the capabilities to handle three sequential sub-tasks: question decomposition for Task 1, and path-level LF generation and graph-level assembly for Task 2.

To achieve this, we reorganize the original datasets into D_{d}, D_{g}, and D_{a} for the decomposition, generation, and assembly tasks, respectively. Specifically, the training sets can be expressed as $D_{\text{d,train}} = \{(p_{\text{d}}, X) | \, q \in p_{\text{d}}\}$, $D_{\text{g,train}} = \{(p_{\text{g}}, \ell) | \, q, X \in p_{\text{g}}\}$ and $D_{\text{a,train}} = \{(p_{\text{a}}, L) | \, q, \ell \in p_{\text{a}}\}$, where p is the prompt, q denotes the original question, $X = \{x | \, e \in x\}$ represents the set of entity-centric sub-questions and explanations x for each topic entities $e \in E$ (E is the set of all topic entities from q), $\ell = \{l | \, e \in l\}$ signifies the set of corresponding path-level LFs l, and L stands for the final graph-level LF.

For the LFs, we process the original SPARQL query S by splitting into reasoning paths Z, each starting with topic entity e. Each $z \in Z$ is converted to its corresponding path-level LF $l \in \ell$, and S is transformed to the graph-level LF L using function $\texttt{convert}(\cdot)$ mentioned in the Preliminary section, as shown in Figure 2. The details of the conversion are provided in Appendix A.

In terms of intermediate expressions X decomposed from q, we apply a rule-based arrangement to each z, specifically using hop-wise linearization to prevent potential hop error accumulation in subsequent processes. Rather than performing a rough linearization by directly concatenating per-hop information into a subject-predict-object sequence as in (Oguz et al., 2022; Yu et al., 2022), we leverage the semantics of triplets to generate intermediate expressions that are closer to comprehensible natural language. As triplet "(e_0, r, e_1)" expresses the statement "*The r of e_0 is e_1.*", we define the rules as demonstrated in Figure 2. The expression of a question or statement depends on the direction of retrieval, in other words, whether the required entity is the object or the subject. Then, the hierarchical relation $r = r_1.r_2.r_3$ can be reconstructed into the phrase "*the r_3 of r_2*" to preserve and clearly convey the underlying semantic structure. Finally, the given entity is integrated with the relation phrase into the intermediate expression. If

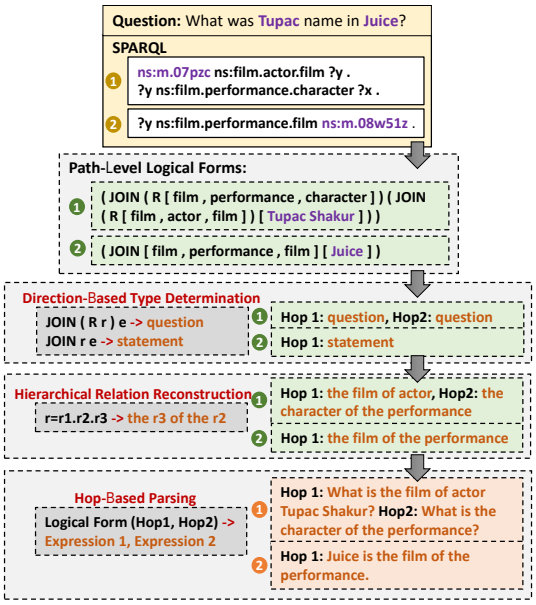

Figure 2: Preparation of training set data for path-level LFs and decomposed intermediate expressions.

multiple hops exist, the given entity only appears in the initial hop.

These contents mentioned above are incorporated into the prompts based on the templates in Appendix B to generate the dataset. Notably, the expressions and path-level LFs for each topic entity are combined in a single prompt to avoid additional computational overhead during further beam search in each process.

4.2 Instruction Tuning

Building on $D_{\text{train}} = D_{\text{d,train}} \cup D_{\text{g,train}} \cup D_{\text{a,train}}$, we perform instruction tuning (Wei et al., 2022a) with low-rank adaption (LoRA) (Hu et al., 2022a) parameter-efficient fine-tuning (PEFT) to equip LLM with task-specific abilities grounded in the target KBs.

Since the three tasks are sequential processes, each with entity-centric contents combined in a single prompt, the probability can be expressed as:

$$P_\Theta(L|q,\mathcal{K}) = \underbrace{P_\Theta(L|\ell)}_{\text{assembly}} \underbrace{P_\Theta(\ell|X)}_{\text{generation}} \underbrace{P_\Theta(X|q,\mathcal{K})}_{\text{decomposition}}, \tag{1}$$

where Θ signifies the parameters of the LLM and $\mathcal{K}$ denotes the KB. $P_\Theta(X|q,\mathcal{K})$ expresses that original question q is decomposed into expressions X grounded in $\mathcal{K}$. $P_\Theta(\ell|X)$ and $P_\Theta(L|\ell)$ describe that the path-level LFs ℓ are generated based on X and then assembled into graph-level LF L.

We formulate the tasks into an optimization problem, aiming to maximize the probability. When combined with LoRA (Hu et al., 2022a), the objective can be described as:

$$\max_\theta \sum_{(p,o)\in D_{\text{train}}} \sum_{t=1}^{|o|} log(P_{\Phi_0+\Delta\Phi(\theta)}(o_t|p,o_{<t})), \tag{2}$$

where o summarizes for the output corresponding to each prompt (with o being either X, ℓ or L), Φ_0 signifies the pre-trained weight of LLM, θ represents the set of trainable parameters updated during fine-tuning (which is small amount for LoRA), and $\Delta\Phi(\theta)$ expresses the task-specific parameter adjustment after PEFT.

