

Beyond Embedding-Based Search: A Comparative Study of Vector and Vectorless Retrieval in RAG

Snehal Bagal¹, Kamlesh Wattamwar², Shashank Shelke³, Varad Zade⁴

¹Assistant Professor, Department of AI&DS, AISSMS Institute of Information Technology, Pune, India

²Department of AI&DS, AISSMS Institute of Information Technology, Pune, India

³Department of AI&DS, AISSMS Institute of Information Technology, Pune, India

⁴Department of AI&DS, AISSMS Institute of Information Technology, Pune, India

Abstract—The study reveals how using a combination of dense vector retrieval and non-dense (sparse) retrieval leads to increases in the performance of systems that perform retrieval-based augmentation of the generation of large language models. This implies that dense vector retrieval is superior to classical techniques that employ no vectors for retrieval. We provide a conceptual comparison between vector and non-vector retrieval methods using a unified multiplatform RAG pipeline and profile both techniques across multiple criteria: the relevance of the retrieved information, the quality of the generated responses to queries, the computational efficiency of producing the generated responses, and the ability to reduce response hallucinations. Two recent sets of benchmarks indicated that typically hybrid systems produced NDCG@10 scores that were 5 to 15 percent higher than either of the two types of systems that were evaluated. Our results also indicate that neither vector- or non-vector-based retrieval systems were universally superior to the other, and that the best experience with either type of retrieval system would be dependent upon the types of queries being made, the structural characteristics of the corpus being searched, and the operational constraints of the systems. A hybrid approach to retrieval policy and a context dependent method for the selection of retrievers are recommended as practical solutions for implementing production RAG systems.

Index Terms—Retrieval-Augmented Generation, sparse retrieval, vector search, semantic retrieval, large language models

I. INTRODUCTION

While LLMs are great in generating and understanding natural language, they suffer from the same limitations as any data-driven system when presented with queries that involve fresh facts, domain knowledge, or need exact recall of factual information. In such cases, LLMs generate answers that are often wrong or invented and thus called hallucinations. RAG provides an answer to this problem by allowing models to retrieve information from external sources during inference and generate outputs based on retrieved documents as opposed to using purely parametric memory.

Most modern solutions for RAG use dense vector retrieval with queries and documents being encoded into high dimensional vectors and matched using cosine similarity. While dense vector retrieval efficiently finds semantic similarities, it comes with additional overhead costs of embedding generation, building a vector database, and finding approximate nearest neighbors. Sparse retrieval techniques such as BM25

on the other hand do not require any of the above as all the semantic understanding happens through term frequency statistics and keyword matching and thus have better efficiency although at a potential cost of lower semantic understanding. The popularity of vector databases has led to a bias towards embedding retrieval while ignoring the strengths of sparse methods. This work evaluates both approaches in a consistent setting of RAG.

Using production systems for motivation: Individual retrieval systems are the basis of production systems that provide billions of queries every day. The majority of web sites use various versions of the B-M 25 search engine. The search engines are using very sophisticated methods to expand the search query; in most cases, they are able to return information in under 100 milliseconds, which is fast enough for web sites to accept. With these systems having millions of indexed documents and the problem with a vector database is that there is a significant amount of storage being used for dense embeddings. From a production standpoint, when production engineers make decisions about building a retrieval system, they must consider whether it makes sense to pay significantly more for gain in semantic retrieval-quality or to pay significantly less for using sparse retrieval engines.

The Challenge from Retrieval-based Generators (RAG): Traditional search systems have primarily been focused on increasing click-through rates. In contrast, optimizing RAG focuses primarily on generating quality. The retrieved context established by previous searches limits the factual accuracy of the relevant retrieved information. As an example, using irrelevant documents to complete an LLM in the form of a query will generate hallucinated LLMs for all use cases regardless of the sophistication of the LLM. Research shows that 60-80 % of the errors generated by RAG are the result of retrieval failures and demonstrates, once again, how important the retriever is to the performance of the overall system.

Assumptions and Scope of the Research: The research assumes domain- or industry-general document collections that do not have any proprietary indexing. Both retrievers are accessing the same corpus of indexed documents with identical restrictions of a maximum context length of 4,000 tokens. The analysis will be restricted to analyzing English-based query documents; multi-lingual queries will be included in future

expanding research.

