

Redefining Information Retrieval of Structured Database via Large Language Models

1st Mingzhu Wang

School of Data Science

University of Science and Technology of China *University of Science and Technology of China*

ACAM and FinChina Joint Laboratory

Hefei, China

wangmz@mail.ustc.edu.cn

2nd Yuzhe Zhang

School of Management

Hefei, China

zyz2020@mail.ustc.edu.cn

3rd Qihang Zhao

Kuaishou Technology

Beijing

zhaoqihang@kuaishou.com

4th Junyi Yang

School of Mathematics

University of Science and Technology of China

Hefei, China

rrrita@mail.ustc.edu.cn

5^{*} Hong Zhang

School of Management

University of Science and Technology of China

Hefei, China

zhangh@ustc.edu.cn

Abstract—Retrieval augmentation is critical when Language Models (LMs) exploit non-parametric knowledge related to the query through external knowledge bases before reasoning. The retrieved information is incorporated into LMs as context alongside the query, enhancing the reliability of responses towards factual questions. Prior researches in retrieval augmentation typically follow a retriever-generator paradigm. In this context, traditional retrievers encounter challenges in precisely and seamlessly extracting query-relevant information from knowledge bases. To address this issue, this paper introduces a novel retrieval augmentation framework called ChatLR that primarily employs the powerful semantic understanding ability of Large Language Models (LLMs) as retrievers to achieve precise and concise information retrieval. Additionally, we construct an LLM-based search and question answering system tailored for the financial domain by fine-tuning LLM on two tasks including Text2API and API-ID recognition. Experimental results demonstrate the effectiveness of ChatLR in addressing user queries, achieving an overall information retrieval accuracy exceeding 98.8%.

Index Terms—Information Retrieval, Retrieval Augmentation, Large Language Models, Structured Database.

I. INTRODUCTION

Knowledge-intensive tasks, such as question answering and dialogue generation [1]–[3], which demand intricate memorization of factual knowledge, face challenges in achieving optimal performance directly on LLMs. This is attributed to the fact that a standalone Large Language Model (LLM) can only incorporate limited knowledge up to the point when it was trained, and it is a significant cost to update its parameters with the latest knowledge [1]. Consequently, relying on implicit reasoning process of LLMs can sometimes lead to inaccuracies and complexities, hindering their application in knowledge-intensive tasks [4]–[6]. Another critical issue arises from the well-known “hallucination problem” in LLMs, where they tend to generate factually inaccurate or erroneous responses

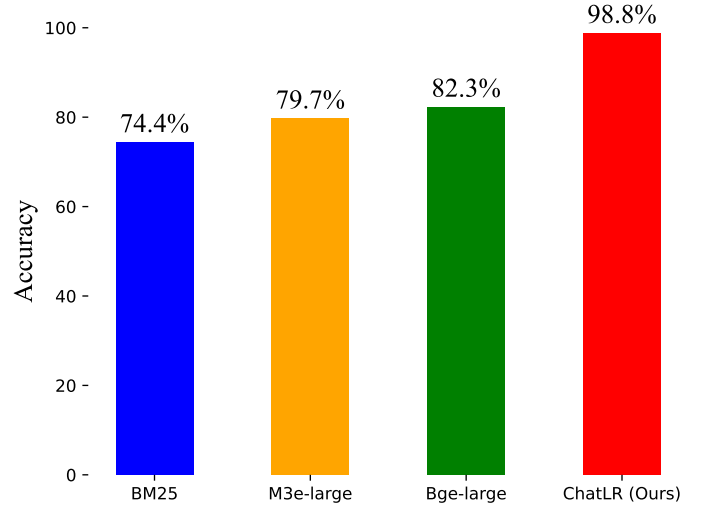


Fig. 1. Performance of ChatLR (Ours).

[7]–[9]. This problem persists across various scenarios involving LLMs and becomes more pronounced when generating responses in vertical domains or with private data.

A promising approach to address the aforementioned challenges is Retrieval-Augmented Generation (RAG) [5]. Instead of relying solely on the model’s parameterized knowledge for inference, RAG enhances LMs by extracting knowledge relevant to the query from an external corpus using a retriever model [10]. Subsequently, the retrieved information serves as prior knowledge, combined with the query, and is then fed into the LLMs to guide its reasoning during the generation process. In this way, the retrieved knowledge significantly aids in overcoming challenges stemming from the reliance on parameterized knowledge stored within the parameters of LLM [11]. The RAG approach essentially equips the LLM with a

*Corresponding Author

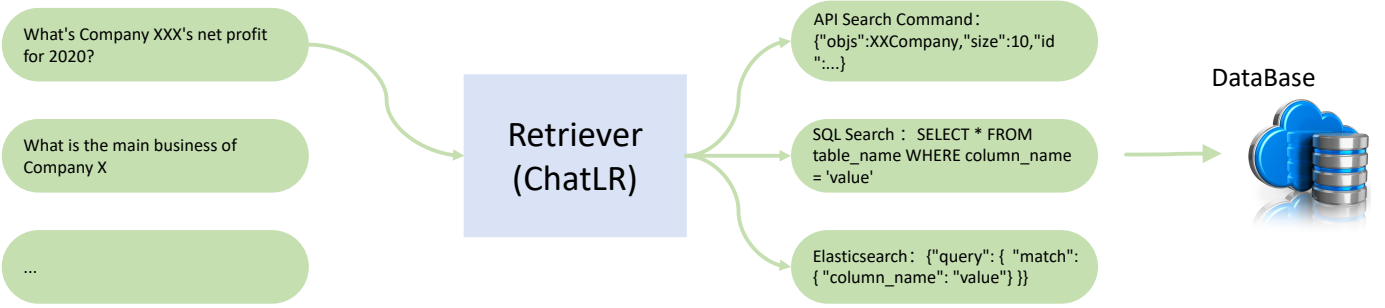


Fig. 2. An example of ChatLR. ChatLR facilitates the retrieval of relevant information within a database by transforming queries (e.g., “What’s Company XXX’s net profit for 2020?”) into search command statements, such as API search commands or SQL queries.

knowledge base containing accurate and extensive knowledge. When the LLM encounters challenges in solving problems outside its expertise, it can consult this knowledge base to find relevant information before proceeding with the resolution. Moreover, the cost of updating the knowledge base is much smaller compared to retraining the LLM, emphasizing the practical advantages of timely knowledge updates.

A typical RAG framework consists of two main components: the retriever and the generator. The retriever is responsible for extracting information relevant to the query from a database, while the generator, usually a Language Model (LM), analyzes the retrieved information and the user’s query to produce the final response. In previous RAG frameworks, the retriever’s operation typically involves embedding external information and the query into vector, calculating the similarity between them, and selecting the most helpful information for answering the query based on vector similarities [12], [13]. Recent advancements have focused more on using LMs to provide supervisory signals for training the retriever. This involves the joint training of the retriever and the LM: if the LM identifies useful documents during output generation, the retriever’s objective should encourage higher rankings for those documents [11]. In cases where the LM is a parameter-frozen black box, supervised signals generated by the LM are utilized to optimize only the retriever to make it inclined to select information that reduces the perplexity of the text generated by the LM [14].

