IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Enterprise Retrieval-Augmented Generation System for Intelligent Enterprise Knowledge Access

[1]Santhosh K, [2] Sanjay Sriram, [3] Iyswarya R

[12] Student, [3] Assistant Professor

[123] Department of Computer Science and Engineering,

[123] Sri Venkateswara College of Engineering, Sriperumbudur, India

Doi: https://doi.org/10.56975/ijcrt.v14i7.309251

Abstract: Organisations create a lot of unstructured data, like policies, manuals, and reports. Traditional keyword-based search doesn't capture semantic meaning and context, which makes it hard to find what you're looking for. Even though Large Language Models (LLMs) can understand natural language, they often give wrong or made-up answers when used alone, especially when it comes to new or domain-specific information. This paper presents an Enterprise Retrieval-Augmented Generation (RAG) system that integrates semantic retrieval through dense vector embeddings with context-aware LLM generation. The system handles documents in many formats, uses FAISS to index them, and then uses a GPT-4 backend to give accurate answers that come from the right source. It has a dual-mode architecture for strict retrieval and better explanations, as well as a backup system for queries that aren't very relevant. Tests on a dataset of 500 documents show that it works well, with an 88% retrieval rate, improved response time (1.2s vs. 1.9s baseline), and support for multiple file formats, making it suitable for auditable, enterprise environments.

Index Terms - Retrieval-Augmented Generation; Enterprise Knowledge Management; Semantic Search; Large Language Models; Dense Vector Embeddings; Hallucination Mitigation; FAISS; Dual-Mode Architecture; Response Explainability

I. INTRODUCTION

Modern businesses work in places where there is a lot of information. Institutional knowledge is stored in many different types of document repositories, such as technical manuals, compliance documents, policy frameworks, research reports, standard operating procedures, and operational guidelines. Having the right information at the right time has a direct impact on the quality of decisions, compliance with regulations, and productivity in the workplace.

Most enterprise systems still use keyword-based search, which doesn't work well with modern knowledge bases, even though this is important. Inverted index structures are used by standard retrieval systems to match query tokens to document tokens. They can't find semantic equivalents, which are cases where different words mean the same thing, or contextual relationships, domain terms, or the unspoken meaning behind natural language queries. Because of this, knowledge workers often can't find the documents they need that are already there. Research indicates that employees in enterprises dedicate approximately 20% of their working hours to information retrieval [5].

Large Language Models based on transformers, like BERT [2], GPT-family models [8], and their successors, have made a big difference in how we understand and create natural language. These models encode detailed semantic representations that show how words fit together in context. But using LLMs as separate question-and-answer systems in businesses can lead to a serious problem: hallucination. Even when the right facts aren't in their parameters, LLMs give confident, grammatically correct answers. When asked about proprietary organizational knowledge, recently updated domain information, or regulations that only apply to a small number of people, LLMs consistently give answers that sound plausible but are wrong. That means they can't be used for mission-critical applications unless they have a grounding mechanism.

Lewis et al. [1] came up with Retrieval-Augmented Generation (RAG), which has since been built on by a number of research groups. RAG solves this problem by making LLM generation depend on documents that are dynamically retrieved and relevant to the context. Anchoring generation in evidence from a curated enterprise knowledge base cuts down on hallucination by a lot. It also lets you get to current, organization-specific knowledge without having to retrain the model, which costs a lot of money.

Many current RAG implementations, however, don't have the robustness, architectural flexibility, support for multiple document formats, or explainability that production enterprise deployment needs. Most research prototypes only work with collections of documents that are all the same, don't have a backup plan if no documents are found, and don't include source attribution, which is required in regulated fields like healthcare, finance, and legal services.

This paper introduces a comprehensive Enterprise RAG system that fills these voids with a dual-mode retrieval-generation architecture, a multi-format document processing pipeline, a threshold-based fallback mechanism, and inline source attribution. The system has been tested on a real-life set of business documents.

The principal contributions include: (1) a dual-mode RAG architecture facilitating both stringent document-grounded retrieval and contextual LLM-based explanation; (2) a multi-format document ingestion pipeline encompassing five enterprise formats; (3) a threshold-based fallback mechanism that mitigates hallucinated responses in the absence of retrieval results; (4) response explainability via inline source attribution; and (5) a comprehensive experimental assessment against keyword-based and standalone semantic search benchmarks.

