Optimizing RAG Retrieval Systems for Real-Time Voice Assistants: A Comparative Analysis of Retrieval Techniques

Alessio Bernardini[1*]

[1*]Department of Computer Science, University of Milan, Milan, Italy.

Corresponding author(s). E-mail(s):
alessio.bernardini@studenti.unimi.it;

Abstract

Current Retrieval-Augmented Generation (RAG) systems are not optimized for voice assistants, which require both extremely fast information retrieval to maintain conversational fluency and adequate accuracy to correctly answer the posed question. To efficiently manage the RAG pipeline, particularly the retrieval component, processing times must be minimized to achieve response times of one second or less. This study presents RAG-Document, a centralized architecture for managing the creation and querying of vectorstores accessible by multiple conversational agents, and evaluates ten different retrieval techniques to identify the optimal trade-off between retrieval quality and speed. Through systematic evaluation on diverse document types using 80 test questions, we find that ParentDocumentRetriever and BM25Retriever achieve the best balance, with accuracies of 86.25% and 82.50% respectively, while maintaining response times below one second. These results provide practical guidance for implementing efficient RAG systems in latency-critical voice assistant applications.

Keywords: Retrieval-Augmented Generation, Voice Assistants, Information Retrieval, Natural Language Processing, Real-time Systems

1 Introduction

Voice assistants have become increasingly prevalent in customer service applications, requiring sophisticated information retrieval systems to provide accurate and timely responses. The architecture of modern voice assistants typically consists of sequential components: Automatic Speech Recognition (ASR), Speech-to-Text (STT), Large Language Model (LLM), and Text-to-Speech (TTS). Each component processes information sequentially, making it critical that every stage operates efficiently. For a voice assistant to maintain natural conversational flow, the total response time must not exceed 3-4 seconds, placing stringent requirements on all pipeline components, particularly the information retrieval system.

Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm for enhancing LLM responses with external knowledge [1]. However, current RAG systems are primarily designed for applications where latency is not extremely critical, such as text-based chatbots, document question-answering, or knowledge base querying. When applied to voice assistants, these systems often fail to meet both the real-time requirements necessary for natural conversation and the adequate quality needed to answer the posed question.

The challenge of optimizing RAG for voice assistants is compounded by the need to balance two competing objectives: retrieval quality and response speed. While sophisticated retrieval techniques can improve accuracy, they often introduce additional computational overhead that violates latency constraints. Recent work has explored various approaches to improve retrieval quality, including dense passage retrieval [3], hybrid search methods combining sparse and dense representations [7], and cross-encoder reranking techniques [4]. However, few studies have systematically evaluated these techniques in the context of strict real-time constraints.

This work addresses this gap by presenting a comprehensive evaluation of retrieval techniques specifically designed for voice assistant applications. We introduce RAG-Document, a centralized architecture that manages the entire RAG pipeline from document ingestion to information retrieval. The system is designed to handle simultaneous requests from multiple conversational agents while maintaining low latency through intelligent caching and pipeline management.

Our research contributions are threefold: (1) RAG-Document, a scalable architecture for managing RAG pipelines in multi-agent voice assistant systems; (2) we conduct a systematic evaluation of ten state-of-the-art retrieval techniques across diverse document types and query patterns; and (3) we identify the optimal retrieval strategies that achieve the best trade-off between accuracy and latency for real-time voice applications to be leveraged in the architecture described in point one.

2 Research Question and Methodology

2.1 Research Question

The primary research question addressed in this work is:

> *What is the best retrieval technique that provides the optimal trade-off between quality and response speed for use in a real-time RAG system designed for voice assistants?*

This question is motivated by the specific requirements of voice assistant applications, where response latency directly impacts user experience. While sophisticated retrieval techniques may offer superior accuracy, they are only practical if they can operate within strict time constraints.

2.2 RAG-Document Architecture

To support our evaluation and provide a practical framework for deploying RAG in voice assistant applications, we developed RAG-Document, a centralized architecture for creating and managing vectorstores queryable by multiple conversational agents. The system comprises the following components:

- **Service Manager**: The core component that orchestrates the entire pipeline creation process, from document loading and chunking to embedding generation and vectorstore creation. It also handles retrieval requests from conversational agents.
- **Neo4j Database**: Maintains snapshots of all created vectorstores, enabling tracking and persistence of the system state.
- **API Server**: Provides endpoints for creating, deleting, and modifying vectorstores, enabling dynamic management of the knowledge base.
- **MCP Server**: Serves as the access point for conversational agents, which can request information using unique keys.
- **Caching System**: Maintains a fixed number of pipelines loaded in memory while persisting others to disk. Pipelines are loaded on-demand when requests are received, optimizing memory usage while maintaining responsiveness.

