An Interactive Multi-modal Query Answering System with Retrieval-Augmented Large Language Models

Mengzhao Wang
Zhejiang University
wmzssy@zju.edu.cn

Haotian Wu
Zhejiang University
jxk706060666@gmail.com

Xiangyu Ke
Zhejiang University
xiangyu.ke@zju.edu.cn

Yunjun Gao
Zhejiang University
gaoyj@zju.edu.cn

Xiaoliang Xu
Hangzhou Dianzi University
xxl@hdu.edu.cn

Lu Chen
Zhejiang University
luchen@zju.edu.cn

ABSTRACT

Retrieval-augmented Large Language Models (LLMs) have reshaped traditional query-answering systems, offering unparalleled user experiences. However, existing retrieval techniques often struggle to handle multi-modal query contexts. In this paper, we present an interactive **M**ulti-modal **Q**uery **A**nswering (MQA) system, empowered by our newly developed multi-modal retrieval framework and navigation graph index, integrated with cutting-edge LLMs. It comprises five core components: Data Preprocessing, Vector Representation, Index Construction, Query Execution, and Answer Generation, all orchestrated by a dedicated coordinator to ensure smooth data flow from input to answer generation. One notable aspect of MQA is its utilization of contrastive learning to assess the significance of different modalities, facilitating precise measurement of multi-modal information similarity. Furthermore, the system achieves efficient retrieval through our advanced navigation graph index, refined using computational pruning techniques. Another highlight of our system is its pluggable processing framework, allowing seamless integration of embedding models, graph indexes, and LLMs. This flexibility provides users diverse options for gaining insights from their multi-modal knowledge base. A preliminary video introduction of MQA is available at **https://youtu.be/xvUuo2ZIqWk**.

PVLDB Reference Format:
Mengzhao Wang, Haotian Wu, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, and Lu Chen. An Interactive Multi-modal Query Answering System with Retrieval-Augmented Large Language Models. PVLDB, 17(12): 4333 - 4336, 2024.
doi:10.14778/3685800.3685868

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/ZJU-DAILY/MQA.

1 INTRODUCTION

Query Answering (QA) systems are pivotal in extracting insights from vast knowledge bases, offering intuitive and real-time information retrieval functionality. Traditional QA systems rely on simplistic keyword matching, which lacks semantic comprehension. Recent advancements in Large Language Models (LLMs), such as ChatGPT, have equipped QA systems with sophisticated context-understanding capabilities [6]. Major tech companies, like Microsoft and Baidu, have launched their own QA applications based on LLMs, such as New Bing and ERNIE Bot, respectively. Despite their success, challenges such as hallucinations and outdated knowledge persist in LLM-based QA systems, impacting their performance [3]. The introduction of retrieval-augmented LLMs offers a promising solution by incorporating vector search techniques [7]. It enables QA systems to provide answers using external knowledge sources, thereby promoting factually consistent and reliable responses [4]. However, current retrieval methods, which cater mainly to single-modality situations, struggle in multi-modal QA contexts. Given the growing complexity of user interactions, multi-modal QA has become increasingly important. It allows for a comprehensive understanding and response to queries by considering various input forms, including text, images, etc. To address this, we have recently developed a novel multi-modal retrieval framework, `MUST` [8], specifically designed to facilitate efficient and accurate multi-modal retrieval. In this demonstration proposal, we focus on the multi-modal QA task, leveraging the capabilities of `MUST` alongside LLMs.

Figure 1: An example of multi-modal QA.

Figure 1 illustrates how users can engage in multi-round dialogues with a QA system, incorporating multi-modal information. For instance, a user may initially submit an inquiry for a "long-sleeved top for older women" in text or audio form. Subsequently, based on the images returned in response, the user can choose a preference and suggest alterations, such as adding a "floral pattern" to the selected image. This interaction process continues until the user is satisfied with the outcomes. Multi-modal QA diverges from the conventional text-only QA model by offering more complex and tailored results, significantly enhancing the user experience [2]. Despite the demonstrated proficiency of advanced LLMs in

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685868

understanding multi-modal content (such as GPT-4 [1]), the retrieval techniques currently employed by LLMs are inadequate for capturing multi-modal contexts. This is evident in two prevalent multi-modal retrieval frameworks, Multi-streamed Retrieval (MR) [5] and Joint Embedding (JE) [2]. These frameworks either combine the results of individual vector searches for each modality or carry out a single-channel vector search by jointly encoding all modalities. However, as empirically confirmed in our MUST paper [8], both baselines exhibit limitations in efficiency and accuracy due to their inability to consider the varying importance of fusing information across modalities and the absence of a dedicated indexing and search method for multi-modal data.

