International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Comparative Evaluation of Vector Embedding Frameworks for Scalable Sematic Retrieval in PDF-Based RAG Systems

Olasehinde Omolayo[a], Odunayo Babatope[b]*

"aGeorgia State University, 33 Gilmer Street SE Atlanta 30303, United States"
"bObafemi Awolowo University, Ile-Ife, Osun State 233129, Nigeria"

A B S T R A C T

We present a comparative analysis of using two different vector embedding frameworks for the implementation of Retrieval-Augmented Generation (RAG) pipelines to enable interactive question and answering result delivery over PDF documents containing structured text and tables. Both solutions adopt the LangChain framework with OpenAI's GPT-4o-mini for summarization and response generation, while leveraging Redis for storing raw documents for multi-turn conversation. We utilize two vector embedding storage strategy: the first pipeline utilises Facebook AI Similarity Search (FAISS), an in-memory vector similarity search engine optimized for local inference and the second employs the PostgreSQL extension, PGVector, for persistent, scalable vector storage. We implement a uniform document ingestion process, multi-vector retrieval mechanism, and a Streamlit-based chat interface across both systems. To evaluate their effectiveness, we propose an assessment framework measuring retrieval quality known as the generation quality (correctness, relevance, and faithfulness) using an LLM-as-a-Judge approach. The experimental results demonstrate that FAISS consistently delivers higher semantic retrieval precision and generation quality, with lower latency compared to PGVector. These findings suggest that FAISS is a more suitable choice for high-performance RAG pipelines, particularly in resource-constrained environments.

Keywords: Retrieval-Augmented Generation (RAG); Vector Embeddings; FAISS; PGVector; Semantic Retrieval

Introduction

Vector embeddings is another important innovation of the human-machine solution that helped to bridge the gap in ability of computers to understand human communication. Vector embeddings give humans the ability to represent words, sentences, and images in a way that computers would understand the context. Large language models (LLMs) have significantly transformed how we interact with and derive insights from vast textual corpora. When combined with Retrieval-Augmented Generation (RAG), LLMs can access external documents to ground their responses in verifiable facts. This architecture is particularly powerful in complex, high-volume domains such as policy analysis, scientific research, or regulatory compliance, where relevant information may span thousands of PDF documents, tables, charts, and infographics.

With the emergence of multimodal large language models (MLLMs), the ability to query both textual and non-textual information such as embedded charts or structured tables within PDFs has opened new frontiers in intelligent document understanding. For example, a query about public attitudes toward climate policy may require the synthesis of survey text and graphical result summaries. RAG architectures enable such queries by retrieving semantically relevant segments from underlying source documents before generating a final response. However, a critical challenge in deploying scalable RAG systems lies in how and where vector embeddings representations of document content are stored and retrieved. Efficient vector storage directly affects retrieval accuracy, system responsiveness, and the memory footprint of the overall pipeline. Our work addresses this bottleneck by comparatively evaluating two popular vector embedding frameworks: FAISS, known for its high-speed approximate nearest neighbour search, and PGVector, a PostgreSQL-native extension that supports persistent, SQL-query able vector storage.

We build and benchmark a semantic retrieval pipeline that processes PDF documents, extracts and summarizes their content, stores the resulting embeddings in both FAISS and PGVector, and retrieves relevant context during LLM inference. Our evaluation leverages both the generation quality (correctness, relevance, faithfulness), along with cosine-based semantic similarity between the LLM outputs and ground truth answers (using LLM-as-a-Judge) to learn about the performance of the two vector embedding frameworks.

Related Work

The use of Retrieval-Augmented Generation (RAG) for information retrieval has drawn considerable attention, as it has proven valuable to researchers and industry experts in various ways. For example, RAG has been applied to open-domain Question-and-Answer, fact verification, long-form reasoning, and general reasoning tasks [1]. It has also been employed in customer service question answering through the construction of knowledge graphs from issue-tracking tickets, improving response accuracy and reducing resolution times [19, 18]. RAG has emerged as a powerful framework for enhancing the factuality and domain grounding of large language models (LLMs) by incorporating relevant external documents into the generative process. This is also affirmed in the work of Lewis et al. [11], where a wide range of RAG toolkits and frameworks like LangChain, Haystack, and LlamaIndex have enabled developers to build intelligent systems for applications such as legal search, clinical question answering, and policy analysis. The use of RAG in medical question answering and clinical reasoning has also been demonstrated [7, 8]. Self-BioRAG selectively retrieves medical documents and self-reflects to verify correctness, thereby improving reasoning and explanation generation in healthcare tasks using domain-specific corpora.