Algorithm 1 Generation and Retrieval

Require: KB $\mathcal{K}$, original question q
$\hat{\mathcal{X}} \leftarrow \texttt{LLM}(q|\mathcal{K})$ $(\hat{X} \in \hat{\mathcal{X}}, \hat{x} \in \hat{X}, \hat{e} \in \hat{x})$
for each $\hat{X}$ in $\hat{\mathcal{X}}$ **do**
  $\hat{\ell}' \leftarrow \texttt{LLM}(\hat{X})$ $(\hat{\ell} \in \hat{\ell}', \hat{l} \in \hat{\ell}, \hat{e} \in \hat{l})$
  for each $\hat{\ell}$ in $\hat{\ell}'$ **do**
    $\hat{\mathcal{L}} \leftarrow \texttt{LLM}(\hat{\ell})$ $(\hat{L} \in \hat{\mathcal{L}})$
  end for
end for
$\mathcal{A} \leftarrow \emptyset$
for each $\hat{L}$ in $\hat{\mathcal{L}}$ **do**
  $A_{\text{e}} \leftarrow \texttt{retrieve\_entity}(\hat{L}|\mathcal{K})$
  $\mathcal{A} += A_{\text{e}}$
end for
if $\mathcal{A} == \emptyset$ **then**
  for each $\hat{L}$ in $\hat{\mathcal{L}}$ **do**
    $A_{\text{r}} \leftarrow \texttt{retrieve\_relation}(\hat{L}|\mathcal{K})$
    $\mathcal{A} += A_{\text{r}}$
  end for
end if
return $\mathcal{A}$

4.3 Logical Form Generation

After the LLM has been fine-tuned, we adopt beam search to perform the multiple tasks.

As shown in Algorithm 1, a series of grouped expressions $\hat{\mathcal{X}}$ are initially generated by the LLM based on the original question q as a result of the multiple beams, where each $\hat{X} \in \hat{\mathcal{X}}$ contains individual expressions $\hat{x}$ for each possible topic entity $\hat{e}$. Utilizing the same LLM, each $\hat{X}$ is mapped to the set of path-level LFs $\hat{\ell}'$ where each $\hat{\ell} \in \hat{\ell}'$ is composed of corresponding path-level LF $\hat{l}$ for each expression $\hat{x}$. Finally, the candidate graph-level LFs $\hat{\mathcal{L}}$ are assembled with each $\hat{\ell}$, where operators are arranged, including intersections for different paths and constraints for values.

4.4 Unsupervised Retrieval and Execution

Following previous work (Luo et al., 2024a), we implement the unsupervised retrieval process with

off-the-shelf SimCSE (Gao et al., 2021) retriever to refine the generated LFs, as also illustrated in Algorithm 1.

The function $\texttt{entity\_retrieve}(\cdot)$ summarizes the entity linking and execution procedures. We compare the similarity with $s_{\text{e}} = \texttt{sim}(\hat{e}, e)$, where $\hat{e}$ is the possible entity predicted and e denotes the entity in entity set $\mathcal{E}$. Candidates with top-k scores, constrained by a certain threshold, are substituted into the generated LF $\hat{L}$, resulting in the refined $\hat{L}'_{\text{e}}$. FACC1 annotation (Gabrilovich et al., 2013) is primarily used for entity linking. Based on this, two scenarios are considered: one with oracle entity linking and the other without. $\hat{L}'_{\text{e}}$ is then transformed back into a SPARQL query and executed in KB $\mathcal{K}$ for answers A_{e}, which can be expressed as:

$$A = \texttt{execute}(\texttt{convert}^{-1}(\hat{L}'|\mathcal{K})). \quad (3)$$

Each A_{e} is added to answer set $\mathcal{A}$. If $\mathcal{A} = \emptyset$, indicating that none of the refined LFs are executable after entity retrieval, relation retrieval $\texttt{relation\_retrieval}(\cdot)$ is then performed for further alignment. We compute the similarity with $s_{\text{r}} = \texttt{sim}(\hat{r}, r)$, where $\hat{r}$ is the generated relation, and r indicates the relation in relation set $\mathcal{R}$. Specifically, we adopt the neighboring relations for topic entity $\hat{e}$ as $\mathcal{R}$. The subsequent processes mirror those of entity retrieval, where the top-k candidates, contrained by a threshold, are used to update the LFs and the corresponding $\hat{L}'_{\text{r}}$ are executed with Equation 3 to retrieve answers A_{r}, which are then added to $\mathcal{A}$. Finally, $\mathcal{A}$ is returned as the set of final answers.

5 Experiments

5.1 Experiment Setup

Datasets. We use WebQSP (Yih et al., 2016b) and CWQ (Talmor and Berant, 2018) as representative KBQA benchmarks. Both datasets can be reasoned on the Freebase KB (Bollacker et al., 2008), with each question paired with a SPARQL query. WebQSP contains 4,737 simple natural language questions, while CWQ includes 34,689 more complex questions with higher hop and entity number. Detailed statistics of the datasets are provided in Appendix C.

Baselines. We adopt both IR-based and SP-based methods as baselines, with each category further subdivided into non-LLM and LLM-based methods. For IR-based methods, we select NSM (He et al., 2021), Rigel (Sen et al., 2021), UniK-QA (Oguz et al., 2022) and UniKGQA (Jiang et al., 2023) as non-LLM IR methods, and RoG (Luo et al., 2024b), ToG (Sun et al., 2024) and FiDeLis (Sui et al., 2024) as LLM-based IR methods. For SP-based methods, we select HGNet (Chen et al., 2022), TIARA (Shu et al., 2022) and FC-KBQA (Zhang et al., 2023b) as non-LLM SP methods, as well as DecAF (Yu et al., 2022), ARG-KBQA (Tian et al., 2024), TFS-KBQA (Wang and Qin, 2024) and ChatKBQA (Luo et al., 2024a) as LLM-based SP methods. Most of the baseline data are directly sourced from the corresponding works.