The information selected by the retriever plays a crucial role in the ability to respond to queries of LM. When the reference material provided to the LLM contains inaccurate content, the LM can hardly generate correct answers. Therefore, the retriever’s search results should not only be highly relevant but also contain minimal noise. What we require, in essence, is a retrieval model capable of providing precise and concise information. This need is particularly pronounced when addressing questions in vertical domains such as finance and law, where accuracy and conciseness in information retrieval results are paramount. However, current information retrieval frameworks, even after fine-tuning for downstream tasks, still cannot reach satisfactory accuracy in real applications.

To address the aforementioned issue, this paper introduces a novel retrieval augmentation framework termed : LLM Re-

trieval with Chat (ChatLR), which leverages LLM for retrieval operations. ChatLR is specifically designed to cater to factual inquiries concerning structured databases. The integration of LLMs significantly enhances the accuracy of knowledge retrieval related to queries. Building upon the precision of data relationships stored in structured databases, our information retrieval framework achieves accurate and efficient searching of relevant information within the structured database by mapping natural language queries to precise database search commands. In the event of changes to the scenario or database, one can effortlessly generalize the framework to various vertical domains by modifying the relevant content in accordance with the ChatLR framework.

In the context of structured databases, we leverage the inherent characteristic of databases possessing structured search commands. We delineate the objective of the retriever as the generation of JSON-formatted search command statements, rather than directly retrieving data. This approach transforms the retriever’s task into a sequence-to-sequence task amenable to targeted fine-tuning by LLMs. Leveraging the powerful semantic understanding and generation capabilities of LLMs, we can directly employ an LLM to generate specific commands based on the query text and sufficient information about the database search commands. In cases where the database utilizes raw SQL queries, we perform the Text2SQL task [15]. This framework is adaptable to various types of queries in different scenarios. When dealing with LLMs like GPT-4 as a black-box model, a common approach is to utilize few-shot learning to stimulate the model’s reasoning abilities [15], though the drawback is that the accuracy of few-shot learning may not be satisfactory. As for the case that LLMs are trainable, fine-tuning can be applied for specific tasks to achieve higher accuracy.

The contribution of this paper can be summarized as follows:

- We propose a novel retrieval augmentation framework named ChatLR. Utilizing ChatLR, we have engineered an intelligent system based on LLM for search and question-answering, interfacing seamlessly with structured databases. By decomposing the targets generated by queries into API-ID recognition task and Text2API task, we conducted experiments to evaluate the effectiveness

of ChatLR.

- For the tasks of API-ID recognition and Text2API, we devised a novel instruction fine-tuning data format. The data generation process was executed utilizing the GPT3.5 API. Subsequently, an extensive manual review and refinement process was undertaken to enhance the quality of the data, resulting in nearly 70,000 instances of high-quality fine-tuning data.
- Finally, we conducted fine-tuning experiments on the open-sourced LLM named Chinese-Alpaca-33B-Pro [16]. The final testing results revealed that the fine-tuned ChatLR achieved accuracies exceeding 99.9% and 98.9% for two specific tasks, with an overall retrieval accuracy of 98.8%. Furthermore, it showcased superiority over other information retrieval algorithms such as BM25 [17], M3e-large [18], and Bge-large [19] (see Figure 1). These findings underscore the efficacy of the ChatLR framework in structured data retrieval, validating its performance against established alternatives in the field.

II. RELATED WORK

A. Retrieval Augmentation

Incorporating retrieved information from external sources into LMs has proven to be effective in a wide range of knowledge-intensive tasks, including factual question answering, fact checking, etc. [1], [2], [4]. Previous works encapsulated the retriever-generator paradigm, leveraging the advantage of generative models [5] and neural retrievers [4], [20], [21]. Rather than solely relying on inherent knowledge and reasoning capabilities, retrieval augmentation approaches enhance the LM by integrating a retriever component capable of sourcing knowledge from the external corpus [4], [5], [22]. Specifically, previous retrieval augmentation methods [11], [23] required fine-tuning the core fractions of LM to adapt to the retriever on specific downstream tasks. With the emergence of LLMs, there have been endeavors to employ LLMs as generators for question-answering tasks under few-shot and zero-shot setting after the information retrieval process [1], [10], [24]. Since then, fine-tuning LLMs becomes prohibitively expensive as the number of unique demands continue to increase [25]. Moreover, many state-of-the-art LLMs can only be accessed via black-box APIs [26], [27]. These APIs enable users to submit queries and receive responses but generally do not support fine-tuning. To enhance the accuracy of LLMs in answering user queries, substantial prompt engineering or fine-tuning retrievals using small-scaled LMs is required is necessary [10], [24].

B. Structured Knowledge-Base Retrieval Augmentation

Retrieval augmentation techniques can be applied to both structured and unstructured knowledge bases. For retrieval from structured knowledge bases, Sun et al. [28] explored open-domain question answering only from web tables without leveraging the unstructured text data. Recent studies have delved into Machine Reading Comprehension (MRC) with

tables, although without a retrieval module [29]–[31]. Furthermore, Chen et al. [32] conducted research on open-domain question answering utilizing both tabular data and textual sources.

Retrieval Augmentation of primary unstructured databases is the process of retrieving relevant information from unstructured data, such as text and knowledge graph, according to the user queries, and then making LMs generate responses. This line of research is commonly referred to as TextQA, primarily designed for open-domain question answering tasks from textual data. Previous approaches often relied on graph neural networks for fine-tuning or adopted linearization of structured knowledge and combine it with texts [33]–[36]. By framing user queries, structured knowledge, and outputs in the text-to-text format [37], our work aims to advance the field of retrieval augmentation for structured database.

In this paper, we focus on a structured knowledge base (SKB) that restore substantial amounts of authoritative data, often presented in tabular form. It is worth noting that, in contrast to relational databases and traditional KBs, tables can be best described as semi-structured information. In the financial field, structured tabular data can, to a certain extent, ensure data accuracy and consistency. Therefore, when performing retrieval augmentation on SKB, it is crucial to prioritize the retrieval performance. Previous studies have employed various methods, including relational chains and deep neural networks [28], beam search [29], iterative retrievers [32], among others. However, this paper primarily focuses on utilizing LLMs for retrieval augmentation for structured databases.

III. METHOD

In this section, we first present our ChatLR framework in subsection III-A, followed by an introduction to the fine-tuning data generation paradigm in subsection III-B, and concluding with an overview of the fine-tuning methodology in subsection III-C.

A. The Overview of ChatLR Framework

Previous information retrieval algorithms such as BERT [13] and Contriever [12], typically embed external knowledge and queries into dense vectors, and select top k similar results from the external knowledge base. Generally, retrieval algorithms can be fine-tuned for downstream tasks. For instance, REPLUG [14] utilizes LMs to generate supervised signals, optimizing the retrieval component to align with the LM. However, even after fine-tuning, the algorithms that rely on dense space retrieval struggle to achieve satisfactory accuracy and concise information retrieval.