The RAG architecture typically operates in two key stages: indexing and querying. During indexing, document files containing both textual and non-textual content are parsed, chunked, summarised, and embedded as high-dimensional vectors. These embeddings are stored in a vector store to facilitate semantic similarity search. The query stage then retrieves the most relevant chunks based on a user prompt and passes them to an LLM to generate a grounded response. Recent developments in multimodal RAG (MuRAG) have extended these principles beyond plain text to include visual data such as infographics, charts, and images [3, 20]. Multimodal LLMs (MLLMs) are increasingly capable of fusing information across modalities, but evaluation methodologies often rely heavily on Visual Question Answering (VQA) datasets [12], which are not tailored to document-scale semantic retrieval tasks.

Additionally, several approaches have explored agent-based conversational pipelines with memory and reasoning layers to handle complex question-answering workflows [5]. These include modeling retrieval agents independently for image and text inputs, enabling better alignment between user intent and structured document segments. From a storage and performance perspective, the vector database backend plays a critical role in deter- mining the efficiency and scalability of semantic retrieval. Facebook AI Similarity Search is widely adopted for approximate nearest neighbor (ANN) search and optimized for large-scale, low-latency environments [9]. In contrast, PGVector, a PostgreSQL extension for vector similarity search, offers tight integration with relational databases, supporting persistent storage and SQL-based filtering, but with no ANN support. Although several benchmarking efforts have compared ANN algorithms [2] and evaluated hybrid retrievers [10], few studies have directly compared vector stores like FAISS and PGVector within a real-world, PDF- based RAG pipeline. Furthermore, research on integrating semi-structured visual data such as that explored in LayoutLM [18] and DocVQA [13] has largely focused on layout understanding rather than end-to-end semantic retrieval and generation.

This paper addresses these gaps by presenting a comparative evaluation of FAISS and PGVector as vector embedding frameworks within a scalable, multimodal RAG architecture. Our focus extends beyond retrieval accuracy to include generation quality, memory efficiency, and operational integration, particularly in the context of document-grounded question answering at scale.

Methodology

This study presents a Retrieval-Augmented Generation (RAG) framework designed for scalable and semantically precise information retrieval from complex PDF documents containing text, and tables. Our primary objective is not only to enable grounded generation but also to comparatively evaluate the retrieval performance of two vector embedding backends, FAISS and PgVector, in real-world document settings. The system is organised into five functional phases: document chunking and summarization, vector embedding and index construction, memory management, retrieval and RAG chain configuration, and user interface interaction. These phases collectively support efficient document processing and grounded response generation. Each component in the methodology is described below.

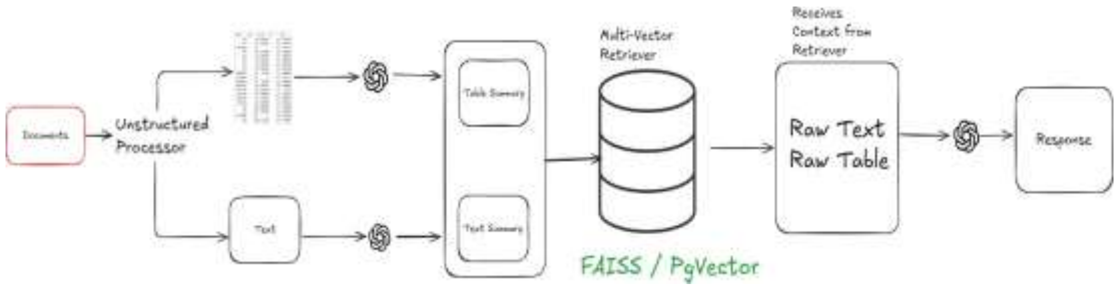


Fig. 1 - Multi-vector retriever using FAISS or PgVector.