Evaluation Metrics. Following previous works (Luo et al., 2024b; Jiang et al., 2023; Sui et al., 2024), we use F1 score and Hits@1 metric to evaluate the coverage of all answers and the accuracy of the top-ranked answer, respectively. Furthermore, following prior work (Luo et al., 2024a), we also apply the beam match (BM), which checks whether the ground truth LF exists within the generated beams to assess the quality of LLM generation.

Models. We employ LLMs for LF generation and pre-trained language models (PLMs) for retrieval. For LLMs, we use the Llama 2 (Touvron et al., 2023) 7B model as the primary backbone throughout the experiments. Additionally, we include Llama 2 13B model for parameter comparison, along with DeepSeek LLM (Bi et al., 2024) 7B (DeepSeek 7B) and DeepSeek R1 (Guo et al., 2025) Distill Llama 8B (DeepSeek R1 8B) models for validation across different backbones. For similarity comparison during retrieval, we adopt off-the-shelf unsupervised RoBERTa-based (Liu et al., 2019) SimCSE (Gao et al., 2021).

Hyperparameters and Environment. Hyperparameters are involved in LLM fine-tuning and beam search. For LLM fine-tuning, we adopt a training batch size of 4 and a learning rate of 5e-5 based on the Llama 2 7B backbone for the main experiments. For model comparison, we retain the hyperparameters for both DeepSeek 7B and DeepSeek R1 8B models, while increasing the learning rate to 7e-5 for the Llama 2 13B backbone. The LLMs are trained for 50 epochs on WebQSP dataset, and for 5 epochs on CWQ dataset. Regarding PEFT, we employ all linear layers (all), as well as query and value projection layers (q_proj, v_proj) for target modules and a LoRA (Hu et al., 2022a) rank of 8. For beam search, the beam sizes range from {3, 5, 8}. All experiments are performed on a single NVIDIA A40 GPU. More details are presented in

Appendix D.

5.2 Experiment Results

We explore the following research questions (RQs) and provide the corresponding results and analyses. **RQ1:** Does the performance of KaeDe outperform that of other baseline methods? **RQ2:** What are the impacts of models and parameters? **RQ3:** How does each part of KaeDe function? **RQ4:** How does the hyperparameter impact the performance? **RQ5:** How does the method operate in detail? Additional experiments and analyses are provided in Appendix E-G.

5.2.1 Overall Comparison (RQ1)

The overall comparison is shown in Table 2. We present the results of KaeDe fine-tuned on the Llama 2 7B model, with a beam size of 8 on WebQSP and 5 on CWQ, respectively, along with oracle entity linking annotation. In general, our proposed method outperforms other baseline methods. When using Hits@1 as the metric, KaeDe surpasses the previous optimal ChatKBQA by 5.73% on WebQSP and 6.91% on CWQ. In terms of F1 score, KaeDe shows a slight decline of 0.26% on WebQSP but an improvement of 3.33% on CWQ. The comparison with other methods and the relatively higher performance increment on CWQ demonstrate the overall effectiveness of KaeDe in representing complex subgraph structures. This is attributed to the decomposed task accomplishment, guided by knowledge-aware question decomposition, which provides benefits even with models that have smaller parameters. However, the beam search involved in intermediate LLM operations and relation retrieval introduces additional noise, leading to a relatively lower F1 score.

5.2.2 Model and Parameter Analysis (RQ2)

To demonstrate the impacts of models and parameters, we conduct experiments using varying parameters (including total LLM parameters and those involved in PEFT) and different backbone models, as depicted in Table 3. When the model parameters increase from 7B to 13B with the same Llama backbone, as reflected in the overall performance, there is even a slight decline of Hits@1 by 0.88% under the condition that all target modules are trained. The presence of simpler questions with fewer hops in CWQ likely explains this, as noted in the work (Xu et al., 2025), where larger models may overfit and be less effective on simpler decomposed tasks.

Method	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
Non-LLM Information Retrieval				
NSM (He et al., 2021)	74.3	67.4	48.8	44.0
Rigel (Sen et al., 2021)	73.3	-	48.7	-
UniK-QA (Oguz et al., 2022)	79.1	-	-	-
UniKGQA (Jiang et al., 2023)	77.2	72.2	51.2	49.0
Non-LLM Semantic Parsing				
HGNet (Chen et al., 2022)	76.9	76.6	68.9	68.5
TIARA (Shu et al., 2022)	75.2	78.9	-	-
FC-KBQA (Zhang et al., 2023b)	-	76.9	-	56.4
LLM-Based Information Retrieval				
RoG (Luo et al., 2024b)	85.7	70.8	62.6	56.2
ToG (Sun et al., 2024)	82.6	-	72.5	-
FiDeLiS (Sui et al., 2024)	84.39	78.32	71.47	64.32
LLM-Based Semantic Parsing				
DecAF (Yu et al., 2022)	82.1	78.8	70.4	-
ARG-KBQA (Tian et al., 2024)	-	75.6	-	-
TFS-KBQA (Wang and Qin, 2024)	79.8	79.9	-	63.6
ChatKBQA* (Luo et al., 2024a)	85.36	**81.20**	81.37	76.82
KaeDe	**91.09**	80.94	**88.28**	**80.15**

Table 2: Comparison results with baseline methods.* denotes the experiments being reproduced on our device, both with Llama 2 7B model and a beam size of 8, along with oracle entity linking annotation.