In response to factual inquiries directed from structured databases, we propose a novel information retrieval framework called ChatLR that achieves both precise and refined information retrieval by mapping natural language queries to precise database search commands. As show in Figure 2, specifically, the query interfaces for structured databases include several types: specific API commands, which provide fixed output information based on specific input arguments;



Prompt: You are now a professional data annotation staff, now you want to annotate the data application scenarios are: the output results of the financial and financial exchange field and the query API interface information into the user for these results of the real scenarios of the question. Below I will introduce you to the basic information of the API and the complete data format information, and you generate the corresponding user questions based on this information, i.e., instruction.

API name : Announcement Information, **API ID :** /announcement/announcementlist, **API Synopsis:** This API is used to search: bring together the latest and most complete announcement information of enterprises/securities. Returns the title, date, source, attachments, associated companies / securities and other content. Let you be the first to know the latest trends and letter disclosure content of listed companies and debt issuers.

Input arguments for announcement information include: field name: objs, value type: string, argument description: search term enterprise code/full name of enterprise/security code/keyword...

A complete data format:

```
{
  "instruction": text,
  "output": {
    {
      "Name": API Name,,
      "objs": search keyword, {
        "from": number of records skipped by the query,, { "from": number of records skipped by the query, { "from": number of records skipped by the query, {
        "size": Maximum data returned by the word query, ...
        ... (all input arguments used by this API, specific API arguments are different, such as objs, from, size above)
        "info": ["CompanyName", "ShareholderName"...] , (the name of the field in the question asked by the user that corresponds to the API output, e.g. company name, shareholder name, etc. as described above)
        "apild": ID of the API.
      }
    }
  }
}
```

According to the above mentioned API information, complete all the "output" information to fill in: which "Name" fill in the name of the API, in the "objs range of values" in the random selection of the name of the enterprise, fill in to get a number of different "objs", followed by the rest of the input arguments, I hope that you according to the above introduction to the input arguments of the API, will be in addition to the "objs" of all the arguments other than According to the above introduction of the API input arguments, all the arguments except "objs" will be filled in randomly according to their meanings and types of values to meet the requirements of the argument values...

According to the above information, you are now required to simulate the generation of the above "output" corresponding to the real scene of the user's possible questions "instruction", **the requirements to meet the following conditions:**

- 1、 Only generate complete data information (json format), do not generate other redundant content.
- 2, "info" generated in strict accordance with the user "instruction" in the requirements to generate only the user "instruction" special designated return field content, do not generate redundant information; for the user "instruction" does not have a special designated return field, at this time, "instruction" should belong to all the information return In this case, we always set "info" to ["ALL"].
- 3, according to the previously obtained multiple "objs", "info" accordingly generate "output", the "output" combination of information and then accordingly generate "instruction", the same data in the "objs" must be completely taken from the content of the "instruction"...

A few examples of complete data are given below:

```
{
  "instruction": "Please check the title, date and source of the announcement about China Life Insurance Company Limited Pingdingshan Weidong Branch published from May 1 to May 15, 2023.",
  "output": {
    "Name": "Announcement Information",
    "objs": "China Life Insurance Company Limited Pingdingshan Weidong Sub-Company",
    "bdate": "20230501",
    "edate": "20230515",
    "from": 0,

    "info": ["Title", "Date", "Source"],
    "apild": "/announcement/announcementlist"
  }
}...
```

Generate data:



GPT: {"instruction": "Please inquire about the title, date and source of the announcement regarding the Elmwood Sunrise Road Securities Office issued between May 1 and May 15, 2023.", "output": { "Name": "Announcement Information", "objs": "Elmwood Sunrise Road Securities Office ", "bdate": "20230501", "edate": "20230515", "from": 0, "info": ["Title", "Date", "Source"], "apild": "/announcement/announcementlist"}}...

Fig. 3. An example of data generation process.

SQL searches, which involve operations such as returning table data and calculations through SQL codes; full-text searches, which utilize query keywords to index the database for word matching, and so forth.

For the aforementioned retrieval commands, we can leverage LLMs to generate specific commands based on text data, such as performing tasks like Text2SQL [15]. With this framework, we delegate information retrieval tasks to LLMs, capitalizing on the extensive semantic understanding capabilities of large language models. This framework can adapt to various types of questions in different scenarios. When the LLM is a black-box model like GPT-4, a common approach is to utilize few-shot learning to stimulate the model’s reasoning abilities [15], though accuracy remains challenging to ensure. In the case of open-source trainable models like LLAMA [38], fine-tuning for specific tasks can be performed to achieve higher accuracy. In the following sections, we will demonstrate the effectiveness of our framework through specific task applications in particular scenarios.

B. Data Generation Paradigm

With the release of LLMs such as ChatGPT [26] and GPT-4 [27], researchers can utilize their APIs for the sake of data generation and augmentation on domain-specific tasks with their powerful language comprehension and generation capabilities. The data used for fine-tuning are generated through the ChatGPT API with prompt constructed by concatenating information from database search interfaces with specific commands. In essence, what we require is for the GPT model to simulate questions that a user might pose in real-world scenarios regarding information in the database. These questions serve as inputs to the model, and the corresponding database search commands are annotated as model outputs.

Our data generation process closely follows the self-instruct approach [39]. An example of our data generation process is shown in Figure 3. Initially, we prepare a seed question file, which guides data generation via GPT-3.5 API. Additionally, we incorporate prompt engineering techniques tailored for ChatGPT. Firstly, we establish the context and role, for instance, instructing the model: “You are currently a data annotation expert in a specific domain.” Secondly, we provide sufficient reference information and offer clear explanations. This includes specific interpretations of database search command arguments, the range of argument values, required data formats, etc. Lastly, following the In-context Learning (ICL) method [40], [41], we introduce several concrete data examples, enabling the model to emulate these examples and generate standard, meaningful, and accurate data. Through iterative improvements in data generation quality, we observed that summarizing the shortcomings of previously generated data and explicitly addressing these in the prompt significantly enhanced the quality of data generated after prompt updates. Ultimately, with the support of prompt engineering, we utilized the GPT-3.5 API to generate a batch of fine-tuned data that played a crucial role in subsequent experiments, aligning well with our requirements. Prior research has highlighted the

significance of sample quality over quantity in fine-tuning LLMs [42]. Consequently, following the completion of data generation, a substantial human effort was invested in meticulously reviewing and modifying the fine-tuning data. This rigorous review aimed to ensure the rationality of the data, including considerations such as the feasibility of generated *instructions* in real-world scenarios and the correspondence between the content of *instructions* and the command contents in the *output*. Through this meticulous process, a substantial volume of high-quality fine-tuning data was obtained.