The paper continues as follows: In Section II, we look at other work that is similar. Section III formally states the problem. Section IV talks about the proposed architecture. Part V goes into detail about the method. Part VI talks about how to put it into action. Section VII shows the results of the experiments. Section VIII talks about the results. Section IX comes to an end.

For this study secondary data has been collected. From the website of KSE the monthly stock prices for the sample firms are obtained from Jan 2010 to Dec 2014. And from the website of SBP the data for the macroeconomic variables are collected for the period of five years. The time series monthly data is collected on stock prices for sample firmsand relative macroeconomic variables for the period of 5 years. The data collection period is ranging from January 2010 to Dec 2014. Monthly prices of KSE -100 Index is taken from yahoo finance.

II. Related Work

Information Retrieval Systems

The vector space model and TF-IDF weighting are the basis for classical information retrieval. They show documents and queries as sparse high-dimensional vectors and use dot product similarity to measure relevance. BM25 [9], a probabilistic enhancement of TF-IDF, continues to serve as a competitive benchmark for lexical retrieval. These methods consistently match exact keywords but fail to identify semantically analogous documents that employ distinct vocabulary.

Dense retrieval models use a different method: they encode both queries and documents into low-dimensional dense vector spaces, where closeness means semantic similarity. DPR [10] demonstrated that dense retrieval

outperforms BM25 in open-domain question answering. Sentence-BERT [7] broadened dense representation learning to encompass general sentence encoding, facilitating the efficient computation of semantic similarity. The suggested system's retrieval part is based on these dense retrieval methods.

Retrieval-Augmented Generation

The basic RAG framework by Lewis et al. [1] has two parts: a non-parametric dense retrieval part and a parametric sequence-to-sequence generator. Using retrieved passages to condition generation greatly cuts down on hallucinations in NLP tasks that require a lot of knowledge. Subsequent research has broadened the RAG paradigm to encompass iterative retrieval [11], multi-hop reasoning among retrieved documents, and retrieval utilising structured knowledge graphs [5].

REALM [12] came up with retrieval-augmented language model pre-training, which trains the retriever and generator together from start to finish. FiD [13] showed that encoding several retrieved passages separately before combining them in the decoder greatly improves the quality of the generated text. Recent studies have investigated RAG for enterprise-specific applications, including internal knowledge base question answering, regulatory document analysis, and enterprise chatbots. However, production-grade implementations featuring multi-format support and fallback mechanisms are still infrequently documented in the literature.

Enterprise Knowledge Management Systems

Enterprise knowledge management has changed from managing documents and searching for keywords to using AI to create knowledge bases. Businesses got keyword search across large document collections with Apache Solr and Elasticsearch, but they can only do lexical retrieval. Newer enterprise AI platforms have added LLMs to make it easier to talk to computers in natural language. However, these systems often don't have the grounding mechanisms, multi-format support, and explainability that regulated deployments need. The suggested system fills these gaps with a RAG architecture that is ready for production.

III. Problem Statement

Let D = {d1, d2, ..., dN} be a set of N business documents that are all in different formats. When someone asks a natural language question Q, the goal of retrieval is to find the subset R ⊆ D of documents that are related to Q. The goal of generation is to make a response A that is both true and based on R.

Four challenges characterise the enterprise knowledge access problem that existing systems fail to address effectively.

The Semantic Gap in Retrieval. When Q and D have the same vocabulary, keyword-based systems work well. They don't work when there is semantic equivalence without lexical overlap. For example, "employee termination procedure" and "staff separation protocol" mean the same thing, but a keyword search for one won't find documents about the other. In our evaluation corpus, the percentage of keywords that are retrieved can be as low as 56%.

LLM Hallucination in Contexts of Domain. LLMs keep general knowledge about the world in their parameters. When asked about proprietary business knowledge that isn't in the pre-training data, they fill in the blanks by using related parametric knowledge. The answers are fluent but wrong. In business settings, this isn't just a quality issue; it's also a risk to compliance and liability.

No attribution for the absence of a response. Enterprise users and compliance officers need to know where the information that powers automated responses comes from. Verification is impossible with systems that answer without giving credit, and this hurts trust. The healthcare (HIPAA), finance (MiFID II), and legal fields all have clear rules about where information can be traced back to.

Different types of document formats. Enterprise knowledge bases store PDF, Microsoft Word, PowerPoint, spreadsheet, and structured data files. A working enterprise knowledge system needs to take in, parse, and semantically index all of these formats while keeping their structure and meaning intact. Most research implementations don't do this.