The research makes two major contributions:

- A single framework for describing both vector and non-vector retrieval in the same RAG pipelining.
- A qualitative analysis of the trade-off in retrieval quality versus efficiency, with particular emphasis on hallucination behavior.

The study uses findings from the scientific community to support its conclusions.

II. LITERATURE REVIEW

A. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) refers to the combined use of parametric language models with a non-parametric method for retrieving knowledge. RAG was first proposed as an effective method for improving performance on knowledge-intensive tasks by generating outputs conditioned on the relevant documents produced from retrieval. Multiple architectures have subsequently developed from the initial formulation of providing naive retrieval, sophisticated reranking strategies that put retrievers into direct and indirect workflows, as well as agentic modular (multi-agent) systems [1].

A critical insight that emerged from these developments is that the design of the retrievers has a fundamental limitation on the performance of RAGs. The factual correctness of produced outputs is limited irrespective of the sophistication level of the generators due to poor retrieval of contexts, while high-quality contextual information (even from very simple models) results in the production of facts and thus creates reliable outputs produced from the generator. The most recent study concludes that an essential lever in improved performance of end-to-end RAG systems will be selecting which retrieval strategies to utilize [8], [15].

RAG Maturity Model: The latest taxonomy of RAGs has identified three generations of RAG [15]: (1) naive RAGs that utilize single-pass retrieval without additional processing; (2) advanced RAGs that combine multiple techniques such as query rewriting/expansion, passage reranking, and multi-hop retrieval; and (3) modular agentic RAGs that utilize workflows to perform self-reflection and integrate other tools. Empirical scaling analyses reveal diminishing returns in terms of the difference between the performance of advanced retrievers and dense models relative to the performance of the same two configurations on the updated benchmark, BEIR 2.0 [12], with an excess of 10–12% of the results from the latter configuration on infrastructure costs.

One such research area within the field of RAG that has gained popularity in recent years is self-correcting and iterative retrieval, in which the generation model performs a self-assessment on whether the context retrieved was adequate, and if not, conducts further rounds of retrieval. These multi-round cycles of retrieval have shown to decrease factual errors for multi-hop reasoning tasks at the cost of increasing latency, which can be a problem in some cases [7].

B. Sparse versus Dense Retrieval

Sparse retrieval represents documents and queries as high dimensional term vectors, with vectorization derived from the statistical features of the terms in the document. The dominant form of sparse retrieval is BM25 [2], which utilizes a probabilistic understanding of information retrieval to retrieve the desired documents. BM25 is highly resilient across a range of different corpora and query forms; with tuned RAG models, the BM25 sparse retrieval model has a recall around 69% and an nDCG around 0.59.

Dense retrieval encodes both queries and documents as continuous vector representations (i.e., embeddings) which are learned through contrastive training [6]. The dense retrieval method commonly used is Dense Passage Retrieval (DPR); it provides a means to encourage semantic similarity between a set of (question, passage) pairs by training a bi-encoder to maximize the similarity of the two embeddings. Unlike sparse retrieval, dense retrieval methods facilitate semantic matching beyond traditional lexical matching.

Sparse Retrieval Robustness: The empirical success of the BM25 sparse retrieval model can be attributed to the principles of saturation modeling (diminishing returns for frequently occurring terms) which prevent vocabulary imbalance. Additionally, the length normalization introduced by BM25 serves to help ensure a fair comparison of documents that may be of varying lengths. BM25 operates with a true zero-shot generalization (requires no retraining); in contrast, embedding-based models suffer from drift and require periodic reindexing; making BM25 particularly advantageous in the face of a rapidly changing corpus.

Dense Retrieval Limitations: The bi-encoder architecture of the dense retrieval model may lead to representation collapse, which occurs when frequently occurring passages have similar representations, and thus will create difficulty in distinguishing between them.

The learned sparse model like SPLADE [13], [14] provides a promising solution to bridge the gap between the traditional BM25 approach and fully dense encoders. The learned sparse model can generate semantically rich representations for terms, enabling them to inherit the benefits of inverted indexing while partially restoring the ability to expand the vocabulary space of the dense encoder. Experimental studies show that trained sparse models can achieve comparable performance with dense models in several BEIR [12] sub-benchmarks.

