KAG: Boosting LLMs in Professional Domains via Knowledge Augmented Generation

Lei Liang[*,1], Mengshu Sun[*,1], Zhengke Gui[*,1], Zhongshu Zhu[1], Ling Zhong[1], Peilong Zhao[1], Zhouyu Jiang[1], Yuan Qu[1], Zhongpu Bo[1], Jin Yang[1], Huaidong Xiong[1], Lin Yuan[1], Jun Xu[1], Zaoyang Wang[1], Zhiqiang Zhang[1], Wen Zhang[2], Huajun Chen[2], Wenguang Chen[1], Jun Zhou[†,1]

{leywar.liang, mengshu.sms, zhengke.gzk, jun.zhoujun}@antgroup.com

[1]**Ant Group Knowledge Graph Team,** [2]**Zhejiang University**

Github:**https://github.com/OpenSPG/KAG**

Abstract

The recently developed retrieval-augmented generation (RAG) technology has enabled the efficient construction of domain-specific applications. However, it also has limitations, including the gap between vector similarity and the relevance of knowledge reasoning, as well as insensitivity to knowledge logic, such as numerical values, temporal relations, expert rules, and others, which hinder the effectiveness of professional knowledge services. In this work, we introduce a professional domain knowledge service framework called Knowledge Augmented Generation (**KAG**). KAG is designed to address the aforementioned challenges with the motivation of making full use of the advantages of knowledge graph(KG) and vector retrieval, and to improve generation and reasoning performance by bidirectionally enhancing large language models (LLMs) and KGs through five key aspects: (1) LLM-friendly knowledge representation, (2) mutual-indexing between knowledge graphs and original chunks, (3) logical-form-guided hybrid reasoning engine, (4) knowledge alignment with semantic reasoning, and (5) model capability enhancement for KAG. We compared KAG with existing RAG methods in multi-hop question answering and found that it significantly outperforms state-of-the-art methods, achieving a relative improvement of 19.6% on hotpotQA and 33.5% on 2wiki in terms of F1 score. We have successfully applied KAG to two professional knowledge Q&A tasks of Ant Group, including E-Government Q&A and E-Health Q&A, achieving significant improvement in professionalism compared to RAG methods. Furthermore, we will soon natively support KAG on the open-source KG engine OpenSPG, allowing developers to more easily build rigorous knowledge decision-making or convenient information retrieval services. This will facilitate the localized development of KAG, enabling developers to build domain knowledge services with higher accuracy and efficiency.

1 Introduction

Recently, the rapidly advancing Retrieval-Augmented Generation (RAG)[1, 2, 3, 4, 5] technology has been instrumental in equipping Large Language Models (LLMs) with the capability to acquire

[1], *: These authors contributed equally to this work.

[2], †: Corresponding author.

domain-specific knowledge. This is achieved by leveraging external retrieval systems, thereby significantly reducing the occurrence of answer hallucinations and allows for the efficient construction of applications in specific domains. In order to enhance the performance of the RAG system in multi-hop and cross-paragraph tasks, knowledge graph, renowned for strong reasoning capabilities, have been introduced into the RAG technical framework, including GraphRAG[6], DALK[7], SUGRE[8], ToG 2.0[9], GRAG[10], GNN-RAG [11] and HippoRAG[12].

Although RAG and its optimization have solved most of the hallucination problems caused by a lack of domain-specific knowledge and real-time updated information, the generated text still lacks coherence and logic, rendering it incapable of producing correct and valuable answers, particularly in specialized domains such as law, medicine, and science where analytical reasoning is crucial. This shortcoming can be attributed to three primary reasons. Firstly, real-world business processes typically necessitate inferential reasoning based on the specific relationships between pieces of knowledge to gather information pertinent to answering a question. RAG, however, commonly relies on the similarity of text or vectors for retrieving reference information, which may lead to incomplete and repeated search results. secondly, real-world processes often involve logical or numerical reasoning, such as determining whether a set of data increases or decreases in a time series, and the next token prediction mechanism used by language models is still somewhat weak in handling such problems.

In contrast, the technical methodologies of knowledge graphs can be employed to address these issues. Firstly, KG organize information using explicit semantics; the fundamental knowledge units are SPO triples, comprising entities and the relationships between them[13]. Entities possess clear entity types, as well as relationships. Entities with the same meaning but expressed differently can be unified through entity normalization, thereby reducing redundancy and enhancing the interconnectedness of knowledge [14]. During retrieval, the use of query syntax (such as SPARQL[15] and SQL[16]) enables the explicit specification of entity types, mitigating noisy from same named or similar entities, and allows for inferential knowledge retrieval by specifying relationships based on query requirements, as opposed to aimlessly expanding into similar yet crucial neighboring content. Meanwhile, since the query results from knowledge graphs have explicit semantics, they can be used as variables with specific meanings. This enables further utilization of the LLM's planning and function calling capabilities [17], where the retrieval results are substituted as variables into function parameters to complete deterministic inferences such as numerical computations and set operations.

To address the above challenges and meet the requirements of professional domain knowledge services, we propose **Knowledge Augmented Generation(KAG)**, which fully leverages the complementary characteristics of KG and RAG techniques. More than merely integrating graph structures into the knowledge base process, it incorporates the semantic types and relationships of knowledge graph and the commonly used Logical Forms from KGQA (Knowledge Graph Question Answering) into the retrieval and generation process. As shown in Figure 1, this framework involves the optimization of the following five modules:

- **We proposed a LLM friendly knowledge representation framework LLMFriSPG**. We refer to the hierarchical structure of data, information, and knowledge of DIKW to upgrade SPG to be friendly to LLMs, named LLMFriSPG, to make it compatible with schema-free information extraction and schema-constrained expert knowledge construction on the same knowledge type (such as entity type, event type), and supports the mutual-indexing representation between graph structure and original text chunks, which facilitates the construction of graph-structure-based inverted index and facilitates the unified representation, reasoning, and retrieval of logical form.
- **We proposed a logical-form-guided hybrid solving and reasoning engine**. It includes three types of operators: *planning, reasoning* and *retrieval*, transforming natural language questions into a problem-solving process that combines language and symbols. Each step in the process can utilize different operators such as exact match retrieval, text retrieval, numerical computation, or semantic reasoning, thereby achieving the integration of four distinct problem-solving processes: retrieval, KG reasoning, language reasoning, and numerical computation.
- **We proposed a knowledge alignment approach based on semantic reasoning**. Define domain knowledge as various semantic relations such as *synonyms, hypernyms*, and *inclusions*. Semantic reasoning is performed in both offline KG indexing and online retrieval

phases, allowing fragmented knowledge generated through automation to be aligned and connected through domain knowledge. In the offline indexing phase, it can improve the standardization and connectivity of knowledge, and in the online Q&A phase, it can serve as a bridge between user questions and indexing accurately.

- **We proposed a model for KAG**. To support the capabilities required for the operation of the KAG framework, such as index construction, retrieval, question understanding, semantic reasoning, and summarization, we enhance the three specific abilities of general LLMs: Natural Language Understanding (NLU), Natural Language Inference (NLI), and Natural Language Generation (NLG) to achieve better performance in each functional module.

We evaluated the effectiveness of the system on three complex Q&A datasets: 2WikiMultiHopQA[18], MuSiQue[19] and HotpotQA[20]. The evaluation focused on both end-to-end Q&A performance and retrieval effectiveness. Experimental results showed that compared to HippoRAG[12], KAG achieved significant improvements across all three tasks, with F1 scores increasing by 19.6%, 12.2% and 12.5% respectively. Furthermore, retrieval metrics also showed notable enhancements.

KAG is applied in two professional Q&A scenarios within Ant Group: E-Government and E-Health. In the E-Government scenario, it answers users' questions about administrative processes based on a given repository of documents. For E-Health, it responds to inquiries related to diseases, symptoms, treatments, utilizing the provided medical resources. Practical application results indicate that KAG achieves significantly higher accuracy than traditional RAG methods, thereby enhancing the credibility of Q&A applications in professional fields. We will soon natively support KAG on the open source KG engine OpenSPG, allowing developers to more easily build rigorous knowledge decision-making or convenient information retrieval services.

In summary, we propose a knowledge-augmented technical framework, KAG, targeting professional question-answering scenarios and validate the effectiveness of this framework based on complex question-answering tasks. We present two industry application cases based on Ant Group's business scenarios and have open-sourced the code to assist developers in building local applications using KAG.

2 Approach

In this section, we will first introduce the overall framework of KAG, and then discuss five key enhancements in sections 2.1 to 2.5. As shown in Figure 1, the KAG framework consists of three parts: KAG-Builder, KAG-Solver, and KAG-Model. The KAG-Builder is designed for building offline indexes, in this module, we proposed a *LLM Friendly Knowledge Representation framework* and *mutual-indexing between knowledge structure and text chunk*. In the module KAG-Solver we introduced a *Logical-form-guided hybrid reasoning solver* that integrates LLM reasoning, knowledge reasoning, and mathematical logic reasoning. Additionally, *knowledge alignment by semantic reasoning* is used to enhance the accuracy of knowledge representation and retrieval in both KAG-Builder and KAG-Solver. The KAG-Model optimizes the capabilities needed by each module based on a general language model, thereby improving the performance of all modules.

2.1 LLM Friendly Knowledge Representation

In order to define a more friendly knowledge semantic representation for LLMs, we upgrade SPG from three aspects: deep text-context awareness, dynamic properties and knowledge stratification, and name it **LLMFriSPG**.

$$\mathscr{M} = \{\mathscr{T}, \rho, \mathscr{C}, \mathscr{L}\}$$

where, $\mathscr{M}$ represents all types defined in LLMFriSPG, $\mathscr{T}$ represents all **EntityType**(e.g., Person in Figure 2), **EventType** classes and all pre-defined properties that are compatible with LPG syntax declarations. $\mathscr{C}$ represents all **ConceptType** classes, concepts and concept relations, it is worth noting that the root node of each concept tree is a **ConceptType** class that is compatible with LPG syntax(e.g., TaxoOfPerson in Figure 2.), each concept node has a unique **ConceptType** class. ρ represents the inductive relations from instances to conecepts. $\mathscr{L}$ represents all executable rules defined on logical relations and logical concepts. For $\forall t \in \mathscr{T}$:

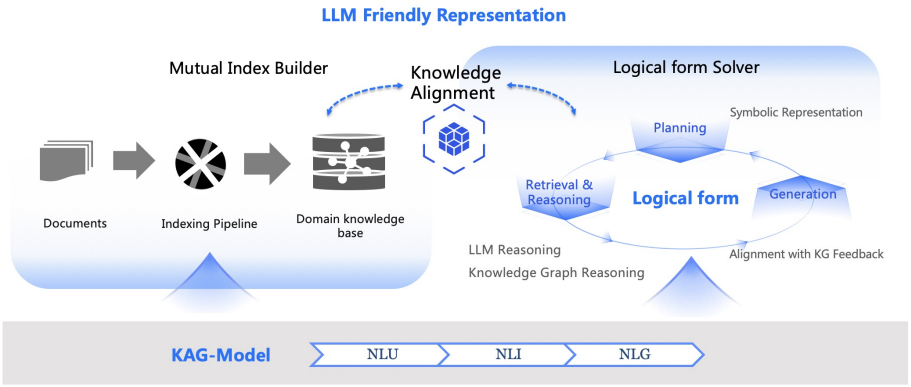

Figure 1: The KAG Framework. The left side shows KAG-Builder, while the right side displays KAG-Solver. The gray area at the bottom of the image represents KAG-Model.

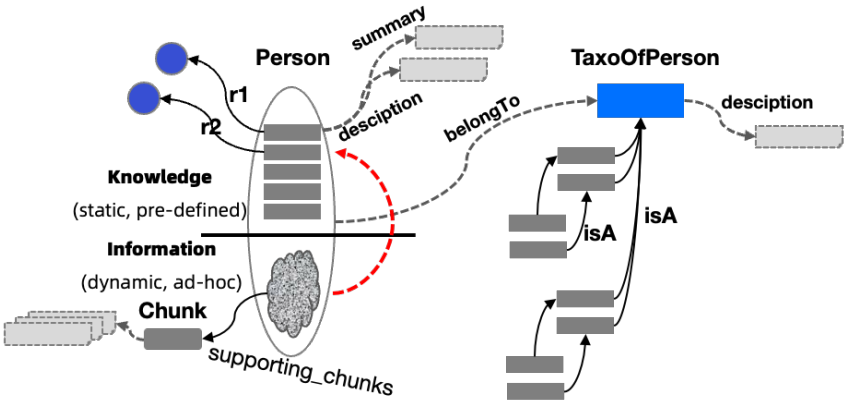

Figure 2: LLMFriSPG:A knowledge representation framework that is friendly to LLMs. Instances and concepts are separated to achieve more effective alignment with LLMs through concepts. In this study, entity instances and event instances are collectively referred to as instances unless otherwise specified. SPG properties are divided into knowledge and information areas, also called static and dynamic area, which are compatible with decision-making expertise with strong schema constraints and document retrieval index knowledge with open information representation. The red dotted line represents the fusion and mining process from information to knowledge. The enhanced document chunk representation provides traceable and interpretable text context for LLMs.

$$p_t = \{p_t^c, p_t{}^f, p_t^b\}$$

As is show in Figure 2, where, p_t represents all properties and relations of type t, and p_t^c represents the domain experts pre-defined part, p_t^f represents the part added in an ad-hoc manner, p_t^b represents the system built-in properties, such as *supporting_chunks, descripiton, summary* and *belongTo*. For any instance e_i, denote $typeof(e_i)$ as t_k, and *supporting_chunks* represents the set of all text chunks containing instance e_i, the user defines the chunk generation strategy and the maximum length of the chunk in KAG builder phase, *description* represents the general descriptive information specific to class t_k. It is worth noting that the meaning of *description* added to the type t_k and the instance e_i is different, when *description* is attached to t_k, it signifies the global description for that type. Conversely, when it is associated with an instance e_i, it represents the general descriptive information for e_i consistent with the orignal document context, *description* can effectively assist LLM in understanding the precise meaning of a specific instance or type, and can be used in tasks such as information extraction, entity linking, and summary generation. *summary* represents the summary of e_i or r_j in the original document context. *belongTo* represents the inductive semantics from instance to concept. Each **EntityType** or **EventType** can be associated with a **ConceptType**

through *belongTo*. It is worth noting that, **1)** $\mathscr{T}$ and $\mathscr{C}$ **have different functions.** The statement t adopts the object-oriented principle to better match the representation of the LPG[21], and $\mathscr{C}$ is managed by a text-based concept tree. This article will not introduce the SPG semantics in detail. **2)** p_t^c **and** p_t^f **can be instantiated separately**. That is, they share the same class declaration, but in the instance storage space, pre-defined static properties and realtime-added dynamic properties can coexist, and we also support instantiating only one of them. This approach can better balance the application scenarios of professional decision-making and information retrieval. General information retrieval scenarios mainly instantiate dynamic properties, while professional decision-making application scenarios mainly instantiate static properties. Users can strike a balance between ease of use and professionalism based on business scenario requirements. **3)** p_t^c **and** p_t^f **share the same conceptual terminology**. Concepts are general common sense knowledge that is independent of specific documents or instances. Different instances are linked to the same concept node to achieve the purpose of classifying the instances. We can achieve semantic alignment between LLM and instances through concept graphs, and concepts can also be used as navigation for knowledge retrieval. the details are shown in section 2.4 and 2.3.