Model	Target	Overall		Hop=3		Hop=4	
		Hits@1	F1	Hits@1	F1	Hits@1	F1
Parameter Comparison							
Llama 2 7B	q_proj,v_proj	87.10	76.76	83.74	71.25	75.85	60.09
Llama 2 7B	all	88.28	80.15	88.47	78.15	78.87	64.41
Llama 2 13B	all	87.40	80.17	88.66	79.29	80.38	71.03
Backbone Comparison							
DeepSeek 7B	all	87.89	80.17	86.83	78.16	79.14	66.07
DeepSeek R1 8B	all	87.37	80.66	86.01	78.89	78.87	69.88

Table 3: The impacts of models and parameters. All results are based on a beam size of 5 using CWQ dataset.

On the other hand, when separately considering the complex questions with different hops, a greater improvement is observed in questions with higher hops, particularly a 1.51% increase in Hits@1 and a 6.62% increase in F1 for 4-hop questions. This is consistent with the model size reliance, as the task of parsing decomposed expressions into path-level LFs for higher hops is more complex. In terms of the PEFT parameters, when target modules extend to all linear layers, Hits@1 and F1 increase by 1.18% and 3.39%, respectively. This indicates that larger trained parameters primarily optimize performance by reducing noise during each intermediate beam search. Futhermore, slight differences are observed between the Llama-based and DeepSeek-based models, both with similar total parameters

Phase	Method	CWQ		
		Hits@1	F1	BM
	KaeDe	88.28	80.15	70.63
Generation	w/o decomposition	87.75	79.21	69.99
	w/o decomposition+generation	85.23	79.26	65.58
Retrieval	w/o oracle entity linking	87.49	79.35	70.63
	w/o relation retrieval	83.91	77.06	70.63
	w/o all retrieval	80.15	72.86	70.63
Other	w/o all	76.18	71.16	65.58

Table 4: Ablation studies on CWQ dataset conducted using the same Llama 2 7B model with a beam size of 5, trained with all PEFT target modules. "w/o all" represents the direct LF generation result without any retrieval.

(7B or 8B), highlighting the generalization of the proposed method across different backbones.

5.2.3 Ablation Studies (RQ3)

To explore the function of each section, we conduct ablation studies on the generation and retrieval phases, as presented in Table 4. Regarding the generation, when question decomposition is removed and path-level LFs are directly generated based on the original question, Hits@1, F1, and BM drop by 0.53%, 0.94%, and 0.64%, respectively. This demonstrates the positive guidance provided by the sub-questions and explanations. While the question decomposition, path-level LF generation, and graph-level LF assembly are all removed, indicating a direct LF generation followed by retrieval, there is a relatively large decline of 3.05% in Hits@1, 0.89% in F1, and 5.05% in BM. Since BM directly reflects the quality of LF generation, this decline underscores the significant role of decomposed tasks. For the retrieval, the results reveal limited reliance on golden entity linking from oracle annotation. However, without relation retrieval, the Hits@1 and F1 decrease by 4.37% and 3.09%, respectively. Furthermore, removing the entire retrieval phase results in a significant performance drop, with the impact being more pronounced for direct LF generation. This highlights the critical importance of grounding LLM generation to KBs.

5.2.4 Hyperparameter Analysis (RQ4)

The impact of beam size, the main hyperparameter, is shown in Figure 3. On both datasets, as the beam size increases, Hits@1 continues to rise, while F1 slightly improves initially but declines after a beam size of 5, with the reduction being more severe for CWQ. This reflects that larger beam sizes provide more candidates that might hit the expected LF, but noise is introduced as well, especially during the retrieval process, which turns non-executable LFs into executable ones, while also bringing in executable but incorrect LFs. Therefore, we select a beam size of 8 for WebQSP and 5 for CWQ as the best performing results, balancing answer accessibility and noise.

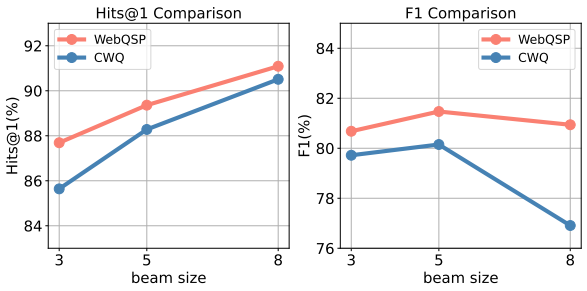

Figure 3: The impact of hyperparameters on WebQSP and CWQ. The beam size ranges from 3, 5, and 8.

Question: William Morris is religions head in which region that is part of the United Kingdom?
Ground Truth LF: (AND (JOIN [religion , religious leadership jurisdiction , leader] (JOIN [religion , religious organization leadership , leader] [William Morris])) (JOIN (R [base , aareas , schema , administrative area , administrative children]) [United Kingdom]))
Answer: Wales

Direct LF Generation: (AND (...?) (JOIN [religion , religion , founding figures] [William Morris]) (AND (JOIN [common , topic , notable types] [Country]) (JOIN (R [location , location , contains]) [United Kingdom]))) → entity error, relation error, hop error (x)→ non-executable

KaeDe:

Question Decomposition:
(1) Who is the administrative children of administrative area United Kingdom?
(2) William Morris is the leader of the religious organization leadership. He/She is the leader of the religious leadership jurisdiction.

Path-Level LF Generation:
(1) (JOIN (R [base , aareas , schema , administrative area , administrative children]) [United Kingdom])
(2) (JOIN [religion , religious leadership jurisdiction , leader] (JOIN [religion , religious organization leadership , leader] [William Morris]))

Graph-Level LF Assembly: (AND (JOIN [religion , religious leadership jurisdiction , leader] (JOIN [religion , religious organization leadership , leader] [William Morris])) (JOIN (R [base , aareas , schema , administrative area , administrative children]) [United Kingdom]))

Executed Result: Wales

Table 5: Case study for questions with multiple topic entities.