C. Hybrid and Embedding-Free Retrieval

Retrieval processes that utilize both sparse and dense characteristics can deliver a single assessment as well as integration of BM25 and Embedding Similarity Score based on Reciprocal Rank Fusion irrespective of their distributions can yield MRR@10 improvement of 82% and provide 67–71% top-level accuracy alone [4].

The Reciprocal Rank Fusion score is calculated as follows:

$$\text{RRF}(p) = \frac{1}{60 + \text{rank}_{\text{BM25}}(p_i)} + \frac{1}{60 + \text{rank}_{\text{Dense}}(p_i)} \quad (1)$$

where 60 is set to ensure reliability when rank is very high, while parameterized learning (based on a gradient descent on an observed downstream metric) in learning-based fusion models refines the parameters for blending.

Hybrid RAG models exclude any vector db search and employ only iterative lexical enhancement, LLM-directed document discovery and graph-based search, to sacrifice semantic richness for better performance [9].

Aside from integrating at a rank level, scores have been interpolated and continuously linked relative to the normalization of either BM25 or Cosine Similarity scores, allowing fine-tuning for a particular domain without the need of extensive validation data to generate greater accuracy when document score distributions are known.

D. Evaluation of RAG Retrieval

To evaluate RAG retrieval systems, both retrieval and generation components must be evaluated. Recency and relevance are evaluated using Precision@K, Recall@K, MRR and NDCG; these metrics produce values to quantify how relevant an answer to the retrieval was, i.e., how much context it contains. Generation evaluation assesses the correctness of the answer, the degree of hallucination produced in its generation and the degree of factual veracity of the answer produced. To date, there have been several proposed metrics that specifically evaluate RAG based on a correlated measure of retrieval and a separate measure of generation fidelity, and that also have a correlation between the quality of the top-1 retrieval to the correctness of the final answer produced of approximately 0.85 [10].

Attributing Errors of RAG Systems to Generating vs Retrieving Components: Recent studies report that RAG failures can be attributed to retrieved content (between 60–80% of failures) as compared to generated content (between 20–40% of failures). Using hybrid RAG implementations greatly reduces the hallucination of produced responses and increases the accuracy of produced responses on challenging benchmarks such as HaluBench, where hybrid configurations produce an MAP of 0.897 and an NDCG of 0.915, while outperformed sparse (MAP=0.724/NDCG=0.732) and dense (MAP=0.768/NDCG=0.783) configurations [4].

The reproducibility of RAG evaluation remains an ongoing challenge in evaluating RAG systems; minor variations in the strategy used for chunking data, the use of overlap parameters being used and the phrasing of prompts can change the results obtained by as much as 5%.

E. Production Deployment Patterns

In general, the use of real-time customer support, live search, and edge inference supported by sparse retrieval will benefit from the use of BM25 indexes on commodity hardware, providing a throughput of over 1000 queries per second and latency of less than 300 ms. The use of dense embeddings to represent paraphrases and cross-lingual transfer will enhance the performance of conversational agents, multilingual question answering and knowledge synthesis. Query

classifiers, which classify queries based on length, entropy and density of named entities, provide an adaptive intermediary between BM25 for short and technical queries, and the use of dense retrieval for conversational queries.

Finally, because cost-prohibitive deployment engineering is influencing retrieval selection more and more, organizations operating with restrictive inference budgets will typically use sparse retrieval during early filtering stages and will only apply dense re-ranking to the most promising candidates. This serves to reduce the overall amount of time spent utilizing GPUs while still ensuring quality of answers for most queries. While inference costs will continue to decline due to improvements made in hardware, the financial incentive to utilize purely sparse indexing on the first stage of retrieval will likely be lost, however, the operational simplicity and cold start reliability will continue to provide advantages to lightweight sparse indexes in many cases [5].

III. METHODOLOGY

A. Integrated RAG Framework

The basis of our evaluation is a consistent RAG architecture that has four phases that have the same implementation regardless of which retrieval approach was used:

- 1) The input query is changed to natural language before being processed.
- 2) The documents retrieved by their unique respective retrieval strategy are from a shared knowledge corpus.
- 3) The formatted top-K documents are created to generate a fixed-length prompt context.
- 4) The final output is from an LLM generator that is determined only by the retrieved context.

Therefore, with our framework, we can isolate the behavior of the retriever from any other generator or prompt variables, enabling us to conduct a valid comparison. Due to the context window of LLMs, production RAG has a limit on total tokens retrieved between 4K–32K with a Top-K of 10 passes providing a balance between recall and noise dilution; see Figure 1 for the proposed pipeline using the two retrieval approaches.