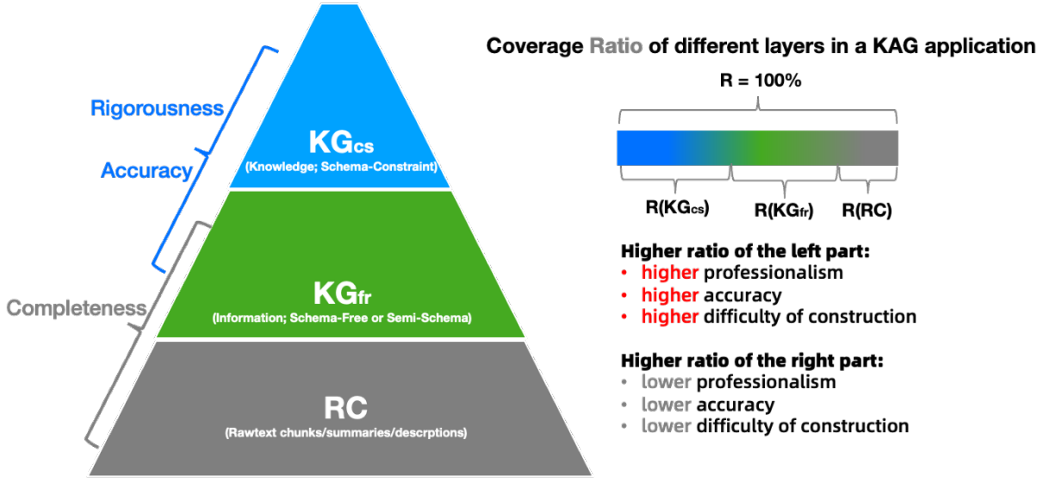

Figure 3: Hierarchical representation of knowledge and information.

In order to more accurately define the hierarchical representation of information and knowledge, as shown in 3, we denote KG_{cs} as *knowledge layer*, which represents the domain knowledge that complies with the domain schema constraints and has been summarized, integrated, and evaluated. denote KG_{fr} as *graph information layer*, which represents the graph data such as entities and relations obtained through information extraction. denote *RC* as *raw chunks layer*, which represents the original document chunks after semantic segmentation. the KG_{cs} layer fully complies with the SPG semantic specification and supports knowledge construction and logical rule definition with strict schema constraints, SPG requires that domain knowledge must have pre-defined schema constraints. It has high knowledge accuracy and logical rigor. However, due to its heavy reliance on manual annotation, the labor cost of construction is relatively high and the information completeness is insufficient. KG_{fr} shares the same EntityTypes, Eventtypes and Conceptual system with KG_{cs}, and provides effective information supplement for KG_{cs}. Meanwhile, the *supporting_chunks, summary*, and *description* edges built between KG_{fr} and *RC* form an inverted index based on graph structure, making *RC* an effective original-text-context supplement for KG_{fr} and with high information completeness. As is show in the right part of figure 3, in a specific domain application, $R(KG_{cs})$, $R(KG_{fr})$, and $R(RC)$ respectively represent their knowledge coverage in solving the target domain problems. If the application has higher requirements for knowledge accuracy and logic rigorousness, it is necessary to build more domain structured knowledge and consume more expert manpower to increase the coverage of $R(KG_{cs})$. On the contrary, if the application has higher requirements for retrieval efficiency and a certain degree of information loss or error tolerance, it is necessary to increase the coverage of $R(KG_{fr})$ to fully utilize KAG's automated knowledge construction capabilities and reduce expert manpower consumption.

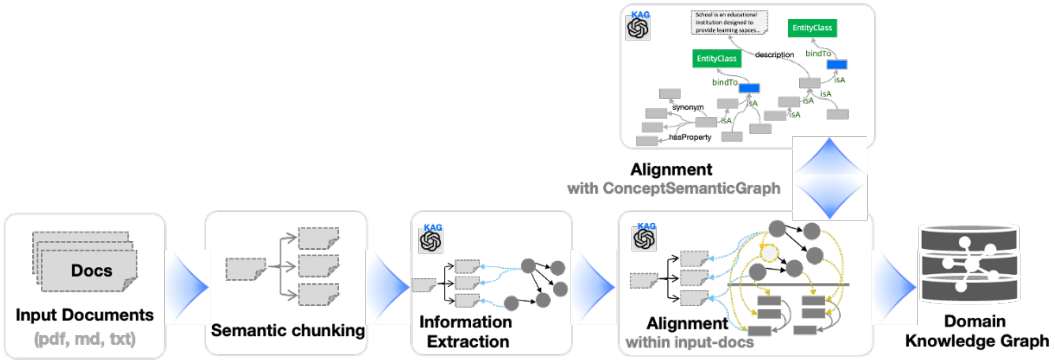

Figure 4: The Pipeline of KAG Builder for domain unstructured documents. From left to right, first, phrases and triples are obtained through information extraction, then disambiguation and fusion are completed through semantic alignment, and finally, the constructed KG is written into the storage.

2.2 Mutual Indexing

As illustrated in Figure 4, KAG-Builder consists of three coherent processes: structured information acquisition, knowledge semantic alignment and graph storage writer. The main goals of this module include: 1) building a mutual-indexing between the graph structure and the text chunk to add more descriptive context to the graph structure, 2) using the concept semantic graph to align different knowledge granularities to reduce noise and increase graph connectivity.

2.2.1 Semantic Chunking

According to the document's structural hierarchy and the inherent logical connections between paragraphs, a semantic chunking process is implemented based on system-built-in prompts. This semantic chunking produces chunks that adhere to both length constraints (specifically for LLM's context window size constraints) and semantic coherence, ensuring that the content within each chunk is thematically cohesive. We defined **Chunk EntityType** in *RC*, which includes fields such as *id, summary, and mainText*. Each chunk obtained after semantic segmentation will be written into an instance of **Chunk**, where *id* is a composite field consisting of *articleID, paraCode, idInPara* concatenated by the connector *#* in order to ensure that consecutive chunks are adjacent in the *id* space. *articleID* represents the globally unique article ID, *paraCode* represents the paragraph code in the article, and *idInPara* is the sequential code of each chunk in the paragraph. Consequently, an adjacency in the content corresponds to a sequential adjacency in their identifiers. Furthermore, a reciprocal relation is established and maintained between the original document and its segmented chunks, facilitating navigation and contextual understanding across different granularities of the document's content. This structured approach to segmentation not only optimizes compatibility with large-scale language models but also preserves and enhances the document's inherent semantic structure and association.

2.2.2 Information Extraction with More Descriptive Context

Given a dataset, we use fine-tuning-free LLM(such as GPT-3.5, DeepSeek, QWen, etc,.) or our fine-tuned model Hum to extract entities, events, concepts and relations to construct KG_{fr}, subsequently, construct the mutual-indexing structure between KG_{fr} and *RC*, enabling cross-document links through entities and relations. This process includes three steps. First, it extracts the entity set $E = \{e_1, e_2, e_3, ...\}$ chunk by chunk, second, extracts the event set $EV = \{ev_1, ev_2, ev_3, ...\}$ associated to all entities and iteratively extracts the relation set $R = \{r_1, r_2, r_3, ...\}$ between all entities in E, finally, completes all hypernym relations between the instance and its *spgClass*. To provide more convenience for the subsequent Knowledge Alignment phase, and overcome the problem of low discrimination of knowledge phrases such as Wikidata[22] and ConceptNet[23], in the entity extraction phase, we use LLMs to generate built-in properties *description, summary, semanticType, spgClass, descripitonOfSemanticType* by default for each instance *e* at one time, as shown in Figure 2, we store them in the *e* instance storage according to the structure of *e.description,e.summary, <e, belongTo, semanticType> and <e, hasClass, spgClass>*.

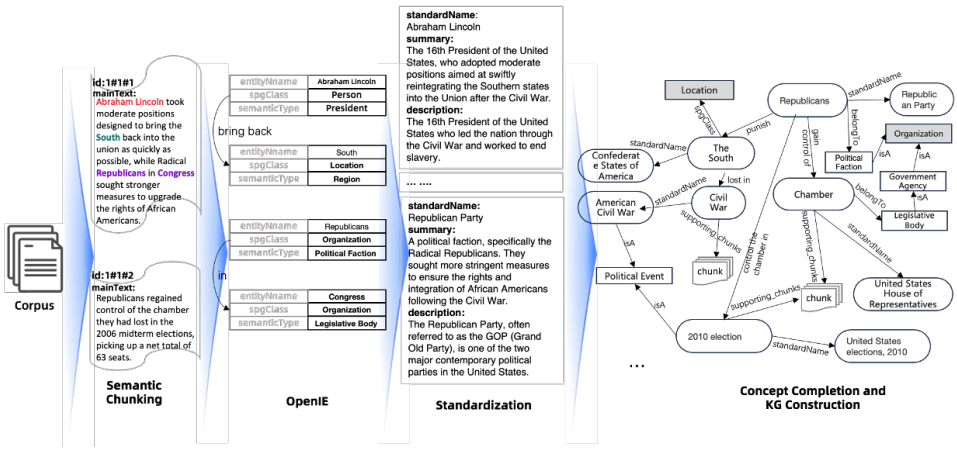

Figure 5: An Example of KAG-Builder pipeline

2.2.3 Domain Knowledge Injection And Constraints

When openIE is applied to professional domains, irrelevant noise will be introduced. Previous researches[3, 5, 24] have shown that noisy and irrelevant corpora can significantly undermine the performance of LLMs. It is a challenge to align the granularity of extracted information and domain knowledge. The domain knowledge alignment capabilities in KAG include: **1) Domain term and concept injection**. We use an iterative extraction approach, First, we store domain concepts and terms with *description* in KG storage. Second, we extract all instances in the document through openIE, then we perform vector retrieval to obtain all possible concept and term sets E_d. Finally, we add E_d to the extraction prompt and perform another extraction to obtain a set E_d^a that is mostly aligned with the domain knowledge. **2) Schema-constraint Extraction**. In the vertical professional domains, the data structure between multiple documents in each data source such as drug instructions, physical examination reports, government affairs, online order data, structured data tables, etc. has strong consistency, and is more suitable for information extraction with schema-constraint, structured Extraction also makes it easier to do knowledge management and quality improvement. For detailed information about knowledge construction based on Schema-constraint, please refer to the SPG[1] and OneKE[25]. This article will not introduce it in detail. It is worth noting that, as shown in figure 2, for the same entity type, such as **Person**, we can pre-define properties and relations such as *name, gender, placeOfBirth, (Person, hasFather, Person), (Person, hasFriend, Person)*, and can also extract tripples directly such as *(Jay Chou, spgClass, Person), (Jay Chou, constellation, Capricorn), (Jay Chou, record company, Universal Music Group)* through openIE. **3) Pre-defined Knowledge Structures By Document Type**. Professional documents such as drug instructions, government affairs documents, and legal definitions generally have a relatively standardized document structure. Each type of document can be defined as an entity type, and different paragraphs are different properties of the entity. Taking government affairs as an example, we can pre-define the GovernmentAffair EntityType and properites such as *administrative divisions, service procedures, required materials, service locations, and target groups*. The divided chunks are the values of different properties. If the user asks *"What materials are needed to apply for housing provident fund in Xihu District?"*, you can directly take out the chunk corresponding to property *required materials* to answer the question, avoiding the possible hallucinations caused by LLM re-generation.