This architecture enables centralized management of vectorstores while providing a single endpoint (MCP Server) for conversational agents to access information. The caching system is particularly important for maintaining low latency, as it ensures that frequently accessed vectorstores remain readily available in memory.

The system is designed to integrate with voice assistant architectures by serving as the knowledge retrieval layer and providing access to company information and policies.

2.3 Retrieval Techniques

We evaluated ten retrieval techniques, selected from current state-of-the-art approaches while excluding overly complex methods that would introduce prohibitive latency. The techniques are organized into three families:

Basic Techniques:

- *Semantic Search*: Dense vector retrieval using embedding similarity [3, 9].
- *Key-based Search (BM25)*: Sparse retrieval based on term frequency and inverse document frequency [2].
- *Hybrid Search*: Combines semantic and keyword-based approaches using score fusion methods [7].

Advanced Techniques:

- *Relevant Segment Extraction (RSE)*: Identifies clusters of contiguous chunks that maximize total relevance.
- *Parent Document Retrieval*: Retrieves small chunks for matching but returns larger parent chunks for context, balancing precision with contextual completeness.

LLM-Enhanced Techniques:

- *Contextual Chunk Headers*: Adds a description of the document containing each chunk within the chunk itself.
- *Hierarchical Indices*: Uses a high-level index (summaries) to filter documents and a low-level index for specific chunks.
- *Query Transformations*: Decomposes or reformulates queries using an LLM before retrieval [8].
- *Multi-Query RAG*: Generates query variants to cover different areas of the semantic space.
- *Reranking*: Performs an initial semantic search (Top-20) and then reorders results using an LLM for surgical precision.

For all experiments, we used `sentence-transformers/all-MiniLM-L6-v2` as the embedding model due to its balance of speed and quality. For techniques requiring LLM calls, we used Groq as the provider with the `gpt-oss-120b` model, selected for its excellent performance in both speed and accuracy.

Implementation Note: Several of the retrieval techniques evaluated in this study were implemented based on approaches documented in the RAG Techniques repository [10], which provides a comprehensive collection of state-of-the-art RAG implementations.

2.4 Problem Formalization

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ represent a collection of documents, and let q denote a user query. The retrieval task can be formalized as finding a function $R : \mathcal{Q} \times \mathcal{D} \to \mathcal{C}$ that maps a query and document collection to a set of relevant chunks $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$, where each chunk c_i is a segment of a document in $\mathcal{D}$.

For voice assistant applications, we must optimize two objectives:

$$\text{Minimize: } T(R, q, \mathcal{D}) \tag{1}$$

$$\text{Maximize: Accuracy}(R, q, \mathcal{D}, \mathcal{A}) \tag{2}$$

where $T(R, q, \mathcal{D})$ represents the retrieval latency and $\text{Accuracy}(R, q, \mathcal{D}, \mathcal{A})$ measures the quality of retrieved chunks against ground truth answers $\mathcal{A}$.

The constraint imposed by voice assistant applications is:

$$T_{\text{total}} = T_{\text{ASR}} + T_{\text{STT}} + T_{\text{RAG}} + T_{\text{LLM}} + T_{\text{TTS}} \leq 3 \text{ seconds} \tag{3}$$

Furthermore, to ensure that the retrieval module does not compromise the overall response time, we impose the following operational constraint:

$$T_{\text{RAG}} \leq 1 \text{ second} \tag{4}$$

3 Experimental Results

3.1 Dataset Description

To comprehensively evaluate the retrieval techniques, we assembled a diverse collection of documents representing various formats and content types commonly encountered in customer service applications:

- **Increso.pdf**: 19-page PDF containing Q&A about the Increso company
- **Unipol.pdf**: 52-page PDF with Q&A about insurance company policies
- **Amazon.docx**: 20-page document detailing shipping and return policies
- **Apple.pdf**: 193-page Apple user manual
- **Etsy.pdf**: 43-page Etsy tax guide
- **Energy Company Documents**: Four text files covering billing (*Fatturazione.txt*), payment (*Pagamento.txt*), installment (*Rateizzazione.txt*), and utility (*Utenza.txt*) policies
- **WhatsApp.pdf**: 63-page application usage manual
- **Hotel.pdf**: 4-page hotel FAQ document

For each vectorstore, we selected ten questions that could be answered using the documents it contains, resulting in 80 different test questions with corresponding ground truth answers. This enables systematic evaluation across different document types and query patterns.

