LLM for Data Management

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

Xuanhe Zhou
Tsinghua University
zhouxuan19@mails.tsinghua.edu.cn

Xinyang Zhao
Tsinghua University
xy-zhao20@mails.tsinghua.edu.cn

ABSTRACT

Machine learning techniques have been verified to be effective in optimizing data management systems and are widely researched in recent years. However, traditional small-sized ML models often struggle to generalize to new scenarios, and have limited context understanding ability (e.g., inputting discrete features only). The emergence of LLMs offers a promising solution to these challenges. LLMs have been trained over a vast number of scenarios and tasks and acquire human-competitive capabilities like context understanding and summarization, which can be highly beneficial for data management tasks (e.g., natural language based data analytics). In this tutorial, we present how to utilize LLMs to optimize data management systems and review new techniques for addressing these technical challenges, including hallucination of LLMs, high cost of interacting with LLMs, and low accuracy for processing complicated tasks. First, we discuss retrieval augmented generation (RAG) techniques to address the hallucination problem. Second, we present vector database techniques to improve the latency. Third, we present LLM agent techniques for processing complicated tasks by generating multi-round pipelines. We also showcase some real-world data management scenarios that can be well optimized by LLMs, including query rewrite, database diagnosis and data analytics. Finally, we summarize some open research challenges.

PVLDB Reference Format:
Guoliang Li, Xuanhe Zhou, and Xinyang Zhao. LLM for Data Management. PVLDB, 17(12): 4213 - 4216, 2024.
doi:10.14778/3685800.3685838

PVLDB Artifact Availability:
The artifacts are available at https://github.com/code4DB/LLM4DB.

1 INTRODUCTION

Machine learning algorithms have shown promising performance in many data management tasks, such as data processing [23], database optimization [30] [17], data analytics [3]. However, traditional machine learning algorithms are hard to adapt to changing environments (e.g., systems, query workloads, and hardware), making them unable to resolve the generalizability and inference problems in data management tasks. In addition, traditional machine learning algorithms cannot meet the requirements for context understanding and multi-step reasoning required in some optimization scenarios such as database diagnosis and root-cause analysis. Fortunately, the excellent capabilities for language understanding and generalization of large language models enable them to effectively overcome these limitations [5, 6]. For example, LLMs can analyze abnormal database metrics and report the root causes and potential solutions to DBAs. LLMs can also serve as natural language (NL) interface for data analytic tasks, such as translating NL request into executable query over their database. In this tutorial, we will present existing LLM techniques for data management [14, 31, 32].