2.2.4 Mutual indexing between text chunk vectors and knowledge structures

KAG's mutual-indexing is a knowledge management and storage mechanism that conforms to the LLMFriSPG semantic representation. As is described in section 2.1, it includes four core data structures: **1) Shared Schema**s are coarse-grained-types pre-defined as SPG Classes at project level, it includes *EntityType*s, *ConceptType*s, and *EventType*s, they serve as a high-level categorization such as *Person, Organization, GEOLocation, Date, Creature, Work, Event*. **2) Instance Graph** include all event and entity instances in KG_{cs} and KG_{fr}. that is, instances constructed through

[1] Official site of SPG: https://spg.openkg.cn/en-US

openIE with schema-free or structured extraction with schema-constraint are both stored as instances in KG storage. **3) Text Chunks** are special entity node that conforms to the definition of the *Chunk EntityType*. **4) Concept Graph** is the core component for knowledge alignment. it consists of a series of concepts and concept relations, concept nodes are also fine-grained-types of instances. Through relation prediction, instance nodes can be linked to concept nodes to obtain their fine-grained semantic types. , and two storage structures: **1) KG Store**. Store KG data structures in LPG databases, such as TuGraph, Neo4J. **2) Vector Store**. Store text and vectors in a vector storage engine, such as ElasticSearch, Milvus, or the vector storage embedded in the LPG engine.

2.3 Logical Form Solver

In the process of solving complex problems, three key steps are involved: *planning, reasoning* and *retrieval*. Disassembling question is a planning process to determine the next problem to be tackled. Reasoning includes retrieving information based on the disassembled question, inferring the answer to the question according to the retrieved results, or re-disassembling the sub-question when the retrieved content cannot answer the question. Retrieval is to find the content that can be used as reference for the original question or the disassembled sub-question. Since interactions between

Algorithm 1 Logical Form Solver

$memory \leftarrow []$
$query_{cur} \leftarrow query$
for $round \in (0, n)$ **do**
    $lf_{list} \leftarrow$ LFPlanner($query_{cur}$)
    $history \leftarrow []$
    for $lf \in lf_{list}$ **do**
        $lf_{subquery}, lf_{func} \leftarrow lf$
        $retrievals_{sub}, answer_{sub} \leftarrow$ Reasoner($lf_{subquery}$, lf_{func})
        $history.append([lf_{subquery}, retrievals_{sub}, answer_{sub}])$
    end for
    $memory \leftarrow$ Memory(*query*, *history*)
    if not Judge(*query*, *memory*) **then**
        $query_{cur} \leftarrow$ SupplyQuery(*query*, *memory*)
    end if
end for
answer $\leftarrow$ Generator(*query*, *memory*)
return *answer*

different modules in traditional RAG are based on vector representations of natural language, inaccuracies often arise. Inspired by the logical forms commonly used in KGQA, we designed an executable language with reasoning and retrieval capabilities. This language breaks down a question into multiple logical expressions, each of which may include functions for retrieval or logical operations. The mutual indexing described in Section 2.2 makes this process possible. Meanwhile, we designed a multi-turn solving mechanism based on reflection and global memory, inspired by ReSP[26]. The KAG solving process, as referenced in Figure 6 and Algorithm 17, first decomposes the current question $query_{cur}$ into a list of subquestions lf_{list} represented in logical form, and performs hybrid reasoning to solve them. If an exact answer can be obtained through multi-hop reasoning over structured knowledge, it returns the answer directly. Otherwise, it reflects on the solution results: storing the answers and retrieval results corresponding to lf_{list} in global memory and determining whether the question is resolved. If not, it generates supplementary questions and proceeds to the next iteration. Section 2.3.1, 2.3.2 and 2.3.3 introduce logical form function for planning, logical form for reasoning and logical form for retrieval respectively. In general, the proposed logical form language has the following three advantages:

- The use of symbolic language enhances the rigor and interpretability of problem decomposition and reasoning.
- Make full use of LLMFriSPG hierarchical representation to retrieve facts and texts knowledge guided by the symbolic graph structure

- Integrate the problem decomposition and retrieval processes to reduce the system complexity.

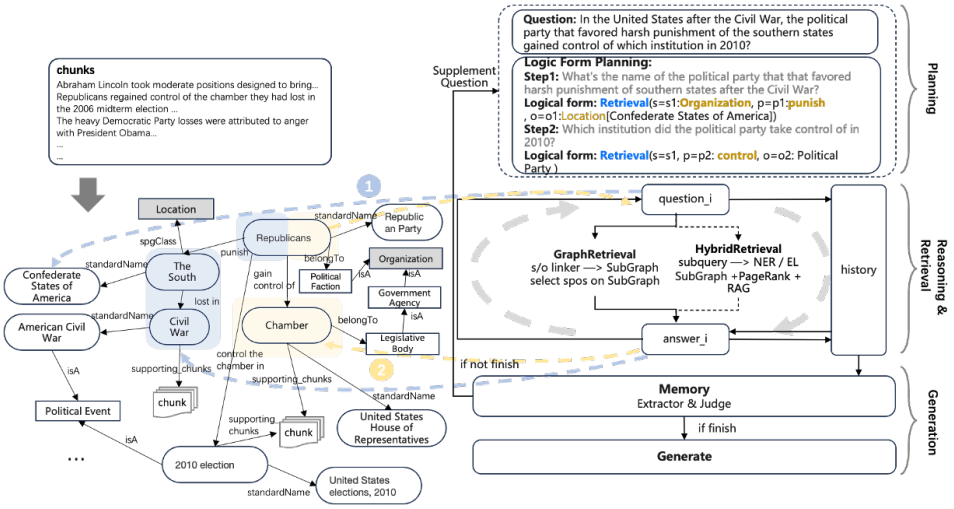

Figure 6: An Example of logical form execution. In this figure, the construction process of KG on the left is shown in Figure 5, and the overall reasoning and iteration process is on the right. First, a logical form decomposition is performed based on the user's overall question, and then logical-form-guided reasoning is used for retrieval and reasoning. Finally, Generation determines whether the user's question is satisfied. If not, a new question is supplied to enter a new logical form decomposition and reasoning process. If it is determined to be satisfied, Generation directly outputs the answer.

Table 13 illustrates a multi-round scenario consistent with pseudocode 17. Although first round the exact number of *plague occurrences* couldn't be determined, but we can extracted information indicates: *"Venice, the birthplace of Antonio Vivaldi, experienced the devastating Black Death, also known as the Great Plague. This pandemic caused by Yersinia pestis led to 75 to 200 million deaths in Eurasia, peaking in Europe from 1347 to 1351. The plague brought significant upheavals in Europe. Although specific occurrence records in Venice aren't detailed, it's clear the city was impacted during the mid-14th century."*. As is shown in Table 13,After two iterations, the answer determined is: ***22 times***.

2.3.1 Logical Form Planning

Logical Functions are defined as Table 1, with each function representing an execution action. Complex problems are decomposed by planning a combination of these expressions, enabling reasoning about intricate issues.

Function Name	Function Declaration
Retrieval	$Retrieval(s = s_i : type[name], p = p_i : edge, o = o_i : type[name],$ $s.prop = value, p.prop = value, o.prop = value)$
Sort	$Sort(A, direction = min\|max, limit = n)$
Math	$math_i = Math(expr)$, expr is in LaTeX syntax and can be used to perform operations on sets. e.g. count: $\|\|A\|\|$, sum: $\sum A$
Deduce	$Deduce(left = A, right = B, op = entailment\|greater\|less\|equal)$
Output	$Output(A, B, ...)$

Table 1: Functions of logical form.

Retrieval. According to the the knowledge or information retrieved from SPO, *s, p, o* should not repeatedly appear multiple times in the same expression. Constraints can be applied to the *s, p, o* for

querying. For multi-hop queries, multiple retrievals are required. When the current variable refers to a previously mentioned variable, the variable name must be consistent with the referenced variable name, and only the variable name needs to be provided. The knowledge type and name are only specified during the first reference.

Sort. Sort the retrieved results. A is the variable name for the retrieved subject-predicate-object(SPO) (s_i, o_i, or $s.prop$, $p.prop$, $o.prop$). $direction$ specifies the sorting direction, where $direction = min$ means sorting in ascending order and $direction = max$ means sorting in descending order. $limit = n$ indicates outputting the topN results.

Math. Perform mathematical calculations. $expr$ is in LaTeX syntax and can be used to perform calculations on the retrieved results (sets) or constants. $math_i$ represents the result of the calculation and can be used as a variable name for reference in subsequent actions.

Deduce. Deduce the retrieval or calculation results to answer the question. A, B can be the variable names from the retrieved SPO or constants. The operator $op = entailment|greater|less|equal$ represents A entails B, A is greater than B, A is less than B, and A is equal to B, respectively.

2.3.2 Logical Form for Reasoning

When the query statement represented by natural language is applied to the search, the logic is often fuzzy, such as *"find a picture containing vegetables or fruits"* and *"find a picture containing vegetables and fruits"*. Whether text search or vector search is used, the similarity between the two queries is very high, but the corresponding answers are quite different. The same is true for problems involving logical reasoning processes such as *and* or *not*, and *intersection* differences. To this end, we use logical form to express the question, so that it can express explicit semantic relations. Similar to IRCOT, we decompose complex original problem and plan out various execution actions such as multi-step retrieval, numerical reasoning, logical reasoning, and semantic deduce. Each sub-problem is expressed using logical form functions, and dependencies between sub-questions are established through variable references. The inference resolution process for each sub-question is illustrated as Algorithm 9. In this process, the **GraphRetrieval** module performs KG structure retrieval according to the logical form clause to obtain structured graph results. Another key module, **HybridRetrieval**, combining natural language expressed sub-problems and logical functions for comprehensive retrieval of documents and sub-graph information. To understand how logical functions can be utilized to reason about complex problems, refer to the following examples as Table 14.

Output. Directly output $A, B, ...$ as the answers. Both A and B are variable names that reference the previously retrieved or calculated

Algorithm 2 Logical Form Reasoner

Require: Each sub-query resulting from the decomposition of a question based on the logical form, along with their respective logical function, are denoted as $lf_{subquery}$ and lf_{func}

Ensure: The retrievals and answer of each sub-query, are denoted as $retri_{sub}$ and $answer_{sub}$

$retri_{kg} \leftarrow$ GraphRetrieval($lf_{subquery}$, lf_{func})
if $retri_{kg} \neq None$ and $retri_{kg} > threshold$ **then**
  $retri_{sub} \leftarrow retri_{kg}$
else
  $retri_{doc} \leftarrow$ HybridRetrieval($lf_{subquery}$, $retri_{kg}$)
  $retri_{sub} \leftarrow retri_{kg}, retri_{doc}$
end if
$answer_{sub} \leftarrow$ Generator($lf_{subquery}$, $retri_{sub}$)
return $retri_{sub}, answer_{sub}$

2.3.3 Logical Form for Retrieval

In naive RAG, retrieval is achieved by calculating the similarity (e.g. cosine similarity) between the embeddings of the question and document chunks, where the semantic representation capability of embedding models plays a key role. This mainly includes a sparse encoder (BM25) and a dense retriever (BERT architecture pre-training language models). Sparse and dense embedding approaches

capture different relevance features and can benefit from each other by leveraging complementary relevance information.

The existing method of combining the two is generally to combine the scores of the two search methods in an ensemble, but in practice different search methods may be suitable for different questions, especially in questions requiring multi-hop reasoning. When query involves proper *nouns, people, places, times, numbers, and coordinates*, the representation ability of the pre-trained presentation model is limited, and more accurate text indexes are needed. For queries that are closer to the expression of a paragraph of text, such as scenes, behaviors, and abstract concepts, the two may be coupled in some questions.

In the design of logical form, it is feasible to effectively combine two retrieval methods. When keyword information is needed as explicit filtering criteria, conditions for selection can be specified within the retrieval function to achieve structured retrieval.

For example, for the query *"What documents are required to apply for a disability certificate at West Lake, Hangzhou?"*, the retrieval function could be represented as: *"Retrieval(s=s1:Event[applying for a disability certificate], p=p1:support_chunks, o=o1:Chunk, s.location=West Lake, Hangzhou)"*. This approach leverages the establishment of different indices (sparse or dense) to facilitate precise searches or fuzzy searches as needed.

Furthermore, when structured knowledge in the form of SPO cannot be retrieved using logical functions, alternative approaches can be employed. These include semi-structured retrieval, which involves using logical functions to search through chunks of information, and unstructured retrieval. The latter encompasses methods such as Retrieval-Augmented Generation (RAG), where sub-problems expressed in natural language are used to retrieve relevant chunks of text. This highlights the adaptability of the system to leverage different retrieval strategies based on the availability and nature of the information.

2.4 Knowledge Alignment

Constructing KG index through information-extraction and retrieving based on vector-similarity has three significant defects in knowledge alignment:

- **Misaligned semantic relations between knowledge**. Specific semantic relations, such as *contains, causes* and *isA*, are often required between the correct answer and the query, while the similarity relied upon in the retrieval process is a weak semantic measure that lacks properties and direction, which may lead to imprecise retrieval of content.
- **Misaligned knowledge granularity**. The problems of knowledge granularity difference, noise, and irrelevance brought by openIE pose great challenges to knowledge management. Due to the diversity of language expressions, there are numerous synonymous or similar nodes, resulting in low connectivity between knowledge elements, making the retrieval recall incomplete.
- **Misaligned with the domain knowledge structure**. There is a lack of organized, systematic knowledge within specific domains. Knowledge that should be interrelated appears in a fragmented state, leading to a lack of professionalism in the retrieved content.

To solve these problems, we propose a solution that leverages concept graphs to enhance offline indexing and online retrieval through semantic reasoning. This involves tasks such as *knowledge instance standardization, instance-to-concept linking, semantic relation completion, and domain knowledge injection*. As described in section 2.2.2, we added descriptive text information to each instance, concept or relation in the extraction phase to enhance its interpretability and contextual relevance. Meanwhile, as described in section 2.2.3, KAG supports the injection of domain concepts and terminology knowledge to reduce the noise problem caused by the mismatch of knowledge granularity in vertical domains. The goal of concept reasoning is to make full use of vector retrieval and concept reasoning to complete concept relations based on the aforementioned knowledge structure to enhance the accuracy and connectivity of the domain KG. Refer to the definition of SPG concept semantics[2], as is shown in Table 2, we have summarized six semantic relations commonly required

[2]Semantic Classification of Concept: https://openspg.yuque.com/ndx6g9/ps5q6b/fe5p4nh1zhk6p1d8

for retrieval and reasoning. Additional semantic relations can be added based on the specific requirements of the actual scenario.