5.2.5 Case Study (RQ5)

We provide an example of how KaeDe operates in Table 5. Given the original question, the direct LF generation exhibits errors in entity, relation, and hop, as well as a redundant entity-centric reasoning path, which leads to failure in execution. For KaeDe, the question is decomposed into

a sub-question x_1 for "*United Kingdom*" and two sentences of explanation x_2 for "*William Morris*", implying a one-hop forward retrieval for the first entity and a two-hop retrieval, both backward, for the second. Then, path-level LFs l_1 and l_2 are generated according to x_1 and x_2, respectively and assembled into the complete graph-level LF L. Finally, the correct answer "*Wales*" is retrieved based on the accurate LF.

6 Conclusion

In this paper, we propose KaeDe, a novel LLM-based generate-then-retrieve approach for KBQA that optimizes complex LF generation through knowledge-aware question decomposition and the subsequent progressive LF generation. Specifically, by using a fine-tuned LLM, the original question is decomposed into simpler entity-centric sub-questions or explanations grounded in KB. Path-level LFs are then mapped from these intermediate expressions and assembled into the final graph-level LF using the same LLM. In conclusion, KaeDe reduces the complexity of LF generation by breaking it down into simpler decomposed tasks, achieving SOTA performance on WebQSP and CWQ datasets, particularly with fewer model parameters.

Limitations

Additional Computational Overhead. We utilize beam search during question decomposition, path-level LF generation, and graph-level assembly. In contrast to the single process in direct LF generation, KaeDe employs three sequential processes, where each subsequent process is generated based on the beam search results of the previous one. This intermediate beam search introduces additional computational overhead. In future work, we plan to address this issue by implementing pruning strategies during intermediate steps to reduce the complexity.

Noise from beam search and retrieval phase. Another issue stemming from beam search is the introduction of noise. During the retrieval phase, non-executable LFs are transformed into executable ones based on the skeleton by replacing the semantically similar entities and relations within the generated LFs. However, some irrelevant LFs produced during beam search may also be calibrated into executable forms, leading to inaccurate results. In future work, we aim to mitigate this issue through specific intermediate determination strategies.

Acknowledgments

This work is supported by the Interdisciplinary Program of Shanghai Jiao Tong University (project number YG2024QNB05).

References

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

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

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. Retrack: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing: system demonstrations*, pages 325–336.

Yongrui Chen, Huiying Li, Guilin Qi, Tianxing Wu, and Tenggou Wang. 2022. Outlining and filling: hierarchical query graph generation for answering complex questions over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8343–8357.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. Uhop: An unrestricted-hop relation extraction framework for knowledge-based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 345–356.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew Mccallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611.

Prayushi Faldu, Indrajit Bhattacharya, et al. 2024. Retinaqa: A robust knowledge base question answering model for both answerable and unanswerable questions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6643–6656.

Tengfei Feng and Liang He. 2025. Rgr-kbqa: Generating logical forms for question answering using knowledge-graph-enhanced large language model. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3057–3070.

Evgeniy Gabrilovich, Michael Ringgaard, Amarnag Subramanya, et al. 2013. Facc1: Freebase annotation of clueweb corpora.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022a. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xixin Hu, Yiheng Shu, Xiang Huang, and Yuzhong Qu. 2021. Edg-based question decomposition for complex question answering over knowledge bases. In *The Semantic Web–ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings 20*, pages 128–145. Springer.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022b. Logical form generation via multi-task learning for complex question answering over knowledge bases. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1687–1696.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065.

Xiang Huang, Sitao Cheng, Yiheng Shu, Yuheng Bao, and Yuzhong Qu. 2023. Question decomposition tree for answering complex questions over knowledge bases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12924–12932.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18608–18616.

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

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. 2024a. ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2039–2056. Association for Computational Linguistics.

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

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109.

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. 2024. Code-style in-context learning for knowledge-based question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18833–18841.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1535–1546.

Mayur Patidar, Riya Sawhney, Avinash Singh, Biswajit Chatterjee, Indrajit Bhattacharya, et al. 2024. Few-shot transfer learning for knowledge base question answering: Fusing supervised models with in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9147–9165.

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. *Journal of machine learning research*, 21(140):1–67.

Priyanka Sen, Armin Oliya, and Amir Saffari. 2021. Expanding end-to-end question answering on differentiable knowledge graphs with intersection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8805–8812.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. Tiara: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2024. Fidelis: Faithful reasoning in large language model for knowledge graph question answering. *arXiv preprint arXiv:2405.13873*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Yuhang Tian, Dandan Song, Zhijing Wu, Changzhi Zhou, Hao Wang, Jun Yang, Jing Xu, Ruanmin Cao, and Haoyu Wang. 2024. Augmenting reasoning capabilities of llms with graph structures in knowledge base question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11967–11977.

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

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

Shouhui Wang and Biao Qin. 2024. No need for large-scale search: Exploring large language models in complex knowledge base question answering. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12288–12299.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-KBQA: Multi-turn interactions for knowledge base question answering with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10561–10582. Association for Computational Linguistics.

Derong Xu, Xinhang Li, Ziheng Zhang, Zhenxi Lin, Zhihong Zhu, Zhi Zheng, Xian Wu, Xiangyu Zhao, Tong Xu, and Enhong Chen. 2025. Harnessing large language models for knowledge graph question answering via adaptive multi-aspect retrieval-augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25570–25578.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016a. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016b. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Peng Yixing, Quan Wang, Licheng Zhang, Yi Liu, and Zhendong Mao. 2024. Chain-of-question: A progressive question decomposition approach for complex knowledge base question answering. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4763–4776.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019. Complex question decomposition for semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4477–4486.