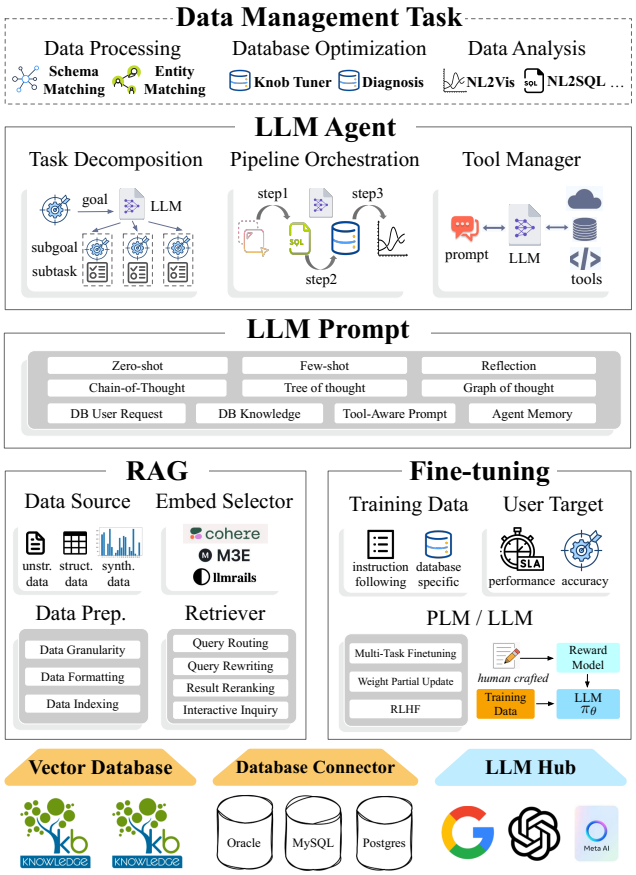

Figure 1: An LLM for Data Management Architecture

Tutorial Overview. We plan to provide a 1.5-hour tutorial to thoroughly review existing LLM techniques for data management.

Background of LLM for Data Management (10 mins). There are three limitations of directly using LLMs for data management: hallucination, high cost, and low accuracy for complicated tasks. There are several methods have been applied to overcome these challenges, such as chains of thought [22, 27, 28] and tool-calling functionalities [21]. Although these works are impressive, they still reveal several limitations. First, they heavily rely on LLMs and the knowledge contained within them to support almost every task (e.g., designing, coding, and testing in LLM-based software development [10, 19]), which may lead to instability and a high error

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685838

rate. Second, for complex tasks like tool calling, extensive training data for specific APIs are required to fine-tune the LLM. This approach is vulnerable to API changes and can result in significant cost inefficiencies [20]. Third, LLM agents still lack the capability to fully utilize knowledge extracted from multiple sources, which are vital to mitigate hallucination issues in LLMs.Therefore, there are three main challenges should be considered during the developing of a LLM-enhanced data management system. First, *how to effectively utilize data sources* (e.g., tabular data and domain-specified knowledge) to reduce LLM hallucination problems (e.g., through knowledge-augmented answering). Second, the execution of complex data management tasks may involve multiple operations. *How to efficiently manage these operations and optimize pipelines* to enhance both execution effectiveness and efficiency. Third, *how to reduce the LLM overhead?* It is rather expensive to call LLMs for every request. It is important to accurately interpret the intent behind user requests and capture the domain knowledge in order to reduce the iterations with LLMs. There are some techniques that use LLMs to optimize data management. First, retrieval augmented generation (RAG) based methods can be used to address the hallucination problem by LLM fine-tuning and prompt engineering, which embeds domain-specific knowledge to vertical domains. Second, LLM agents are used to provide multiple-round inference and pipeline executions to process complicated tasks. Third, vector databases are used to reduce the high cost of LLMs which provide semantic search and caching abilities.

Retrieval Augmented Generation (20 mins). In this part, we introduce the RAG techniques. To capture vertical domain data and knowledge preparation, RAG is used to provide the knowledge and APIs required to each data management system, which will be used to generate prompts to augment the query. The data and knowledge preparation first collects relevant knowledge and APIs. Then for the knowledge (and text explanations for APIs), RAG splits the knowledge into text chunks based on their semantics. Then for each text chunk, RAG selects an embedding model, generates an embedding, and inserts the embedding into vector databases. Later, for online query processing, RAG generates embedding for the query, uses the vector databases to search relevant knowledge and APIs in order to generate effective prompts and input the prompt to the LLMs in order to improve the inference quality.

LLM Prompt and LLM Agent (20 mins). Given an online query, an embedding model is used to transform queries into corresponding vectors. Furthermore, LLM analyzes the query intent and decides whether to use a single-round processing or a multiple-round processing based on the task difficulty. If it can be answered in a single round, LLM prompts are generated by searching the domain knowledge and APIs using the vector databases and inputting these prompts to LLMs. If the query has to be answered by multiple rounds, LLM agents are used to generate a multiple-round pipeline. Moreover, to reduce the overhead of frequently interacting with LLMs, a cache layer is used to improve the performance.