Formal Expression	Description	Example
<*var1, synonym, var2*>	A *synonym* relation means that a word or phrase *var2* that has the same or nearly the same meaning as another word or phrase *var1* in the same language and given context.	*Fast* is a synonym of *quick*.
<*var1, isA, var2*>	An *isA* relation means that a hypernym *var2* that is more generic or abstract than a given word or phrase *var1* and encompasses a broader category that the given word belongs to.	*Car* isA *Vehicle*.
<*var1, isPartOf, var2*>	An *isPartOf* relation means that something *var1* is a component or constituent of something *var2* larger. This relation shows that an item is a part of a bigger whole.	*Wheel* isPartOf *car*.
<*var1, contains, var2*>	A *contains* relation means that something *var1* includes or holds *var2*, something else within it. This indicates that one item has the other as a subset or component.	*Library* contains *books*.
<*var1, belongTo, var2*>	An *belongTo* relation means that something *var1* is an instance of concept *var2*.	*Chamber* belongTo *Legislative Body*.
<*var1, causes, var2*>	A *causes* relation means that one event or action *var1* brings about another *var2*. This indicates a causal relation where one thing directly results in the occurrence of another.	*Fire* causes *smoke*.

Table 2: Commonly used semantic relations.

2.4.1 Enhance Indexing

The process of enhancing indexing through semantic reasoning, as shown in Figure 5 , specifically implemented as predicting semantic relations or related knowledge elements among index items using LLM, encompassing four strategies:

- *Disambiguation and fusion of knowledge instances*. Taking entity instance e_{cur} as an example, first, the one-hop relations and description information of e_{cur} are used to predict *synonymous* relations to obtain the synonym instance set E_{syn} of e_{cur}. Then, the fused target entity e_{tar} is determined from E_{syn}. Finally, the entity fusion rules are used to copy the properties and relations of the remaining instances in E_{syn} to e_{tar}, and the names of these instances are added to the *synonyms* of e_{tar}, the remaining instances will also be deleted immediately.
- *Predict relations between instances and concepts*. For each knowledge instance (such as event, entity), predict its corresponding concept and add the derived triple $< e_i, belongTo, c_j >$ to the knowledge index. As is shown in Figure 5, <*Chamber, belongTo, Legislative Body*> means that the Chamber belongs to Legislative Body in classification.
- *Complete concepts and relations between concepts*. During the extraction process, we use concept reasoning to complete all *hypernym* and *isA relation*s between semanticType and spgClass. As is shown in Figure 5 and Table 2, we can obtain the semanticType of *Chamber* is *Legislative Body*, and its *spgClass* is *Organization* in the extraction phase. Through semantic completion, we can get <*Legislative Body, isA, Government Agency*>, <*Government Agency, isA, Organization*>. Through semantic completion, the triple information of KG_{fr} space is more complete and the connectivity of nodes is stronger.

2.4.2 Enhance Retrieval

In the retrieval phase, we utilize semantic relation reasoning to search the KG index based on the phrases and types in the logical form. For the types, mentions or relations in the logical form, we

employ the method of combining semantic relation reasoning with similarity retrieval to replace the traditional similarity retrieval method. This retrieval method makes the retrieval path professional and logical, so as to obtain the correct answer. First, the hybrid reasoning performs precise type matching and entity linking. If the type matching fails, then, semantic reasoning is performed. As shown in Figure 6, if the type *Political Party* fails to match, semantic reasoning is used to predict that *Political Party* contains *Political Faction*, and reasoning or path calculation is performed starting from *Political Faction*.

Take another example. If the user query q_1 is *"Which public places can cataract patients go for leisure?"* and the document content d_2 is *"The museum is equipped with facilities to provide barrier-free visiting experience services such as touch, voice interpretation, and fully automatic guided tours for the visually impaired."*, It is almost impossible to retrieve d_2 based on the vector similarity with q_1. However, it is easier to retrieve d_2 through the semantic relation of <*cataract patient, isA, visually impaired*>.

2.5 KAG-Model

KAG includes two main computational processes: *offline index building* and *online query and answer generation*. In the era of *small language models*, these two tasks were typically handled by two separate pipelines, each containing multiple task-specific NLP models. This results in high complexity for the application system, increased setup costs, and inevitable cascading losses due to error propagation between modules. In contrast, large language models, as a *capability complex*, can potentially integrate these pipelines into a unified, simultaneous end-to-end reasoning process.

As shown in Figure 7, the processes of indexing and QA each consist of similar steps. Both of the two pipelines can be abstracted as *classify, mention detection, mention relation detection, semantic alignment, embedding*, and *chunk, instance, or query-focused summary*. Among these, *classify, mention detection*, and *mention relation detection* can be categorized as NLU, while *semantic alignment* and *embedding* can be grouped under NLI. Finally, the *chunk, instance or query-focused summary* can be classified under NLG. Thus, we can conclude that the three fundamental capabilities of natural language processing that a RAG system relies on are NLU, NLI, and NLG.

We focused on exploring methods to optimize these three capabilities, which are introduced in subsections 2.5.1, 2.5.2, and 2.5.3 respectively. Additionally, to reduce the cascade loss caused by linking models into a pipeline, we further explored methods to integrate multiple inference processes into a single inference. Subsection 2.5.4 will discuss how to equip the model with retrieval capabilities to achieve better performance and efficiency through one-pass inference.

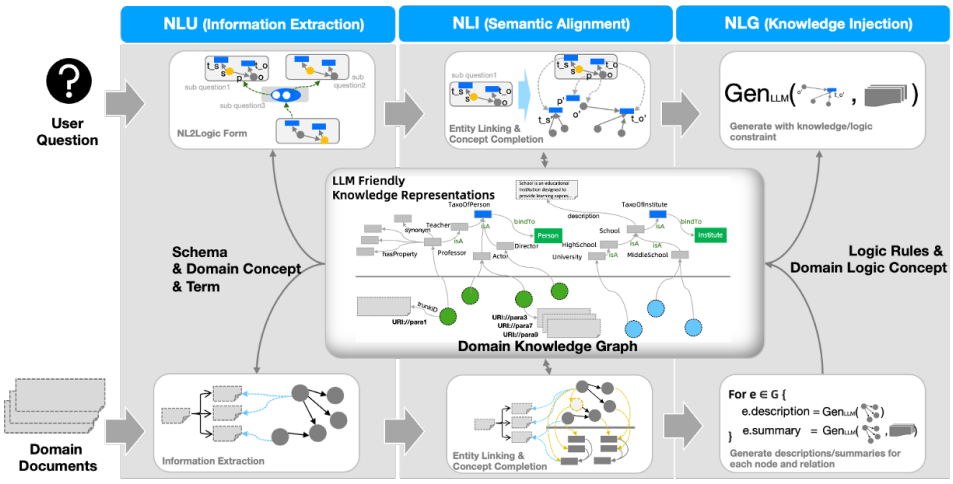

Figure 7: The model capabilities required for KAG.

2.5.1 Natural Language Understanding

NLU is one of the most common foundational tasks in natural language processing, including *text classification, named entity recognition, relation Extraction, subject and object extraction, trigger detection, event argument extraction, event extraction*, and *machine reading comprehension*. We have collected over 30 public datasets to enhance understanding capabilities. Experiments found that simply transforming the original datasets into instruction datasets can achieve comparable results to specialized models on trained tasks, but this approach does not improve the model's NLU capabilities on unseen domains. Therefore, we conducted large-scale instruction reconstruction, designing various instruction synthesis strategies to create an NLU instruction dataset with over 20,000 diverse instructions. By utilizing this dataset for supervised fine-tuning on a given base model, the model has demonstrated enhanced NLU capabilities in downstream tasks. The instruction reconstruction strategy mainly consists of the following three types.

- **Label bucketing:** [25]This strategy focuses on label-guided tasks, where the aim is to extract text based on labels or map text to specified labels, including *classification, NER, RE*, and *EE*. When labels in a dataset collectively co-occur in the training set, the model may learn this pattern and overfit to the dataset, failing to independently understand the meaning of each label. Therefore, during the instruction synthesis process, we adopt a polling strategy that designates only one label from each training sample as part of a bucket. Additionally, since some labels have similar semantics and can be confused, we group easily confused labels into a single bucket, allowing the model to learn the semantic differences between the two labels more effectively.
- **Flexible and Diverse Input and Output Formats:** The LLM employs an instruction-following approach for inference, and a highly consistent input-output format may cause the model to overfit to specific tasks, resulting in a lack of generalization for unseen formats. Therefore, we have flexibly processed the input and output formats. The output is handled as five different formatting instructions, as well as two types of natural language instructions. Additionally, the output format can dynamically be specified as markdown, JSON, natural language, or any format indicated in the examples.
- **Instructoin with Task Guildline:** Traditional NLP training often employs a "sea of questions" approach, incorporating a wide variety of data in the training set. This allows the model to understand task requirements during the learning process, such as whether to include job titles when extracting personal names. For the training of LLMs, we aim for the model to perform tasks like a professional annotator by comprehending the task description. Therefore, for the collected NLU tasks, we summarize the task descriptions using a process of self-reflection within the LLM. This creates training data that includes task descriptions within the instructions. Additionally, to enhance task diversity, we implement heuristic strategies to rephrase the task descriptions and answers. This enables the model to understand the differences between task descriptions more accurately and to complete tasks according to the instructions.

We fine-tuned six foundational models: qwen2, llama2, baichuan2, llama3, mistral, phi3, and used six understanding benchmarks recorded on OpenCompass for performance validation. The table 3 shows that the KAG-Model has a significant improvement in NLU tasks.

2.5.2 Natural Language Inference

The NLI task is used to infer the semantic relations between given phrases. Typical NLI tasks include *entity linking, entity disambiguation, taxonomy expansion, hypernym discovery*, and *text entailment*. In the context of knowledge base Q&A, due to the diversity and ambiguity of natural language expressions, as well as the subtle and different types of semantic connections between phrases, it often requires further alignment or retrieval of related information through NLI tasks based on NLU. As described in section 2.4, we categorize the key semantic relation in knowledge base applications into six types. Among these, relations such as *isA, isPartOf* and *contains* exhibit directional and distance-based partial order relations. During the reasoning process, it is crucial to accurately determine these semantic relations to advance towards the target answer. In traditional approaches, separate training of representation pre-training models and KG completion(KGC) models is often employed to reason about semantic relations. However, these KGC models tend to focus

Models	C3	WSC	XSum	Lambda	Lcsts	Race	Average
GPT4	95.10	74.00	20.10	65.50	12.30	92.35	59.89
Qwen2	92.27	66.35	18.68	62.39	13.07	**88.37**	56.86
KAG_{Qwen2}	**92.88**	**70.19**	**31.33**	**66.16**	**18.53**	88.17	**61.21**
Llama2	81.70	50.96	23.29	63.26	15.99	55.64	48.47
KAG_{Llama2}	**82.36**	**63.46**	**24.51**	**65.22**	**17.51**	**68.48**	**53.59**
Baichuan2	**84.44**	66.35	20.81	62.43	16.54	76.85	54.57
$KAG_{Baichuan2}$	84.11	**66.35**	**21.51**	**62.64**	**17.27**	**77.18**	**54.84**
Llama3	**86.63**	**65.38**	25.84	36.72	0.09	**83.76**	49.74
KAG_{Llama3}	83.40	62.50	**26.72**	**54.07**	**18.45**	81.16	**54.38**
Mistral	**67.29**	30.77	21.16	59.98	0.78	73.46	42.24
$KAG_{Mistral}$	47.29	**39.42**	**21.54**	**69.09**	**17.14**	72.42	**44.48**
Phi3	68.60	**42.31**	**0.60**	**71.74**	3.47	73.18	43.32
KAG_{Phi3}	**85.21**	25.94	0.36	71.24	**15.49**	**74.00**	**45.37**

Table 3: Enhancement of natural language understanding capabilities in different LLMs by KAG. The experimental results are based on the open-compass framework and tested using the "gen" mode. The evaluation metrics for C3, WSC, Lambda, and Race are ACC. XSum and Lcsts are measured using ROUGE-1. Race includes Race-middle and Race-high, and their average is taken.

on learning graph structures and do not fully utilize the essential textual semantic information for semantic graph reasoning. LLMs possess richer intrinsic knowledge, and can leverage both semantic and structural information to achieve more precise reasoning outcomes. To this end, we have collected a high-quality conceptual knowledge base and ontologies from various domains, creating a conceptual knowledge set that includes 8,000 concepts and their semantic relations. Based on this knowledge set, we constructed a training dataset that includes six different types of conceptual reasoning instructions to enhance the semantic reasoning capabilities of a given base model, thereby providing semantic reasoning support for KAG.

Semantic reasoning is one of the core ability required in KAG process, we use NLI tasks and general reasoning Q&A tasks to evaluate the ability of our model, the results are as shown in Table 4 and Table 5. The evaluation results indicates that our KAG-Model demonstrates a significant improvement in tasks related with semantic reasoning: First, Table 5 shows that on the Hypernym Discovery task(which is consistent in form with the reasoning required in semantic enhanced indexing and retrieval.), our fine-tuned KAG-llama model outperforms Llama3 and ChatGPT-3.5 significantly. In addition, the better performance of our model on CMNLI, OCNLI and SIQA compared with Llama3 in Table 4 shows that our model has good capabilities in general logical reasoning.