To address the aforementioned challenge, we develop an interactive **M**ulti-modal **Q**uery **A**nswering (MQA) system based on our meticulously designed multi-modal retrieval framework and navigation graph index [8], integrated with cutting-edge LLMs [1]. This system offers advanced multi-modal data interaction capabilities via an intuitive interface. The key features are outlined below:

User-friendliness. MQA presents an intuitive interface that accommodates multiple modalities, such as text and images, facilitating customizable searching within a multi-modal knowledge base. A key aspect of the system is its iterative refinement process, empowering users to fine-tune initial results through ongoing dialogue. In image retrieval, MQA transforms descriptive text into visuals, establishing a feedback loop where additional details refine the outcomes, guiding the system towards more relevant outputs, even with vague initial inputs. Moreover, our system integrates advanced LLMs, enhancing the intelligence of the interaction process.

Accuracy. MQA improves query accuracy by utilizing an innovative multi-vector representation technique across multi-modal data. It encodes objects and queries using standalone unimodal encoders or a complex multi-modal encoder, achieving comprehensive vectorized representation. Additionally, the method is refined by our unique, effective vector weight learning model [8], capturing individual modality importance through contrastive learning for better similarity evaluations. This empowers users to meticulously adjust searches, hence aligning results with their distinct preferences.

Flexibility. MQA demonstrates flexibility, embracing a wide range of encoders, weight configurations, index algorithms, and retrieval frameworks. Firstly, it supports seamless encoder integration, such as LSTM, ResNet, and CLIP [8]. Additionally, it allows tailored weight adjustments using intrinsic vector weight learning or user-specific inputs for search refinement. Moreover, it enables index deployment via an intricate navigation graph framework, effortlessly incorporating existing components. Lastly, it accommodates various retrieval frameworks—MR, JE, and MUST—as well as advanced LLMs like GPT-4, fostering a robust ecosystem for multi-modal QA.

Scalability. To meet efficiency requirements in large-scale data retrieval, MQA employs an advanced navigation graph index designed for multi-modal data [8]. This approach assigns multiple vectors per object to a unified index, capturing object similarities. The resulting structure, with vertices representing objects and edges reflecting similarity, narrows the search space, ensuring direct retrieval with minimal traversal. Additionally, MQA optimizes both the index structure and multi-vector computations to enhance scalability over a vast knowledge base.

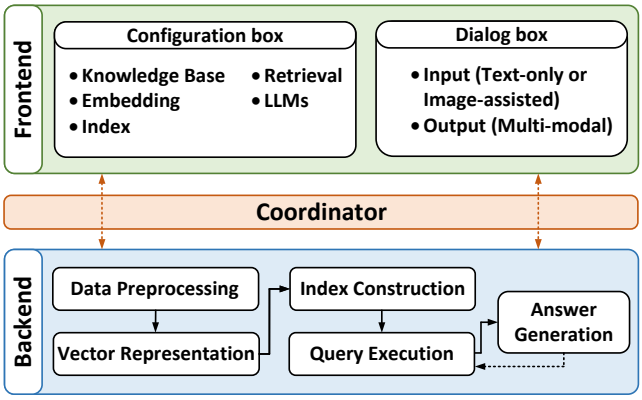


Figure 2: The overall architecture of MQA.

2 SYSTEM OVERVIEW

Figure 2 illustrates the system architecture of MQA, including both the frontend and backend. The frontend offers a user interface for configuring the knowledge bases, embedding models, index algorithms, retrieval parameters, and LLMs. It also offers a dialogue box for users to engage in multi-modal QA interactions. The backend handles data preprocessing, vector representation, index construction, query execution, and answer generation. A coordinator supervises all components' operations and their data transitions.

Data Preprocessing. This component integrates a multi-modal knowledge base into MQA. Data is stored as an object collection with unique IDs for indexing, allowing an expansive data representation. For instance, a movie's film, poster, and synopsis can be stored as a singular object with multiple modalities. Once ingested, data becomes usable within MQA, which enables users to leverage the system's embedding and indexing techniques for more effective data storage and retrieval. Importantly, external knowledge ingestion is optional, and disabling it means MQA relies solely on chosen LLMs for responses. This setting is adjustable in the knowledge base section of the configuration box (the frontend of Figure 2).

