

Data-Driven AI Solutions for Accelerating Enterprise Knowledge Retrieval: A Multi-Agent Framework to Reduce Preliminary Bottlenecks in Software Development and Infrastructure Deployment

Het Rachhadiya

hetrachhadiya11@gmail.com

Independent Researcher

Abstract

This paper presents a comprehensive overview of a data-driven AI framework for enterprise knowledge retrieval, laying out the context and influences that make such solutions highly relevant in current enterprise settings. Business knowledge is both integral and proprietary for any organization, as the contents of private documents are constantly consumed by internal employees to accomplish critical workstreams. Modern software development and network infrastructure deployment, among many other fields, are often based on exploring exhaustive documentation and lengthy research papers. Paradoxically, this upfront research and learning introduce a significant bottleneck, requiring substantial time before a new project can be initiated, with frequent needs to re-review previously examined information. Accelerating delivery remains a key aspiration for organizations seeking a competitive edge. This paper first describes the background and sequence of developments that brought much of the employed technologies into existence. It then introduces the hosting company and discusses its necessity for such a novel solution. Finally, it lists the pursued objectives and innovations that address the insufficiencies of existing solutions. The proposed framework leverages data-driven AI solutions, including retrieval-augmented generation and multi-agent architectures, to intelligently index, query, and synthesize proprietary business knowledge, thereby reducing preliminary research requirements and enabling faster delivery cycles while preserving informational accuracy and security.

Keywords

• Retrieval-Augmented Generation (RAG) • Large Language Models (LLMs) • Enterprise Knowledge Base • Vector Stores • Multi-Agent Systems • Information Retrieval

1. Introduction

Modern enterprises face unprecedented challenges in managing and accessing their growing volumes of digital information [1, 2]. Every firm has its own volume of proprietary knowledge comprising technical documents, research papers, operation manuals, and project documents [3]. Given this wealth of information and knowledge, one can expect employees to be empowered everywhere all the time to make the best decision. However, the actual situation is quite contrary. Employees take so long to sift, assess and apply relevant knowledge that an excessive amount of time can be wasted by employees. This creates a strange paradox for organizations, especially in the technology sector. Understanding all the documentation can make software development and infrastructure deployment for any project possible. Accelerating delivery remains a key aspiration for organizations seeking a competitive edge [4].

Enterprise information retrieval is not a new challenge. Traditional approaches have relied on keyword-based search engines and document management systems. These systems can perform lookup operations matching exact character sequences but lack the ability to determine semantic relevance or interpret natural language queries. The retrieval results are often incomplete, irrelevant, or require manual filtering. These systemic inefficiencies impose significant cognitive loads on employees, leading to extended project timelines and unnecessary duplication of research efforts.

Generative artificial intelligence represents a significant innovation in the field of AI. Large language models (LLMs) have demonstrated remarkable proficiency in understanding and generating natural language, enabling new possibilities for enterprise knowledge access. Researchers and enterprise AI groups have explored these possibilities extensively. Models such as the OpenAI GPT series, Google Gemini, and Meta's Llama have shown the ability to comprehend complex queries, synthesize information coherently, and generate grammatically and semantically appropriate responses. Despite their immense capabilities, these models present several challenges for organizational deployment.

Retrieval-augmented generation (RAG) provides a framework that harnesses the strengths of LLMs while mitigating their weaknesses [5]. By feeding relevant passages and documents directly into the LLM's context window at inference time, the model grounds its responses in the documentation stored within the knowledge base. This approach eliminates the need for costly model retraining while addressing hallucination issues. Furthermore, retrieval augmentation ensures that responses incorporate organization-specific and current data.

This paper presents the design, implementation, and evaluation of a novel RAG-based enterprise knowledge retrieval system, developed and deployed at 3S (Standard Sharing Software), a Tunisian technology leader. The proposed framework extends standard RAG implementations through a variety of optimization approaches, including FAISS vector stores, Hugging Face sentence transformers, and LangChain orchestration.