Models	CMNLI	OCNLI	SIQA
Llama3	35.14	32.1	44.27
KAG-Llama3	49.52	44.31	65.81

Table 4: Enhancement of natural language Inference capabilities in different LLMs by KAG. The evaluation metrics for CMNLI, OCNLI, SIQA are measured with accuracy.

	1A.English	2A.Medical	2B.Music
ChatGPT-3.5	30.04	26.12	28.47
Llama3-8B	23.47	24.26	18.73
KAG-Llama3	**38.26**	**55.14**	**30.16**

Table 5: Hypernym Discovery performance comparison on SemEval2018-Task9 dataset, measured in MRR.

2.5.3 Natural Language Generation

Models that have not undergone domain adaptation training often exhibit significant differences from the target text in domain logic and writing style. Moreover, acquiring sufficient amounts of annotated data in specialized domains frequently poses a challenge. Therefore, we have established

two efficient fine-tuning methods for specific domain scenarios, allowing the generation process to better align with scene expectations: namely, K-Lora and AKGF.

Pre-learning with K-LoRA. First of all, we think that using knowledge to generate answers is the reverse process of extracting knowledge from text. Therefore, by inverting the previously described extraction process, we can create a 'triples-to-text' generation task. With extensive fine-tuning on a multitude of instances, the model can be trained to recognize the information format infused by the KG. Additionally, as the target text is domain-specific, the model can acquire the unique linguistic style of that domain. Furthermore, considering efficiency, we continue to utilize LoRA-based SFT. We refer to the LoRA obtained in this step as K-LoRA.

Alignment with KG Feedback. The model may still exhibit hallucinations in its responses due to issues such as overfitting. Inspired by the RLHF(Reinforcement Learning with Human Feedback) approach[27, 28], we hope that the KG can serve as an automated evaluator, providing feedback on knowledge correctness of the current response, thereby guiding the model towards further optimization. First, we generate a variety of responses for each query by employing diverse input formats or random seeds. Subsequently, we incorporate the KG to score and rank these responses. The scoring process compare generated answer with knowledge in KG to ascertain their correctness. The reward is determined by the number of correctly matched knowledge triples. The formula for calculating the reward is represented by Formula 1.

$$reward = \log(rspo + \alpha \times re) \quad (1)$$

where α is a hyperparameter, *rspo* represents the number of SPO matches, and *re* represents the number of entity matches.

We select two biomedical question-answering datasets, CMedQA[29] and BioASQ[30], for evaluating our model. CMedQA is a comprehensive dataset of Chinese medical questions and answers, while BioASQ is an English biomedical dataset. We randomly choose 1,000 instances from each for testing. For CMedQA, we employ the answer texts from the non-selected Q&A pairs as corpora to construct a KG in a weakly supervised manner. Similarly, with BioASQ, we use all the provided reference passages as the domain-specific corpora. Experimental results, as shown in Table 6, demonstrate significant enhancement in generation performance. For more details on the specific implementation process, please refer to our paper[31]

Model	CMedQA		BioASQ	
	Rouge-L	BLEU	Rouge-L	BLEU
ChatGPT-3.5 0-shot	14.20	1.78	21.14	5.93
ChatGPT-3.5 2-shot	14.66	2.53	21.42	6.11
Llama2	14.02	2.86	23.47	7.11
KAG_{Llama2}	**15.44**	**3.46**	**24.21**	**7.79**

Table 6: Performance comparison on CMedQA & BioASQ. "CP" indicates "continual pre-trained". We consider continual pre-training as a basic method of domain knowledge infusion, on par with other retrieval-based methods. Consequently, we do not report on the outcomes of hybrid approaches.

2.5.4 Onepass Inference

Most retrieval enhanced systems operate in a series of presentation models, retrievers, and generation models, resulting in high system complexity, construction costs, and the inevitable concatenation loss caused by error transfer between modules. We introduces an efficient one-pass unified generation and retrieval (OneGen) model to enable an arbitrary LLM to generate and retrieve in one single forward pass. Inspired by the latest success in LLM for text embedding, we expand the original vocabulary by adding special tokens (i.e. retrieval tokens), and allocate the retrieval task to retrieval tokens generated in an autoregressive manner. During training, retrieval tokens only participate in representation fine-tuning through contrastive learning, whereas other output tokens are trained using language model objectives. At inference time, we use retrieval tokens for efficient retrieving on demand. Unlike the previous pipeline approach where at least two models are needed for retrieval and generation, OneGen unified them in one model, thus eliminating the need for a separate retriever and greatly reducing system complexity.

As shown in experiment results in Table 7, we draw the following conclusions: (1) OneGen demonstrates efficacy in $R \rightarrow G$ task, and joint training of retrieval and generation yields performance gains on the RAG task. The Self-RAG endows LLMs with self-assessment and adaptive retrieval, while OneGen adds self-retrieval. Our method outperforms the original Self-RAG across all datasets, especially achieving improvements of 3.1pt on Pub dataset and 2.8pt on ARC dataset, validating the benefits of joint training. (2) OneGen is highly efficient in training, with instruction-finetuned LLMs showing strong retrieval capabilities with minimal additional tuning. It requires less and lower-quality retrieval data, achieving comparable performance with just 60K noisy samples and incomplete documents, without synthetic data. For more details on the specific implementation process, please refer to paper[32]

		Generation Performance				**Retrieval Performance**	
		HotpotQA		2WikiMultiHopQA		HotpotQA	2WikiMultiHopQA
BackBone	**Retriever**	EM	F1	EM	F1	Recall@1	Recall@1
Llama2-7B	Contriever	52.83	65.64	70.02	74.35	73.76	68.75
	self	54.82	67.93	75.02	78.86	75.90	69.79
Llama3.1-7B	Contriever	53.72	66.46	70.92	75.29	69.79	66.80
	self	55.38	68.35	75.88	79.60	72.55	68.98
Qwen2-1.5B	Contriever	48.55	**61.02**	68.32	72.66	72.41	67.70
	self	48.75	60.98	73.84	77.44	72.70	69.27
Qwen2-7B	Contriever	53.32	66.22	70.80	74.86	74.15	69.01
	self	55.12	67.60	76.17	79.82	75.68	69.96

Table 7: In RAG for Multi-Hop QA settings, performance comparison across different datasets using different LLMs.

3 Experiments

3.1 Experimental Settings

Datasets. To evaluate the effectiveness of the KAG for knowledge-intensive question-answering task, we perform experiments on 3 widely-used multi-hop QA datasets, including HotpotQA [20], 2WikiMultiHopQA [18], and MuSiQue [19]. For a fair comparison, we follow IRCoT [33] and HippoRAG [12] utilizing 1,000 questions from each validation set and using the retrieval corpus related to selected questions.

Evaluation Metric. When evaluating QA performance, we use two metrics: Exact Match (EM) and F1 scores. For assessing retrieval performance, we calculate the hit rates based on the Top 2/5 retrieval results, represented as Recall@2 and Recall@5.

Comparison Methods. We evaluate our approach against several robust and commonly utilized retrieval RAG methods. **NativeRAG** using ColBERTv2 [34] as retriever and directly generates answers based on all retrieved documents [35]. **HippoRAG** is a RAG framework inspired by human long-term memory that enables LLMs to continuously integrate knowledge across external documents. In this paper, we also use ColBERTv2 [34] as its retriever [12]. **IRCoT** interleaves chain-of-thought (CoT) generation and knowledge retrieval steps in order to guide the retrieval by CoT and vice-versa. This interleaving allows retrieving more relevant information for later reasoning steps. It is a key technology for implementing multi-step retrieval in the existing RAG framework.

3.2 Experimental Results

3.2.1 Overall Results

The end-to-end Q&A performance is shown in Table 8. Within the RAG frameworks leveraging ChatGPT-3.5 as backbone model, HippoRAG demonstrates superior performance compared to NativeRAG. HippoRAG employs a human long-term memory strategy that facilitates the continuous integration of knowledge from external documents into LLMs, thereby significantly enhancing Q&A capabilities. However, given the substantial economic costs associated with utilizing ChatGPT-3.5, we opted to use the DeepSeek-V2 API as a viable alternative. On average, the performance of the IRCoT + HippoRAG configuration utilizing the DeepSeek-V2 API slightly surpasses that of ChatGPT-

3.5. Our constructed framework KAG shows significant performance improvement compared to IRCoT + HippoRAG, with EM increases of 11.5%, 19.8%, and 10.5% on HotpotQA, 2WikiMultiHopQA, and MuSiQue respectively, and F1 improvements of 12.5%, 19.1%, and 12.2%. These advancements in end-to-end performance can largely be attributed to the development of more effective indexing, knowledge alignment and hybrid solving libraries within our framework. We evaluate the effectiveness of the single-step retriever and multi-step retriever, with the retrieval performance shown in Table 9. From the experimental results, it is evident that the multi-step retriever generally outperforms the single-step retriever. Analysis reveals that the content retrieved by the single-step retriever exhibits very high similarity, resulting in an inability to use the single-step retrieval outcomes to derive answers for certain data that require reasoning. The multi-step retriever alleviates this issue. Our proposed KAG framework directly utilizes the multi-step retriever and significantly enhances retrieval performance through strategies such as mutual-indexing, logical form solving, and knowledge alignment.

Framework	**Model**	HotpotQA		2WikiMultiHopQA		MuSiQue	
		EM	F1	EM	F1	EM	F1
NativeRAG [35, 34]	ChatGPT-3.5	43.4	57.7	33.4	43.3	15.5	26.4
HippoRAG [12, 34]	ChatGPT-3.5	41.8	55.0	46.6	59.2	19.2	29.8
IRCoT+NativeRAG	ChatGPT-3.5	45.5	58.4	35.4	45.1	19.1	30.5
IRCoT+HippoRAG	ChatGPT-3.5	45.7	59.2	47.7	62.7	21.9	33.3
IRCoT+HippoRAG	DeepSeek-V2	51.0	63.7	48.0	57.1	26.2	36.5
KAG *w/* LFS_{ref_3}	DeepSeek-V2	<u>59.8</u>	<u>74.0</u>	<u>66.3</u>	<u>76.1</u>	<u>35.4</u>	<u>48.2</u>
KAG *w/* $LFSH_{ref_3}$	DeepSeek-V2	**62.5**	**76.2**	**67.8**	**76.2**	**36.7**	**48.7**

Table 8: The end-to-end generation performance of different RAG models on three multi-hop Q&A datasets. The values in **bold** and <u>underline</u> are the best and second best indicators respectively.

	Retriever	**HotpotQA**		**2Wiki**		**MuSiQue**	
		Recall@2	Recall@5	Recall@2	Recall@5	Recall@2	Recall@5
Single-step	BM25 [36]	55.4	72.2	51.8	61.9	32.3	41.2
	Contriever [37]	57.2	75.5	46.6	57.5	34.8	46.6
	GTR [38]	59.4	73.3	60.2	67.9	37.4	49.1
	RAPTOR [39]	58.1	71.2	46.3	53.8	35.7	45.3
	Proposition [40]	58.7	71.1	56.4	63.1	37.6	49.3
	NativeRAG [35, 34]	64.7	79.3	59.2	68.2	37.9	49.2
	HippoRAG [12, 34]	60.5	77.7	70.7	89.1	40.9	51.9
Multi-step	IRCoT + BM25	65.6	79.0	61.2	75.6	34.2	44.7
	IRCoT + Contriever	65.9	81.6	51.6	63.8	39.1	52.2
	IRCoT + NativeRAG	<u>67.9</u>	82.0	64.1	74.4	41.7	53.7
	IRCoT + HippoRAG	67.0	<u>83.0</u>	**75.8**	**93.9**	<u>45.3</u>	<u>57.6</u>
	KAG	**72.8**	**88.8**	<u>65.4</u>	<u>91.9</u>	**48.5**	**65.7**

Table 9: The performance of different retrieval models on three multi-hop Q&A datasets

3.3 Ablation Studies

The objective of this experiment is to deeply investigate the impact of the knowledge alignment and logic form solver on the final results. We conduct ablation studies for each module by substituting different methods and analyzing the changes in outcomes.

3.3.1 Knowledge Graph Indexing Ablation

In the graph indexing phase, we propose the following two substitution methods:

1) Mutual Indexing Method. As a baseline method of KAG, according to the introduction in Sections 2.1 and 2.2, we use information extraction methods (such as OpenIE) to extract phrases and triples in document chunks, and form the mutual-indexing between graph structure and text

chunks according to the hierarchical representation of LLMFriSPG, and then write them into KG storage. We denote this method as **M_Indexing**.

2) Knowledge Alignment Enhancement. This method uses knowledge alignment to enhance the KG mutual-indexing and the logical form-guided reasoning & retrieval. According to the introduction in Section 2.4, it mainly completes tasks such as the classification of instances and concepts, the prediction of hypernyms/hyponyms of concepts, the completion of the semantic relationships between concepts, the disambiguation and fusion of entities, etc., which enhances the semantic distinction of knowledge and the connectivity between instances, laying a solid foundation for subsequent reasoning and retrieval guided by logical forms. We denote this method as **K_Alignment**.

3.3.2 Reasoning and Retrieval Ablation

Multi-round Reflection. We adopted the multi-round reflection mechanism from ReSP[26] to assess whether the Logical Form Solver has fully answered the question. If not, supplementary questions are generated for iterative solving until the information in global memory is sufficient. We analyzed the impact of the maximum iteration count n on the results, denoted as ref_n. If $n = 1$, it means that the reflection mechanism is not enabled. In the reasoning and retrieval phase, we design the following three substitution methods:

1) Chunks Retriever. We define KAG's baseline retrieval strategy with reference to HippoRAG's[12] retrieval capabilities, with the goal of recalling the top_k chunks that support answering the current question. The Chunk score is calculated by weighting the vector similarity and the personalized pagerank score. We denote this method as *ChunkRetri*, we denote ChunkRetri with n-round reflections as CR_{ref_n}.