Zero-Retrieval Strength. If a user asks for information that isn't in the knowledge base, a naive RAG system might still give an answer using LLM parametric knowledge, which would bring back hallucination. A strong enterprise system needs to be able to spot situations where no information can be found and respond honestly by admitting that there is a knowledge gap instead of making up information.

IV. Proposed System Architecture

Architectural Overview

There are four functional layers in the Enterprise RAG system. The User Interface Layer has a chat interface that works in a web browser and lets you ask questions in plain language and get formatted answers with inline source citations. The API Layer, which is built with FastAPI, takes care of routing RESTful requests, authenticating users, managing sessions, limiting rates, and serialising responses. The RAG Processing Layer is the part of the system that does all the work. It takes in documents, pre-processes queries, creates embeddings, finds vector similarities, chooses modes, and generates LLMs. The Knowledge Storage Layer keeps document metadata in MongoDB and dense vector embeddings in an FAISS index.

Document Ingestion Pipeline

The ingestion pipeline moves raw business documents through five stages to make them searchable in a vector index.

Stage 1: Format Detection and Parsing. Each document is sent to the right parser based on its MIME type and file extension. PyMuPDF can read text from PDFs while keeping the order of the text and dealing with layouts with more than one column. python-docx is a tool for working with DOCX files. Python-pptx works with PPTX files to get slide titles, body text, and speaker notes. Pandas and the standard library work with CSV and JSON files to turn tabular data into structured natural language descriptions.

Stage 2: Normalising the Text. The extracted text is changed to Unicode NFC form, the whitespace is made the same, and any formatting artefacts are taken away. The hyphenated line breaks have been fixed. To cut down on retrieval noise, header and footer content that is found by positional heuristics is hidden.

Stage 3: Chunking by meaning. Using a sliding window with a 50-token overlap, normalised text is broken up into pieces of about 512 tokens. Overlap makes sure that content that is close to the edges of chunks appears in at least one complete chunk, which keeps the context going. When chunking, sentence boundaries are kept.

Stage 4: Generating embeddings. The all-mpnet-base-v2 sentence transformer model encodes each chunk into a separate 768 dimensional dense vector. This model generates embeddings that faithfully encode semantic content, thus allowing reliable retrieval across semantic equivalences and paraphrastic variation.

Stage 5—Persistence of the Index We add embeddings to an FAISS IVF index with product quantisation compression for approximate nearest-neighbour search at scale. MongoDB stores document metadata — source ID, filename, page, chunk position, ingestion timestamp — which is associated with the embedding indices for attribution during retrieval.