C. Retrieval System Based on LLM for Structured Databases

To validate the effectiveness of the ChatLR retrieval framework, we integrated a specific structured database from financial domain, undertaking the entire process from data construction to fine-tuning LLM instructions. This comprehensive approach allows us to explore the practical application of the ChatLR retrieval framework for structured information retrieval in a specific financial context. Should there be alterations in the scenario or database, adapting the pertinent content in line with the ChatLR framework allows for the seamless application of the framework to different vertical domains. The database utilized specific API search commands, where users can choose specific API functions (e.g., key domestic indicators API providing functionality to retrieve specific arguments such as operating income). Inputting specific argument values yields precise result tables within the diverse functionalities provided by the API.

Figure 4 shows the practical application of the ChatLR framework as a retrieval system for structured databases. Specifically, ChatLR observes real-world user queries to identify key arguments. The arguments include specific API functionality required to address the query, the filling values for the API input arguments, and specific return values needed from the multitude of results provided by the API. We integrate these arguments into JSON-formatted labels, forming fine-tuning data in conjunction with user queries.

As for the process of converting user instructions into specific API command information, it is imperative to provide the LLM with the comprehensive information of respective API as a prerequisite. However, due to the substantial number of APIs, incorporating the entire set of API information into the LLM prompt tokens at once would significantly surpass the maximum context length of the LLM. To address this challenge, inspired by the methods employed in tasks such as Text2SQL using LLMs like GPT-4 [27], we adopt the strategy of initially mapping the user’s query to a specific table. Subsequently, we furnish the LLM with the information about that table, including attributes like headers, in conjunction with the query, facilitating the completion of Text2SQL reasoning [43].

To refine the fine-tuning task of mapping user queries to API command labels, we decompose it into two distinct sub-tasks within our framework: API-ID recognition and Text2API. Specifically, by first determining the particular API to reference, it becomes unnecessary to incorporate other irrelevant

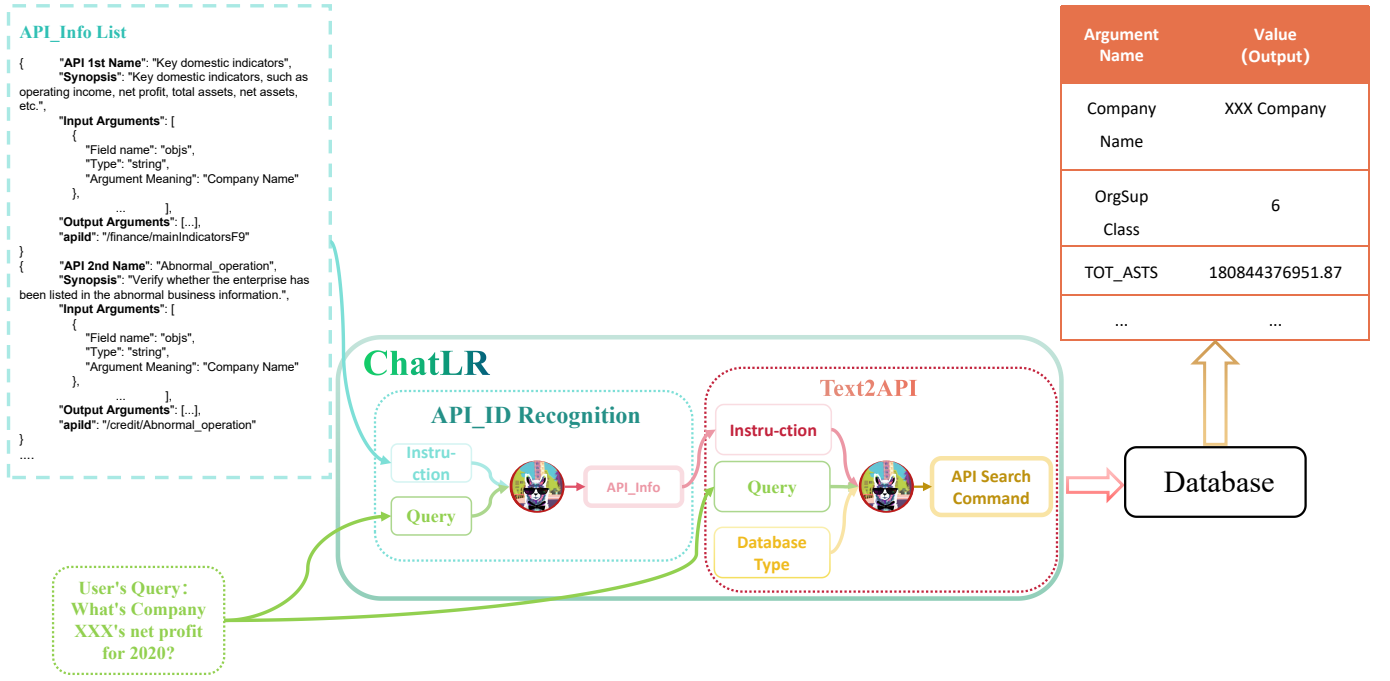


Fig. 4. A Retrieval System for Structured Databases. ChatLR undertakes two distinct tasks. Initially, it engages in API-ID recognition, parsing the user’s query to identify the specific API-ID. Subsequently, leveraging the API-Info list, it retrieves detailed information about the identified API, including input and output argument names. With the acquired API information, ChatLR proceeds to execute the Text2API task, generating the precise API search command statement. Finally, the search command is employed to retrieve accurate results from the database, which are then presented to the user.

API information into the prompt, thereby effectively mitigating the maximum context length constraint. This approach is feasible due to the unique identifier assigned to each API, which serves as a distinguishing factor. Consequently, we leverage the API-ID recognition to precisely position the user’s inquiry onto the specific API-ID capable of flawlessly resolving the search task. This method ensures the systematic and efficient functioning of our framework in handling token length constraints, offering a viable solution to the computational challenges posed by the vast API landscape.

The specific procedure involves concatenating a command prompt template with specific instructions, along with all API-IDs and their corresponding Chinese names, with the user’s query. The concatenated prompt is then fed into ChatLR. Subsequently, ChatLR produces a specific API-ID. Upon completing API-ID recognition, the obtained result is fed into API-info, a database containing fundamental information about all APIs as illustrated in the Figure 5. This information encompasses ID, all required input arguments along with their value types, argument meanings, and output argument details. This information primarily serves to provide sufficient external knowledge for LLM in subsequent prompt engineering.

We extract the information of the identified API, compile it into an API-INFO prompt template, concatenate it with the user’s query, and further supplement it with special command information. This composite input is then reintroduced into ChatLR. This time, due to the distinct command, the model outputs a standardized JSON-formatted API command. This command includes API-ID, API input arguments with cor-

rect values, and info (a list comprising all necessary user-desired output argument names). Finally, the corresponding argument values are input into the identified API retrieval function with the help of API command information file. This process involves querying the external database to retrieve relevant information, extracting output argument information included in info, and reorganizing it to present to users. To summarize, the framework, facilitated by LLM, serves as a bridge, transforming user natural language queries into precise search commands, resulting in accurate and refined financial data.

IV. EXPERIMENT

In this section, we introduce our experiments in detail, including datasets in subsection IV-A, models in subsection IV-B, implementation details in subsection IV-C, and experimental results in subsection IV-D.