2) Logical Form Solver (Enable Graph Retrieval). Next, we employ a Logical Form Solver for reasoning. This method uses pre-defined logical forms to parse and answer questions. First, it explores the reasoning ability of the KG structure in KG_{cs} and KG_{fr} spaces, focusing on accuracy and rigor in reasoning. Then, it uses supporting_chunks in *RC* to supplement retrieval when the previous step of reasoning has no results. We denote this method as LFS_{ref_n}. The parameter n is maximum number of iteration parameter.

3) Logical Form Solver (Enable Hybrid Retrieval). In order to make full use of the mutual-indexing structure between KG_{fr} and *RC* to further explore the role of KG structure in enhancing chunk retrieval, we modify the LFS_{ref_n} by disabling the Graph Retrieval functionality for direct reasoning. Instead, all answers are generated using the Hybrid Retrieval method. This approach enables us to evaluate the contribution of graph retrieval to the performance of reasoning. We denote this method as $LFSH_{ref_n}$.

Through the design of this ablation study, we aim to comprehensively and deeply understand the impact of different graph indexing and reasoning methods on the final outcomes, providing strong support for subsequent optimization and improvement.

3.3.3 Experimental Results and Discussion

Graph Index	Reasoning	HotpotQA		2Wiki		MuSiQue	
		EM	F1	EM	F1	EM	F1
M_Indexing	CR_{ref_3}	52.4	65.4	48.2	56.0	24.6	36.6
K_Alignment	CR_{ref_3}	54.7	69.5	62.7	72.5	29.6	41.1
	LFS_{ref_1}	59.1	73.4	65.2	74.4	31.3	43.4
	LFS_{ref_3}	59.8	74.0	<u>66.3</u>	<u>76.1</u>	<u>35.4</u>	<u>48.2</u>
	$LFSH_{ref_1}$	<u>61.5</u>	<u>76.0</u>	66.0	75.0	33.5	44.3
	$LFSH_{ref_3}$	**62.5**	**76.2**	**67.8**	**76.2**	**36.7**	**48.7**

Table 10: The end-to-end generation performance of different model methods on three multi-hop Q&A datasets. The backbone model is DeepSeek-V2 API. As is described in Algorithm 17, ref_3 represents a maximum of 3 rounds of reflection, and ref_1 represents a maximum of 1 round, which means that no reflection is introduced.

Graph Index	Reasoning	HotpotQA		2Wiki		MuSiQue	
		R@2	R@5	R@2	R@5	R@2	R@5
M_Indexing	CR_{ref_3}	61.5	73.8	54.6	59.7	39.3	52.8
K_Alignment	CR_{ref_3}	56.3	83.0	**66.3**	88.1	40.0	62.3
	LFS_{ref_1}	/	/	/	/	/	/
	LFS_{ref_3}	/	/	/	/	/	/
	$LFSH_{ref_1}$	55.1	85.0	65.9	**92.4**	36.1	58.4
	$LFSH_{ref_3}$	**72.7**	**88.8**	65.4	91.9	**48.4**	**65.6**

Table 11: The recall performance of different methods across three datasets is presented. The answers to some sub-questions in the LFS_{ref_n} method use KG reasoning without recalling supporting chunks, which is not comparable to other methods in terms of recall rate. BackBone model is DeepSeek-V2 API.

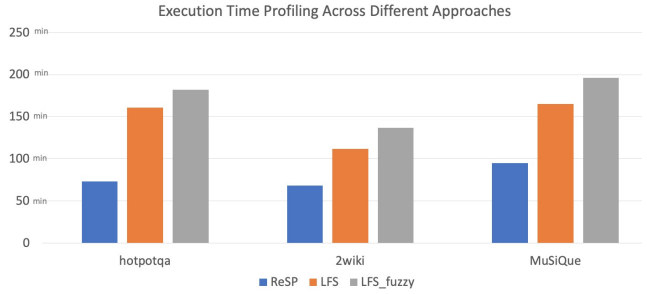

Figure 8: Each of the three test datasets comprises 1000 test problems, with 20 tasks processed concurrently and maximum number of iterations n is **3**. CR_{ref_3} method exhibits the fastest execution, whereas $LFSH_{ref_3}$ method is the slowest. Specifically, CR_{ref_3} method outperforms $LFSH_{ref_3}$ method by 149%, 101%, and 134% across the three datasets. In comparison, on the same dataset, the LFS_{ref_3} method outperforms the $LFSH_{ref_3}$ method by 13%, 22%, and 18%, respectively, with F1 relative losses of 2.6%, 0.1%, and 1.0%, respectively.

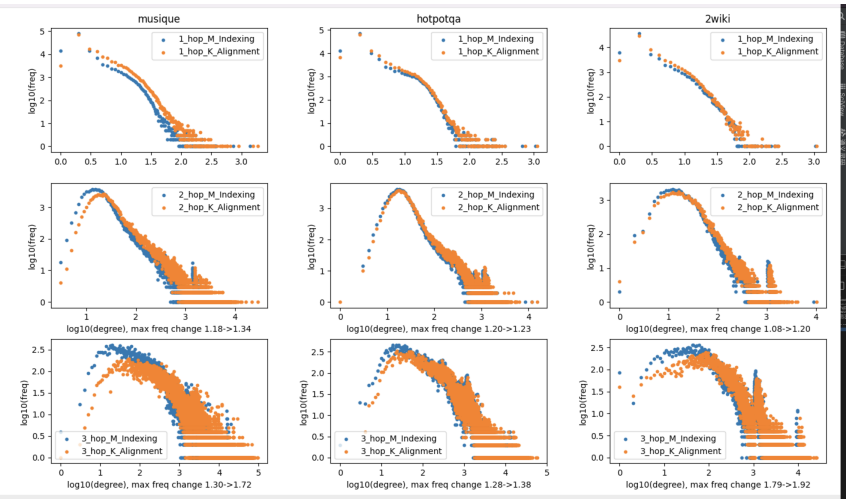

Figure 9: The connectivity of the graph exhibits a notable rightward shift after applying **K_Alignment**,the distribution changes of 1-hop, 2-hop, and 3-hop neighbors are shown.

The analysis of the experimental outcomes can be approached from the following two perspectives:

1) Knowledge Graph Indexing. As is shown in Table 11, after incorporation Knowledge Alignment into the KG mutual-indexing, the top-5 recall rates of CR_{ref_3} improved by 9.2%, 28.4%, and 9.5% respectively, with an average improvement of 15.7%. As shown in Figure 9, after enhancing knowledge alignment, the relation density is significantly increased, and the frequency-outdegree graph is shifted to the right as a whole

- The 1-hop graph exhibits a notable rightward shift, indicating that the addition of semantic structuring has increased the number of neighbors for each node, thereby enhancing the graph's density.

- The 2-hop and 3-hop graphs display an uneven distribution, with sparse regions on the left and denser regions on the right. When comparing before and after **K_Alignment**, it is evident that the vertices in each dataset have shifted rightward, with the left side becoming more sparse. This suggests that nodes with fewer multi-hop neighbors have gained new neighbors, leading to this observed pattern.

This signifies that the newly added semantic relations effectively enhance graph connectivity, thereby improving document recall rates.

2) Graph Inference Analysis. In terms of recall, $LFSH_{ref_3}$ achieves improvements over CR_{ref_3} under the same graph index, with increases in top-5 recall rates by 15%, 32.2%, and 12.7%, averaging an improvement of 19.9%. This enhancement can be attributed to two main factors:

- $LFSH_{ref_3}$ decomposes queries into multiple executable steps, with each sub-query retrieving chunks individually. As shown in the time analysis in Figure 8, both $LFSH_{ref_3}$ and LFS_{ref_3} consume more than twice the time of $LFSH_{ref_3}$, indicating that increased computational time is a trade-off for improved recall rates.

- $LFSH_{ref_3}$ not only retrieves chunks but also integrates SPO triples from execution into chunk computation. Compared to $LFSH_{ref_3}$, it retrieves additional query-related relationships.

Due to the subgraph-based query answering in LFS_{ref_3}, it cannot be compared directly in recall rate analysis but can be examined using the F1 metric. In comparison to $LFSH_{ref_3}$, LFS_{ref_3} answered questions based on the retrieved subgraphs with proportions of 33%, 34%, and 18%,respectively. LFS_{ref_3} shows a decrease in the F1 metric by 2.2%, 0.1%, and 0.5%, while the computation time reduces by 12%, 22%, and 18%.

The analysis of the cases with decreased performance reveals that errors or incomplete SPOs during the construction phase lead to incorrect sub-query answers, resulting in wrong final answers. This will be detailed in the case study. The reduction in computation time is primarily due to the more efficient retrieval of SPOs compared to document chunks.

In industrial applications, computation time is a crucial metric. Although LFS_{ref_n} may introduce some errors, these can be improved through graph correction and completion. It is noteworthy that in the current experiments, the slight decrease in metrics has been traded off for reduced computation time, which we consider a feasible direction for industrial implementation.

For analyze the impact of the maximum number of iterations parameter n on the results, LFS_{ref_1} compared to LFS_{ref_3} , the F1 scores decreased by 0.6%, 1.6%, and 4.8%, respectively. Based on the experiments of LFS_{ref_3}, the proportions for an iteration count of 1 were analyzed to be 97.2%, 94.8%, and 87.9%; $LFSH_{ref_1}$ compared to $LFSH_{ref_3}$, the F1 scores decreased by 0.2%, 1.2%, and 4.4%, respectively. Based on the experiments of $LFSH_{ref_3}$, the proportions for an iteration count of 1 were analyzed to be 98.3%, 95.2%, and 84.1%; showing a positive correlation with the F1 score reduction. Table 13 provides a detailed analysis of the effect of iteration rounds on the solution of the final answer. Increasing the maximum number of iterations parameter facilitates the re-planning of existing information when LFS_{ref_n} is unable to complete the solution, thereby addressing some unsolvable case.

4 Applications

4.1 KAG for E-Goverment

We used the KAG framework and combined it with the Alipay E-government service scenario to build a Q&A application that supports answering users' questions about service methods, required materials, service conditions, and service locations. To build the e-government Q&A application, we first collected 11,000 documents about government services, and based on the methods described in section 2, implemented functional modules such as index building, logical-form-guided reasoning and solving, semantic enhancement, and conditional summary generation.

During the offline index construction phase, the semantic chunking strategy is used to segment government service documents to obtain specific matters and their properties such as the administrative region, service process, required materials, service location, target audience, and the corresponding chunks.

In the reasoning and solving phase, a logical function is generated based on the given user question and graph index structure, and the logical form is executed according to the steps of the logical function. First, the index item of the administrative area where the user is located is accurately located. Then, the item name, group of people, etc. are used for search. Finally, the corresponding chunk is found through the *required materials* or *service process*. specifically inquired by the user.

In the semantic enhancement phase, we added two semantic relations, *synonymy and hypernymy*, between items. A synonymous relation refers to items in two different regions with different names but the same meaning, such as *renewal of social security card* and *application for lost social security card*; a co-hypernymy relation refers to two items belonging to different subcategories under the same major category of items, such as *applying for housing provident fund loan for construction of new housing* and *applying for housing provident fund loan for construction and renovation of new housing*, the two items have a common hypernymy *applying for housing provident fund loan*.

We compared the effects of the two technical solutions, NaiveRAG and KAG, as shown in the table below. It is evident that KAG shows significant improvements in both completeness and accuracy compared to NaiveRAG.

Methods	SampleNum	Precision	Recall
NaiveRAG	492	66.5	52.6
KAG	492	91.6	71.8

Table 12: Ablation Experiments of KAG in E-Goverment Q&A.

4.2 KAG for E-Health

We have developed a medical Q&A application based on the Alipay Health Manager scenario, which supports answering user's questions regarding popular science about disease, symptom, vaccine, operation, examination and laboratory test, also interpretation of medical indicators, medical recommendation, medical insurance policy inquires, hospital inquires, and doctor information inquires. We have sorted out authoritative medical document materials through a team of medical experts, and produced more than 1.8 million entities and more than 400,000 term sets, with a total of more than 5 million relations. Based on this high-quality KG, we have also produced more than 700 DSL[3] rules for indicator calculations to answer the questions of indicator interpretation.

During the knowledge construction phase, a strongly constrained schema is used to achieve precise structural definition of entities such as diseases, symptoms, medications, and medical examinations. This approach facilitates accurate answers to questions and generates accurate knowledge, while also ensuring the rigor of relations between entities. In the reasoning phase, the logical form is generated based on the user's query, and then translated to DSL form for the query on KG. The query result is returned in the form of triples as the answer. The logical form not only indicates how to query the KG, but also contains the key structural information in the user's query (such as city, gender, age, indicator value, etc.). When parsing the logical form for query in graph,

[3]DSL: https://openspg.yuque.com/ndx6g9/ooil9x/sdtg4q3bw4ka5wmz

the DSL rules which produced by medical expert will also be triggered, and the conclusion will be returned in the form of triples. For example, if a user asks about *"blood pressure 160"*, it will trigger the rules as:

```
Define (DiseaseSeverity/`Grade 1 Hypertension`) {
  SystolicPressure >= 140 OR DiastolicPressure >= 90
}

Define (DiseaseSeverity/`Grade 2 Hypertension`) {
  SystolicPressure >= 160 OR DiastolicPressure >= 100
}

Define (Disease/`Hypertension`) {
  DiseaseSeverity/`Grade 1 Hypertension` OR DiseaseSeverity/`Grade 2 Hypertension`
}
```

, which strictly follows the defination of $\mathscr{L}$ in LLMFriSPG, and the conclusion that the person may have hypertension will be obtained.