Vector Representation. This module converts multi-modal objects into vectorized forms, establishing a standardized mathematical expression [8]. The frontend embedding configuration includes a universal vector support function, endorsing diverse libraries and models, such as OpenAI CLIP. MQA demonstrates remarkable versatility in representing an object or query using high-dimensional vectors derived from a range of encoders. Notably, MQA introduces a vector weight learning model to discern the importances of different modalities for similarity measurement between objects [8]. The learning process adaptively adjusts weights to reflect different modalities' importances in similarity measurement. Ultimately, MQA outputs the learned weights for vector concatenation, accomplishing a comprehensive representation of multi-modal data.

Index Construction. The index construction component builds a unified navigation graph index based on objects' multi-vector representation, utilizing modality weights from the vector representation component. The navigation graph corresponds each vertex to an object and forges connections between object pairs through edges denoting object similarity. This index narrows the search space, directing the query to the target object by probing a subset of the collection. We propose a general pipeline for constructing

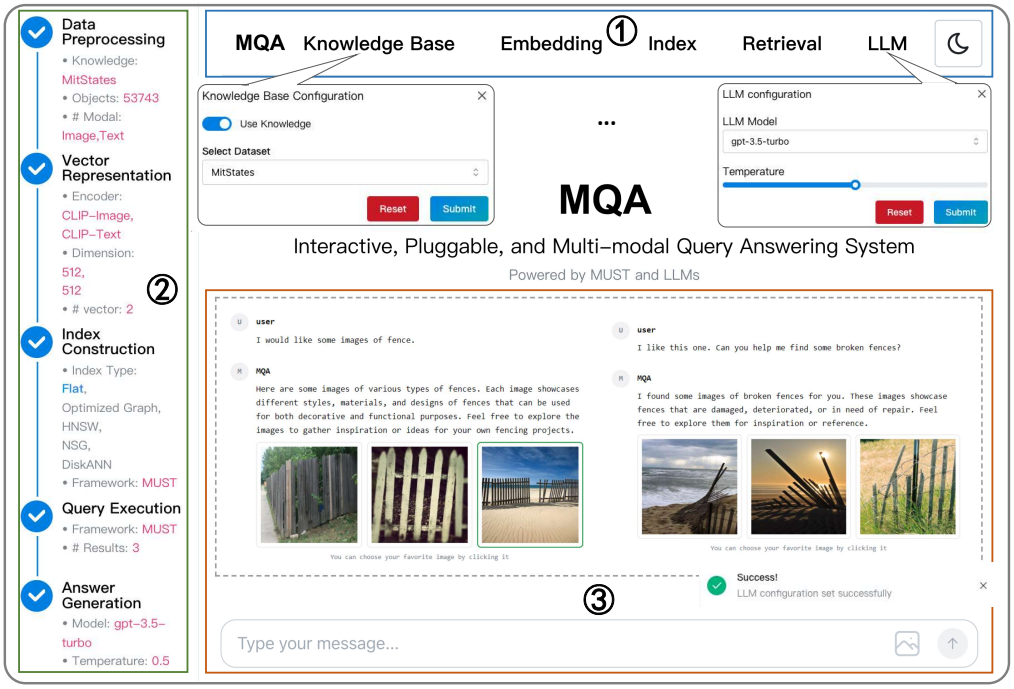
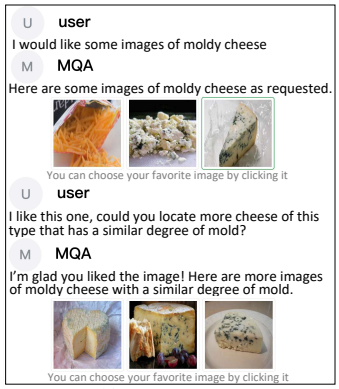
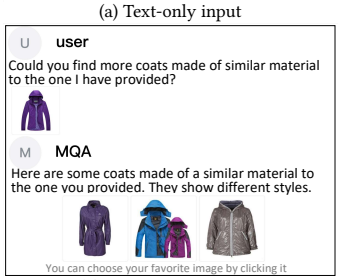


Figure 3: User interface of MQA.



