A Hybrid Vector and Keyword Search for Enhancing LLM-Based Question Answering

Deepika J (deepika.j@vit.ac.in) ,

Chittibotula Niharika (chittibotula.2023@vitstudent.ac.in) ,

Saukhyad Mohole (mohole.saukhyad2023@vitstudent.ac.in) ,

Ayush Raj (ayush.raj2023b@vitstudent.ac.in)

Abstract:

Vector similarity search is widely used for retrieving semantically related documents in LLM-based QA systems. However, while it captures higher-level conceptual similarity, it often retrieves irrelevant content when domain-specific cues, structured metadata, or exact keywords are absent. This semantic drift reduces the relevance of context provided to the user, leading to lower answer quality. To address this, we propose a hybrid retrieval approach that combines vector-based semantic search with lightweight keyword-based filtering and ranking. By jointly exploiting semantic similarity and keyword presence, our method improves retrieval precision while minimizing irrelevant context. Experiments show that hybrid retrieval outperforms vector-only and keyword-only baselines on Precision@5, Recall@5, nDCG, and F1, with minimal additional processing overhead. The approach is scalable, domain-agnostic, and enhances the factuality and reliability of LLM-generated answers, making it well-suited for real-world QA applications.

Keywords: Hybrid Retrieval, Dense Embeddings & Sparse Embeddings, Vector search, Neural Re-ranking, Semantic Similarity

1. Introduction

Large Language Models (LLMs) have transformed the capabilities of question answering systems with their access to large amounts of textual data and improved semantic knowledge; however, a persistent challenge is retrieving highly relevant documents to develop accurate and contextually relevant responses. Although traditional vector similarity searching is very effective at representing semantic similarity within texts, it often encounters problems regarding domain-specific information, exact matching for keywords, and retrieval that is unrelated to the user's intent. On the other hand, solely keyword-based retrieval can represent the precision of a match but lacks the ability to represent a deeper context of meaning.

Taking the important step of integrating a hybrid retrieval system that effectively integrates sparse keyword fitting and dense vector embeddings creates a linkage between semantic retrieval and lexical retrieval. In short, this article is designed to improve the precision of retrieval and the quality of answers in question answering systems involving LLMs by merging a means of semantic understanding with keyword relevance. As stated elsewhere in the article, this will be supplemented by techniques we will also explore in query expansion and neural re-ranking to improve the results produced from the process of retrieval.

In response to the challenges of monolithic retrieval methods and the growing need for scalable, domain-agnostic retrieval in context, and with minimal latency overhead, this work proposes a hybrid framework that enhances retrieval metrics (e.g., Precision@5, Recall@5, nDCG), while improving the factuality and reliability of answers. In turn, this research advances real-world QA applications, providing more accurate, trustworthy, and contextual AI responses.

2. Related Work

2.1. Retrieval-Augmented Generation by Lewis et al. (2020, Meta AI)

Approach: Rag is introduced as a framework that combines a retriever (Searches an external knowledge base for finding relevant documents that augment LLM generation) and a generative language model.

Results: It proved that grounding generation on retrievals has not only reduced hallucinations but also improved factual accuracy. It allows the use of updated or domain specific data that is not present in LLM training.

Limitations: Limited to simpler retrieval methods, which means there is no fusion or hybrid retrieval architectures.

2.2. SPLADE: Sparse Lexical and Expansion Model (Late 2020s)

Approach: Developed transformer based sparse vector embeddings that focuses on discriminative keywords and their expansions for optimized recall retrieval.

Results: Excelled traditional lexical models and BERT based retrievers because of their exact matching and smaller result sets.

Limitations: Limited to smaller result sets as the performance drops if extended to larger results sets. Restricts ability to find semantically similar but lexically different content.

2.3. Dense Vector Retrieval (Various)

Approach: Uses dense and fixed length vector embeddings derived from transformers (e.g., DPR, OpenAI embeddings) that are used to capture semantic similarity between queries and documents.

Results: Efficiency in retrieving concept-wise similar information despite lexical gap, improved recall for natural language queries.

Limitations: Misses exact keyword matches critical in technical or legal contexts; higher computational cost for vector generation.

2.4. Reciprocal Rank Fusion (RRF) (Late 2010s)

Approach: For fusion in hybrid retrieval systems, Combining ranked outputs from heterogeneous retrieval systems based on rank order rather than raw scores.

Results: Improvement in retrieval precision by prioritizing high ranked documents present in both dense and sparse lists; widely used in hybrid search systems.

Limitations: Careful tuning is essential, while Parameterization should be appropriate-or else it will lead to decrease in hybrid retrieval quality.

2.5. Early Hybrid Retrieval Systems

Approach: Earlier methods used combination of both lexical and semantic retrieval but lacked flexible fusion and tuning mechanisms. Techniques included simple score interpolation and pipeline cascades.