A. Datasets

The structured database used in our experiments is an exhaustive repository encompassing financial, legal, and associated data facets. This comprehensive resource restores essential information pertaining to major companies and organizations, including financial indicators, financing details, judicial case specifics, etc. The dataset is securely stored in an SQL database maintained by FinChina Company. Nevertheless, owing to the substantial volume and extensive breadth of the data, the efficiency of SQL query statements may occasionally fall short of practical application demands. In response to

/credit/Abnormal_operation

Verify whether the enterprise has been listed in the abnormal business information.

/finance/mainIndicatorsF9

Key domestic indicators, such as operating income, net profit, total assets, net assets, etc.

/credit/tax_violations

Check whether the enterprise has tax violations.

/financing/Trust_financing

Obtain data on trust financing in which the enterprise is involved as a financier.

/financing/PEVC

Access to corporate venture capital information and funding history data

Argument Name	Value (Input)
objs	XXX Company
from	0
size	10
...	...

DataBase



Argument Name	Value (Output)
CompanyName	XXX Company
OrgSupClass	6
AuditYear	2021
ReportDateType	Annual Report
TOT_ASTS	180844376951.87
YOY_TOT_ASTS	-6.64
TOT_LIABS	105957304667.32
YOY_TOT_CMN_EQTY	2.93
...	...

Fig. 5. Structured database API search framework. Users select the appropriate API-ID based on the functional descriptions of various APIs, inputting all required argument values for the chosen API. Subsequently, a database query is performed to obtain comprehensive tabular data.

this challenge, a strategic solution has been implemented by integrating 60 API query interfaces tailored to the specific type and scope of the database information. This integration allows users to rapidly and precisely retrieve required information within the expansive database through the API interface.

As show in Figure 5, during the information retrieval process, each API mandates particular argument values (e.g., an organization’s name). Depending on the API’s functionality and the input argument values, it returns a list of output arguments with accurate information. Our objective is to leverage LLMs for the retrieval of user queries related to pertinent knowledge from the database. This endeavor seeks to deliver results that are not only accurate but also refined.

According to the data generation method mentioned in subsection III-B, we generated approximately 30,000 pieces of raw data for 60 APIs. After two rounds of filtration, including program-based filtering and manual review and modification (conducted by FinChina Company), a total of 11,000 pieces of raw data were obtained. Program-based filtering ensured the correctness of the data in JSON-format and verified whether the data for specific APIs included all their input arguments and whether the output argument names were correct. Subsequent manual review was conducted to validate the correlation between the instructions generated by GPT and the content in the output. In case of discrepancies, necessary modifications and adjustments were made. This meticulous process was implemented to ensure the accuracy and reliability of the generated data.

Upon the foundation of the original data, we curated distinct fine-tuning datasets for the API-ID recognition task and the Text2API task. Our fine-tuning data are structured in the format of $\{instruction, input, output\}$. For the API-

ID recognition task, the instruction is composed of specific instruction content concatenated with the ID and Chinese name information of all APIs. The input corresponds to the user’s query, and the output represents the ID of a specific API. Regarding the Text2API task, the instruction is constructed by combining special instruction content with the overall information of a specific API. The input is the user’s query, and the output is the complete JSON-formatted API search command. The design of the specific instruction content in the instruction for both tasks closely resembles the prompt structure in data generation. We incorporated prompt engineering techniques, including scenario and role setting, detailed information interpretation, and specific examples referencing ICL, into both instruction designs. Subsequent experimental results emphasize the importance of prompt content, even in the context of fine-tuning data, for LLMs.

In accordance with the outlined structure, each of the two tasks yielded approximately 11,000 instances of fine-tuning data. However, since our current data is entirely generated based on GPT, it does not fully consider the variety of queries that users might pose in real-world scenarios. After training the initial model, we subjected it to internal testing at FinChina Company, concurrently collecting data information. Subsequently, we supplemented the fine-tuning data with this additional information to enhance our model’s performance and robustness in handling real-world scenarios.

During internal testing, we identified two significant issues in the API-ID recognition and Text2API tasks. For API-ID recognition, real-world user queries may not always align with solving them using one of the 60 APIs. Hence, we introduced negative samples where the query content is broad or meaningless. In such cases, we guided the model to output

TABLE I
MAIN RESULTS ON TEXT SUMMARY

Model	BLEU-4	Rouge-1	Rouge-2	Rouge-L
ChatGLM-6B	16.5	36.1	19.0	23.5
ChatGLM-6B/FT	65.2	78.5	70.6	74.3
BELLE-13B	15.3	38.2	25.2	20.4
BELLE-13B/FT	62.1	80.3	72.3	73.7
Chinese-Alpaca-13B	16.7	39.6	27.2	22.3
Chinese-Alpaca-13B/FT	66.1	85.8	76.7	78.3

TABLE II
MAIN RESULTS ON API-ID RECOGNITION AND TEXT2API

Setting	Method	Parameter	API-ID recognition	Text2API
Few-shot	GPT3.5-turbo-16k	175B	37.3	15.3
	Chinese-Alpaca33B-pro	33B	24.2	10.2
Zero-shot	ChatGLM2-6B	6B	10.2	3.4
	Chinese-Alpaca33B-pro	33B	15.6	7.2
	ChatGLM2-6B/ChatLR	6B	75.3	74.4
	BELLE13B/ChatLR	13B	78.1	77.3
	Chinese-Alpaca13B-plus/ChatLR (Ours)	13B	89.1	87.3
	Chinese-Alpaca33B-pro//ChatLR (Ours)	33B	99.9	98.9

a negative label, providing users with a prompt indicating that the question lacks practical significance or is too broad, prompting them to rephrase for clarity and specificity. By learning from diverse types of negative sample labels, LLM can recognize complex instructions that are challenging to decompose, refusing to execute them. In such cases, the model prompts the user for more specific intent. For instance, when faced with a vague user query like “Tell me some information,” the LLM may output a prompt seeking more precise details: “Please provide specific details for the information (company name, type of information, etc.) you want to inquire about.”

Furthermore, the financial enterprise names in our original 11,000 training instances were populated from FinChina Company’s comprehensive enterprise name list. However, in reality, users often prefer to inquire about specific enterprises using abbreviations or aliases. To address this, we added around 20,000 instances to the original data, using enterprise abbreviations as search keywords. With this addition, we now have approximate 30,000 original data instances in total, which were further expanded into two subtasks using distinct instructions. The final fine-tuning dataset for Text2API consists of 30,000 instances, and after introducing negative samples, the API-ID recognition fine-tuning dataset comprises 40,000 instances.