Jiajie Zhang, Shulin Cao, Tingjian Zhang, Xin Lv, Juanzi Li, Lei Hou, Jiaxin Shi, and Qi Tian. 2023a. Reasoning over hierarchical question decomposition tree for explainable question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14556–14570.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784.

Kun Zhang, Jiali Zeng, Fandong Meng, Yuanzhuo Wang, Shiqi Sun, Long Bai, Huawei Shen, and Jie Zhou. 2024. Tree-of-reasoning question decomposition for complex question answering with large language models. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 38, pages 19560–19568.

Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023b. Fc-kbqa: A fine-to-coarse composition framework for knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1002–1017.

Yaowei Zheng, Richong Zhang, Junhao Zhang, YeYanhan YeYanhan, and Zheyan Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410.

A Conversion of Logical Forms

In this section, we provide a detailed description of $\texttt{convert}(\cdot)$ function, composed of entity-centric per-hop projection, operator arrangement, and entity substitution.

A.1 Entity-Centric Per-Hop Projection

The main SPARQL lines are split into reasoning paths depending on the topic entities involved. Then, for each reasoning path, the projection for a single hop follows the expressions in Table 6. For instance, a single SPARQL line "$?x\ r\ e$." can be mapped to $(\ JOIN\ r\ e\ )$. If a second hop exists for the topic entity e, it can be iteratively converted based on the projection from the previous hop. When the iteration terminates, it forms a path-level LF that represents a reasoning path originating from e.

LF Projection	Direction	SPARQL	Triplet
$(\ JOIN\ r\ e\ )$	backward	$?x\ r\ e$.	$(?x, r, e)$
$(\ JOIN\ (\ R\ r\ )\ e\ )$	forward	$e\ r\ ?x$.	$(e, r, ?x)$

Table 6: Single Hop Projection.

A.2 Operator Arrangement

Once the path-level LFs are constructed for each topic entity e mentioned in the SPARQL, the graph-level LF is organized based on the operators, with some commonly used ones shown as examples in Table 7. For example, AND is utilized to represent the intersection between different reasoning paths, and $ARGMAX$ describes the constraints of values.

A.3 Entity Substitution

After the graph structure is arranged, the entity IDs starting with "*m.*" or "*g.*", which are in the format stored in Freebase, are replaced by their corresponding labels in the original SPARQL for improved comprehension.

A.4 Specific Example

We provide a specific example to explain the conversion.

Main Lines in SPARQL:
ns:m.09l3p ns:film.actor.film ?y .
?y ns:film.performance.character ?x .
?y ns:film.performance.film ns:m.0ddt_ .

The first two lines can be grouped into reasoning path z_1 for entity "*m.09l3p*", while the last line corresponds to reasoning path z_2 for entity "*m.0ddt_*".

Operator	Definition
	Operator for Entity
AND	$(\ AND\ (\ \mathcal{E}_1\ \mathcal{E}_2\ )\ )$ denotes intersecting between entity sets $\mathcal{E}_1$ and $\mathcal{E}_2$
$COUNT$	$(\ COUNT\ (\ \mathcal{E}\ )\ )$ denotes counting the size of entity set $\mathcal{E}$
	Operator for Value
$ARGMAX$	$(\ ARGMAX\ \mathcal{E}\ r\ )$ denotes obtaining the maximum literal after the projection $\{(\ JOIN\ r\ e\ )\|e \in \mathcal{E}\}$
$ARGMIN$	$(\ ARGMIN\ \mathcal{E}\ r\ )$ denotes obtaining the minimum literal after the projection $\{(\ JOIN\ r\ e\ )\|e \in \mathcal{E}\}$
GT	$(\ GT\ \mathcal{E}\ i\ )$ denotes the subset of $\mathcal{E}$ which is greater than literal i
GE	$(\ GE\ \mathcal{E}\ i\ )$ denotes the subset of $\mathcal{E}$ which is greater than or equal to literal i
LT	$(\ LT\ \mathcal{E}\ i\ )$ denotes the subset of $\mathcal{E}$ which is less than literal i
LE	$(\ LE\ \mathcal{E}\ i\ )$ denotes the subset of $\mathcal{E}$ which is less than or equal to literal i
TC	$(\ TC\ \mathcal{E}\ i\ )$ denotes the subset of $\mathcal{E}$ which is constrained by temporal literal i

Note: GT, GE, LT, LE are abbreviations for the operators $GREATER\ THAN$, $GREATER\ EQUAL$, $LESS\ THAN$, $LESS\ EQUAL$, respectively.

Table 7: Main Operators.

The first line of z_1 can be mapped to $e_{11} = (\ JOIN\ (\ R\ [\ film\ ,\ actor\ ,\ film\ ]\ )\ [\ m.09l3p\ ]\ )$. For the second hop, a nested expression is constructed based on e_{11} (the $?y$ in z_1), represented as $(\ JOIN\ (\ R\ [\ film\ ,\ performance\ ,\ character\ ]\ )\ e_{11}\ )$. z_2 can be generated with the same method. Then, the path-level LFs can be expressed as follows. (Notably, the path-level LFs in the paper are represented with entity labels instead of entity IDs. Here, we refer to the expressions as path-level LFs simply to illustrate the intermediate steps.)