3.2 Evaluation Metrics and Methodology

The evaluation process consists of three phases:

Phase 1: Pipeline Creation Time. We measure the time required for each retriever to create its pipeline, including document processing, chunking, and any technique-specific preprocessing. While this metric is less critical for production systems (as pipelines are created offline), it provides insight into the computational requirements of each approach.

Phase 2: Retrieval Latency. We execute all 80 questions against each retriever and record the average retrieval time and the chunks returned. This generates 800 test cases (10 retrievers $\times$ 80 questions).

Phase 3: Accuracy Evaluation. We employ an LLM-as-Judge methodology to evaluate whether each retriever returned the correct chunks and whether the question can be answered correctly using those chunks. This provides a measure of retrieval quality that correlates with end-user experience.

3.3 Results

Table 1 presents the comprehensive results of our evaluation across all ten retrieval techniques.

Table 1 Performance comparison of retrieval techniques

Retriever	Avg Pipeline Time (s)	Avg Retriever Time (s)	Accuracy
BM25Retriever	37.72	**0.00076**	66/80 (82.50%)
ContextualHeaderRetriever	45.91	0.02001	50/80 (62.50%)
HierarchicalRetriever	49.51	0.02085	52/80 (65.00%)
HybridRetriever	48.85	0.02269	63/80 (78.75%)
MultiQueryRetriever	47.21	0.65429	52/80 (65.00%)
ParentDocumentRetriever	46.78	0.01861	69/80 (86.25%)
QueryTransformRetriever	43.32	1.28019	45/80 (56.25%)
RelevantSegmentRetriever	48.47	**0.01542**	30/80 (37.50%)
RerankingRetriever	47.40	5.71086	71/80 (**88.75%**)
SemanticRetriever	42.11	0.01738	54/80 (67.50%)

3.4 Analysis

Pipeline Creation Time. BM25Retriever exhibits the fastest pipeline creation time (37.72 seconds), benefiting from its simpler indexing requirements compared to embedding-based approaches. Other techniques show relatively similar creation times ranging from approximately 42 to 49 seconds, with SemanticRetriever (42.11 seconds) and QueryTransformRetriever (43.32 seconds) being slightly faster than techniques requiring more complex preprocessing. However, pipeline creation time is of limited practical importance since this process occurs offline before the voice assistant is deployed.

Retrieval Latency. This metric is critical for voice assistant applications. BM25Retriever achieves the lowest latency (0.00076 seconds), followed closely by RelevantSegmentRetriever (0.01542 seconds) and ParentDocumentRetriever (0.01861 seconds), with several other techniques maintaining sub-0.03 second response times. In contrast, RerankingRetriever (5.71 seconds) and QueryTransformRetriever (1.28 seconds) introduce latencies that are prohibitive for real-time voice applications. The MultiQueryRetriever also shows elevated latency (0.65 seconds) due to multiple retrieval operations.

Accuracy. RerankingRetriever achieves the highest accuracy (88.75%), validating the effectiveness of cross-encoder reranking. However, its latency makes it impractical for our application. ParentDocumentRetriever demonstrates the second-highest accuracy (86.25%), followed by BM25Retriever (82.50%). Notably, RelevantSegmentRetriever shows the lowest accuracy (37.50%), suggesting that the segment extraction approach may not be well-suited for the diversity of queries in our test set.

Trade-off Analysis. When considering the joint requirements of latency and accuracy, ParentDocumentRetriever and BM25Retriever emerge as the most suitable choices for voice assistant applications. ParentDocumentRetriever offers the best balance with 86.25% accuracy and minimal latency (0.01861 seconds), while BM25Retriever provides exceptional speed (0.00076 seconds) with solid accuracy (82.50%). The slightly higher accuracy of ParentDocumentRetriever comes with the benefit of retrieving larger context windows, which may improve LLM response generation.