Domain-Specific LLM Fine-tuning (10 mins). General LLMs may lack the domain knowledge (e.g., complex query optimization skills) required to understand user intent or retrieve data accurately. Therefore, we need to fine-tune either the general LLM or an LLM designed based on task requirements, using user-provided data sources. The fine-tuned LLM can provide more accurate answers for specific domain questions (e.g., offering query rewriting advice).

Vector Database (10 mins). To facilitate the domain knowledge retrieval, the vector databases are used to accelerate the efficiency. Given a query embedding, the vector database can efficiently find the most similar data embeddings based on embedding similarity functions. Moreover, the vector databases also need to support both predicate filter and vector search to improve the recall. We discuss effective vector indexes and search algorithms.

LLM for Data Management Applications (10 mins). We introduce three typical LLM based data management applications, including database optimization (e.g., system diagnosis), data processing (e.g., data standardization), and data analysis (e.g., nl2vis).

Open Challenges (10 mins). We present the open research challenges in LLM for data management.

Target Audience. The intended audience include all VLDB attendees from research and industry communities that are interested in data management, machine learning and LLM. Our tutorial will be self-contained and not require any prior background knowledge.

Difference with Existing Audience. There are tutorials on machine learning and databases [4, 13, 25, 29, 33]. However, there is no tutorial on LLM for data management. Thus different from them, we focus on fundamental techniques of LLM for data management.

2 TUTORIAL OUTLINE

2.1 LLM for Data Management Overview

As shown in Figure 1, there are five important components in an LLM-based data management system, including *RAG, domain-specific fine-tuning, LLM prompt management, LLM agent, and vector databases*. Specifically, *RAG* incorporates domain knowledge like concept explanation and tool API description, retrieves relevant knowledge based on the context, and improves the accuracy of LLM outputs with the retrieved knowledge (e.g., as the input prompt). *Vector databases* can be utilized to speed up the knowledge retrieval procedure in RAG, especially when there are numerous textual knowledge chunks. *Domain-specific fine-tuning* utilizes user-supplied data sources to update the pre-trained LLM, such that achieving better performance in one or multiple specific data management tasks. *LLM prompt management* maintains a set of carefully crafted prompts, which are fed into the LLM to better understand the user intent and enhance the result quality. Furthermore, for complex tasks, an *LLM agent* (e.g., a fine-tuned LLM equipped with tools, memory, and prompts) decomposes a user request into multiple sub-tasks (e.g., split/transform/validation operations for data standardization) and generates an optimized pipeline to effectively execute these sub-tasks.

General Workflow. Given a data management task, we prepare the LLM agent to collect RAG knowledge into vector database, tool suite, and general or fine-tuned LLMs (according to the task difficulty and available domain data). Next, for a user request, the agent utilizes LLM to (1) understand user intent, (2) synthesize an execution pipeline (where some operations may call RAG or tool APIs), and (3) carry out, evaluate, refine the pipeline until generating the final output (e.g., well-verified result or "fail to answer").

RAG. This module uses RAG to enhance the query with deterministic external knowledge [8]. Given a user request, RAG retrieves the top-k relevant knowledge chunks and augments LLM with the information of the retrieved chunks. It uses vector databases to speed up the process and then generates prompts based on the information. In scenarios such as translating natural language (NL) to SQL, traditional LLMs might not capture the specific local data schema and data values. RAG addresses this by searching for and retrieving the relevant schemas and values, which are then used to generate prompts for the LLM. Additionally, RAG identifies the query intent using LLMs. Based on the detected intent, RAG searches for relevant APIs, such as pandas API for data analytics, and instructs the LLM to call the appropriate APIs to handle the query request. We thoroughly summarize existing RAG techniques.

Vector Databases. This module aims to improve the execution effectiveness and efficiency from multiple aspects. When a knowledge query is issued by LLM worker, the vector database (i) integrates context and intent analysis to enrich the query; and (ii) utilizes advanced similarity search algorithms (e.g., graph-network-based embedding for relational knowledge patterns) to ensure high relevance in search results by understanding the semantic difference of queries. It can cache hot user requests and their answers, and when a similar request is posted, vector database directly answers the request, without involving LLMs.