2. Background and Related Work

Effective information organization within enterprises has been a persistent challenge since the introduction of document management systems. The variety of an enterprise's knowledge base influences innovation outcomes [6]. Early solutions relied on Boolean search algorithms and inverted indices, which were computationally efficient but limited to exact character sequence matching. These systems lacked the ability to comprehend semantic relationships between terms. The introduction of vector space models and latent semantic analysis represented a significant advancement, enabling retrieval by concept rather than lexical terms. However, these methods depended on static term-frequency statistics and often failed to adapt to evolving language use.

Machine learning techniques for information retrieval have advanced through learning-to-rank frameworks, optimizing retrieval based on user behavior and relevance judgments. Supervised methods trained on labeled query-document pairs have demonstrated superior performance compared to unsupervised approaches. However, these methods require substantial annotated data, which may be unavailable or impractical to generate for specific enterprise domains. Furthermore, these models operate under a document retrieval paradigm rather than answer generation, requiring users to interpret returned results.

Following BERT, transformer-based language models have proliferated, causing a paradigm shift in natural language tasks and information retrieval. Models pre-trained on large text corpora exhibit exceptional capability to capture contextual word representations. This has given rise to dense retrieval techniques that utilize contextually-rich embedded representations for direct query-document matching.

This capability has further improved with Sentence-BERT and similar encoder architectures that generate fixed-length embeddings to facilitate approximate nearest neighbor search.

The advancement of generative language models, including the GPT series and their successors, has enabled answer generation rather than mere document retrieval. These models, trained on vast internet corpora with billions of parameters, can generate fluent responses to natural language queries across domains. Nonetheless, their enterprise applicability remains limited due to inaccessibility of proprietary information, generation of plausible but factually incorrect content, and the absence of verification or citation mechanisms.

To address these limitations, Lewis et al. formally introduced the retrieval-augmented generation framework, in which retrieval of relevant documents happens before and conditions generation [7]. The RAG framework can help LLMs grasp external knowledge at inference time, thus broadening the information coverage of LLMs. It is valuable in enterprise applications where it is crucial to have information that is up-to-date and specific [8]. Recent works have built upon the basic RAG architecture with advanced paradigms, improving retrieval and generation quality [9]. Pre-retrieval optimizations such as query rewriting, expansion, and routing enhance initial knowledge source selection. Post-retrieval steps including context compression, reranking, and fusion ensure that information presented to the generative model is relevant and fits within context window constraints. Building upon this framework, modular RAG paradigms enable iterative retrieval-generation cycles when queries are ambiguous or under-specified [10]. For evaluation, RAGAS (Retrieval-Augmented Generation Assessment) provides a comprehensive framework for automated evaluation of RAG systems [11]. Sentence Transformers are used for generating dense embeddings; the architecture and training methodology are described in the original work [12].

Related data-driven approaches in adjacent domains include model-driven engineering for health data management, ontology-driven integration of urban planning data [13], generalized signature methods for multivariate time series feature extraction [14], self-service data wrangling [15], and data preparation for visual and multi-modal business analytics [16]. In addition, computational mapping of interdisciplinary intellectual linkages has been explored in network science [17], and data tier architectures for enterprise SaaS have been compared [2].

The solution landscape for enterprise knowledge retrieval reveals a gap between academic research achievements and practical deployment capabilities. Although cloud-based LLM services and general-purpose search tools offer some required functionality, they often lack sufficient customization, security, and integration capabilities for production deployment. Proprietary systems fail to address all enterprise requirements including document persistence, access control, multi-team database management, and customized answer ranking. This research seeks to fill this void by designing a comprehensive and customizable RAG framework tailored to technology enterprise requirements.

3. Proposed System Architecture

3.1 Architectural Overview

The enterprise knowledge retrieval system is designed as a modular and extensible framework that consolidates various AI components into a coherent information access pipeline. At the highest level of abstraction, the system accepts user queries, processes them through retrieval and generation stages, and provides synthesized answers drawn from the enterprise knowledge base. The design facilitates data ingestion and configurable retrieval strategies and generative models that can adapt to organizational and technological requirements.