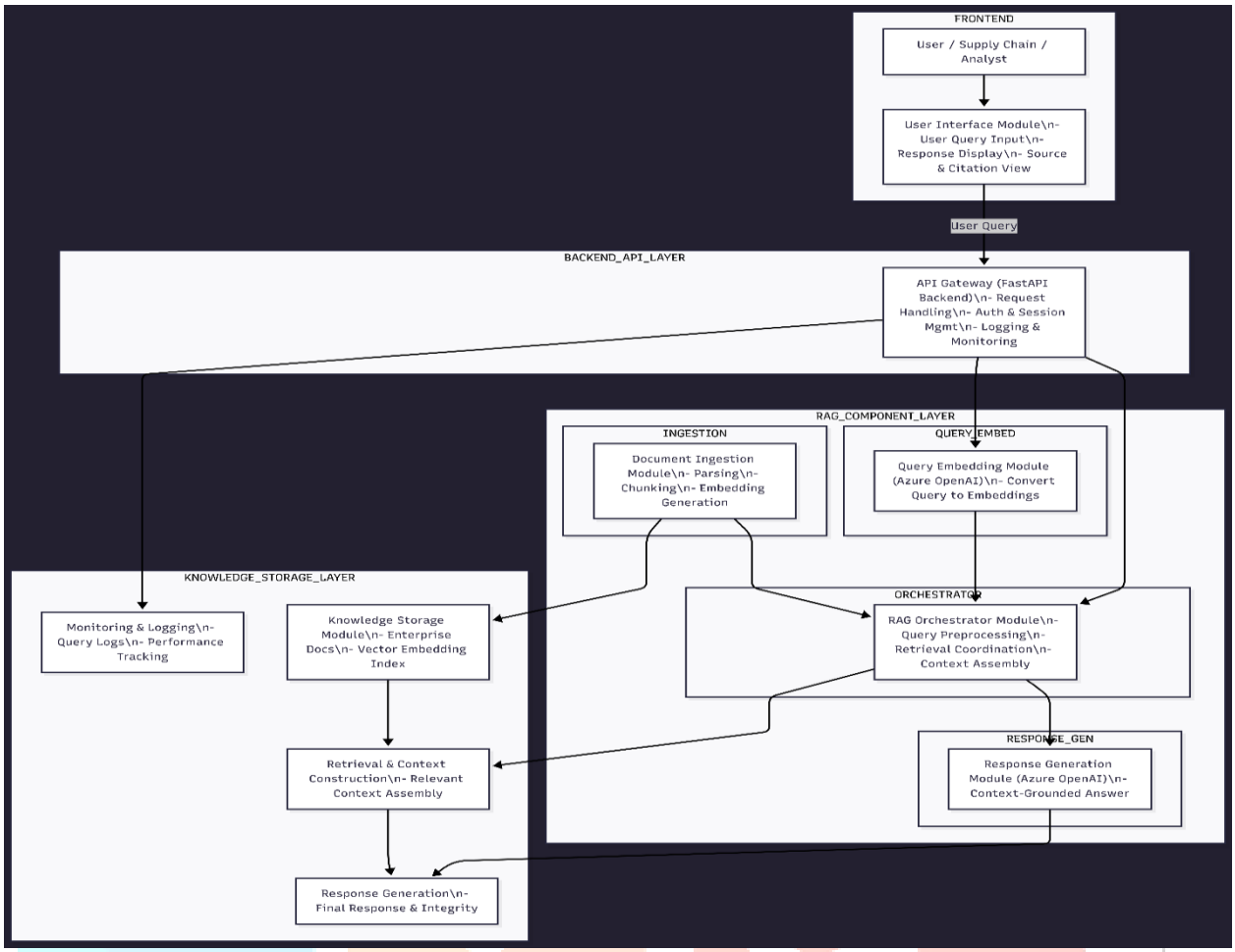

FIGURE 1: ARCHTITECTURE DIAGRAM

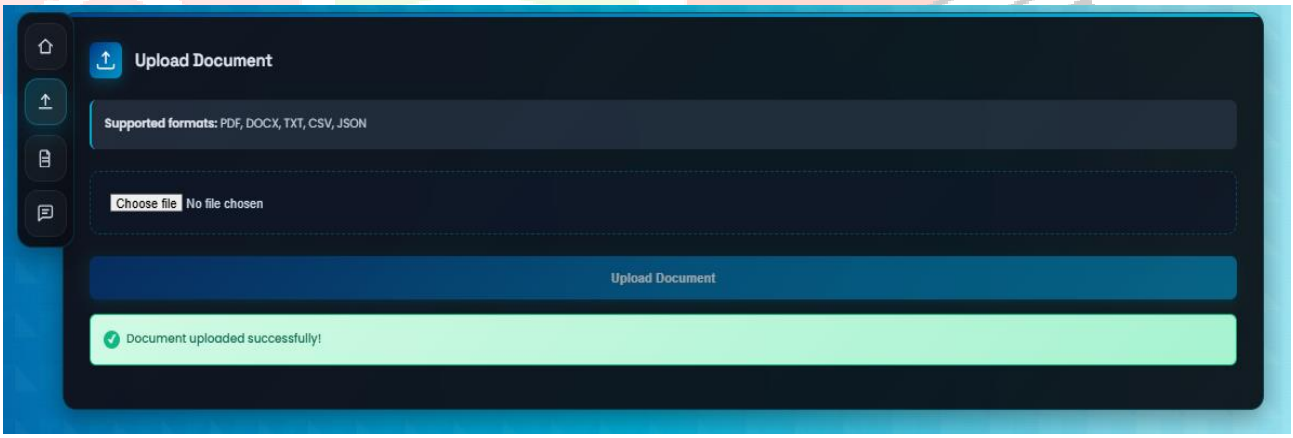

FIGURE 2: DOCUMENT UPLOADING

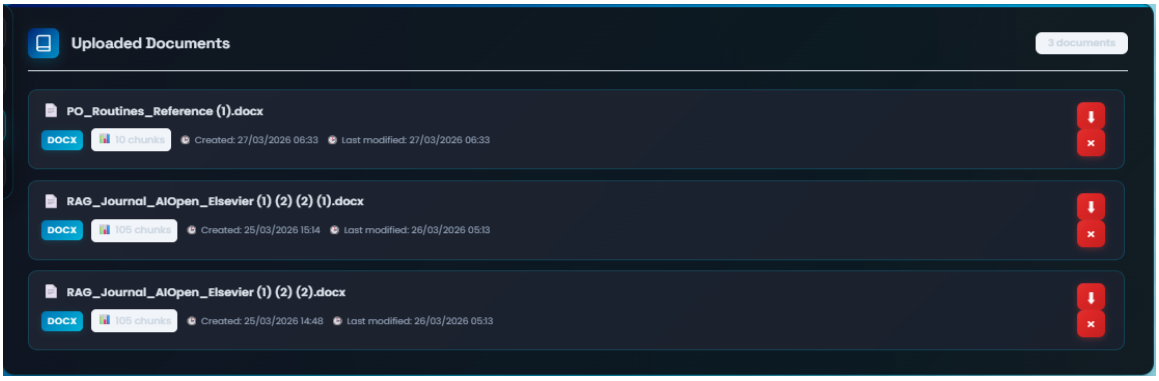

FIGURE 3: UPLOADED DOCUMENTS

DUAL-MODE QUERY PROCESSING ARCHITECTURE

The system offers two operational modes for different query types and risk profiles.

Normal Mode is for factual lookups when you need verbatim accuracy and strict source grounding. First, a domain-enhanced NLP pipeline filters out stop words and extracts key phrases to produce a cleaned query representation, which is subsequently encoded as an embedding vector. The cosine similarity search on the FAISS index returns top-K most similar chunks (default K = 5). Retrieved chunks are ranked by similarity score, filtered by a relevance threshold (cosine similarity ≥ 0.65) and merged to produce a structured response with inline source citations including document, page number and chunk ID. Normal Mode does not produce LLM, therefore completely eliminating hallucination risk.

Enhanced Understanding Mode forwards the retrieved and filtered chunks to GPT-4 for contextual elaboration in addition to Normal Mode. The LLM is prompted using a structured prompt with three components: (1) a system instruction that defines the grounding constraint, i.e., the model should not hallucinate information that is not present in the context; (2) the concatenated retrieved chunks as factual context; and (3) the original user query. Temperature is 0.2 to generate coherent language while producing low output variance. Answers contain citations to sources. This mode is suitable for analytical questions that need synthesis, summarisation or explanation of retrieved content.

FALLBACK MECHANISM