Document Chunking and Summarization

The indexing pipeline begins by decomposing each PDF into its fundamental components such as raw text blocks, figures, and tables using a high-resolution, layout-aware parsing strategy designed to preserve page-level structural fidelity, as described in [16, 4, 14]. To support fine-grained semantic retrieval, we implement a chunking mechanism that segments documents based on structural heuristics, particularly section headings and visual markers. Each resulting chunk is annotated with metadata, including block type and layout context, to retain interpretability during downstream tasks. Chunk-level summaries are optionally generated using OpenAI's GPT-4o-mini model to enhance semantic richness and improve retrieval precision in multi-vector query pipelines.

Vector Embedding and Index Construction

Following the chunking and summarization process, each text and table summary is transformed into a high-dimensional vector representation using OpenAI's embedding model. These embeddings capture the underlying semantic content of the chunks and enable efficient retrieval based on similarity scoring. This vectorization step is essential for enabling fast and accurate retrieval of semantically relevant passages within a document. By mapping each chunk from the original textual space into a dense vector space, the system supports scalable similarity search during the querying phase. To access the storage and retrieval opportunities, we evaluate two distinct vector database backends, FAISS and PGVector. All embeddings are stored with unique doc id references that map back to the corresponding original content in a separate Redis-based document store.

Memory Management with Redis

To facilitate efficient mapping between the vector embeddings and their source content, all original document chunks including raw text and table representations are stored in a Redis-based document store. Each chunk is indexed by a unique doc id, which is shared with its corresponding vector embedding in FAISS or PGVector. This enables the retrieval pipeline to fetch the full textual context once a vector match is found. In addition to document storage, Redis helps us to supports the conversational memory needed for multi-turn question answering. We used LangChain's *ConversationBufferMemory* to persist chat history, allowing the system to generate coherent, context-aware responses across successive queries.

Retrieval and RAG Chain Configuration

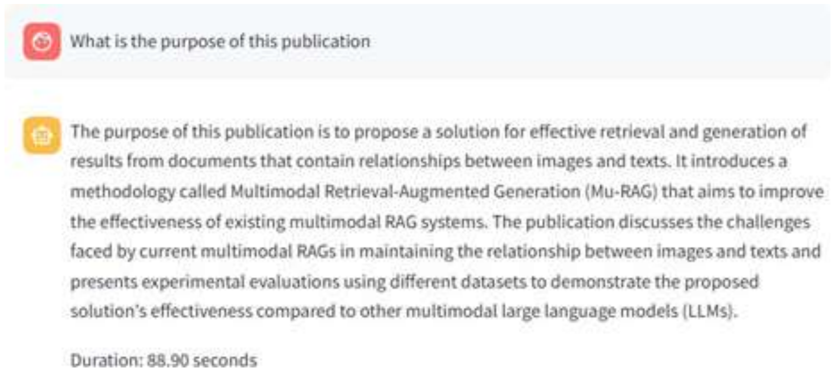
The core question-answering workflow is implemented using LangChain's composable chaining architecture, enabling modular orchestration of retrieval and generation steps. Upon receiving a user query, the *MultiVectorRetriever* searches the vector store to identify semantically relevant document chunks. These chunks are combined and formatted into a unified context using a custom *RunnableLambda* function. The query and the retrieved context are then injected into a structured prompt via *ChatPromptTemplate*, which is subsequently processed by OpenAI's GPT-4o-mini model through the ChatOpenAI interface. The model output is parsed using *StrOutputParser* to extract a clean, user-ready response. This modular design supports seamless substitution of embedding models, vector databases, and prompt templates, enabling flexible experimentation and optimization of the RAG pipeline.

User Interface and Interaction Flow

Users interact with the system through a Streamlit web interface just as shown in Fig. 2 and Fig. 3. They upload PDFs, which are parsed, summarized, embedded, and indexed. After indexing, users submit queries via *st.chat_input*. With that, each query is embedded and matched against stored vectors using cosine similarity. The top results and prior conversation history, with the help of *ConversationBufferMemory* are passed to the LLM to generate a grounded response. The responses are displayed in real-time, along with metadata such as response time and confidence indicators.