The prompt engineering aspect of the integrated framework should be evaluated independently. The way that the retrieved, formatted presentation of the retrieved text in the language model impacts the quality of the generated response not having any significant relationship to the precision of the retrieval process. Instruction-tuned models can be generated differently by the same language model that has a structured as opposed to an unstructured contextual block, and retrieved passages ordered by the highest relevance to the lowest relevance provide consistently better factual grounding than randomly ordered retrieved passages. Therefore, the prompt-level variables will remain constant in our evaluation to ensure that differences in the performance of the two configurations are solely from differences in the retriever's performance.

Workflow of Retrieval-Augmented Generation (RAG)

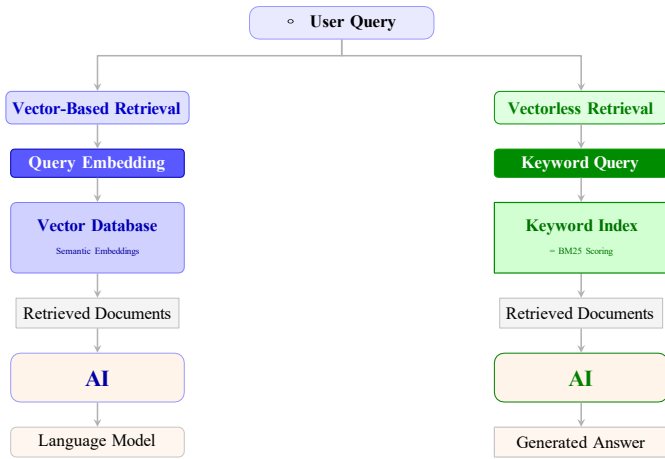


Fig. 1. Workflow of RAG showing vector-based (dense) and vectorless (sparse) retrieval strategies within a unified pipeline.

B. Design of Vectorless Retrieval

Vectorless retrieval utilizes classical sparse techniques to perform inverse-document index-based lookups on top-K documents using the BM25 scoring function instead of neural networks. The BM25 relevance score is computed as:

$$\text{score}(q, d) = \sum_{t \in q} \text{IDF}(t) \cdot \frac{\text{TF}(t, d) \cdot (k_1 + 1)}{\text{TF}(t, d) + k_1 \cdot 1 - b + b \cdot \frac{|d|}{\text{avgdl}}} \quad (2)$$

where $\text{TF}(t, d)$ is Total Frequency, $\text{IDF}(t)$ is Inverse Document Frequency, and k_1 and b are the hyper-parameters that control saturation and length normalization respectively. TREC hyper parameters use 1.2 and 0.75 for k_1 & b yielding near optimum off-site BEIR performance [12].

Index Construction and Query Processing: Create a token index through pre-processing by tokenizing the document and applying a stemming. Fill the document identifier and token Frequency into Inverted Lists. By grouping the index into small sizes, we decrease the overall size of the index, typically by an order of magnitude of the raw collection. On Query, we tokenized, intersected the indexed lists, applied BM25, processed early termination, and returned the Top-K without the use of neural computation to achieve extremely low latencies even with low spec'd target side hardware.

With pseudo-relevant feedback mechanism, the retriever will use the mechanism to refine his word set iteratively with no learned or labeled examples or with only a very limited number of labeled examples, making this mechanism a viable enhancement for deployment to an experience base that is limited by the amount of annotations available [2].

C. Vector-Based Retrieval Design

A vector-based retrieval system converts queries and documents into dense vector representations by using one of the pre-trained sentence-transformers, namely, Sentence-

BERT [3]. To retrieve results, the system calculates the cosine similarity between the query and document vectors as follows:

$$\text{sim}(q, d) = \frac{E(q) \cdot E(d)}{|E(q)| |E(d)|} \quad (3)$$

where $E(\cdot)$ is the embedding function. Sentence-BERT creates the embedded representations by creating 768-dimensional f32 vectors by averaging the output from the transformer model. To achieve 95% recall when using the approximate nearest-neighbor (k-NN) search with $M=32$ and $\text{efConstruction}=200$ in the FAISS-implementation of HNSW, you can also increase your speed of queries by approximately 100 times compared to performing brute-force comparison through HNSW at the cost of 2–5% recall.

