Before Proceeding, do the literature survey (current year – 5 years) and give the table based on the study.

Columns could be the following in the table

- Author(s), Year
- Title of the paper
- Methodology
- Key Findings
- Relevance to Our Work

Prepare the gap identified and feasible solutions based on the study. This will give you confidence that you are on the right track of extension works based on the existing state of art in the hypothesis chosen.

Recommended implementation phases — Hybrid Vector + Keyword Search

Phase 1 — Problem Scoping & Data Preparation

1.1 Scope & Corpus
- Define domain: PDFs, manuals, academic articles, enterprise docs.
- Corpus size categories: Small (≤10k), Medium (10k–500k), Large (≥1M).

1.2 Text Extraction
- Tools: Apache Tika / pdfplumber / BeautifulSoup.
- Remove boilerplate, normalize whitespace, retain punctuation.

1.3 Chunking Strategy
- Chunk size ~512 tokens, 20–30% overlap.
- Store metadata: chunk_id, doc_id, file_type, section, timestamp.

1.4 Preprocessing for Keyword Index
- Tokenization, lowercase, stopword removal, stemming/lemmatization.
- Keep raw text for embeddings.

1.5 Benchmark Queries & Ground Truth
- Create semantic and keyword-sensitive queries with gold-standard answers.

Phase 2 — Baseline Systems Setup

Vector Retrieval Pipeline:
- Embedding Models: all-MiniLM-L6-v2 (384 dims), all-mpnet-base-v2 (768 dims).
- Vector Index: FAISS HNSW, Milvus, Pinecone.
- HNSW Params: M=32, efConstruction=200–400, efSearch=128.
- Retrieval: cosine similarity, topN=100.

Keyword Retrieval Pipeline:
- Elasticsearch BM25 ranking.
- Analyzer: lowercase, stopword removal, stemming.
- Field mapping boosts: title^3, doc_type^2, body^1.
- BM25 Params: k1≈1.2, b≈0.75.
- Retrieval: topN=100.

Baseline Evaluation:
- Metrics: Precision, Recall, nDCG, MRR.

- Vector-only weakness: semantically related but wrong type.
- Keyword-only weakness: strict matches, misses synonyms.

Phase 3 — Hybrid Retrieval Engine

3.1 Fusion Strategies:
- Hard Filtering: discard candidates without required keywords/doc_type.
- Soft Reranking: Weighted score (α=0.7 initially).

3.2 Candidate Set:
- Retrieve topN=100 from both indices, merge, normalize scores.

3.3 Boosting Rules:
- Title match, exact phrase match, doc_type match → higher weight.

3.4 Output:
- TopK=3–8 chunks (fit LLM context window).

Phase 4 — Integration with LLM

4.1 Context Assembly:
- Attach metadata, ensure token budget compliance.

4.2 Prompting Strategy:
- Constrained instruction: “Answer only using provided context.”

4.3 LLM Settings:
- Temperature=0.0, fixed max_tokens.

4.4 Evaluation of LLM Output:
- Factual accuracy, hallucination rate, response depth (1–5).

Phase 5 — Evaluation & Benchmarking

5.1 Retrieval Metrics:
- Precision, Recall, nDCG, MRR.

5.2 LLM Answer Metrics:
- Exact Match, F1 overlap, BERTScore, human evaluation.

5.3 Experimental Conditions:
- Compare vector-only, keyword-only, hybrid (α=0.5–0.9).

5.4 Statistical Validation:
- Paired tests (Wilcoxon).

5.5 Sensitivity Studies:
- Chunk size variations, topN variations, embedding model comparisons.

Phase 6 — Optimization & Scalability

6.1 Vector Index Tuning:
- Adjust efSearch for recall/latency tradeoff.
- IVF+PQ for memory savings, GPU acceleration.

6.2 Keyword Index Tuning:
- Elasticsearch sharding for >1M docs.
- Domain-specific synonyms.

6.3 Parallelization & Caching:
- Parallel vector + keyword retrieval, Redis caching.

6.4 Practical Parameters:
- Small (≤10k): FAISS + single ES node.
- Medium (≤500k): FAISS multi-thread + 2–3 ES shards.
- Large (≥1M): Distributed Milvus + ES cluster.