Experiments and Analysis

To evaluate the effectiveness of the proposed Retrieval-Augmented Generation (RAG) pipeline, we conducted a series of experiments comparing two vector embedding backends, FAISS and PGVector, across three representative PDF documents. Each document was selected to reflect varying structural and semantic complexities: HBS papers focused on nutritional data and included both prose and tabular content; the *Layout Parser Paper* was structurally dense and layout-sensitive; and RAG Chatbots for Education represented a conceptual academic manuscript with abstract relationships and minimal formatting cues. For each document, text and table embeddings were generated using OpenAI's embedding model and stored using either FAISS or PGVector. Retrieval was performed using a multi-vector retriever, and the top-*k* most relevant chunks were passed to the GPT-4o-mini model for response generation. To assess performance, we evaluated both retrieval and generation quality. Generation quality was assessed using correctness, relevance, and faithfulness scores, obtained through an LLM-as-a-Judge framework. Additionally, semantic similarity was computed using cosine distance between the generated answers and predefined gold-standard responses using LLM-as-a-Judge. Together, these metrics provide a comprehensive evaluation of each backend's ability to support accurate and grounded semantic retrieval across diverse document types.

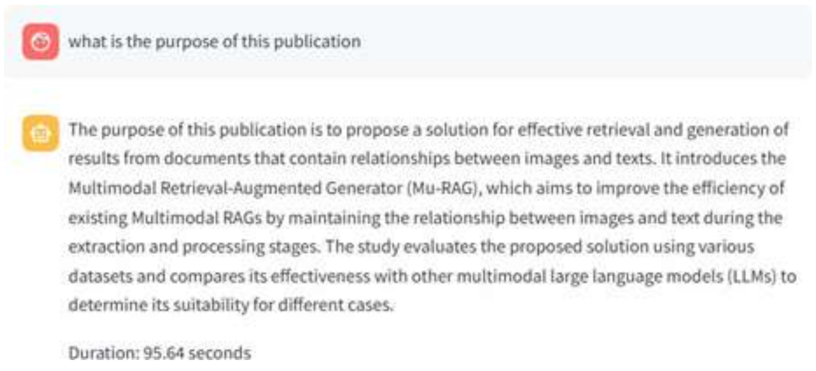


What is the purpose of this publication

The purpose of this publication is to propose a solution for effective retrieval and generation of results from documents that contain relationships between images and texts. It introduces a methodology called Multimodal Retrieval-Augmented Generation (Mu-RAG) that aims to improve the effectiveness of existing multimodal RAG systems. The publication discusses the challenges faced by current multimodal RAGs in maintaining the relationship between images and texts and presents experimental evaluations using different datasets to demonstrate the proposed solution's effectiveness compared to other multimodal large language models (LLMs).

Duration: 88.90 seconds

Fig. 2 - RAG response generation using FAISS for semantic vector retrieval.


what is the purpose of this publication

The purpose of this publication is to propose a solution for effective retrieval and generation of results from documents that contain relationships between images and texts. It introduces the Multimodal Retrieval-Augmented Generator (Mu-RAG), which aims to improve the efficiency of existing Multimodal RAGs by maintaining the relationship between images and text during the extraction and processing stages. The study evaluates the proposed solution using various datasets and compares its effectiveness with other multimodal large language models (LLMs) to determine its suitability for different cases.

Duration: 95.64 seconds

Fig. 3 - RAG response generation using PGVector for semantic vector retrieval.

A summary of the experimental results is shown in Table 1. Across all documents, both FAISS and PGVector failed to retrieve exact matches for gold-standard answers, with the Hit Rate and MRR of 0.000 in every case. This outcome does not imply failure at the generation stage but rather suggests that retrieved chunks were semantically relevant yet did not contain literal replicas of the ground truth with ChatGPT-as-a-Judge. Such behaviour is expected in long-form document retrieval settings, where relevant context may be fragmented or paraphrased. Despite the uniform retrieval scores, meaningful differences emerged in correctness, relevance, and faithfulness of generated responses. On the HBS Papers dataset, both vector stores performed comparably well, with FAISS slightly outperforming PGVector in correctness and faithfulness, while maintaining identical relevance and semantic similarity. The results indicate that for documents with well-structured textual and tabular content, either backend is sufficiently robust. The Layout Parser document, on the other hand, highlighted a more pronounced divergence. PGVector significantly outperformed FAISS across all qualitative metrics, particularly in correctness (4.70 versus 0.00), suggesting that PGVector's tighter integration with relational metadata may enhance retrieval fidelity in layout-rich contexts. FAISS, by contrast, failed to retrieve us- able content, resulting in poor generation quality despite relatively high semantic similarity, which further suggests the model was approximating a response based on minimal context.