B. Models

In order to investigate the label alignment capability of LLMs after fine-tuning on a small yet high-quality dataset, we conducted an experiment in text summarization. Given that both our data corpus and usage context are in Chinese, we compared several state-of-the-art LLMs pre-trained on Chinese. We manually annotated summaries for 10,000 Chinese news articles and fine-tuned ChatGLM [44], [45], BELLE

[46]–[48], and Chinese-Alpaca [16] using this labeled data. The experimental results (refer to the Table I) revealed that, irrespective of the model, all exhibited a high label alignment capability post fine-tuning. Among the three models, Chinese-Alpaca demonstrated the highest Rouge scores [49] on the test set after fine-tuning. Consequently, we selected it as the foundational model for ChatLR. This model, an extension of the LLaMA [38] architecture, incorporates an expanded Chinese vocabulary and undergoes continual pre-training with Chinese corpus, thereby enhancing its Chinese semantic understanding capability. Furthermore, the Chinese-Alpaca model fine-tuned with Chinese instruction data, significantly improving its understanding and execution capabilities in handling instructions [16]. Our model design supports adaptation with other LLMs, providing versatility and flexibility in its implementation.

C. Implementation Details

In order to optimize training time and memory usage, we employed a highly efficient fine-tuning technique called LoRA [50] for model training. Following a multitask fine-tuning approach [51], we simultaneously fine-tuned the model for both API-ID recognition and Text2API tasks on a dataset comprising a total of 70,000 instances. The fine-tuning process was executed on four A800 GPUs for a duration of 60 hours, resulting in the development of our ChatLR model.

D. Results

1) *Metrics*: We use accuracy as evaluation metric for both API-ID recognition and Text2API tasks. For API-ID recognition task, a prediction is considered correct if it exactly matches the ground-truth label ID. Regarding Text2API task, a prediction is considered correct if the input and output

argument information in the predicted result matches the ground-truth API search command information.

2) *Overall Performance*: From Table II, it is evident that with the assistance of the ChatLR framework, different LLMs achieve significantly higher accuracy on the test sets of both tasks compared to their standalone counterparts after fine-tuning, especially under zero-shot setting. Notably, ChatLR even improves the performance of GPT-3.5 by 62.6% in API-ID recognition and nearly 83.6% in Text2API. For Text2API task, since we require the model to output complete JSON-formatted data and accurately infer all input arguments contained in the API along with their correct values in the given query context, it puts a higher demands on both the model’s basic database-related knowledge and financial domain-specific knowledge. In light of this, our few-shot setting includes more detailed instruction explanations and incorporates multiple complete data output examples to compensate for the knowledge gap in financial domain expertise.

Due to the introduction of abbreviated and negative samples during fine-tuning, ChatLR can recognize more colloquial instructions, aligning better with the requirements for real-world applications. For instances where the user’s intent is overly vague, ChatLR refrains from generating instructions and prompts the user to provide more detailed intent clarification. It exhibits the ability to consider the rationality of instructions akin to human experts, rather than rigidly transforming instructions. Furthermore, ChatLR proves to be efficient, as it only requires fine-tuning on a small-scaled pre-trained LM, ChatGLM2-6B, to achieve results that significantly surpass the 175B GPT-3.5. With high accuracy on both tasks, ChatLR allows for the accurate and rapid retrieval of target structured data based on API search commands.

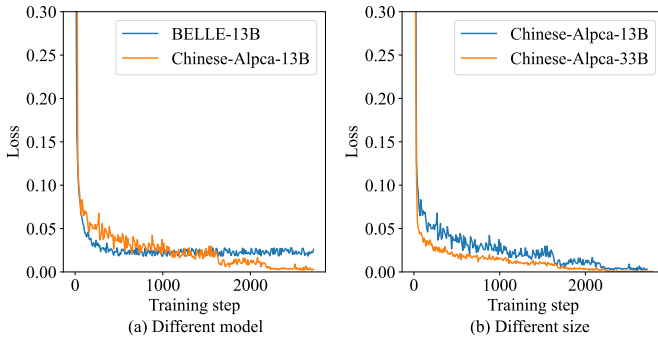


Fig. 6. Training processes of ChatLR.

TABLE III
ACCURACIES OF VARIOUS METHODS FOR INFORMATION RETRIEVAL FROM
STRUCTURED DATABASES

Methods	Accuracy
BM25	74.4
M3e-large	79.7
Bge-large	82.3
ChatLR (Ours)	98.8

TABLE IV
LLM-PREFERRED INSTRUCTION ANALYSIS.

Task	LLM-preferred Instruction
Instruction Fine-tuning	No characterization required Simple and clear instructions ICL not helpful
Data Generation	Requires character setup Detailed cleaning instructions ICL helps a lot

Figure 6 (a) illustrates that under the condition of equal model sizes, Chinese-Alpaca is more suitable for training the ChatLR framework. It can be seen that Chinese-Alpaca converges faster to a relatively small loss value than BELLE with continual training process. Figure 6 (b) shows that, larger foundation models lead to faster convergence. This is demonstrated by the fact that Chinese-Alpaca-33B model converges faster to smaller loss values than its 13B counterparts with identical training steps.

3) *Comparative Experiments*: In this experiment, we conducted comparative studies on ChatLR and other baseline information retrieval models, including BM25 [17], M3e-large [18], and Bge-large [19]. BM25 is a static sparse retrieval algorithm which does not need training. Both M3e-large and Bge-large are dense retrieval algorithms that are pre-trained on large unlabeled corpora and fine-tuned for specific tasks. Bge, which is inspired by the strategy of instruction tuning add [47], improves its general capabilities in multitask scenarios.

As demonstrated in Table III, ChatLR exhibits a significantly superior performance compared to other retrieval algorithms, achieving an overall accuracy of 98.8%. This high precision in structured data information retrieval establishes a robust foundation for various knowledge-intensive question-answering domains.

4) *Analysis of LLM-Preferred Instruction*: In the context of both data generation and LLM instruction fine-tuning, similar prompt content is employed. Specifically, in the inference instruction and instruction fine-tuning data, prompts are strategically used to stimulate the learning capabilities of LLM. Through extensive experiments, we note that the impact of identical prompt engineering techniques varies significantly between the inference and fine-tuning processes (see Table IV). We conducted multiple experiments by adjusting character setup, the level of detail in instruction content, and whether to add ICL examples separately during the fine-tuning of instructions, as illustrated in Figure 7. The experiments were performed on three foundation models: BELLE-13B, Chinese-Alpaca-13B, and Chinese-Alpaca-33B. Conclusions regarding the fine-tuning accuracy based on different fine-tuning models and parameter scales for the three instruction settings were drawn from the observations. Primarily, role-setting techniques, universally applied to LLMs, prove inconsequential during fine-tuning. Removing role-setting statements from the instructions in fine-tuning data has a negligible effect on model performance. Fine-tuning instructions necessitate clarity