Path-Level LFs l_1 for entity m.09l3p:
(JOIN (R [film , performance , character]) (JOIN (R [film , actor , film]) [m.09l3p]))
Path-Level LFs l_2 for entity m.0ddt_:
(JOIN [film , performance , film] [m.0ddt_])

According to the SPARQL description, the intersection of z_1 and z_2 occurs at "$?y$", which is at the first hop of z_1 and the last hop of z_2. Therefore, the graph-level LF can be formulated as the expression below. (It's worth noting that the graph-level LF here is the same as the path-level LF mentioned above, shown only to display the intermediate process.)

Graph-Level LFs L:
(JOIN (R [film , performance , character]) (AND (JOIN [film , performance , film] [m.0ddt_]) (JOIN (R [film , actor , film]) [m.09l3p])))

Finally, the entity IDs in the graph-level LF are replaced with the corresponding labels to form the

final LF.

Final LF:
(JOIN (R [film , performance , character]) (AND (JOIN [film , performance , film] [Star Wars Episode I: The Phantom Menace]) (JOIN (R [film , actor , film]) [Natalie Portman])))

B Prompt Template

In this section, we provide prompt templates for both training and inference. The LLMs are fine-tuned for two primary tasks: (1) Task 1, knowledge-aware question decomposition and (2) Task 2, progressive LF generation. These tasks can be further broken down into three sub-tasks: question decomposition for Task 1, path-level LF generation, and graph-level LF assembly for Task 2.

For question decomposition, the prompt template p_{d} is presented as follows.

Template p_{d} for Question Decomposition

You are an expert in KBQA. Given an original question {**original question**}. Please parse the original question into the corresponding simple questions or explanations based on each topic entity.

Then, the prompt template for path-level LF generation p_{g} is illustrated below.

Template p_{g} for Path-Level LF Generation

You are an expert in KBQA. Given an original question {**original question**} and the corresponding simple questions or explanations {**simple questions or explanations**}. Please generate the corresponding path-level logical forms.

Finally, in terms of the prompt template for graph-level LF assembly p_{a} is shown in the box.

Template p_{a} for Graph-Level LF Assembly

You are an expert in KBQA. Given an original question {**original question**} and all path-level logical forms {**path-level LFs**}. Please generate the final complete logical form for the original question.

The expressions in the prompt template, marked with "**", serve as placeholders for the corresponding content to be filled in. To avoid the additional computational overhead and noise induced by beam search, the placeholders are replaced with the combined intermediate results before proceeding to the next step. For instance, simple questions x_{q}^1, x_{q}^2 and explanation x_{ex}^1 are generated from the question decomposition task. Then, the path-level LF generation task will include all of these expressions in a single prompt, represented as $x_{\mathrm{q}}^1, x_{\mathrm{q}}^2, x_{\mathrm{ex}}^1 \in p_{\mathrm{g}}$.

C Details of Datasets

In this section, we provide details about the datasets utilized in this paper.

We employ the WebQSP (Yih et al., 2016b) and CWQ (Talmor and Berant, 2018) datasets as representative benchmarks for simple and complex questions, respectively. Both datasets can be referenced against the Freebase (Bollacker et al., 2008) KB. The statistics are provided in Table 8. Since both datasets include questions with multiple entities, we define the hop number of a question as the maximum hop count across all topic entity-centric reasoning paths.

WebQSP dataset is composed of 4,737 simple questions. In terms of hop number, most of these questions are single-hop. Regarding the answers, the distribution between single-answer and multiple-answer questions is approximately equal.

CWQ dataset contains 34,689 more complex questions generated from WebQSP, encompassing four types: conjunction, composition, comparative, and superlative. Regarding the hop number, two-hop questions make up the majority, while single-hop and multi-hop questions (those with more than two hops) exhibit a relatively balanced distribution.As for the answers, single-answer questions constitute the largest proportion in the dataset.

Dataset	Split	Proportion(%)	Hop			Answers	
			1(%)	2(%)	>2 (%)	1(%)	>1 (%)
WebQSP	Train	65.40	62.09	37.91	0	53.22	46.78
	Test	34.60	63.81	36.19	0	50.86	49.14
CWQ	Train	79.68	19.09	56.11	24.80	70.59	29.41
	Val	10.14	23.47	50.12	26.41	73.12	26.88
	Test	10.18	21.47	55.29	23.24	74.31	25.69

Table 8: Statistics of WebQSP and CWQ datasets.

D Details of Experiments

In this section, we present the details of the hyperparameter settings and model information used throughout the experiments.

We demonstrate the detailed experiment settings in Table 10. The parameters in bold represent the settings corresponding to the best-performing results, if other parameter options exist.

Furthermore, we provide the details of the models in Table 9, including the model size, the train-

Model Name	Backbone	Model Size (B)	Params for All (%)	Params for q, v (%)	Training Time (h)
Llama 2 7B	Llama-2-7b-chat	6.7426	0.2958	0.0622	{17 (WebQSP), 18 (CWQ)}
Llama 2 13B	Llama-2-13b-chat	13.0472	0.2398	0.0503	{31 (WebQSP), 33 (CWQ)}
DeepSeek 7B	DeepSeek-llm-7b-chat	6.9291	0.2704	/	{25 (CWQ)}
DeepSeeK R1 8B	DeepSeek-R1-Distill-Llama-8B	8.0512	0.2605	/	{26 (CWQ)}

Table 9: Details of Models.