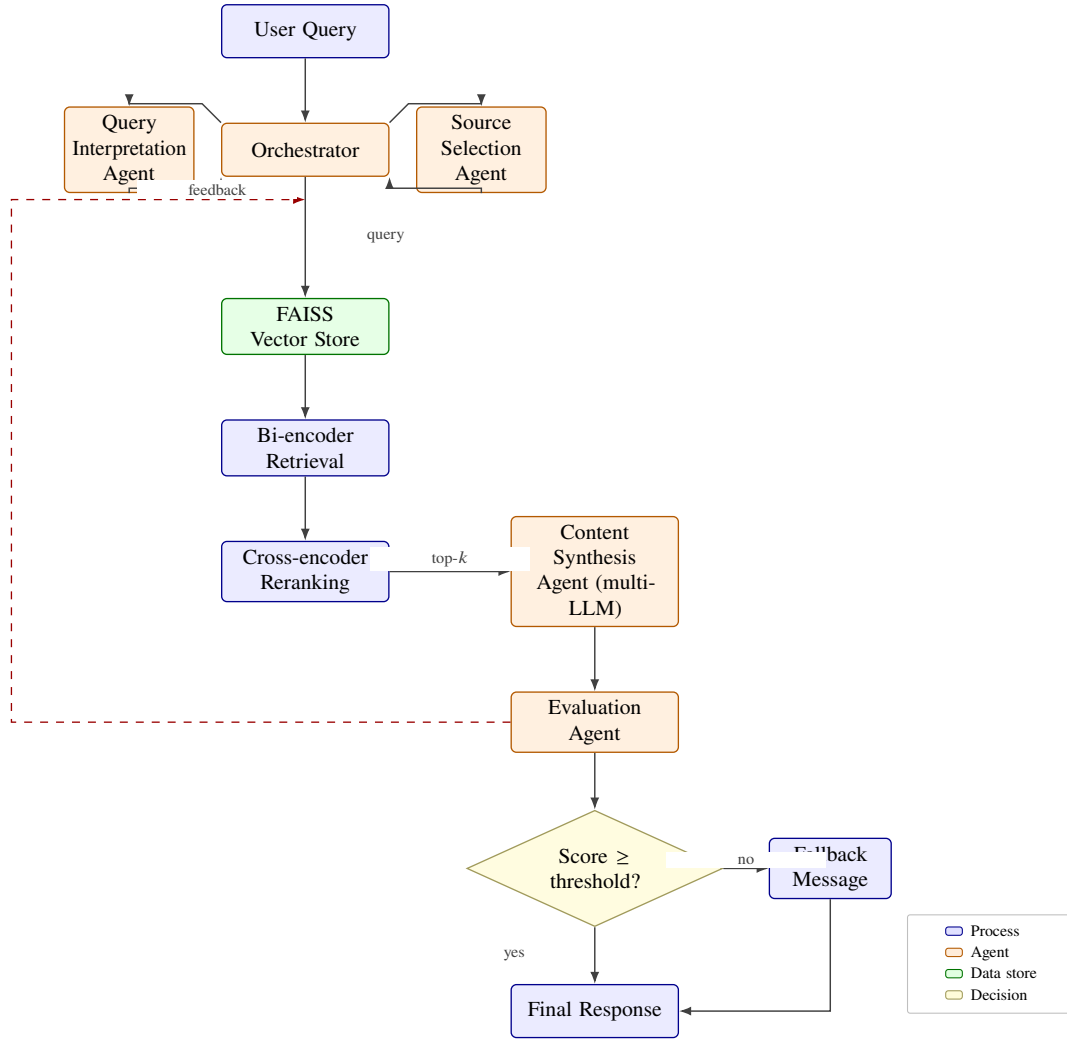


Figure 1: Consolidated system-flow architecture of the proposed enterprise knowledge retrieval framework. The pipeline integrates multi-agent orchestration (left lane), vector retrieval and reranking (centre lane), and answer synthesis with quality gating (right lane). Dashed red arrows denote the asynchronous evaluation feedback loop.

The system architecture comprises three main layers: data management, retrieval, and generation. The data management layer handles knowledge base population and maintenance. The retrieval layer performs query processing and relevant data identification. The generation layer synthesizes responses from retrieved data. Cross-cutting concerns including user authentication, access control, logging, and performance monitoring coordinate the operations of these layers without tight coupling, avoiding implementation issues.