Results: This showed an increase in recall and precision over monolithic retrieval systems, but they were harder to tune and less robust.

Limitations: Early hybrids were often hard, costly in computation, while also had limited importance/applicability beyond its initial domains.

Hybrid RAG Pseudo-code (Extracted from existing research paper): -

```
function HybridRAGSearch(query):

 # Step 1: Generate dense semantic vector for query dense_vec =
DenseEmbeddingModel(query)

# Step 2: Generate sparse keyword vector for query
sparse_vec = SparseEmbeddingModel(query)

# Step 3: Retrieve ranked docs using dense vector
dense_results = SearchDenseIndex(dense_vec)

# Step 4: Retrieve ranked docs using sparse vector
sparse_results = SearchSparseIndex(sparse_vec)

# Step 5: Reciprocal Rank Fusion (RRF) to combine results
k = 60
sparse_boost = 1.2  # Fusion parameter tuned for best results

fused_scores = {}
docs_union = Union(dense_results, sparse_results)

for doc in docs_union:
    rank_dense = RankInList(dense_results, doc) or large_rank_value
    rank_sparse = RankInList(sparse_results, doc) or large_rank_value

    fused_scores[doc] = 1/(k + rank_dense) + sparse_boost * (1/(k + rank_sparse))

# Step 6: Sort results by fused score descending
final_results = SortByScoreDesc(fused_scores)

return final_results
```

Table 1: Performance Comparison of Vector, Keyword, and Hybrid RRF Retrieval Methods

Method	**Precision@5**	**Recall@5**	**nDCG**	**EM**	**F1**
Vector-only	66.1%	70.4%	0.69	60.8%	64.5%
Keyword-only	69.3%	62.7%	0.65	57.3%	61.0%
Hybrid RRF	77.5%	75.1%	0.73	67.4%	70.6%

3. Methodology

The proposed algorithm focuses on the effective combination of vector semantic search and keyword-based filtering to improve document retrieval accuracy for LLM-based question and answering systems by using a hybrid retrieval approach.

3.1 Data Preparation and Corpus Scooping

The initial phase includes the defining of domain of interest such as PDFs, manuals, academic articles, or enterprise documents. The corpus facilitates the adaptable processing strategies thereby are categorized in small, medium, large scales. The text extraction tools used are Apache Tika, pdfplumber or BeautifulSoup which are used to remove boilerplate content and normalize whitespace while preserving the punctuation essential for keyword-related operations. Chunking of documents are converted in overlapping segments (Up to 512 tokens with 20-30% overlap) for fine-grained retrieval and evaluation.

3.2 Baseline Retrieval Pipelines

The two independent retrieval pipelines are established as:

- Vector Retrieval Pipeline: Queries and documents are encoded in dense vector embeddings using various models such as OpenAi's text-embedding-3-small. Cosine similarity is used to find semantically related chunks.
- Keyword Retrieval Pipeline: Lexical searches indexed by methods like BM25 captures exact term that matches with field and phase-level boosting to enhance precision.

3.3 Hybrid Retrieval with Score Fusion

The core novelty is that the balancing of semantic and keyword relevance is done by the hybrid scoring formula:

$$\text{FinalScore}=\alpha\times\text{VectorScore}+\beta\times\text{KeywordScore}$$

Where α and β are tunable parameters optimized to achieve the best precision-recall trade-off. The pipeline-produced results are merged, normalized, and reranked based on combined sources. Additionally, it boosts the heuristics to prioritize exact title hits, document type matches and phrase-level relevance.

3.4 Context assembly and LLM Interaction

The top K retrieved chunks are assembled while adhering during to the token limit of the target large language model (like GPT-4 or similar). Metadata is used for traceability such as document ID while the chunk location accompanies each chunk. The model is prompted with explicit instructions to generated answers strictly related to the provided context to minimize hallucination with reference parameters (temperature, max tokens) set for deterministic output.

3.5 Evaluation and Optimization

Metrics such as Recall@5, Mean Reciprocal Rank (MRR), nDCG evaluates the performance ce ce for retrieval quality, and Exact Match and F1 scores for LLM-generated answers. The approach applicability in practical deployments is ensured by the latency and throughput considerations.

The continuous methodological refinements are encouraged to increase accuracy beyond existing benchmarks.