If the maximum cosine similarity among all retrieved chunks is less than 0.65, the fallback mechanism is activated. The system skips Normal Mode retrieval and Enhanced Mode generation and returns a structured fallback response. The response confirms that there is no relevant content in the knowledge base, reports the similarity score of the best matching chunk retrieved from the knowledge base, and suggests query reformulation options. These options suggest alternative vocabulary or more specific phrasing. In zero-retrieval situations, this mechanism fully prevents hallucination, a non-negotiable requirement for enterprise deployment.

V. METHODOLOGY

FORMAL QUERY PROCESSING ALGORITHM

The Q-RAG query processing algorithm processes each user query Q as follows:

Step 1: Preprocessing. 1. Remove stopwords from Q using domain-augmented stopword list L. 2. Q0 = Q \ L Apply statistical n-gram key-phrase extraction $K(Q')$ to extract semantically important phrases and obtain phrase set P.

Step 2 – Embedding . Encode Q to a dense embedding e_Q=E(Q) in R^768 using sentence transformer encoder E. The entire original query Q is used for embedding . P is used for optional keyword pre-filtering to narrow down candidate sets before vector search .

Step 3 – Retrieval for each chunk embedding ei in index I do Calculate cosine similarity sim(eQ, ei). end Return Ck, the top-K chunks ordered by decreasing similarity. Apply threshold filter: C_filtered = {c in C_K | sim(e_Q, e_c) >= τ} where τ=0.65.

Step 4: Send the Mode. If C_filtered is empty, call FALLBACK(Q, max_sim). If the mode is NORMAL, call COMPOSE(C_filtered). Call GENERATE(Q, C_filtered) if mode is ENHANCED.

Step 5—Putting together the response. Add CITE(C_filtered) source citations to the answer. Send the attributed response back to the API layer so it can be sent to the user interface.