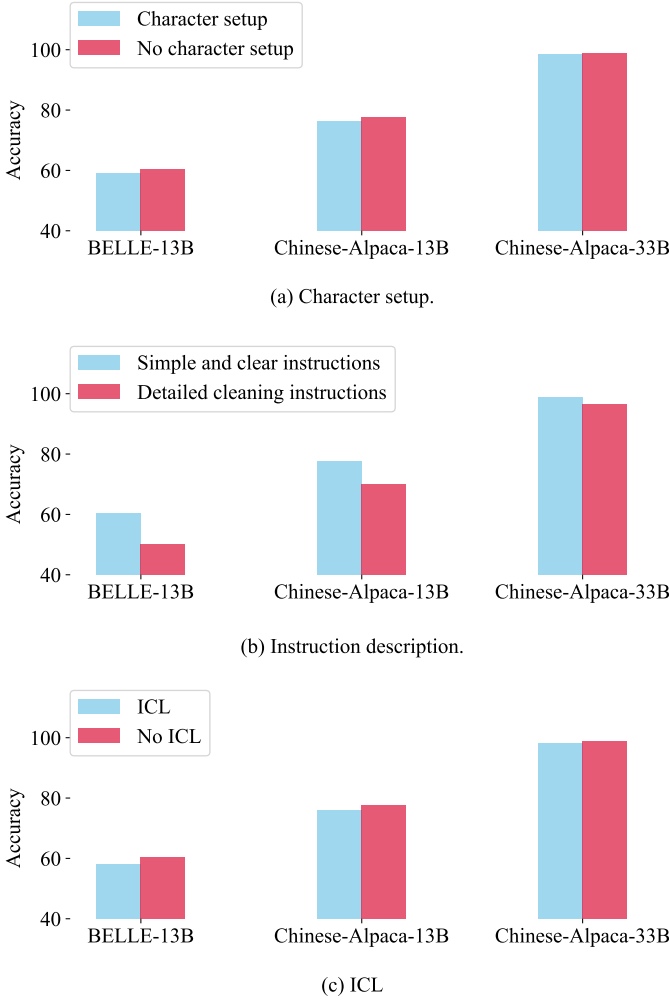


Fig. 7. Accuracies of BELLE-13B, Chinese-Alpaca-13B, and Chinese-Alpaca-33B under three different instruction settings.

and directness, emphasizing simplicity. As shown in Figure 7 (b), compared to the other two aspects (Figure 7 (a) and (c)), this factor has a significant impact on final accuracy. For LLMs, concise instructions imply a reduced token consumption. Conversely, during data generation, comprehensive prompts are required to elaborate on how the model should assist in the task. Additionally, inspired by the principles of In-context Learning (ICL) [40], we leveraged previously accumulated high-quality data to create specific data reference examples, significantly enhancing the quality of our generated data. In contrast, the inclusion of examples in fine-tuning data instructions did not yield a notable improvement in the model's training and inference processes. Additionally, as the model scale increases, the impact of adjustments to instruction settings on the results diminishes. In other words, the larger the model scale is, the higher the robustness of this framework can achieve. For reference, we expose the specific instructions used for fine-tuning ChatLR in the Figure 8, and the instructions for data generation can be seen in the Figure 3.

API-ID recognition

```
{ "instruction": "Please convert the user's real scenario question to the corresponding API_ID that needs to be queried, if only based on the question can not determine which API can return the user's required information, then do not generate a specific API_ID, the output prompts the user to enter more detailed information can be prompted. All desirable API_IDs and their Chinese names are as follows: ..."
```

```
"input": "Please inquire about the title, date and source of the announcement regarding the Elmwood Sunrise Road Securities Office issued between May 1 and May 15, 2023." ,
"output": "/announcement/announcementlist" }
```

Text2API

```
{ "instruction": "Please convert the user's real-life questions into query API interface information, and output the user's natural language questions into corresponding output information, the generated output must meet the json format, where the output information includes: all the input parameter values of the API obtained according to the user's real-life questions, and the return parameter information obtained according to the user's real-life questions. Parameter information info, if the user does not specify which fields to return, or require the return of universal information, all information, then \"info\": [\"ALL\"]. API details as follows :/announcement/announcementlist... [generating by API-ID recognition]"
```

```
"input": "Please inquire about the title, date and source of the announcement regarding the Elmwood Sunrise Road Securities Office issued between May 1 and May 15, 2023." ,
"output": { "objs": "Elmwood Sunrise Road Securities Office" . "bdate": "20230501" , "edate": "20230515" , "from": 0 , "info": [ "Title" , "Date" , "Source" ] , "apid": "/announcement/announcementlist" } }
```

Fig. 8. Two instructions for fine-tuning ChatLR.

Computational Instruction Design

```
{ "input": "Help me calculate the sum of XXX company's profits for the first two quarters of 2022." ,
"output": {
"retrieve command 1 (1st quarter)",
"retrieve command 2 (2nd quarter)",
"calculation command (sum)"
} }
```

Fig. 9. An Example of Computational Instruction Design.

V. CONCLUSION AND FUTURE WORK

This work redefines the information retrieval framework for structured databases by incorporating LLMs and suggests fine-tuning the LLM to enhance retrieval accuracy. LLMs play a more indispensable role within the ChatLR framework, instead of a pure generator in previous RAG paradigm. Leveraging its formidable semantic comprehension capabilities, LLM can construct the mapping between queries and search commands. Consequently, it can generate precise answer utilizing the interaction between factual user queries and structured database. The effectiveness of the proposed framework has demonstrated its effectiveness on a real-world information retrieval task using domain-specific structured databases. The extensibility of this framework can be verified by moderate modification to accommodate to various scenarios and databases. Such modifi-

cations adhere to the prescribed structure, facilitating seamless generalization across diverse vertical domains. This not only substantiates the versatility of our proposed framework, but also underscores its practical utility in real-world industrial applications.

Our research predominantly focuses on optimizing queries for structured databases. However, the user queries often encompass complex intentions and requirements, such as computing maximum budgets and sorting by annual profits, which involve multi-stage mathematical reasoning steps. LLMs encounter challenges in numerical reasoning, leading to computational errors and illusions [52]. As depicted in Figure 8, our proposed framework incorporates a Json-formatted output design, ensuring precise identification of commands across modules. An extended concept arising from this design involves the incorporation of calculation and statistical operation units in the output. Leveraging the natural language understanding ability of LLMs, commands related to computations and sorting are identified and outputted (as illustrated in Figure 9). Consequently, our approach relieves the LLM from direct operational and computational tasks, presenting a series of pre-packaged function commands for LLM invocation. By comprehending data information and user inputs, LLM generates a command sequence for execution by the backend system, thereby producing accurate computational outcomes following structured data queries, sorting, and related operations. It is noteworthy that our framework exclusively addresses the information retrieval module within the RAG framework. This exclusivity stems from the fact that the databases we utilized are structured, which obviates the need for LM to summarize outputs. For unstructured RAG with diverse data, the prevailing practice involves information retrieval through vector searches, culminating in LM summarization of outputs [1], [10]. In this context, LLM has the potential to optimize other RAG modules by means of its semantic understanding ability. For instance, LLM may serve as a bridge from queries to embedding keywords in retrieval from vector space. In essence, addressing complex queries often requires more than simplistic keywords to retrieve precise contextual information from databases. LLM, through query analysis, can deduce comprehensive keyword information necessary for the retriever, resulting in more accurate outcomes when matching against knowledge repositories [53].