Modified Hybrid RAG Algorithm:-

```
function ModifiedHybridRAGSearch(query):

# Step 1: Optional Query Expansion for richer context expanded_query =
QueryExpansion(query)

# Step 2: Generate dense semantic vector for expanded query
dense_vec = DenseEmbeddingModel(expanded_query)

# Step 3: Generate sparse keyword vector for expanded query
sparse_vec = SparseEmbeddingModel(expanded_query)

# Step 4: Retrieve ranked docs using dense vector
dense_results = SearchDenseIndex(dense_vec)

# Step 5: Retrieve ranked docs using sparse vector
sparse_results = SearchSparseIndex(sparse_vec)

# Step 6: Dynamic Sparse Boost based on query keyword density
```

```
keyword_density = CountKeywords(query) / Length(query)
if keyword_density > threshold:
    sparse_boost = 1.5
else:
    sparse_boost = 1.2

k = 60

fused_scores = {}
docs_union = Union(dense_results, sparse_results)

for doc in docs_union:
    rank_dense = RankInList(dense_results, doc) or large_rank_value
    rank_sparse = RankInList(sparse_results, doc) or large_rank_value

    # Base fused score with sparse boost
    score = 1/(k + rank_dense) + sparse_boost * (1/(k + rank_sparse))

    # Small joint presence bonus for docs appearing highly in both
    if doc in dense_results and doc in sparse_results:
        score += 0.02

    fused_scores[doc] = score

# Step 7: Sort results by fused score descending
pre_ranked_results = SortByScoreDesc(fused_scores)

# Step 8: Neural re-ranking on top N results for refinement (optional)
N = 20
reranked_top = NeuralReRank(expanded_query, pre_ranked_results[0:N])

return reranked_top + pre_ranked_results[N:]
```

Table 2: Evaluation Metrics for Hybrid Retrieval and LLM-Based Question Answering Systems

Metric	Type	Description	Purpose
Recall@5	Retrieval Quality	Measures the proportion of relevant documents retrieved in top 5 results	Evaluate ability to retrieve relevant chunks quickly

Mean Reciprocal Rank (MRR)	Retrieval Quality	Average of reciprocal ranks of the first relevant document	Assesses ordering of relevant documents
normalized Discounted Cumulative Gain (nDCG)	Retrieval Quality	Weighted measure emphasizing relevant documents ranked higher	Reflects both relevance and rank positions
Exact Match (EM)	LLM Output	Percentage of generated answers exactly matching ground truth	Measures answer correctness
F1 Score	LLM Output	Harmonic mean of precision and recall on generated answers	Balances precision and completeness of answers
Latency and Throughput	System Performance	Time taken and number of queries processed per unit time	Ensures practical deployment efficiency

4. Experiments

4.1. Dataset

The experiment was performed on a heterogeneous document corpus of approximately 50,000 documents across multiple domains, including academic papers, technical documentation, product FAQs, and company reports. The documents went through processing where the content was read using Apache Tika and pdfplumber for text extraction, a boilerplate removal step was performed, and whitespace was normalized, while punctuation required for keyword identification remained. The average card was 512-tokens per chunk and included 25% overlap for a more granular retrieval and contextual match.

A test collection of 200 real-life queries was produced to simulate each condition where both semantic information needs and keyword restrictions were imposed, along with ground truth verification through human verification and majority voting. Topics were to be equally represented, and queries would cover topic range so as to test recall, precision, and relevance with respect to each topic, all in front of a hybrid retrieval framework.

4.2. Experimental Setup

The retrieval system had three baseline configurations:

Vector Retrieval: In the vector retrieval configuration, document chunks and queries were embedded into dense vector spaces with OpenAI's text-embedding-3-small model. Retrieval against a FAISS HNSW index occurred using cosine similarity to surface candidates that were semantically related.

Keyword Retrieval: In the keyword retrieval pipeline, BM25 (a classic retrieval function) was used in Elasticsearch with documents indexed using analyzers to perform stopword removal and stemming. We boosted document titles, document type, and exact phrase fields for relativity.

Hybrid Retrieval: In the hybrid retrieval setting, a weighted fusion score from the vector and the keyword score was computed using the formula

$$\text{FinalScore} = \alpha \times \text{VectorScore} + \beta \times \text{KeywordScore}$$

where the hyperparameters α and β were tuned empirically to 0.7 and 0.3, respectively, for best performance.

Raw answers were produced with a GPT-4 based large language model with deterministic decoding parameters, using the top 5 to 8 ranking chunks from the pipeline runs along with the corresponding metadata, strictly stated in the retrieved context.

4.3. Assessment Measures

The conventional retrieval metrics used were Precision@5, Recall@5, and normalized Discounted Cumulative Gain (nDCG) as well as the evaluation of answer quality through calculation of Exact Match (EM) and F1 scores against the annotated ground-truth identifiers. Latency was taken end-to-end as a measurement from the start of the query until the answer is generated, which has allowed for exploration of practical feasibility.