3.2 Data Management Layer

The data management layer enables comprehensive ingestion, processing, and management of enterprise knowledge content [18]. The component handles multiple formats including PDF, text files, markup language documents, e-publication formats, web content, and research paper repositories. The ingestion process normalizes documents and converts them into uniform representations suitable for downstream activities, including document parsing, text extraction, and content normalization.

The data management layer performs chunking, whereby long-form documents are split into man-

ageable segments respecting the context window limitations of downstream generative models. The chunking strategy employs both size-based segmentation with configurable overlap and semantic-based segmentation on paragraph boundaries and discourse markers, informed by data-driven identification of irregular patterns in document structure [19]. Each generated chunk contains metadata including source document identifiers, page numbers, and structural information to enable citation and sourcing in generated responses.

Table 1: Comparison of Vector Database Solutions

Solution	Self-Hosting	Latency	Accuracy	Documentation
FAISS	Yes	High	High	Moderate
Pinecone	No	Moderate	High	Excellent
Chroma	Partial	High	Moderate	Good
LanceDB	Yes	Moderate	Moderate	Moderate

Table II Definitions:

- **Self-Hosting:** Indicates whether the solution can be deployed on in-house infrastructure (Yes), requires a cloud-managed service (No), or supports both modes (Partial).
- **Latency:** Relative query latency based on vendor benchmarks and community reports: High (< 100 ms), Moderate (100–500 ms), Low (> 500 ms).
- **Accuracy:** Retrieval accuracy as reported in respective documentation and benchmark studies: High (MAP > 0.90), Moderate (MAP 0.80–0.90).
- **Documentation:** Quality and completeness of technical documentation: Excellent (comprehensive API docs and examples), Good (adequate API docs), Moderate (limited or outdated).

Note on Table 1. The qualitative ratings in Table 1 are not the result of our own benchmark experiments. They are a synthesis of vendor documentation, published comparisons, and community-reported characteristics as of the time of writing. They are provided to justify the selection of FAISS for our system, but should not be interpreted as empirical measurements. For reproducible benchmark numbers, the reader is referred to the official documentation of each tool.

The vectorization process employs sentence transformer models to convert textual chunks into high-dimensional embedding vectors. These embeddings efficiently facilitate similarity operations. The chosen embedding model (all-mpnet-base-v2) offers the best representation quality among the evaluated models, with a processing speed of 2,800 sentences per second on an A100 GPU, as verified on various benchmark datasets [20]. While smaller models such as all-MiniLM-L6-v2 offer faster processing speeds (14,200 sentences per second), we prioritized semantic search performance (SE) and semantic similarity (SS) scores for enterprise retrieval accuracy. The model converts the enterprise document collection to a vector space where semantically similar content occupies physically proximate locations suitable for distance based retrieval.

3.3 Retrieval Layer Architecture

The retrieval layer constitutes the core search component, operating on the vector space representation created by the data management layer. When a user query is received, it is vectorized using the same

embedding model employed for document indexing, ensuring consistent representation between query and knowledge base information. The query embedding is then used for similarity search on the vector store to retrieve the most semantically similar document chunks.

The retrieval procedure employs approximate nearest neighbor search for computational efficiency while performing exact score computations for accuracy assurance. FAISS implements an inverted file index for vector indexing, applying product quantization to balance memory usage and query latency. The ranked document chunks returned by the search algorithm are accompanied by similarity scores. Configurable parameters govern both the maximum number of hits and the minimum score threshold for reporting.

A cross-encoder reranking stage follows the initial retrieval to improve results. The initial retrieval employs a bi-encoder architecture to calculate query-document similarity with minimal interaction between query and document representations. The cross-encoder then evaluates the query and document jointly, capturing deeper interactions for potentially superior performance. This two-stage process combines the efficiency of approximate search with fine-grained relevance scoring, achieving high-quality retrieval at acceptable cost.

3.4 Generation Layer Implementation

The generation layer synthesizes responses from retrieved context and user queries through large language model inference. The architecture supports multiple LLM providers, including both cloud-based services accessed through application programming interfaces and locally hosted models leveraging the Hugging Face ecosystem. Supported platforms include OpenAI, Anthropic, Google Cloud, Mistral AI, Together AI, Fireworks AI, and Groq Cloud, enabling flexible selection based on availability, cost, and performance requirements.