(a) Text-only input



(b) Image-assisted input

Figure 4: Interaction examples.

fine-grained navigation graphs on CGraph[1], a cross-platform Directed Acyclic Graph (DAG) framework. The pipeline consists of five flexible parts, allowing any current navigation graph to be decomposed and smoothly integrated into MQA. Furthermore, we incorporate components from several state-of-the-art algorithms in the context of concatenated vectors, resulting in a novel indexing algorithm. On the configuration box's index item, users can modify existing navigation graphs (e.g., NSG, HNSW, DiskANN, Starling [9]) or initiate custom graphs via the backend API.

Query Execution. The query execution component navigates efficiently to relevant contexts with user queries using multi-modal vector search methods. Upon receiving a query, MQA launches a merging-free search across modalities within the navigation graph. The query is projected into a high-dimensional vector space, where multi-modal objects are located, forming a query point. MQA traverses the graph, starting at a random or fixed vertex, and explores neighboring vertices closer to the query point. This iterative search terminates when no closer vertex is discovered. In this process, distances are calculated via incremental scanning, enhancing efficiency by circumventing unnecessary calculations. Notably, any previous outcome can be chosen to augment the current user query input (as indicated by the dotted arrow in the backend of Figure 2), promoting an intelligent multi-modal search procedure. Users can modify retrieval settings, like result count, framework, and modality weights at the query point, in the frontend's configuration box.

Answer Generation. This component formulates responses from retrieved results and the user query context. Beyond sourcing relevant information from the knowledge base, MQA generates natural, conversational replies, enhancing user interactions. The output includes additional details like preference markers. Users have the flexibility to select from various LLMs in the configuration box. When an LLM is accessible, it works in coordination with the query execution module to handle queries and responses. The user's query is simultaneously dispatched to both the query execution module and the LLM as a prompt. The search results from the query execution module are then redirected to the LLM. The final user response is a summary from the LLM. In the absence of an available LLM, users can still carry out a multi-modal QA procedure through direct engagement with the query execution module.

Coordinator. The coordinator serves as the system's central nexus, supervising all component operations and facilitating smooth data transition across the system. Both the frontend and backend exclusively interact with the coordinator, which functions as a conduit between them, as demonstrated by two-way arrows in Figure 2. This arrangement fosters a streamlined and efficient codebase where API endpoints engage with a single reference point within MQA.

3 DEMONSTRATION

MQA's backend is built on the Flask framework, coupled with a user-centric frontend developed with React, Remix, and Mantine that ensures an intuitive user experience filled with interactive features. The demonstration begins with an introductory tour that displays the working panels of MQA (refer to Figure 3), providing a foundational understanding of user-system interaction. Subsequently, we delve into a hands-on exploration of MQA's features from the user's perspective (see Figure 4). To exhibit our techniques' superiority, we provide comparative results with three baseline methods within MQA for identical query inputs (see Figure 5).

Working Panels. As depicted in Figure 3, the MQA system encompasses three panels for user interaction: ① configuration, ② status monitoring, and ③ query-answering (QA) engagement.

[1]https://github.com/ChunelFeng/CGraph

① Configuration Panel. This interface empowers users to explore the system's features. It facilitates the selection of domain-specific knowledge bases, customizing the search results. Upon a user's selection of a knowledge base, the system initiates data loading, establishing the necessary knowledge base for retrieval. Through embedding options, various encoders can be adjusted for multi-modal data embedding, a critical process for transforming multi-modal data into a vectorized format. The option to activate vector weight learning customizes index construction and retrieval processes with a unique weighting mechanism for the generated vectors. Indexing configurations provide choices on methods and parameters, setting the stage for efficient data retrieval. Additionally, retrieval settings allow users to dictate the retrieval framework and the size of the result set. LLM options present a selection of models and control over output variability via temperature settings. Configuration feedback is relayed through a pop-up box (located in the bottom right corner of Figure 3), ensuring users are informed of the system's setup status.

② Status Monitoring Panel. This panel displays a comprehensive overview of the system's workflow, from data input to output, providing real-time updates. Milestones such as data preprocessing, vector representation, and index construction are visibly tracked with tick marks and relevant details, encompassing encoder details, modal counts, vector dimensions, index types, retrieval frameworks, and LLM specifics. This feature ensures that users can verify and assess their custom settings at a glance.

③ QA Panel. Serving as the interface for query submissions, this module accepts both textual input and image uploads from users to form a multi-modal query. It promptly returns relevant multi-modal information, using an optimized retrieval mechanism guided by LLM to ensure context accuracy. Users can fine-tune these results by providing additional input, leading to tailored content. Additionally, this module includes a demonstration example, providing a practical reference to enhance user interaction.