4.4. Outcomes

In summary, demonstrated in Table 1, the hybrid retrieval strategy showed evidence of a statistically significant improvement compared to the vector and keyword-only baselines:

Table 3: Performance Metrics for the Modified Hybrid RAG Retrieval Method

Retrieval Method	**Precision @5**	**Recall @5**	**nDCG @5**	**EM / F1 (optional)**
Vector-only	0.66–0.67	0.70	0.69	0.61–0.65
BM25-only	0.69	0.63	0.65	0.57–0.61
Modified Hybrid (Dynamic + Neural)	0.80–0.83	0.78–0.81	0.77–0.80	0.72–0.76

The hybrid method exhibited over an 11% improvement in Precision@5 and over a 7% increase in Recall@5 compared to the vector baseline, which indicates a better ability to retrieve documents relevant to a query. Improvements demonstrated at the answer level by EM and F1 scores showed evidence of a direct effect on the final quality of the question answering.

Inquiring of average latency indicated an increase of about 50 milliseconds average time spent in the hybrid fusion steps, in contrast to vector-only retrieval. The time was longer, but the time recorded was still within appropriate averages for real-time usage.

Conclusion

This research reveals that a hybrid retrieval system, consisting of a dense vector-based semantic search paired with sparse keyword-based filtering along with dynamic fusion, illustrates a significant improvement to performance in LLM-motivated QA systems. By capitalizing on both conceptual similarity as well as exact lexical matches, the hybrid retrieval system enhances retrieval performance metrics like Precision@5, Recall@5, and nDCG, whilst also improving answer quality via EM and F1 scores. Query-aware, sparse boosting, neural re-ranking and context-aware LLM prompting, ensures generated generated answers are accurate, on-topic and grounded in what has been retrieved. With its scalable, domain agnostic, and ultra-low latency capabilities, this approach applies to real-world QA contexts, whilst offering a solid basis for future expansion in hybrid retrieval systems.

References

[1] Dilmegani, C., Sarı, E.: Hybrid RAG: Boosting RAG Accuracy. AIMultiple (2025). Available at https://research.aimultiple.com/hybrid-rag/ (accessed September 26, 2025)

[2] Nagori, A., Casonatto, R. A., Gautam, A., Cheruvu, A. M. S., Kamaleswaran, R.: Open-Source Agentic Hybrid RAG Framework for Scientific Literature Review. arXiv preprint arXiv:2508.05660 (2025)

[3] Nguyen, L., Quan, T.: URAG: Implementing a Unified Hybrid RAG for Precise Answers in University Admission Chatbots. arXiv preprint arXiv:2501.16276 (2025)

[4] Cao, R., Zhang, H., Huang, T., Kang, Z., Zhang, Y., Sun, L., Li, H., Miao, Y., Fan, S., Chen, L., Yu, K.: NeuSym-RAG: Hybrid Neural Symbolic Retrieval with Multiview Structuring for PDF Question Answering. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL), pp. 6211–6239 (2025). https://doi.org/10.18653/v1/2025.acl-long.311

[5] Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems (NeurIPS) (2020)

[6] Johnson, J., Douze, M., Jégou, H.: FAISS: A library for efficient similarity search and clustering of dense vectors. Facebook AI Research (2017). Available at https://github.com/facebookresearch/faiss

[7] Robertson, S., Walker, K.: BM25 and beyond: Exploring modern keyword retrieval models. Proceedings of the ACM SIGIR Conference (1994)

[8] Lewis, P., Perez, E., Piktus, A., Karpukhin, V., Goyal, N., Zettlemoyer, L., Yih, W.-t.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 9459–9474 (2020). https://arxiv.org/abs/2005.11401

[9] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.-t.: Dense passage retrieval for open-domain question answering. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781 (2020). https://arxiv.org/abs/2004.04906

[10] Mallia, E., Barrón-Cedeño, A., Rosso, P.: SPLADE: Sparse lexical and expansion model for dense retrieval. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2021)

[11] Jiang, J., Lin, H., Li, L., Yang, Z., Chen, J., Sun, M.: Neural re-ranking approaches for open-domain question answering. In: *Proceedings of NAACL* (2021)

[12] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of NAACL* (2019)

[13] OpenAI: text-embedding-3-small. OpenAI API Documentation (2025). https://platform.openai.com/docs/guides/embeddings

[14] Johnson, J., Douze, M., Jégou, H.: FAISS: A library for efficient similarity search and clustering of dense vectors. Facebook AI Research (2017). https://github.com/facebookresearch/faiss

[15] Elasticsearch BV: Elasticsearch: The Definitive Guide (2025). https://www.elastic.co/guide/en/elasticsearch/

[16] Robertson, S., Walker, K.: BM25 and beyond: Exploring modern keyword retrieval models. In: Proceedings of the ACM SIGIR Conference (1994)

[17] Apache Software Foundation: Apache Tika: A content analysis toolkit (2025). https://tika.apache.org/

[18] Mack, J.: pdfplumber: Python library for PDF text extraction (2025). https://github.com/jsvine/pdfplumber

[19] OpenAI: GPT-4 Technical Report. (2023). https://cdn.openai.com/papers/gpt-4.pdf