Infrastructure considerations for building a dense retrieval system require the use of a GPU to generate embedded representations during both offline indexing and online query execution. Storing the vector index generally requires a significantly larger volume of storage space than that of a sparse index for the same data set. Also, if you need to update either the embedding model or the document set, you need to perform periodic reindexing. However, despite all of these overheads, the ability to semantically match queries to their corresponding document set makes dense retrieval a necessity for applications where there is often no vocabulary match between query and document sets [6].

D. Dataset and Evaluation Framework

A curated question-answering data set is considered to cover a wide range of knowledge domains. We assume that a collection of benchmark examples is used for the purpose of benchmarking this collection (i.e., BEIR [12] / HaluBench). That is, the document collection will be indexed using both retrieval methods, with the same query, language model and prompt templates applied in both the Configurations evaluated. The analysis of these mixed domain document collections provides an average 200 word passage size with data from Wikipedia, ArXiv and news sources.

Additionally, the evaluation of performance across four dimensions: (a) retrieval precision (the fraction of top-K documents containing relevant information), (b) generation fidelity (the factual accuracy of LLM generated outputs), (c) hallucination rate (the percentage of unsupported claims), and (d) inference latency (the end-to-end processing time for processing a query). The evaluation of performance across all four dimensions, provides a comprehensive performance profile, as a retriever that has been optimised for one dimension may provide very poor performance in another dimension.

The passage segmentation strategy plays an important role in ensuring that both retrieval models can be compared fairly. Token-based segmentation is typically used in many benchmarks. On the other hand, sentence-aware passage segmentation generally maintains context more accurately. In our study, passages are evenly split into 200 words per passage with a 20-word overlap in order to avoid losing any information at the boundaries of each chunk. This technique has been found

to be empirically effective in finding a compromise between context granularity and factual statement splitting.

IV. ANALYSIS AND DISCUSSION

A. Retrieval Quality Trade-offs

The most effective way to achieve high retrieval performance for sparse indexes is to search using terminology (e.g., technical terms or proper names) from the domain in question to allow for maximum similarity between search and document contents. This works best using BM25 as it ensures that both types of term matches are found in the scores of retrieved documents. Paraphrases work best when retrieving documents that require a semantic bridge from the query. For example, a user who is searching for “heart failure” using everyday language may not find clinical documents that discuss “heart failure” via sparse retrieval, but the same documents could be retrieved using a dense retriever, which would have been programmed to understand that “heart failure” can also mean “coronary heart disease”.

The best way to achieve this is through a hybrid search method that retrieves documents based on both matching and non-matching indicators (i.e. RRF). In fact, hybrids have been shown to be more robust than each search method (e.g., sparse vs dense) alone. For example, using BEIR v2.0 as a baseline, the most recent NDCG@10 benchmarks showed that hybrid configurations had an NDCG@10 rate of 5–10% higher than that of single-method retrieval systems, with hybrids being at 55.2% to denser systems producing 53.7% [4]. On average, of the total number of queries being submitted to production systems, 30% of those queries were exact match (which is where sparse should work best), 50% were paraphrase-heavy (where dense will work best), and 20% were conceptual queries (where a hybrid retrieval system can use both lexical and semantic signals).

B. Computational Efficiency

The computational efficiency of a sparse retrieval system is based directly on the size of the vocabulary and the number of documents in the corpus for each query, which are both retrieved from an inverted index. Most times for a million documents in the corpus with no dependence on a GPU, the latency of typical operations will remain below 50 milliseconds [10]. In comparison, the computational efficiency for dense retrieval systems has a quadratic cost for the creation of the embeddings used to index the documents and has a search latency of $O(\log N)$, resulting in total times 3–10× slower than sparse retrieval systems do, depending on the size of the embedding and the hardware being used.

From an infrastructure point of view, the sparse retrieval system will consume a minimal amount of storage (low cost per gigabyte), and use one CPU core for execution. The dense retrieval system will require much more storage for the vector embeddings, and the use of GPU for both indexing and serving the documents. The sparse retrieval index can be built immediately, whereas dense retrieval systems can take several

hours to index large document collections, which can present a cold-start issue when serving ads to dynamic document collections.