In the semantic enhancement phase, we utilize the term set to express the two semantic relations of synonymy and hypernym of concepts. The hypernym supports the expression of multiple hypernyms. During knowledge construction and user Q&A phase, entities are aligned with medical terms. For example, in the concept of surgery type, the hypernym of deciduous tooth extraction and anterior tooth extraction is tooth extraction. When the user only asks questions about tooth extraction, all its hyponyms can be retrieved based on the term, and then the related entity information can be retrieved for answering. With the support of KAG, we achieved a recall rate of 60.67% and a precision rate of 81.32% on the evaluation set which sampling online Q&A queries. In the end-to-end scenario, the accuracy of medical insurance policy inquires (Beijing, Shanghai, Hangzhou) reached 77.2%, and the accuracy rate of popular science intentions has exceeded 94%, and the accuracy rate of interpreting indicator intentions has exceeded 93%.

5 Related Works

5.1 DIKW Pyramid

Following the DIKW pyramid theories[41, 42, 43, 44], after data is processed and contextualised, it becomes information, and by integrating information with experience, understanding, and expertise, we gain knowledge. We usually use information extraction technology to obtain information from the original text[45, 46, 47], and obtain knowledge from the information through linking, fusion, analysis, and learning technology[43, 48, 46]. Information and knowledge are a single entity having different forms. There are no unified language to represent data, information and knowledge, RDF/OWL[49] only provides binary representation in the form of triples, and LPG[21] lacks support for knowledge semantics and classification. SPG[4][50] supports knowledge hierarchy and classification representation, but lacks text context support that is friendly to large language models. Our proposed LLMFriSPG supports hierarchical representation from data to information to knowledge, and also provides reverse context-enhanced mutual-indexing.

5.2 Vector Similarity-based RAG

The external knowledge base use the traditional search engine provides an effective method for updating the knowledge of LLMs, it retrievals supporting documents by calculating the text or vector similarity[1, 4] between the query and document, and then answers questions using the in-context learning method of LLMs. In addition, this method faces great challenges in understanding long-distance knowledge associations between documents. Simple vector-based retrieval is not suitable for multi-step reasoning or tracking logical links between different information fragments. To address these challenges, researchers have explored methods such as fine-grained document segmentation, CoT[33], and interactive retrieval[26, 2]. Despite these optimizations, traditional query-chunks similarity methods still has difficulty in accurately focusing on the relations between key knowledge in complex questions, resulting in low information density and ineffective association of remote knowledge. We will illustrate the logical-form-guided solving method.

[4]Official site of SPG: https://spg.openkg.cn/en-US

5.3 Information Retrieval-based GraphRAG

This type of methods use information extraction techniques to build entity and relation associations between different documents, which can better perceive the global information of all documents. Typical tasks in the knowledge construction phase include: graph information extraction and knowledge construction&enhancement. Methods like GraphRAG[51], ToG 2.0[9], HippoRAG[12] use OpenIE to extract graph-structure information like entities and relations, some of them exploit multi-hop associations between entities to improve the effectiveness of cross-document retrieval[9, 12], methods like DALK[7] use PubTator Central(PTC) annotation to reduce the noise problem of openIE, some of them utilize entity disambiguation technology to enhance the consistency of graph information[12, 52]. GraphRAG[51] generates element-level and community-level summaries when building offline indexes, and it uses a QFS[53] method to first calculate the partial response of each summary to the query and then calculate the final response. This inherent characteristic of GraphRAG's hierarchical summarization makes it difficult to solve questions such as multi-hop Q&A and incremental updates of documents. KGs constructed by openIE contains a lot of noise or irrelevant information[54, 55, 56]. According to the DIKW pyramid hierarchy, these methods only extract the information graph structure and make limited attempts to disambiguate entities in the transformation of information into knowledge,but they do not address issues such as semantic directionality and logical sensitivity. This paper will introduce a method in KAG to enhance information-to-knowledge conversion based on domain concept semantic graph alignment.

5.4 KG-based Question and Answering

Reasoning based on traditional KGs has good explainability and transparency, but is limited by the scale of the domain KG, the comprehensiveness of knowledge, the detailed knowledge coverage, and the timeliness of updates[57]. n this paper, we introduce HybridReasoning to alleviate issues such as knowledge sparsity, inconsistent entity granularity, and high graph construction costs. The approach leverages KG retrieval and reasoning to enhance generation, rather than completely replacing RAG.

To achieve KG-enhanced generation, it is necessary to address KG-based knowledge retrieval and reasoning. One approach is knowledge edge retrieval (IR)[58], which narrows down the scope by locating the most relevant entities, relations, or triples based on the question. Another approach is semantic parsing (SP)[59, 60], which converts the question from unstructured natural language descriptions into executable database query languages (such as SQL, SPARQL[61], DSL[5], etc.), or first generates structured logical forms (such as S-expressions[62, 63]) and then converts them into query languages.

Although conversational QA over large-scale knowledge bases can be achieved without explicit semantic parsing (e.g., HRED-KVM[64]), most work focuses on exploring context-aware semantic parsers[60, 65, 63].

Some papers use sequence-to-sequence models to directly generate query languages[66, 67]. These methods are developed for a specific query language, and sometimes even for a specific dataset, lacking generality for supporting different types of structured data. Others use step-by-step query graph generation and search strategies for semantic parsing[68, 69, 70]. This method is prone to uncontrollable issues generated by LLM, making queries difficult and having poor interpretability. Methods like ChatKBQA[63], CBR-KBQA[71] completely generate S-expressions and provide various enhancements for the semantic parsing process. However, the structure of S-expressions is relatively complex, and integrating multi-hop questions makes it difficult for LLMs to understand and inconvenient for integrating KBQA and RAG for comprehensive retrieval. To address these issues, we propose a multi-step decomposed logical form to express the multi-hop retrieval and reasoning process, breaking down complex queries into multiple sub-queries and providing corresponding logical expressions, thereby achieving integrated retrieval of SPO and chunks.

5.5 Bidirectional-enhancement of LLMs and KGs

LLM and KG are two typical neural and symbolic knowledge utilization methods. Since the pre-trained language model such as BERT [72], well-performed language models are used to help improve the tasks of KGs. The LLMs with strong generalization capability are especially believed to be

[5]DSL: https://openspg.yuque.com/ndx6g9/ooil9x/sdtg4q3bw4ka5wmz

helpful in the life-cycle of KGs. There are a lot of works conducted to explore the potential of LLMs for in-KG and out-of-KG tasks. For example, using LLMs to generate triples to complete triples is proved to be much cheaper than the traditional human-centric KG construction process, with acceptable accuracy for the popular entities [73]. In the past decade, methods for in-KG tasks are designed by learning from KG structures, such as structure embedding-based methods. The text information such as names and descriptions of entities is not fully utilized due to the limited text understanding capability of natural language processing methods until LLMs provide a way. Some works using LLMs for text semantic understanding and reasoning of entities and relations in KG completion [74], rule learning [75], complex logic querying [76], etc. On the other way, KGs are also widely used to improve the performance of LLMs. For example, using KGs as external resources to provide accurate factual information, mitigating hallucination of LLMs during answer generation [9], generating complex logical questions answering planning data to fine-tune the LLMs, improving LLMs planning capability and finally improving its logical reasoning capability [77], using KGs to uncover associated knowledge that has changed due to editing for better knowledge editing of LLMs [78], etc. The bidirectional-enhancement of LLMs and KGs is widely explored and partially achieved.

6 Limitations

In this article, we have proven the adaptability of the KAG framework in Q&A scenarios in vertical and open domains. However, the currently developed version of OpenSPG-KAG 0.5 still has major limitations that need to be continuously overcome, such as:

Implementing our framework requires multiple LLM calls during the construction and solving phases. A substantial number of intermediate tokens required to be generated during the planning stage to facilitate the breakdown of sub-problems and symbolic representation, this leads to computational and economic overhead, as illustrated in Table 14, where the problem decomposition not only outputs sub-problems but also logical functions, resulting in approximately twice as many generated tokens compared to merely decomposing the sub-problems. Meanwhile, currently, all model invocations within the KAG framework, including entity recognition, relation extraction, relation recall, and standardization, rely on large models. This multitude of models significantly increases the overall runtime. In future domain-specific implementations, tasks like relation recall, entity recognition, and standardization could be substituted with smaller, domain-specific models to enhance operational efficiency.

The ability to decompose and plan for complex problems requires a high level of capability. Currently, this is implemented using LLMs, but planning for complex issues remains a significant challenge. For instance, when the task is to compare who is older, the problem should be decomposed into comparing who was born earlier. Directly asking for age is not appropriate, as they are deceased, and "what is the age" refers to the age at death, which doesn't indicate who is older. Decomposing and planning complex problems necessitates ensuring the model's accuracy, stability, and solvability in problem decomposition and planning. The current version of the KAG framework does not yet address optimizations in these areas. We will further explore how pre-training, SFT, and COT strategies can improve the model's adaptability to logical forms and its planning and reasoning capabilities.

> Question: *Which film has the director who is older, God'S Gift To Women or Aldri Annet Enn Bråk?*
> Q1: *Which director directed the film God'S Gift To Women?* A1: **Michael Curtiz**
> Q2: *Which director directed the film Aldri Annet Enn Bråk?* A2: **Edith Carlmar**
> Q3: *What is the age of the director of God'S Gift To Women?* A3: **74 years old.** *Michael Curtiz (December 24, 1886 to April 11, 1962)...*
> Q4: *What is the age of the director of Aldri Annet Enn Bråk?* A4: **91 years old.** *Edith Carlmar (Edith Mary Johanne Mathiesen) (15 November 1911 to 17 May 2003) ...*
> Q5: *Compare the ages of the two directors to determine which one is older.* A5: **Edith Carlmar is older**. *Actually, Michael Curtiz was born earlier.*

OpenIE significantly lowers the threshold for building KGs, but it also obviously increases the technical challenges of knowledge alignment. Although the experiments in this article have shown that the accuracy and connectivity of extracted knowledge can be improved through knowledge alignment. However, there are still more technical challenges waiting to be overcome, such as optimizing the accuracy of multiple-knowledge(such as events, rules, pipeline, etc.) extraction

and the consistency of multiple rounds of extraction. In addition, schema-constraint knowledge extraction based on the experience of domain experts is also a key way to obtain rigorous domain knowledge, although the labor cost is high. These two methods should be applied collaboratively to better balance the requirements of vertical scenarios for the rigor of complex decision-making and the convenience of information retrieval. For instance, when extracting team members from multiple texts and asked about the total number of team members, a comprehensive extraction is crucial for providing an accurate answer based on the structured search results. Incorrect extractions also impair response accuracy.

7 Conclusion and Future Work

In order to build professional knowledge services in vertical domains, fully activate the capabilities and advantages of symbolic KGs and parameterized LLMs, and at the same time significantly reduce the construction cost of domain KGs, we proposed the KAG framework and try to accelerated its application in professional domains. In this article, we introduce in detail the knowledge accuracy, information completeness and logical rigorous are the key characteristics that professional knowledge services must have. At the same time, we also introduce innovations such as LLMs friendly knowledge representation, mutual-indexing of knowledge structure and text chunks, knowledge alignment by semantic reasoning, logic-form-guided hybrid reasoning&solving and KAG model. Compared with the current most competitive SOTA method, KAG has achieved significant improvements on public data sets such as HotpotQA, 2wiki, musique. We have also conducted case verifications in E-goverment Q&A and E-Health Q&A scenarios of Alipay, further proving the adaptability of the KAG framework in professional domains.

In the future, there is still more work to be explored to continuously reduce the cost of KG construction and improve the interpretability and transparency of reasoning, such as multiple knowledge extraction, knowledge alignment based on **OneGraph**, domain knowledge injection, large-scale instruction synthesis, illusion suppression of knowledge logic constraints, etc.

This study does not encompass the enhancement of models for decomposing and planning complex problems, which remains a significant area for future research. In future work, KAG can be employed as a reward model to provide feedback and assess the model's accuracy, stability, and solvability through the execution of planning results, thereby enhancing the capabilities of planning models.

We will also work in depth with the community organization **OpenKG** to continue to tackle key technical issues in the collaboration between LLMs and KGs.

8 Acknowledgements

This work was completed by the AntGroup Knowledge Graph Team, in addition to the authors in the list, other contributors include Yuxiao He, Deng Zhao, Xiaodong Yan, Dong Han, Fanzhuang Meng, Yang Lv, Zhiying Yin, etc, thank you all for your continuous innovation attempts and hard work. This work also received strong support from Professor Huajun Chen, Researcher Wen Zhang of Zhejiang University, and Professor Wenguang Chen of AntGroup Technology Research Institute, thank you all.

References

[1] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[2] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Com-*

putational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023, pages 9248–9274. Association for Computational Linguistics, 2023.

[3] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

[4] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501, 2024.

[5] Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *arXiv preprint arXiv:2311.09210*, 2023.

[6] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2024.

[7] Dawei Li, Shu Yang, Zhen Tan, Jae Young Baik, Sunkwon Yun, Joseph Lee, Aaron Chacko, Bojian Hou, Duy Duong-Tran, Ying Ding, et al. Dalk: Dynamic co-augmentation of llms and kg to answer alzheimer's disease questions with scientific literature. *arXiv preprint arXiv:2405.04819*, 2024.