Currently, we are endeavoring to realize the two aforementioned concepts. In future work, we aim to broaden the categories of compatible database types. Additionally, we intend to investigate an integrative approach to unify the tasks of querying from structured and unstructured databases. This goal is driven by the objective of optimizing the utilization of external database knowledge in the realm of factual knowledge-based questioning.

REFERENCES

- [1] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 9802–9822.
- [2] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard *et al.*, "KILT: a benchmark for knowledge intensive language tasks," *arXiv preprint arXiv:2009.02252*, 2020.
- [3] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.
- [4] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3929–3938.
- [5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-Augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.
- [7] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," *arXiv preprint arXiv:2109.07958*, 2021.
- [8] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," *arXiv preprint arXiv:2005.00661*, 2020.
- [9] M. Huang, X. Zhu, and J. Gao, "Challenges in building intelligent open-domain dialog systems," *ACM Transactions on Information Systems (TOIS)*, vol. 38, no. 3, pp. 1–32, 2020.
- [10] Z. Yu, C. Xiong, S. Yu, and Z. Liu, "Augmentation-Adapted Retriever Improves Generalization of Language Models as Generic Plug-In," *arXiv preprint arXiv:2305.17331*, 2023.
- [11] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Few-Shot learning with retrieval augmented language models," *arXiv preprint arXiv:2208.03299*, 2022.
- [12] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Unsupervised Dense Information Retrieval with Contrastive Learning," *Transactions on Machine Learning Research*, 2022.
- [13] J. D. M.-W. C. Kenton and L. K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.
- [14] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W.-t. Yih, "Replug: Retrieval-augmented black-box language models," *arXiv preprint arXiv:2301.12652*, 2023.
- [15] S. Ma, W. Jiang, X. Ao, M. Tian, X. Feng, Y. Lyu, Q. She, and Q. He, "Semantic-Driven Instance Generation for Table Question Answering," in *International Conference on Database Systems for Advanced Applications*. Springer, 2023, pp. 3–18.
- [16] Y. Cui, Z. Yang, and X. Yao, "Efficient and Effective Text Encoding for Chinese LLaMA and Alpaca," *arXiv preprint arXiv:2304.08177*, 2023. [Online]. Available: <https://arxiv.org/abs/2304.08177>
- [17] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.
- [18] H. s. Wang Yuxin, Sun Qingxuan, "M3E: Moka Massive Mixed Embedding Model," 2023.
- [19] S. Xiao, Z. Liu, P. Zhang, and N. Muennighof, "C-Pack: Packaged resources to advance general chinese embedding," *arXiv preprint arXiv:2309.07597*, 2023.
- [20] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," *arXiv preprint arXiv:1906.00300*, 2019.
- [21] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.
- [22] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark *et al.*, "Improving language models by retrieving from trillions of tokens," *arXiv preprint arXiv:2112.04426*, 2021.
- [23] G. Izacard and E. Grave, "Distilling knowledge from reader to retriever for question answering," *arXiv preprint arXiv:2012.04584*, 2020.
- [24] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlga, A. Shashua, K. Leyton-Brown, and Y. Shoham, "In-context retrieval-augmented language models," *arXiv preprint arXiv:2302.00083*, 2023.

- [25] A. Maronikolakis and H. Schütze, “Multidomain pretrained language models for green NLP,” in *Proceedings of the Second Workshop on Domain Adaptation for NLP*, 2021, pp. 1–8.
- [26] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, “Training language models to follow instructions with human feedback,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.
- [27] R. OpenAI, “GPT-4 technical report,” *arXiv*, pp. 2303–08 774, 2023.
- [28] H. Sun, H. Ma, X. He, W.-t. Yih, Y. Su, and X. Yan, “Table cell search for question answering,” in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 771–782.
- [29] P. Pasupat and P. Liang, “Compositional semantic parsing on semi-structured tables,” *arXiv preprint arXiv:1508.00305*, 2015.
- [30] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, “Semantic parsing via staged query graph generation: Question answering with knowledge base,” in *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*, 2015.
- [31] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang, “Tabfact: A large-scale dataset for table-based fact verification,” *arXiv preprint arXiv:1909.02164*, 2019.
- [32] W. Chen, M.-W. Chang, E. Schlinger, W. Wang, and W. W. Cohen, “Open question answering over tables and text,” *arXiv preprint arXiv:2010.10439*, 2020.
- [33] W. Hwang, J. Yim, S. Park, and M. Seo, “A comprehensive exploration on wikisql with table-aware word contextualization,” *arXiv preprint arXiv:1902.01069*, 2019.
- [34] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang, “K-Bert: Enabling language representation with knowledge graph,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, 2020, pp. 2901–2908.
- [35] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, “A simple language model for task-oriented dialogue,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 179–20 191, 2020.
- [36] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J.-G. Lou, “TAPEX: Table pre-training via learning a neural SQL executor,” *arXiv preprint arXiv:2107.07653*, 2021.
- [37] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang *et al.*, “Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models,” *arXiv preprint arXiv:2201.05966*, 2022.
- [38] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [39] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, “Self-Instruct: Aligning language model with self generated instructions,” *arXiv preprint arXiv:2212.10560*, 2022.
- [40] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
- [41] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui, “A survey for in-context learning,” *arXiv preprint arXiv:2301.00234*, 2022.
- [42] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu *et al.*, “Lima: Less is more for alignment,” *arXiv preprint arXiv:2305.11206*, 2023.
- [43] M. Pourreza and D. Rafiei, “Din-Sql: Decomposed in-context learning of text-to-sql with self-correction,” *arXiv preprint arXiv:2304.11015*, 2023.
- [44] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia *et al.*, “Glm-130b: An open bilingual pre-trained model,” *arXiv preprint arXiv:2210.02414*, 2022.
- [45] Z. Du, Y. Qian, X. Liu, M. Ding, J. Qiu, Z. Yang, and J. Tang, “GLM: General Language Model Pretraining with Autoregressive Blank Infilling,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 320–335.
- [46] BELLEGroup, “BELLE: Be Everyone’s Large Language model Engine,” <https://github.com/LianjiaTech/BELLE>, 2023.
- [47] Y. Ji, Y. Deng, Y. Gong, Y. Peng, Q. Niu, L. Zhang, B. Ma, and X. Li, “Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases,” *arXiv preprint arXiv:2303.14742*, 2023.
- [48] C. Wen, X. Sun, S. Zhao, X. Fang, L. Chen, and W. Zou, “ChatHome: Development and Evaluation of a Domain-Specific Language Model for Home Renovation,” *arXiv preprint arXiv:2307.15290*, 2023.
- [49] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*, 2004, pp. 74–81.
- [50] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [51] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, “Scaling instruction-finetuned language models,” *arXiv preprint arXiv:2210.11416*, 2022.
- [52] S. Imani, L. Du, and H. Shrivastava, “Mathprompter: Mathematical reasoning using large language models,” *arXiv preprint arXiv:2303.05398*, 2023.
- [53] L. Gao, X. Ma, J. Lin, and J. Callan, “Precise zero-shot dense retrieval without relevance labels,” *arXiv preprint arXiv:2212.10496*, 2022.