The strong performance of BM25Retriever is noteworthy and may reflect several factors: (1) our test questions frequently contained keywords directly present in source documents, favoring keyword-based retrieval; (2) the document collection includes structured Q&A content where keyword matching is particularly effective; and (3) the simplicity of BM25 eliminates potential errors introduced by more complex retrieval pipelines. HybridRetriever, which combines semantic and keyword approaches, achieves respectable accuracy (78.75%) with low latency, representing another viable option for applications requiring balanced performance.

4 Concluding Remarks

This study presents a comprehensive evaluation of retrieval techniques for RAG systems in voice assistant applications, addressing the critical challenge of balancing retrieval quality with strict latency constraints. Through systematic evaluation on diverse document types using 80 test questions, we identify ParentDocumentRetriever and BM25Retriever as the optimal choices for real-time voice applications, achieving accuracies of 86.25% and 82.50% respectively while maintaining sub-second response times.

Our results demonstrate that achieving high accuracy does not necessarily require the most sophisticated techniques. ParentDocumentRetriever achieves near-optimal accuracy (86.25%) with minimal latency overhead, while the surprisingly strong performance of BM25Retriever (82.50% accuracy at 0.00076 seconds) suggests that for certain document types and query patterns, particularly those involving well-defined

keywords and structured content, simpler methods can be highly effective. This finding has important practical implications for system designers, who can achieve excellent performance while minimizing computational complexity.

The RAG-Document architecture presented in this work provides a practical framework for deploying RAG systems in multi-agent voice assistant applications. Its centralized design enables efficient management of multiple vectorstores while the caching system ensures that frequently accessed information remains readily available. This architecture can be readily adapted to various voice assistant applications, particularly in customer service domains where access to company policies and documentation is essential.

4.1 Limitations and Future Work

Several limitations of this study suggest directions for future research:

Test Set Bias. Our test questions may favor keyword-based retrieval due to their construction. Future work should include more diverse query types, including paraphrased questions and queries requiring deeper semantic understanding.

Document Diversity. While our evaluation includes various document types, expanding to additional domains (technical documentation, conversational transcripts, multimedia content) would provide more comprehensive insights.

Hybrid Approaches. The moderate performance of HybridRetriever suggests that simple combinations of semantic and keyword methods may not capture the full potential of hybrid approaches. Future work could explore more sophisticated fusion strategies, including learned combination weights and dynamic method selection based on query characteristics.

Adaptive Retrieval. Developing methods that dynamically select retrieval strategies based on query characteristics could optimize the latency-accuracy trade-off on a per-query basis.

Multi-lingual Support. Extending the evaluation to non-English documents and queries would assess the generalizability of our findings across languages.

In conclusion, this work provides practical guidance for implementing efficient RAG systems in latency-critical voice assistant applications. The strong performance of ParentDocumentRetriever, combined with the exceptional speed of BM25Retriever, offers system designers concrete options for achieving the necessary balance between speed and quality. As voice assistants continue to proliferate in customer service and other domains, these insights will become increasingly valuable for delivering responsive and accurate conversational experiences.

Acknowledgements. The authors would like to thank Increso for providing the opportunity to develop and test the RAG-Document system in a production customer service environment.

Declarations

Data availability: The datasets and code used in this study are replicable and available at the following link: https://github.com/abernardini-unimi/nlp-project.

References

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.

[2] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333-389.

[3] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. T. (2020). Dense passage retrieval for open-domain question answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6769-6781.

[4] Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*.

[5] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

[6] Ma, X., Gong, Y., He, P., Zhao, H., & Duan, N. (2023). Query rewriting for retrieval-augmented large language models. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 5303-5315.

[7] Luan, Y., Eisenstein, J., Toutanova, K., & Collins, M. (2021). Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9, 329-345.

[8] Li, M., Zhang, Y., Liu, J., Wang, H., Chen, X., Zhao, Y., ... & Lin, J. (2025). Query expansion in the age of pre-trained and large language models: A comprehensive survey. *arXiv preprint arXiv:2509.07794*.

[9] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982-3992.

[10] Diamant, N. (2024). RAG Techniques: A comprehensive collection of advanced Retrieval-Augmented Generation techniques. GitHub repository. https://github.com/NirDiamant/RAG_Techniques