Interaction Scenarios. MQA enhances user engagement through two primary interaction scenarios, illustrated in Figure 4, with detailed configurations displayed in the corresponding status monitoring panel (Figure 3). (a) Text-only input: In the absence of a reference image, users can initiate a search using a text description, such as "I would like some images of moldy cheese". The system responds by retrieving images that match the description. Users can select a preferred image (by clicking) and refine their request, possibly by adding "I like this one, could you locate more cheese of this type that has a similar degree of mold?". MQA iteratively refines the search based on user feedback until the user is content with the images retrieved. (b) Image-assisted input: When a reference image is available, users upload it and describe their specific requirements, for example, "Could you find more coats made of similar material to the one I have provided?". The MQA system analyzes the visual and textual information, delivering images that align with both the reference image and the textual specifications. Users can further interact by selecting a preferred result for more personalized searches.

Comparative Analysis. In MQA, we compare the outcomes of various retrieval frameworks under identical query conditions. We adjust the index and retrieval techniques by the configuration panel,

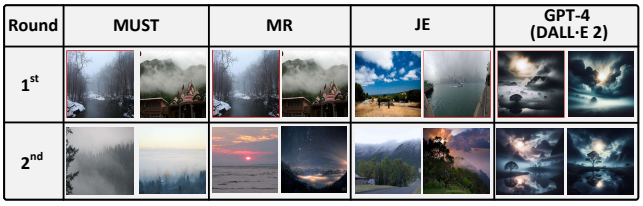


Figure 5: Results from two-round response using different retrieval frameworks, with the user's choice marked in red.

with Figure 5 illustrating the system's two-round response. The user begins with a textual request: "Could you assist me in finding images of foggy clouds?". Subsequently, upon specifying a preference "I like this one, could you provide more similar images of foggy clouds?", MQA returns results based on the selected image and new user feedback. MUST consistently delivers optimal results in both rounds. In contrast, the JE framework underperforms, initially presenting an irrelevant image, followed by two images that do not align with the user's selection. Although MR initially matches MUST's results for text-only input, it fails to maintain alignment with the multi-modal inputs in the subsequent round. GPT-4 (DALL·E 2), lacking multi-modal retrieval configurations, generates synthetic images that miss a touch of realism.

ACKNOWLEDGMENTS

This work was supported in part by the NSFC under Grants No. (62025206, U23A20296, and 62102351), Zhejiang Province's "Lingyan" R&D Project under Grant No. 2024C01259, and Ningbo Yongjiang Talent Introduction Programme (2022A-237-G). Yunjun Gao is the corresponding author of the work.

REFERENCES

[1] 2024. GPT-4 is OpenAI's most advanced system, producing safer and more useful responses. https://openai.com/gpt-4. [Online; accessed 07-April-2024].
[2] Ginger Delmas, Rafael Sampaio de Rezende, Gabriela Csurka, and Diane Larlus. 2022. ARTEMIS: Attention-based Retrieval with Text-Explicit Matching and Implicit Similarity. In *International Conference on Learning Representations (ICLR)*.
[3] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv:2311.05232* (2023).
[4] Zhi Jing, Yongye Su, Yikun Han, Bo Yuan, Chunjiang Liu, Haiyun Xu, and Kehai Chen. 2024. When Large Language Models Meet Vector Databases: A Survey. *arXiv:2402.01763* (2024).
[5] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 2614–2627.
[6] Jiajia Wang, Weizhong Zhao, Xinhui Tu, and Tingting He. 2023. A novel dense retrieval framework for long document retrieval. *Frontiers of Computer Science (FCS)* 17, 4 (2023), 174609.
[7] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science (FCS)* 18, 6 (2024), 186345.
[8] Mengzhao Wang, Xiangyu Ke, Xiaoliang Xu, Lu Chen, Yunjun Gao, Pinpin Huang, and Runkai Zhu. 2024. MUST: An Effective and Scalable Framework for Multimodal Search of Target Modality. In *IEEE International Conference on Data Engineering (ICDE)*.
[9] Mengzhao Wang, Weizhi Xu, Xiaomeng Yi, Songlin Wu, Zhangyang Peng, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, Rentong Guo, and Charles Xie. 2024. Starling: An I/O-Efficient Disk-Resident Graph Index Framework for High-Dimensional Vector Similarity Search on Data Segment. *Proceedings of the ACM on Management of Data (PACMMOD)* 2, 1 (2024), 14:1–14:27.