LLM Fine-tuning. This module prepares LLMs by updating some LLM parameters with the domain data. There are multiple typical techniques in this module, including multi-task fine-tuning, partial update of the LLM parameter, and reinforcement learning from human feedback (RLHF). First, multi-task fine-tuning trains an LLM on several related tasks simultaneously, utilizing shared knowledge to boost performance, improve generalization, and mitigate data scarcity issues [15]. Second, partial update of the LLM parameter utilizes techniques like low-rank adapter (LoRA [11]) to efficiently fine-tune the model by adjusting only a small subset of parameters, reducing computational costs and preventing overfitting. That is extremely important for lightweight update in online database scenarios. Third, RLHF aims to refine the model's responses based on human feedback, improving the relevance and accuracy of outputs [18]. For instance, in an NL2SQL application, RLHF can help the model better understand and respond to user queries by learning from human feedback (e.g., correct errors, and highlight bottlenecks).

LLM Prompt. This module interprets the user's request into instructions that LLMs can easily follow. It first extracts the request intent, generates the prompt by inserting the query intent into the prepared prompt template, and inputs the prompt to LLM to handle the request. In addition to basic techniques like *Tokenization*, the input prompt can be split into meaningful segments. These segments are converted into functional operations (using tool or model APIs) and data access operations (using knowledge or data sources). These operations form a basic execution pipeline that the LLM can follow to execute. Moreover, there are various prompt techniques for accurate LLM inference. Zero-shot and few-shot learning approaches help the model understand the task and respond to queries with minimal examples. Reflection techniques allow the model to review and refine its responses, ensuring higher accuracy. Chain/Tree/Graph of thought strategies enable the model to break down complex queries into logical steps, improving the clarity and coherence of the responses. Additionally, memory mechanisms are employed to retain context from previous interactions, ensuring consistency and continuity in the responses.

LLM Agent. This module is responsible for solving complex tasks that the origin LLMs cannot handle in one-step processing. There are three main relevant techniques, including *task decomposition*, *pipeline orchestration*, and *tool management*. Task Decomposition is responsible for decompose a complex problem into several subtasks, which could increase the ability of understanding and reduce the risk of hallucination. Pipeline Orchestration designs and optimizes the execution pipelines. To avoid generating sub-optimal or infeasible execution pipelines, pipeline orchestration could generate several plans and select the optimal one. Tool management involves selecting and integrating the appropriate tools or models required for each operator in the pipeline. In addition to generating pipelines, Pipeline Orchestration could also optimize the execution pipeline with the help of external information provided by tool management.

Data & Model Management. This module owns a variety of data sources to (i) align the system's internal understanding with external knowledge and (ii) mitigate over-reliance on LLMs, which serves as the intermediary between LLM agent and vertical domain knowledge base, databases, tools and AI models.

2.2 LLM for Data Management Tasks

In configuration tuning, components such as knob tuning require adjustments to a wide range of system settings. However, traditional methods often experience unstable performance and exhibit limited generalizability, even when employing transfer techniques. Thus, we present how to utilize LLMs to extract tuning experience or directly recommend setting values [12, 24].

In query optimization, query rewrite [9] aims to transfer a SQL query to an equivalent but more efficient SQL query. For example, works like [2, 26], aim to identify new rewrite rules from SQL pairs [26] or to develop new domain-specific languages (DSLs) to make it easier to implement these rules [2]. However, these methods are limited because they can only handle simple rules (e.g., with up to three operators); but they don't make use of valuable rewrite experiences in textual documents and DBA experiences. Besides, works like [30] judiciously adjust the order or apply rules to enhance the overall rewrite performance. We present how to utilize LLM for query rewrite, including rewrite rule generation and rewrite order exploration [16].