Table 1 - Evaluation Results across FAISS and PGVector on Three PDF Documents

Documents	**Backend**	**Correctness**	**Relevance**	**Faithfulness**	**Cosine Similarity**
hbspapers 48.pdf	FAISS	5.00	10.00	10.00	0.944
	PGVector	4.85	10.00	9.40	0.943
layout-parser-paper.pdf	FAISS	0.00	2.50	3.50	0.791
	PGVector	4.70	9.70	9.50	0.910
RAG Chatbots for Ed.pdf	FAISS	4.00	9.00	8.00	0.910
	PGVector	0.00	3.90	7.00	0.815

In the case of the RAG Chatbots document, the performance patterns reversed. FAISS achieved strong correctness, relevance, and faithfulness scores, while PGVector exhibited a breakdown in correctness, despite achieving moderate semantic similarity. This contrast underscores the sensitivity of each vector store to document genre and embedding alignment. Taken together, these findings reveal that while both FAISS and PGVector are capable of supporting a functional RAG pipeline, their strengths are context-dependent. FAISS demonstrates superior performance in content-heavy or abstract research documents, where approximate vector search may benefit from looser semantic boundaries. PGVector, on the other hand, offers advantages in documents with hierarchical structure or multimodal formatting, where schema-aware indexing supports more targeted retrieval. Although semantic similarity remained relatively high for both systems in most evaluations, the inability to retrieve precise gold-standard answers reinforces the importance of chunking strategies, summarization quality, and the need for hybrid or hierarchical retrievers. Ultimately, the choice between FAISS and PGVector should be informed by the nature of the documents being processed and the requirements of the downstream application whether speed, structure, or schema-awareness is the dominant constraint.

Conclusion

These results underscore the trade-offs between the two embedding frameworks: FAISS excels in high- performance vector search and performs well in semantically homogeneous documents. PGVector shows promise in layout-intensive or mixed-format documents, offering better integration with SQL-based logic and persistent document metadata. The discrepancy in performance across documents suggests that the optimal vector backend may be application dependent.

Limitation and Future Work

While the proposed framework demonstrates the feasibility of integrating FAISS and PGVector into scalable RAG pipelines for PDF-based retrieval, several limitations remain. The evaluation was performed on a small set of documents and questions, limiting the generalizability of results. Broader domain coverage and more diverse query types are needed to strengthen the findings. The current system treats FAISS and PGVector in isolation. Future work could explore hybrid architectures that combine ANN search with schema-aware SQL filtering to improve both retrieval accuracy and operational flexibility. Finally, while the pipeline handles textual and tabular content well, support for fully multimodal retrieval, such as charts and figures remains limited. Integrating vision-language models and layout-aware embeddings presents a natural direction for future expansion.

References

[1] Asai A, Lin Z, Lee K, Hajishirzi H. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. International Conference on Learning Representations (ICLR).2024.

[2] Aumuller M, Bernhardsson E, Faithfull A. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems. 2020 Jan;87(C):101374. doi:10.1016/j.is.2019.02.006.

[3] Chen W, Hu H, Chen X, Verga P, Cohen W. MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing ; 2022 Dec; Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. p. 5558–5570. doi:10.18653/v1/2022.emnlp-main.375.

[4] Cho J, Mahata D, I˙rsoy O, He Y, Bansal M. M3DocRAG: Multi-modal retrieval is what you need for multi-page multi-document understanding. arXiv preprint arXiv:2411.04952. 2024 Nov 7.