EMBEDDING MODEL SELECTION

We chose the all-mpnet-base-v2 sentence transformer because it did better than other models of similar computational cost on MTEB semantic textual similarity tasks. The model has a 12-layer transformer architecture and uses mean pooling over the last hidden states to turn text sequences of up to 512 tokens into 768-dimensional embeddings. Pre-training on a multilingual corpus with a lot of different types of language makes sure that the embedding quality stays the same across all types of business language.

VECTOR INDEX CONFIGURATION

The FAISS IVF index has 256 inverted lists (n_list = 256) and uses 8-byte codes (PQ8) for product quantisation. During retrieval, 32 inverted lists are checked (n_probe = 32), which strikes a balance between speed and accuracy. This setup gets a recall@10 of 0.94 on the development set and cuts the memory footprint by about 8 times compared to flat exact-index storage. This means it can be used on regular enterprise server hardware without a GPU.

LLM PROMPT ENGINEERING

The structured prompt template for Enhanced Understanding Mode is meant to improve the quality of responses while keeping strict grounding. The system message says, "You are an enterprise knowledge assistant." Only use the information given to answer the user's question. Do not use any information that is not in the context. If the context doesn't have enough information to answer the question, make it clear. "Always give credit to the source document for every piece of information you use." This prompt design, along with a temperature of 0.2, always gives answers that are based on the content that was found and are easy to read.

VI. IMPLEMENTATION DETAILS

TECHNOLOGY STACK

Python 3.10 is used to build the backend. FastAPI 0.104 is used in the REST API layer. It handles requests asynchronously, automatically documents OpenAPI 3.0, and checks requests and responses with Pydantic. MongoDB 6.0 keeps track of document metadata, such as the number of documents, the status of ingestion, chunk metadata, and query logs. FAISS 1.7.4 (CPU version) gives the vector index IVF indexing and product quantisation.

The sentence-transformers 2.2.2 library's all-mpnet-base-v2 model makes sentence embeddings. The LLM part connects to Azure OpenAI Service. It uses a GPT-4 (0613) deployment with a 8,192-token context window for Enhanced Understanding Mode. The frontend is a single-page React app that talks to the FastAPI backend using RESTful JSON APIs. Docker Compose takes care of deploying multiple containers, such as the API server, a MongoDB instance, and the FAISS index server.

MULTI-FORMAT DOCUMENT PROCESSING

PyMuPDF 1.23.6 is used for PDF processing, and it can rebuild the reading order for academic and technical documents with multi-column layouts. Python-docx 0.8.11 is used to process DOCX files. It keeps the content of tables by turning it into structured text. Python-pptx 0.6.21 is used to process PPTX files. It extracts text from slide bodies, titles, speaker notes, and slide sequence metadata. Pandas 2.0 and JSON process CSV and JSON files. Before chunking, tabular records are turned into natural language sentence descriptions. The unicodedata library is used to normalise all of the text that was extracted.

SYSTEM DEPLOYMENT ARCHITECTURE

Docker 24.0 and Docker Compose orchestration are used to containerise the system. The production deployment has three services: (1) a FastAPI application server on Uvicorn with four worker processes; (2) MongoDB on a separate volume with journaling turned on for durability; and (3) an FAISS index server that shows the vector index over gRPC. Nginx is a reverse proxy that takes care of SSL termination, request routing, and rate limiting. Horizontal scaling at the API layer has been tested with document corpus sizes of up to 50,000 pages, and there was no decrease in retrieval latency.

VII. EXPERIMENTAL RESULTS

EXPERIMENTAL SETUP

The experiments used a representative set of 500 business documents in five different formats: technical manuals (180 documents), policy and compliance documents (120), operational reports (100), research summaries (60), and structured data files in CSV and JSON (40 files). The corpus has about 12,400 pages and 6.2 million tokens after preprocessing and normalisation.

Five domain experts made a 200-query evaluation set with three groups: factual lookups (80 queries, 40%), procedural queries that needed multi-step information (70 queries, 35%), and analytical queries that needed synthesis across multiple documents (50 queries, 25%). Two domain expert annotators separately labelled sets of documents that were relevant to the ground truth. Cohen's kappa measured inter-annotator agreement at $\kappa = 0.84$.

The server had an Intel Xeon E5-2680 v4 (14 cores), 64 GB of RAM, and no GPU. It ran all the tests. Latency measurements are the average of 200 query runs after a warm-up of 20 queries. There are two baselines that the proposed system is compared to: (1) Keyword Search: BM25-based retrieval with Elasticsearch 8.10; (2) Semantic Search: standalone dense retrieval with the same sentence transformer model but no LLM generation.