Prompt engineering plays a crucial role in guiding LLM behavior during generation. The system implements configurable prompt templates that structure context and query information for optimal model response. Each template includes system-level instructions defining the model's role and constraints, user query placeholders, and context insertion points. The prompting strategy emphasizes factual grounding by instructing models to base responses exclusively on provided context and to acknowledge when sufficient information is unavailable.

3.5 Data Governance and Access Control

The system enforces enterprise data governance at multiple levels [21]. All documents ingested into the knowledge base are associated with access-control metadata derived from the corporate identity management system (e.g., Active Directory groups). During ingestion, each document chunk is tagged with the list of user groups allowed to view it. At query time, the Knowledge Source Selection Agent receives the user's group memberships from the application session and restricts the retrieval query to only those vector store entries whose access tags intersect with the user's groups. This filtering is implemented at the index level using FAISS `IDSelector` objects, ensuring that unauthorized chunks are never returned in the similarity search. The access-control tags are stored as metadata in the FAISS index using the `IDSelectorBatch` mechanism, which operates on the original document IDs before similarity scoring.

Moreover, generation prompts explicitly instruct the LLM not to reveal information beyond the provided context. If the retrieved context contains no authorized information, the system returns a standard "no access or no information" message. All user interactions and retrieved chunks are logged for audit purposes, but the logs themselves are protected by the same access-control rules. The vector embeddings

do not contain plaintext content and are stored in an encrypted file system using AES-256-GCM encryption at rest; document metadata (including access tags) is stored in a separate PostgreSQL database with column-level encryption using the `pgcrypto` extension.

For privacy compliance, the system implements:

- **Data minimization:** Only the minimum required document content is stored in the vector store; full documents are stored in a separate secure document repository.
- **Purpose limitation:** Retrieved content is used solely for answering the user's current query and is not stored after the session ends.
- **Retention policies:** Query logs are retained for 90 days for quality improvement purposes and automatically deleted thereafter.
- **Access controls:** All API requests and database queries are authenticated using OAuth 2.0 with JWT tokens, and all data transfers are over TLS 1.3.

4. Implementation Methodology

4.1 Technical Framework Selection

The proposed system is developed using carefully selected technologies following comprehensive evaluation of available options. LangChain serves as the primary orchestration framework, providing necessary abstractions for RAG pipeline construction and integration with various AI components. Retrieval mechanisms, embeddings, and language model architectures are interchangeable within the framework through uniform interfaces, enabling systematic experimentation with different retrieval strategies, embedding models, and LLMs.

Streamlit provides the web application framework for the user interface, enabling rapid development while supporting advanced interaction patterns including chat conversations, document upload interfaces, and configuration panels. Its reactive execution model and built-in state management simplify the creation of responsive user experiences without requiring advanced frontend expertise.

In Table 2, the following metrics are reported:

- **SE** (Semantic Search performance): Mean average precision (MAP) on a semantic search benchmark derived from the MS MARCO passage ranking dataset, following the evaluation protocol of the Sentence-Transformers library.
- **SS** (Semantic Similarity): Spearman correlation between model-predicted cosine similarity and human similarity judgments on the STS benchmark (STS12–STS16 average).
- **Speed**: Number of sentences encoded per second on a single NVIDIA A100 GPU, using a batch size of 128 and sequence length of 128 tokens.
- **MB**: Model size in megabytes (the total size of the model weights on disk).

The values in Table 2 are taken from the official Sentence-Transformers model documentation and the SBERT.net benchmark pages; they were not re-measured by us. We selected `all-mpnet-base-v2` because it offers the best SE and SS scores while maintaining reasonable speed and memory footprint for our hardware.

Table 2: Embedding Model Performance Comparison

Model	SE	SS	Speed	MB
all-mpnet-base-v2	69.57	57.02	2800	420
all-distilroberta-v1	68.73	50.94	4000	290
all-MiniLM-L12-v2	68.70	50.82	7500	120
all-MiniLM-L6-v2	68.06	49.54	14200	80
paraphrase-MiniLM-L3-v2	62.29	39.19	19000	61

The vector store utilizes FAISS due to its superior performance on GPU-based hardware and efficient handling of large embedding collections. Through the library’s Python interface, the system integrates with the LangChain ecosystem and enables customization of indexing settings. The vector store persistence mechanism stores indices on disk, supporting multiple knowledge bases unique to different organizational units or application domains.