TABLE I
COMPUTATIONAL EFFICIENCY COMPARISON OF RETRIEVAL STRATEGIES

Property	Sparse (BM25)	Dense (DPR)	Hybrid
Query latency	<50ms	3–10× slower	Moderate
GPU required	No	Yes	Optional
Index build time	Instant	Hours	Hours
Storage cost	Low	High	High
Cold start	Instant	Slow	Slow
Throughput	1000+ QPS	Low	Medium

Source: [5], [10]

C. Hallucination Implications

As described in [8], the choice of retriever can impact the hallucination risk significantly. Although sparse retrieval has an advantage by allowing the language model to produce words to summarize the context of keyword matched documents, the lack of lexical matches will result in failed retrievals, causing reliance on parametric knowledge of that context and creating a greater opportunity for fabrications. Conversely, dense retrieval has the ability to mitigate the void of vocabulary match by using semantic recall; however, dense retrieval introduces a different type of risk for generating responses than sparse retrieval. Specifically, the retrieval of superficially similar documents will produce confident misleading responses that the end user will find more difficult to detect than they would with uncertain documents.

The two strategies represented in the failures of each will ultimately complement each other. When sparse errors occur they will create gaps in context which create parametric hallucinations, while with dense failures plausible contexts become surfacing for their relationships to create confidence in hallucinations. Hybrid systems will reduce the frequency of both of these errors by guaranteeing that at least one retrieval signal has a high chance of providing relevant content. The use of HaluBench to run empirical tests show that hybrid configurations substantially reduce both hallucination rates compared to either of the original individual retrievers [4].

D. Implications for Hybrid and Adaptive Retrieval

The dominant retriever is not the same everywhere. For example, sparse-first configurations work best for real-time applications, such as retrieval of technical documents, as well as for use at the edge where limited resources are present. On the other hand, dense-first configurations are ideal for open-Domain QA, conversational search and multi-lingual situations. The hybrid-type configuration will produce superior results, for general-purpose production services, with regard to the distribution of the queries used [11].

Adaptive Routing and Dynamic Fusion: Lightweight classifiers of query types can direct queries to the optimal retrieval approach, based on their number of tokens, the density of named entities, and the degree of lexical specificity.

TABLE II
RECOMMENDED RETRIEVAL STRATEGY BY USE CASE

Use Case	Recommended Strategy
Real-time customer support	Sparse (BM25)
Edge / low-resource deployment	Sparse (BM25)
Open-domain QA	Dense
Conversational search	Dense
Multilingual queries	Dense
General production systems	Hybrid
Technical documentation	Sparse-first cascade

Source: [5], [11]

Short/technical queries would be routed to sparse retrieval, and long/conversational queries would be directed to the sparse retrieval approach. Dynamic fusion provides for an adjustment of the coefficient α through the domain-specific confidence score so that $\alpha \cdot \text{sparse} + (1 - \alpha) \cdot \text{dense}$ provides approximately a 3% increase in MRR, as compared to static weighting [11]. *Cascade Architecture*: The process of using BM25 to find the top sparse candidates, then using a dense re-ranking approach will be able to keep most of the sparse retrieval efficiency while recovering much of the semantic retrieval quality obtainable from a dense retrieval. Generally, there exist use cases of cascade architecture, as the solution trades off between latency of response time and maintaining a high level of semantic precision; therefore, it creates an effective compromise between using sparse-only and dense-only methodologies [4].

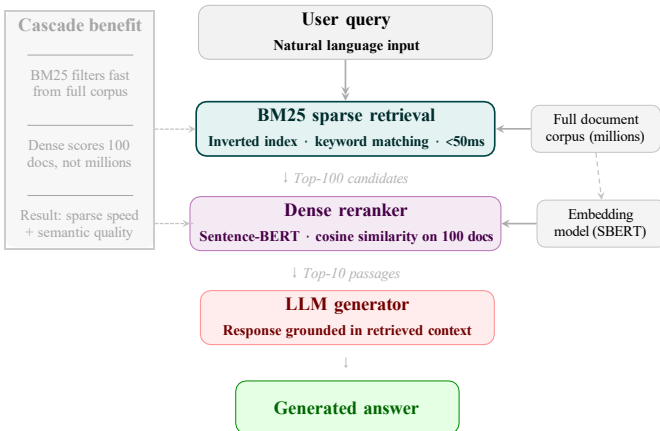


Fig. 2. Cascade retrieval architecture: BM25 performs fast first-stage filtering over the full corpus, reducing candidates to Top-100. The dense reranker then applies semantic scoring only on these candidates, yielding Top-10 high-quality passages for LLM generation.