RETRIEVAL ACCURACY

The suggested RAG system gets 88% of the time right in both operational modes. This is a 32% improvement over keyword-based BM25 search (56%) and a 16% improvement over standalone semantic search (72%). The improvement over BM25 shows that the embedding model can connect words that don't match between queries and documents. The gain over standalone semantic search shows how the query preprocessing pipeline helps reduce retrieval noise by removing stopwords and extracting key phrases.

TABLE I. RETRIEVAL ACCURACY COMPARISON

METHOD	ACCURACY (%)	IMPROVEMENT OVER BM25
Keyword Search (BM25)	56%	—
Standalone Semantic Search	72%	+16 pp
Proposed RAG (Normal Mode)	88%	+32 pp
Proposed RAG (Enhanced Mode)	88%	+32 pp

PRECISION, RECALL, F1-SCORE, AND MAP

The suggested RAG system has a precision of 0.85, a recall of 0.90, an F1-score of 0.87, and a MAP of 0.83. The recall score (0.90) is higher than the precision score (0.85). This means that the semantic retrieval component finds the most relevant documents most of the time, but it also finds some that are only slightly relevant and weren't included in the ground truth. MAP of 0.83 shows that the quality of the rankings is always high across the whole set of results, not just at certain cutoffs.

Factual lookup queries (93% accuracy, F1 0.91) and procedural queries (87% accuracy, F1 0.86) are the best types of queries for performance. Analytical queries, which need information from more than one document, get a lower score (81% accuracy, F1 0.79).

TABLE II. FULL EVALUATION METRICS

METHOD	PRECISION	RECALL	F1-SCRE	MAP
Keyword Search (BM25)	0.54	0.59	0.56	0.51
Standalone Semantic Search	0.70	0.74	0.72	0.67
Proposed RAG System	0.85	0.90	0.87	0.83

Response Latency Analysis

The average response time for RAG Normal Mode is 0.90 seconds, which is much faster than the keyword search baseline of 1.90 seconds. The improvement, even with the extra embedding and vector search steps, shows that FAISS approximate nearest-neighbor retrieval has a lower query complexity than BM25's linear document scan. Enhanced Mode adds about 0.30 seconds to the time it takes to generate an LLM, bringing the total to 1.20 seconds, which is well within the requirements for interactive responses.

TABLE III. Response Latency Statistics (seconds)

System	Mean (s)	Std Dev	P50 (s)	P95 (s)
Keyword Search (BM25)	1.90	0.45	1.85	2.80
Semantic Search	1.10	0.22	1.05	1.55
RAG Normal Mode	0.90	0.18	0.87	1.30
RAG Enhanced Mode	1.20	0.29	1.15	1.85

Hallucination Evaluation

A collection of 40 enquiries regarding information lacking in the knowledge base was formulated to assess hallucination reduction. A standalone LLM baseline without retrieval produced hallucinated responses for 35 of 40 queries (87.5% hallucination rate), generating plausible-sounding but factually incorrect content. The suggested RAG system correctly turned on the fallback mechanism for all 40 queries (100% fallback rate), giving back structured knowledge-gap acknowledgements with no made-up content.

Three expert annotators checked the factual accuracy of Enhanced Mode responses by comparing them to source documents for queries that had relevant knowledge base content. The system got a factual consistency score of 96.2%, which means that 96.2% of the responses that were looked at had all of their factual claims based on and consistent with the source documents that were found.

Scalability Analysis

Table IV shows how long it takes to retrieve data from different corpus sizes. FAISS IVF approximate nearest-neighbor retrieval scales sub-linearly: as the corpus grows from 1,000 to 100,000 documents (a 100× increase), mean retrieval latency goes up from 48 ms to 134 ms (a 2.8× increase). This sub-linear scaling shows that the proposed architecture will work for large enterprise knowledge bases that don't have distributed infrastructure at normal enterprise corpus scales.