Database system diagnosis aims to identify the root cause based on various query/database/system statistics and provide solutions which could fix these root causes. Traditional machine learning methods define this problem as a classification task, which require a large number of high-quality training data and lack the ability to generate human-like analysis reports. In this tutorial, we introduce how to automate the development of LLM-enhanced database system diagnosis [31].

In data processing tasks, we aim to transform raw data into meaningful information by conducting a series of systematic operations, including error detection, data imputation, schema matching, etc.

However, traditional methods generally rely on empirical rules to filter or update data, which requires significant human effort and is typically difficult to generalize across different datasets. Thus, we present how to effectively process data with LLMs [1, 7].

In many data analytical scenarios, users do not have the ability to write SQL queries or data analyzing code. Users prefer to use natural language (NL) to interact with the data directly. Therefore, translating users' natural language queries into SQL queries or code can greatly lower the threshold for data analysis. In this tutorial, we present how to accurately conduct data analysis with LLMs.

2.3 Open Research Challenges

There are still some open research challenges in this paradigm. We will summarize some important open problems in LLM for data management. (1) ***Database Domain LLM.*** How to train a database domain LLM that supports all the typical data management tasks, rather than addressing them on a case-by-case basis. (2) ***Standardizing Model Interfaces.*** How to reduce the development costs of LLM4DB applications. This includes creating standardized APIs and interfaces that can be easily integrated into existing database systems. (3) ***Model Lightweighting.*** How to deploy distilled large models into database kernels. For instance, developing methods to compress and optimize LLMs so they can be efficiently run within the limited computational resources of database environments without sacrificing performance. (4) ***Generalization capability Improvement.*** How to enhance the representativeness of training data to further lower the usage threshold of LLM4DB. This could involve curating diverse and comprehensive datasets that cover a wide range of database scenarios.

3 BIOGRAPHY

Guoliang Li is a full professor in the Department of Computer Science, Tsinghua University. His research interests mainly include data cleaning and integration and machine learning for databases. He got VLDB 2017 early research contribution award, TCDE 2014 early career award, VLDB 2023 Industry Best paper runner-up, Best of SIGMOD 2023, SIGMOD 2023 research highlight award, DASFAA 2023 Best Paper Award, CIKM 2017 best paper award. He will present the background and open problems.

Xuanhe Zhou gets his PhD degree from the Department of Computer Science, Tsinghua University, advised by Guoliang Li. His research interests lie in intelligent database optimization and data governance for AI. He will present LLM agent and fine-tuning.

Xinyang Zhao is a currently a PhD student in the Department of Computer Science, Tsinghua University. Her research interests lie in the interdisciplinary technologies of database and machine learning. She will present RAG.

ACKNOWLEDGMENTS

This paper was supported by National Key R&D Program of China (2023YFB4503600), NSF of China (61925205, 62232009, 62102215), Zhongguancun Lab, Huawei, TAL education, and Beijing National Research Center for Information Science and Technology (BNRist). Xinyang Zhao is the corresponding author.