Parameter	WebQSP	CWQ
Model Size	{**7B**, 8B, 13B}	{**7B**, 8B, 13B}
PEFT	LoRA	LoRA
LoRA Rank	8	8
Target Modules	{**all**, q_proj v_proj}	{**all**, q_proj v_proj}
Learning Rate	{**5e-5** (**7B**, 8B), 7e-5 (13B)}	{**5e-5** (**7B**, 8B), 7e-5 (13B)}
Batch Size	4	4
Beam Size	{3, 5, **8**}	{3, **5**, 8}
Epoch	50	5
Retriever	SimCSE	SimCSE
Top-k Entity	50	50
Top-k Relation	15	15

Table 10: Details of Hyperparameters.

able parameters for different target modules (such as all linear layers, as well as the q_proj and v_proj layers), and the approximate training time. Notably, all linear layers include {q_proj, v_proj, k_proj, o_proj} for the attention matrix, and {gate_proj, up_proj, down_proj} for the multilayer perceptron (MLP) layers in PEFT. The {q_proj, v_proj} setting is a regular configuration for Llama models. As we have observed in the experiments, there is no significant training time increment due to the parameter addition when the trained layers are added to all linear layers.

Throughout this work, we leverage Llama Factory framework (Zheng et al., 2024) to perform the LLM fine-tuning.

E Efficiency Analysis

Method	Beam Size	Generation (s)	Retrieval (s)
direct	3	1.94	9.18
	5	2.15	9.34
	8	2.65	7.56
KaeDe	3	6.29	13.16
	5	8.99	8.93
	8	20.64	29.40

Table 11: Average running time per question on CWQ dataset, with direct LF generation and KaeDe, both based on the Llama 2 7B model, with beam sizes ranging from 3, 5, and 8.

In this section, we analyze the efficiency comparison between KaeDe and direct LF generation, both of which are generate-then-retrieve approaches for semantic parsing.

We randomly sample 200 questions from the CWQ dataset and separately calculate the average running time per question for LLM generation and retrieval. The results are shown in Table 11. In terms of the generation phase, based on LLM beam search, the direct LF generation has $O(n)$ complexity for a single operation, while KaeDe follows $O(n^3)$ complexity due to the three iterative sequential tasks. While for the retrieval phase, the time is highly dependent on the LF quality and number. Thus, a decline is observed in beam size 8 for direct LF generation and in beam size 5 for KaeDe, attributed to the increase in LF quality. However, as the beam size continues to rise in KaeDe, the number of generated LFs begins to dominate, leading to an increase in time.

F Failure Analysis

Failure Type	Failure Rate (%)
Entity Error	32.08
Relation Error	71.18
Hop Error	49.62
Constraint Error	8.02
Graph Structure Error	41.60
Execution Error	5.76
Non-Executable	42.61

Table 12: Failure analysis on CWQ dataset, based on the Llama 2 7B model with a beam size of 5.

In section, we demonstrate the failure analysis for the proposed method.

The statistics are presented in Table 12, where failures encompass errors in entity, relation, hop, constraint, graph structure, and execution. Specifically, graph structure errors refer to cases where reasoning paths are missing or redundant, or where intersections between different paths are misplaced. Execution errors occur when the generated LF matches the ground truth exactly, but either fails to produce any answer or generates an incorrect answer. This issue often arises when a literal is involved. These errors often overlap; for instance, hop errors are frequently related to relation errors.

Relation errors account for the majority of the failures, but the non-executable rate is much lower. This indicates that the retrieval process significantly helps in calibrating non-executable LFs into executable ones, based on the correct predicted LF skeletons. However, it also introduces noise by producing incorrect yet executable LFs. Hop errors and graph structure errors are major contributors to unrecoverable failures due to changes in the LF skeleton, with their proportions aligned with the non-executable rate.

G Additional Case Study

Question: Before the Euro, what was the currency where Louis-Eugene Cavaignac was appointed to governmental position?
Ground Truth LF: (JOIN (R [location , country , currency formerly used]) (JOIN [government , governmental jurisdiction , governing officials] (JOIN [government , government position held , appointed by] [Louis-Eugène Cavaignac])))
Answer: French franc, Assignat
Direct LF Generation: (....?) (AND (JOIN [government , governmental jurisdiction , governing officials] (JOIN [government , government position held , appointed by] [Louis-Eugène Cavaignac])) (JOIN (R [finance , currency , countries used]) [Euro])) → entity error, relation error, hop error (x) → non-executable
KaeDe:
Question Decomposition: (1) Louis-Eugène Cavaignac is the appointed by of the government position held. He/She is the governing officials of the governmental jurisdiction. What is the currency formerly used of the country?
Path-Level LF Generation: (1) (JOIN (R [location , country , currency formerly used]) (JOIN [government , governmental jurisdiction , governing officials] (JOIN [government , government position held , appointed by] [Louis-Eugène Cavaignac])))
Graph-Level LF Assembly: (JOIN (R [location , country , currency formerly used]) (JOIN [government , governmental jurisdiction , governing officials] (JOIN [government , government position held , appointed by] [Louis-Eugène Cavaignac])))
Executed Result: French franc, Assignat

Table 13: Additional case study for a three-hop complex question.

In the Experiment section, we aim to illustrate the full process through a case study. Therefore, we provide the case study with multiple topic entities to show how the method works for generating multiple path-level LFs and their final assembly to produce the graph-level LF. However, the hop number is relatively low, with a maximum of two, due to the constraint of the dataset where multiple hops and multiple answers do not occur simultaneously in one question. In this section, we supplement a case study with a three-hop question.

For direct LF generation, two possible entities are predicted. However, "*Euro*" is redundant, leading to the failure of intersection between the paths starting from "*Louis-Eugène Cavaignac*" and "*Euro*," respectively. Additionally, the "*Louis-Eugène Cavaignac*" path misses a hop.

In terms of KaeDe, expressions involving two explanations and one question are generated for the entity "*Louis-Eugène Cavaignac*". The first two hops represent backward retrieval, while the last hop denotes forward retrieval. With fine-tuned LLMs, the combined explanation is mapped to a path-level LF. Since no other entity or constraints exist, it is directly output as the graph-level LF. The final executed answer turns out to be correct.