Research Title: Hybrid Vector + Keyword Search

1. Topic

Hybrid Vector and Keyword Search for Improving LLM-based Question Answering Systems

2. Problem Statement

Large Language Model (LLM)-based QA systems often rely on vector search to retrieve semantically relevant documents. While effective, this approach can return documents that are semantically related but content-irrelevant, missing explicit cues such as keywords, document types, or required phrases. This leads to reduced precision in context retrieval and ultimately degrades the accuracy and reliability of generated answers.

3. Proposed Solution

We propose a Hybrid Retrieval Approach that integrates:
- Vector-based Search → Captures semantic similarity and broader meaning.
- Keyword Filtering / Ranking Layer → Ensures explicit keyword presence (e.g., PDF Type A vs Type B, mandatory terms).
- Fusion Mechanism → Combines both scores into a final ranking to balance semantic relevance with keyword precision.

This ensures higher retrieval accuracy and more contextually precise answers from the LLM.

4. Objectives

1. Improve retrieval precision by combining semantic similarity with keyword presence.
2. Reduce the inclusion of irrelevant but semantically related documents.
3. Evaluate the impact of hybrid retrieval on LLM QA accuracy.
4. Ensure minimal computational overhead for real-time applications.

5. Data Sources

- Corpus Documents: Academic papers, PDFs, FAQs, product manuals, or domain-specific datasets.
- Queries: User-generated or benchmark QA datasets (e.g., Natural Questions, domain-curated queries).

- Metadata: Document type, keywords, and structured fields for filtering.

6. Technology Stack

- Vector Database: FAISS, Pinecone, Milvus, or Weaviate.
- Embedding Models: OpenAI embeddings, Sentence-Transformers, or domain-specific models.
- Keyword Indexing: Elasticsearch, PostgreSQL full-text search, or in-memory inverted index.

- Fusion & Ranking: Python-based re-ranker combining vector and keyword scores.
- Evaluation: Precision, Recall, nDCG, EM/F1 for QA.

7. Workflow

Step 1: Data Preprocessing
- Chunk documents into 200–500 tokens.
- Compute embeddings and index in vector DB.
- Extract/store keywords and metadata.

Step 2: Vector Retrieval
- Embed query → retrieve top-N candidates by cosine similarity.

Step 3: Keyword Filtering/Scoring
- Extract keywords from query (manual/automated).
- Compute keyword presence score for candidate documents.

Step 4: Fusion Layer
- Combine scores: FinalScore = α × VectorScore + β × KeywordScore
- Re-rank candidates and select top-K for LLM context.

Step 5: QA Generation
- Provide re-ranked top-K documents as context to LLM.
- Generate answer with reduced noise.

8. Expected Outputs

- More precise document retrieval (keyword-sensitive).
- Improved QA performance (Exact Match, F1, reduced hallucination).
- Demonstration of trade-off between α (semantic weight) and β (keyword weight).
- Lightweight system with minimal added latency.

9. Novelty in Approach

1. Hybrid Retrieval: Combines semantic understanding (vectors) with explicit keyword matching.
2. Lightweight Post-processing: Keyword layer adds minimal overhead, making it scalable.
3. Improved QA Depth: Better context leads to more factual and relevant LLM responses.
4. Generalizable: Can be applied to any vector DB and adapted to different domains.

10. Example Experimental Results

Method	Precision@5	Recall@5	nDCG	EM	F1
Vector-only	65%	72%	0.68	58%	62%
Keyword-only	70%	60%	0.64	55%	59%

Hybrid (α=0.7, β=0.3)	78%	75%	0.74	66%	70%
Hybrid (α=0.5, β=0.5)	75%	73%	0.71	63%	67%