[5] Foosherian M, Purwins H, Rathnayake P, Alam T, Teimao R, Thoben KD. Enhancing pipeline-based conversational agents with large language models. In: Proceedings of the 1st Workshop on Taming Large Language Models: Controllability in the Era of Interactive Assistants! ; 2023 Sep; Prague, Czech Republic. Association for Computational Linguistics. p. 56–67.

[6] Harbi Y, Medani K, Gherbi C, Aliouat Z, Harous S. Roadmap of adversarial machine learning in Internet of Things–enabled security systems. Sensors (Basel). 2024 Aug 9;24(16):5150. doi:10.3390/s24165150. PMID: 39204846; PMCID: PMC11359573.

[7] Jeong M, Sohn J, Sung M, Kang J. Improving medical reasoning through retrieval and self-reflection with retrieval-augmented large language models. Bioinformatics. 2024 Jun 28;40(Suppl 1):i119–i129. doi:10.1093/bioinformatics/btae238. PMID: 38940167; PMCID: PMC11211826.

[8] Jeong M, Sohn J, Sung M, Kang J. Improving Medical Reasoning through Retrieval and Self-Reflection with Retrieval-Augmented Large Language Models. Bioinformatics. 2024;40(Suppl 1):i119–i128.

[9] Johnson J, Douze M, J´egou H. Billion-scale similarity search with GPUs. IEEE Trans Big Data. 2021 Jul 1;7(3):535–547. doi:10.1109/TBDATA.2019.2921572.

[10] Karpukhin V, Oguz B, Min S, Lewis P, Wu L, Edunov S, Chen D, Yih WT. Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics; 2020. p. 6769–6781. doi:10.18653/v1/2020.emnlp-main.550.

[11] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Kulkarni V, Sun Y, Yuan X, Chen W, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems (NeurIPS). 2020;33:9459–9474.

[12] Marino K, Rastegari M, Farhadi A, Mottaghi R. OK-VQA: A visual question answering benchmark requiring external knowledge. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019 Jun 15–20; Long Beach, CA, USA. IEEE; 2019. p. 3195–3204. doi:10.1109/CVPR.2019.00331.

[13] Mathew M, Karatzas D, Jawahar CV. DocVQA: A dataset for VQA on document images. In: *Proceed- ings of the 2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Waikoloa, HI, USA: IEEE; 2021. p. 2200–2209. doi:10.1109/WACV48630.2021.00225.

[14] Papageorgiou G, Sarlis V, Maragoudakis M, Tjortjis C. A multimodal framework embedding retrieval-augmented generation with MLLMs for Eurobarometer data. Information. 2024;15(6):279. doi:10.3390/info15060279.

[15] Wu J, Li H, Yasunaga M, Liang P. Self-routing RAG: Binding selective retrieval with knowledge ver- balization. arXiv preprint. 2025;arXiv:2404.17265.

[16] Xie X, Yan H, Yin L, Liu Y, Ding J, Liao M, Liu Y, Chen W, Bai X. PDF-WuKong: A large multimodal model for efficient long PDF reading with end-to-end sparse sampling. arXiv preprint arXiv:2410.05970. 2025 Jan 20.

[17] Xu C, Wu Y, Karmaker Santu S. Retrieval-augmented generation with knowledge graphs for customer service question answering. arXiv preprint. 2024;arXiv:2404.04876.

[18] Xu Y, Li M, Cui L, Huang S, Wei F, Zhou M. LayoutLM: Pre-training of text and lay- out for document image understanding. arXiv preprint arXiv:1912.13318. 2020. Available from: https://doi.org/10.48550/arXiv.1912.13318.

[19] Xu Z, Cruz MJ, Guevara M, Wang T, Deshpande M, Wang X, Li Z. Retrieval-augmented generation with knowledge graphs for customer service question answering. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24); 2024. p. 2905–2909. doi:10.1145/3626772.3661370.

[20] Zhao R, Chen H, Wang W, Jiao F, Do XL, Qin C, Ding B, Guo X, Li M, Li X, Joty S. Retrieving multimodal information for augmented generation: A survey. In: Findings of the Association for Com- putational Linguistics: EMNLP 2023 ; 2023 Dec; Singapore. Association for Computational Linguistics. p. 4736–4756. doi:10.18653/v1/2023.findings-emnlp.314.