As illustrated in Figure 2, the cascade architecture...

The continually learning capabilities can be utilized to further improve adaptive routing in the future by recording per query retrieval results and then re-training the classifier based on the feedback received. In scenarios where no ground truth relevance is available, weak signals such as the correction of the result returned, abandoning of the search process, or posing of further clarifying queries act as a form of weak supervision for updating the boundaries of the routing process.

V. CONCLUSION

This paper presents a conceptual comparison of vector-based dense retrieval and vectorless sparse retrieval within Retrieval-Augmented Generation pipelines. Sparse methods deliver exceptional computational efficiency and keyword precision for structured domains and latency-sensitive deployments. Dense embedding-based retrieval enables semantic matching essential for paraphrased and multilingual queries. Neither approach dominates universally; optimal choice depends on query distribution, corpus characteristics, and infrastructure constraints.

Hybrid and adaptive retrieval architectures that combine both signals represent the most robust solution for general-purpose production RAG systems. Empirical benchmarks consistently show hybrid configurations outperforming individual retrievers across precision, recall, and hallucination metrics, with the infrastructure overhead justified by quality improvements in most application contexts [4].

Future research should focus on learned retriever selection via lightweight classifiers, progressive hybrid indexing for dynamic corpora, and embedding-free graph retrieval for structured domains [9]. Long-term, end-to-end differentiable RAG pipelines jointly optimizing retrieval and generation objectives represent a promising direction for further quality gains [15]. Immediate engineering priorities include deploying query-based routing classifiers and evaluating cascade retrieval architectures in production-grade systems.

Beyond the immediate engineering horizon, the broader challenge of aligning retrieval objectives with human evaluation criteria remains open. Automated retrieval metrics such as NDCG and MRR correlate imperfectly with user satisfaction in deployed conversational systems. Developing richer evaluation protocols that capture not only factual correctness but also response coherence, citation transparency, and user trust calibration will be essential as RAG systems take on increasingly consequential roles in information access and decision support [16].

REFERENCES

- [1] P. Lewis, E. Perez *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [2] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.
- [3] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of EMNLP*, 2019.
- [4] K. Gomez, F. Silberzahn *et al.*, "Blended RAG: Hybrid Sparse–Dense Retrieval for Retrieval-Augmented Generation," *arXiv:2404.07220*, 2024.
- [5] S. Gupta, R. Ranjan *et al.*, "A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions," *arXiv:2410.12837*, 2024.

- [6] V. Karpukhin, B. Oguz *et al.*, “Dense Passage Retrieval for Open-Domain Question Answering,” in *Proceedings of EMNLP*, 2020.
- [7] P. Zhao, H. Zhang *et al.*, “Retrieval-Augmented Generation: A Comprehensive Survey,” *arXiv:2312.10997v1*, 2023.
- [8] H. Peng *et al.*, “Retrieval-Augmented Generation for AI-Generated Content: A Survey,” *arXiv:2402.19473*, 2024.
- [9] J. Maghakian *et al.*, “Embedding-Free RAG,” in *Findings of EMNLP*, 2025.
- [10] C. Xiong *et al.*, “Accelerating Inference of Retrieval-Augmented Generation via Sparse Context Selection,” *arXiv:2405.16178*, 2022.
- [11] K. MoR, “Better Handling Diverse Queries with a Mixture of Retrievers,” *arXiv:2506.15862*, 2025.
- [12] N. Thakur, N. Reimers, A. Rüchle, A. Srivastava, and I. Gurevych, “BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models,” in *Proc. NeurIPS Datasets and Benchmarks Track*, 2021.
- [13] T. Formal, B. Piwowarski, and S. Clinchant, “SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking,” in *Proc. 44th International ACM SIGIR*, pp. 2288–2292, 2021.
- [14] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant, “SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval,” *arXiv:2109.10086*, 2021.
- [15] Y. Gao, Y. Xiong *et al.*, “Retrieval-Augmented Generation for Large Language Models: A Survey,” *arXiv:2312.10997*, 2023.
- [16] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “Ragas: Automated Evaluation of Retrieval Augmented Generation,” in *Proc. EACL (Demo Track)*, pp. 150–158, 2024.