TABLE IV. Scalability: Retrieval Latency vs. Corpus Size

Corpus Size (documents)	Mean Retrieval Latency (ms)	Scaling Factor (vs. 1K)
1,000	48	1.0×
5,000	62	1.3×
10,000	74	1.5×
50,000	108	2.3×
100,000	134	2.8×

VIII. Discussion

Performance Analysis

The experimental results show that the proposed system makes big improvements on all evaluation metrics compared to both baselines. The 32 percentage-point increase in accuracy over BM25 shows how bad the semantic gap is in traditional enterprise search and how useful dense retrieval is for managing organisational

knowledge. The 16 percentage-point gain over standalone semantic search shows that query preprocessing and relevance thresholding add a lot more than just raw embedding-based retrieval.

The 100% fallback activation rate for zero-retrieval queries is a safety feature that is important for businesses to use. The threshold-based fallback gets rid of the worst kind of failure: confident, fluent, and factually wrong answers. This is not an option in regulated fields.

LIMITATIONS

It is still harder to do analytical queries that require multi-hop reasoning across documents, with an accuracy of 81% compared to 93% for factual lookups. The single-pass retrieval architecture gets a fixed top-K chunk set instead of going through a series of reasoning steps. Future research will combine multi-hop reasoning frameworks with iterative retrieval systems.

Text extraction quality diminishes for documents exhibiting subpar OCR, rotated text, mathematical equations depicted as images, or intricate multi-column technical diagrams. Future studies ought to investigate multimodal document comprehension models to address these instances.

The performance of the system depends on how good and how wide the enterprise knowledge base is. The fallback mechanism works correctly for gaps where documents were not ingested. To keep their documents up to date and complete, organisations need document lifecycle management processes. These include adding new content, updating old documents, and getting rid of old content.

PRACTICAL DEPLOYMENT CONSIDERATIONS

When deploying in production, you need to pay attention to a few practical details. You can add access control by storing per-document access control lists (ACLs) in MongoDB metadata and filtering the results of a retrieval by the authenticated user's permission set before sending a response. You can add logging for queries and responses to the FastAPI middleware layer for audit purposes without changing the main retrieval pipeline. The system has been tested with document collections of up to 50,000 pages. At the API layer, horizontal scaling allows for larger deployments through standard container orchestration.

IX. CONCLUSION

This paper introduced an Enterprise Retrieval-Augmented Generation system aimed at overcoming the practical constraints of keyword-based retrieval and independent LLM deployment in organisational knowledge management. The system has a multi-format document ingestion pipeline, FAISS-based semantic retrieval using dense vector embeddings, a dual-mode retrieval-generation architecture that supports both strict document-grounded retrieval and contextual LLM elaboration, a threshold-based fallback for robustness in zero-retrieval scenarios, and inline source attribution for response explainability.

Tests on a 500-document enterprise corpus with a 200-query evaluation set show 88% retrieval accuracy, precision 0.85, recall 0.90, F1-score 0.87, and MAP 0.83. These numbers are much better than the baselines for BM25 keyword search (56%) and standalone semantic search (72%). In Enhanced Mode, the response time is 1.2 seconds, and in Normal Mode, it is 0.9 seconds. This is within the requirements for interactive responses. In cases where there is no retrieval, the threshold-based fallback completely stops hallucination. Retrieval latency scales sub-linearly, allowing for corpus sizes of up to 100,000 documents on standard server hardware.

Future directions encompass: (1) multi-hop iterative retrieval to enhance analytical query performance; (2) multimodal document comprehension for image-intensive and equation-rich documents; (3) personalised retrieval that considers user role, expertise, and access permissions; (4) incremental knowledge base updates without complete reindexing; and (5) assessment on larger, more diverse enterprise document corpora across various organisational domains.

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. Yih, T. Rocktaschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459–9474.

[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, Minneapolis, MN, 2019, pp. 4171–4186.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.

[4] X. Chen, Y. Liang, S. Shi, D. He, and Q. Liu, "Mitigating Language Model Hallucination via Retrieval-Grounded Generation," IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 3, pp. 1482–1495, 2024.

[5] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying Large Language Models and Knowledge Graphs: A Roadmap," IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 7, pp. 3580–3599, 2024.

[6] J. Johnson, M. Douze, and H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.

[7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proc. EMNLP-IJCNLP, Hong Kong, 2019, pp. 3982–3992.

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, and P. Dhariwal, "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 1877–1901.

[9] 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.

[10] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in Proc. EMNLP, 2020, pp. 6769–6781.

[11] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy," in Findings of EMNLP, 2023.

[12] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," in Proc. ICML, 2020, pp. 3929–3938.

[13] G. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering," in Proc. EACL, 2021, pp. 874–880.