[8] Minki Kang, Jin Myung Kwak, Jinheon Baek, and Sung Ju Hwang. Knowledge graph-augmented language models for knowledge-grounded dialogue generation, 2023.

[9] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, and Jian Guo. Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. *arXiv preprint arXiv:2407.10805*, 2024.

[10] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation, 2024.

[11] Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning, 2024.

[12] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*, 2024.

[13] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8:489–508, 2016.

[14] Siwei Wu, Xiangqing Shen, and Rui Xia. Commonsense knowledge graph completion via contrastive pretraining and node clustering, 2023.

[15] Yi-Hui Chen, Eric Jui-Lin Lu, and Kwan-Ho Cheng. Integrating multi-head convolutional encoders with cross-attention for improved sparql query translation, 2024.

[16] Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. Knowledge base question answering: A semantic parsing perspective, 2022.

[17] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[18] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics, 2020.

[19] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022.

[20] Dirk Groeneveld, Tushar Khot, Mausam, and Ashish Sabharwal. A simple yet strong pipeline for hotpotqa. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8839–8845. Association for Computational Linguistics, 2020.

[21] Chandan Sharma and Roopak Sinha. A schema-first formalism for labeled property graph databases: Enabling structured data loading and analytics. In *Proceedings of the 6th ieee/acm international conference on big data computing, applications and technologies*, pages 71–80, 2019.

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

[23] Hugo Liu and Push Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

[24] Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. How easily do irrelevant inputs skew the responses of large language models? *arXiv preprint arXiv:2404.03302*, 2024.

[25] Honghao Gui, Hongbin Ye, Lin Yuan, Ningyu Zhang, Mengshu Sun, Lei Liang, and Huajun Chen. Iepile: Unearthing large-scale schema-based information extraction corpus. *arXiv preprint arXiv:2402.14710*, 2024.

[26] Zhouyu Jiang, Mengshu Sun, Lei Liang, and Zhiqiang Zhang. Retrieve, summarize, plan: Advancing multi-hop question answering with an iterative approach. *arXiv preprint arXiv:2407.13101*, 2024.

[27] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[28] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[29] Xiongtao Cui and Jungang Han. Chinese medical question answer matching based on interactive sentence representation learning. volume abs/2011.13573, 2020.

[30] Anastasios Nentidis, Georgios Katsimpras, Eirini Vandorou, Anastasia Krithara, Antonio Miranda-Escalada, Luis Gasco, Martin Krallinger, and Georgios Paliouras. Overview of BioASQ 2022: The tenth BioASQ challenge on large-scale biomedical semantic indexing and question answering. In *Lecture Notes in Computer Science*, pages 337–361. Springer International Publishing, 2022.

[31] Zhouyu Jiang, Ling Zhong, Mengshu Sun, Jun Xu, Rui Sun, Hui Cai, Shuhan Luo, and Zhiqiang Zhang. Efficient knowledge infusion via KG-LLM alignment. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 2986–2999. Association for Computational Linguistics, 2024.

[32] Jintian Zhang, Cheng Peng, Mengshu Sun, Xiang Chen, Lei Liang, Zhiqiang Zhang, Jun Zhou, Huajun Chen, and Ningyu Zhang. Onegen: Efficient one-pass unified generation and retrieval for llms, 2024.

[33] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics, 2023.

[34] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3715–3734. Association for Computational Linguistics, 2022.

[35] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In

Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[36] Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 232–241. ACM/Springer, 1994.

[37] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022, 2022.

[38] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. Large dual encoders are generalizable retrievers. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 9844–9855. Association for Computational Linguistics, 2022.

[39] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[40] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. Dense X retrieval: What retrieval granularity should we use? *CoRR*, abs/2312.06648, 2023.

[41] Russell L Ackoff. From data to wisdom. *Journal of applied systems analysis*, 16(1):3–9, 1989.

[42] Sasa Baskarada and Andy Koronios. Data, information, knowledge, wisdom (dikw): A semiotic theoretical and empirical exploration of the hierarchy and its quality dimension. *Australasian Journal of Information Systems*, 18(1), 2013.

[43] Jose Claudio Terra and Terezinha Angeloni. Understanding the difference between information management and knowledge management. *KM Advantage*, pages 1–9, 2003.

[44] Jonathan Hey. The data, information, knowledge, wisdom chain: the metaphorical link. *Intergovernmental Oceanographic Commission*, 26(1):1–18, 2004.

[45] Sunita Sarawagi et al. Information extraction. *Foundations and Trends® in Databases*, 1(3):261–377, 2008.

[46] Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 65–76, 2010.

[47] Jakub Piskorski and Roman Yangarber. Information extraction: Past, present and future. *Multi-source, multilingual information extraction and summarization*, pages 23–49, 2013.

[48] Priti Srinivas Sajja and Rajendra Akerkar. Knowledge-based systems for development. *Advanced Knowledge Based Systems: Model, Applications & Research*, 1:1–11, 2010.

[49] Dean Allemang and James Hendler. *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[50] Peng Yi, Lei Liang, Yong Chen Da Zhang, Jinye Zhu, Xiangyu Liu, Kun Tang, Jialin Chen, Hao Lin, Leijie Qiu, and Jun Zhou. Kgfabric: A scalable knowledge graph warehouse for enterprise data interconnection.

[51] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[52] Bhaskarjit Sarmah, Benika Hall, Rohan Rao, Sunil Patel, Stefano Pasquali, and Dhagash Mehta. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. *arXiv preprint arXiv:2408.04948*, 2024.

[53] Hoa Trang Dang. Duc 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, pages 48–55, 2006.

[54] Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. Aser: A large-scale eventuality knowledge graph. In *Proceedings of the web conference 2020*, pages 201–211, 2020.

[55] Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. Codekgc: Code language model for generative knowledge graph construction. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 23(3):1–16, 2024.

[56] Tianqing Fang, Hongming Zhang, Weiqi Wang, Yangqiu Song, and Bin He. Discos: Bridging the gap between discourse knowledge and commonsense knowledge. In *Proceedings of the Web Conference 2021*, pages 2648–2659, 2021.

[57] Ling Tian, Xue Zhou, Yan-Ping Wu, Wang-Tao Zhou, Jin-Hao Zhang, and Tian-Shu Zhang. Knowledge graph and knowledge reasoning: A systematic review. *Journal of Electronic Science and Technology*, 20(2):100159, 2022.

[58] Yiyu Yao, Yi Zeng, Ning Zhong, and Xiangji Huang. Knowledge retrieval (kr). In *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, pages 729–735. IEEE, 2007.

[59] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[60] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2946–2955, 2018.

[61] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora M. Aroyo, editors, *The Semantic Web - ISWC 2006*, pages 30–43, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[62] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488, New York, NY, USA, 2021. Association for Computing Machinery.

[63] Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Luu Anh Tuan. 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*. Association for Computational Linguistics, 2024.

[64] Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. Conversational question answering over knowledge graphs with transformer and graph attention networks. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 850–862. Association for Computational Linguistics, 2021.

[65] Yunshi Lan and Jing Jiang. Modeling transitions of focal entities for conversational knowledge base question answering. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3288–3297. Association for Computational Linguistics, 2021.

[66] Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander G. Gray, Ramón Fernandez Astudillo, Maria Chang, et al. Leveraging abstract meaning representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3884–3894. Association for Computational Linguistics, 2021.

[67] Reham Omar, Ishika Dhall, Panos Kalnis, and Essam Mansour. A universal question-answering platform for knowledge graphs. *Proceedings of the ACM on Management of Data*, 1(1):57:1–57:25, 2023.

[68] Farah Atif, Ola El Khatib, and Djellel Difallah. Beamqa: Multi-hop knowledge graph question answering with sequence-to-sequence prediction and beam search. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 781–790, New York, NY, USA, 2023. Association for Computing Machinery.

[69] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9237–9251. Association for Computational Linguistics, 2023.

[70] Yu Gu, Xiang Deng, and Yu Su. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949, Toronto, Canada, July 2023. Association for Computational Linguistics.

[71] Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 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, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[72] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[73] Blerta Veseli, Simon Razniewski, Jan-Christoph Kalo, and Gerhard Weikum. Evaluating the knowledge base completion potential of GPT. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6432–6443. Association for Computational Linguistics, 2023.

[74] Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. Making large language models perform better in knowledge graph completion. *ACM MM*, 2024.

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

[76] Nurendra Choudhary and Chandan K. Reddy. Complex logical reasoning over knowledge graphs using large language models. *CoRR*, abs/2305.01157, 2023.

[77] Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, Jeff Z. Pan, Wen Zhang, and Huajun Chen. Learning to plan for retrieval-augmented large language models from knowledge graphs. *CoRR*, abs/2406.14282, 2024.

[78] Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. Knowledge graph enhanced large language model editing. *CoRR*, abs/2402.13593, 2024.

A Example of KAG Solver

Round One

Initial Question: How many times did the plague occur in the birth place of Concerto in C Major Op 3 6's composer?

Step1: Who is the composer of Concerto in C Major Op 3 6?.
Logical Form: Retrieval(s=s1:Work[C Major Op 3 6],p=p1:composer,o=o1:Person)
Graph Retrieval: (antonio vivaldi, composed concerto in, c major op 3 6)
Hybird Retrieval: not executed
A1: Antonio Vivaldi

Step2: What is the birthplace of this composer?
Logical Form: Retrieval(s=o1,p=p2:brithPlace,o=o2:Place)
Graph Retrieval: not found
Hybird Retrieval:
doc1:Antonio Vivaldi composed a mandolin concerto (Concerto in C major Op.3 6) ...
doc2:Antonio Lucio Vivaldi (Italian: ... Born in **Venice**...
...
A2: Venice

Step3: How many times did the plague occur in this birthplace?
Logical Form: Retrieval(s=o2,p=p3:plagueOccurrence,o=o3:Plague)
Graph Retrieval: not found
Hybird Retrieval:
doc1:Antonio Vivaldi composed a mandolin concerto (Concerto in C major Op.3 6) ...
doc2:The Black Death, also known as the Great Plague.. in Europe from 1347 to 1351..
...
A3: I don't know

Step4: Count the number of plague occurrences.
Logical Form: Count(o3)->nums
Graph Retrieval: not found
Hybird Retrieval: not found
A4: I don't know

Round Two

Rewrite Question: What specific records or historical accounts detail the number of plague occurrences in Venice during the mid-14th century?

Step1: What specific records or historical accounts detail the number of plague occurrences in Venice during the mid-14th century?
Logical Form: Retrieval(s=s1:City[Venice],p=p3:plagueOccurrencesInMid14thCentury,o=o1:Times)
Graph Retrieval: not found
Hybird Retrieval:
doc1:In 1466, perhaps 40,000 people died of the plague...**Plague occurred in Venice 22 times between 1361 and 1528**...
doc2:The Black Death, also known as the Great Plague...
...
A1: The plague occurred in Venice 22 times between 1361 and 1528. The 1576–77 plague killed 50,000, almost a third of the population.

Final Answer

Question: How many times did the plague occur in the birth place of Concerto in C Major Op 3 6's composer?

Step1: What specific records or historical accounts detail the number of plague occurrences in Venice during the mid-14th century?
A: 22 times

Table 13: An example of using logical-from to guide question planning, reasoning, retrieval, and answer generation, and using multiple rounds of reflection to rephrase questions.

B Example of Logical form Reasoner

Numerical Reasoning

question: Which sports team for which Cristiano Ronaldo played in 2011 was founded last ?
Step1: Identify the Sports Teams Cristiano Ronaldo Played for in 2011 .
Logical Form: Retrieval(s=s1:Player[Cristiano Ronaldo], p=p1:playedFor, o=o1:SportsTeam, p.PlayedForInYear=2011)
Step2: Determine the Foundation Years of Each Identified Team.
Logical Form: Retrieval(s=o1, p=p2:foundationYear, o=o2:Year)
Step3: Which team was founded last?
Logical Form: Sort(set=o1, orderby=o2, direction=max, limit=1)

question: What is the sum of 30 + 6 and the age of the founder of Tesla in 2027 ?
Step1: What is the sum of 30 + 6 ?
Logical Form: math1 = Math(30+6)
Step2: Who is the founder of Tesla?
Logical Form: Retrieval(s=s2:Company[Tesla], p=p2:founder, o=o2)
Step3: In which year was the founder of Tesla born?
Logical Form: Retrieval(s=o2, p=p3:yearOfBirth, o=o3)
Step4: How old will the founder of Tesla be in the year 2027?
Logical Form: math4 = Math(2027-o3)
Step5: What is the sum of math1 and math4?
Logical Form: math5 = Math(math1+math4)

Logical Reasoning

question: Find a picture containing vegetables or fruits.
Step1: Find pictures containing vegetables.
Logical Form: Retrieval(s=s1:Image, p=p2:contains, o=o1:Vegetables)
Step2: Find pictures containing fruits.
Action2: Retrieval(s=s2:Image, p=p2:contains, o=o2:Fruits)
Step3: Output s1, s2.
Logical Form: Output(s1, s2)

question: Find a picture containing vegetables and fruits.
Step1: Find pictures containing vegetables.
Logical Form: Retrieval(s=s1:Image, p=p2:contains, o=o1:Vegetables)
Step2: Find pictures containing fruits.
Logical Form: Retrieval(s=s1, p=p2:contains, o=o2:Fruits)
Step3: Output s1.
Logical Form: Output(s1)

Semantic Deduce

question: Do I need to present the original ID card when applying for a passport?
Step1: What documents are required to apply for a passport?
Logical Form: Retrieval(s=s1:Event[apply for a passport], p=p1:support_chunks, o=o1:Chunk)
Step2: Does this set of documents include the original identity card?
Logical Form: Deduce(left=o1, right=the original identity card, op=entailment)

Table 14: The cases of reasoning with logical form