REFERENCES

[1] A. Albalak, Y. Elazar, S. M. Xie, and et al. A survey on data selection for language models. *CoRR*, abs/2402.16827, 2024.
[2] Q. Bai, S. Alsudais, and C. Li. Querybooster: Improving SQL performance using middleware services for human-centered query rewriting. *Proc. VLDB Endow.*, 16(11):2911–2924, 2023.
[3] C. Chai, G. Li, J. Fan, and Y. Luo. Crowdchart: Crowdsourced data extraction from visualization charts. *IEEE Trans. Knowl. Data Eng.*, 33(11):3537–3549, 2021.
[4] G. Cong, J. Yang, and Y. Zhao. Machine learning for databases: Foundations, paradigms, and open problems. In *SIGMOD*, pages 622–629. ACM, 2024.
[5] F. K. A. et al. Awesome chatgpt prompts. https://github.com/f/awesome-chatgpt-prompts, 2023.
[6] P. L. et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 2023.
[7] M. Fan, X. Han, J. Fan, C. Chai, N. Tang, G. Li, and X. Du. Cost-effective in-context learning for entity resolution: A design space exploration. *CoRR*, abs/2312.03987, 2023.
[8] Y. Gao, Y. Xiong, X. Gao, and et al. Retrieval-augmented generation for large language models: A survey. *CoRR*, 2023.
[9] G. Graefe. Volcano - an extensible and parallel query evaluation system. *IEEE Trans. Knowl. Data Eng.*, 6(1):120–135, 1994.
[10] S. Hong, X. Zheng, J. Chen, and et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.
[11] 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. In *ICLR*. OpenReview.net, 2022.
[12] X. Huang, H. Li, J. Zhang, and et al. Llmtune: Accelerate database knob tuning with large language models. *CoRR*, abs/2404.11581, 2024.
[13] G. Li, X. Zhou, and L. Cao. Machine learning for databases. *Proc. VLDB Endow.*, 14(12):3190–3193, 2021.
[14] J. Li, B. Hui, G. Qu, B. Li, J. Yang, B. Li, B. Wang, B. Qin, R. Cao, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. Chang, F. Huang, R. Cheng, and Y. Li. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. *CoRR*, abs/2305.03111, 2023.
[15] B. Liu, C. Chen, C. Liao, and et al. Mftcoder: Boosting code llms with multitask fine-tuning. *CoRR*, 2023.
[16] J. Liu and B. Mozafari. Query rewriting via large language models. *CoRR*, abs/2403.09060, 2024.
[17] M. Ma, Z. Yin, S. Zhang, and et al. Diagnosing root causes of intermittent slow queries in large-scale cloud databases. *Proc. VLDB Endow.*, 13(8):1176–1189, 2020.
[18] L. Ouyang, J. Wu, X. Jiang, and et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.
[19] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, and et al. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.
[20] Y. Qin, S. Liang, Y. Ye, and et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789, 2023.
[21] T. Schick, J. Dwivedi-Yu, R. Dessì, and et al. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.
[22] N. Shinn and et al. Reflexion: an autonomous agent with dynamic memory and self-reflection. *CoRR*, 2023.
[23] N. Tang, J. Fan, F. Li, J. Tu, X. Du, G. Li, S. Madden, and M. Ouzzani. Rpt: relational pre-trained transformer is almost all you need towards democratizing data preparation. *arXiv preprint arXiv:2012.02469*, 2020.
[24] I. Trummer. DB-BERT: A database tuning tool that "reads the manual". In *SIGMOD*, pages 190–203, 2022.
[25] W. Wang, M. Zhang, G. Chen, and et al. Database meets deep learning: Challenges and opportunities. *SIGMOD Rec.*, 45(2):17–22, 2016.
[26] Z. Wang, Z. Zhou, Y. Yang, H. Ding, G. Hu, D. Ding, C. Tang, H. Chen, and J. Li. Wetune: Automatic discovery and verification of query rewrite rules. In *SIGMOD*, pages 94–107. ACM, 2022.
[27] 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*, 35:24824–24837, 2022.
[28] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models, 2023.
[29] X. Zhou, C. Chai, G. Li, and J. Sun. Database meets artificial intelligence: A survey. *IEEE TKDE*, 34(3):1096–1116, 2020.
[30] X. Zhou, G. Li, C. Chai, and J. Feng. A learned query rewrite system using monte carlo tree search. *Proc. VLDB Endow.*, 15(1):46–58, 2021.
[31] X. Zhou, G. Li, Z. Sun, and et al. D-bot: Database diagnosis system using large language models. *Proc. VLDB Endow.*, 2024.
[32] X. Zhou, Z. Sun, and G. Li. DB-GPT: large language model meets database. *Data Sci. Eng.*, 9(1):102–111, 2024.
[33] R. Zhu, L. Weng, B. Ding, and J. Zhou. Learned query optimizer: What is new and what is next. In *SIGMOD*, pages 561–569. ACM, 2024.