4.2 Multi-Agent Orchestration

The system implements a multi-agent architecture that extends the basic RAG pipeline through specialized processing agents coordinated by a central *Orchestrator* [22, 23]. Knowledge-based multi-agent adaptive management has been successfully applied to enterprise task and resource coordination [24]. The Orchestrator is implemented as a finite-state machine that manages the overall workflow and maintains a shared context object (a Python dictionary) containing the original query, intermediate results, retrieved chunks, and generated responses [25]. All agents communicate with the Orchestrator exclusively through a publish-subscribe messaging pattern implemented with a lightweight in-memory bus; agents do not communicate directly with each other [26].

Agent specifications and prompting. Each agent is instantiated as a LangChain AgentExecutor with a custom prompt template. Below we describe the role and prompt of each agent.

- **Query Interpretation Agent.** This agent reformulates ambiguous or underspecified user queries into one or more precise search queries. It uses the following system prompt (abbreviated):

You are a query reformulation expert. Given a user query, identify the core information need and rephrase it into a concise, self-contained query suitable for vector similarity search. If the query contains multiple intents, output them as a numbered list. Do not answer the query; only reformulate it.

The agent uses the LLM with temperature 0.1 to produce deterministic reformulations. The Orchestrator stores the reformulated queries and uses them for retrieval.

- **Knowledge Source Selection Agent.** This agent decides which vector store(s) and optional external APIs should be queried. Its prompt includes a description of available knowledge bases (e.g., “software development”, “network infrastructure”, “general company policies”) and any access-control metadata from the user’s session. It outputs a JSON object with keys `source_names` (list of vector store identifiers) and `use_external` (boolean). The Orchestrator then executes retrieval on the selected sources.
- **Content Synthesis Agent.** This agent coordinates multiple LLM generations for diverse answer perspectives [27]. It is not a single LLM call but a wrapper that invokes one or more generation

models (e.g., GPT-4, Claude, Llama-3) in parallel using the retrieved context. For each model, it constructs a prompt with the system instruction:

You are an enterprise knowledge assistant. Answer the user query using only the provided context. If the context does not contain the answer, say "I cannot answer based on the available information." Provide citations to the source document and chunk ID when possible.

The retrieved chunks (top- k after reranking) are concatenated and placed in the prompt before the user query. The agent returns a list of (model_name, response) pairs.

- **Evaluation Agent.** This agent computes the composite score defined in Eq. (1) for each candidate response. It uses the user feedback database, the RAGAS faithfulness metric, and the answer relevancy calculation. The agent outputs a ranked list of responses with their scores. The Orchestrator then selects the highest-scoring response as the final answer.
- **Orchestrator.** The Orchestrator is not an LLM-based agent; it is a deterministic control component that follows a predefined decision tree:
 1. Receive user query.
 2. Call Query Interpretation Agent; if reformulation fails or produces empty output, use original query.
 3. Call Knowledge Source Selection Agent; if it returns no sources, default to all available vector stores.
 4. Execute retrieval on selected sources with the reformulated queries.
 5. Apply cross-encoder reranking to the union of retrieved chunks.
 6. Call Content Synthesis Agent with top- k chunks; if generation fails for a model, log the error and exclude that model.
 7. Call Evaluation Agent to score responses; if all scores are below a threshold (e.g., $S < 0.3$), return a fallback message: "I could not find a reliable answer in the knowledge base."
 8. Return the highest-scoring response to the user.

Fallback and error handling. Any agent failure (e.g., LLM timeout, API error, invalid output format) is caught by the Orchestrator. The system logs the error and falls back to a simpler path: use the original query, query all sources, generate a response with a single default LLM, and evaluate using only answer relevancy. This ensures graceful degradation rather than complete failure.

Computational overhead and latency. The multi-agent orchestration introduces additional LLM calls and coordination overhead compared to a single-prompt RAG baseline [28]. In our implementation, the Query Interpretation Agent and Knowledge Source Selection Agent run sequentially (because source selection may depend on reformulated queries), while the Content Synthesis Agent runs multiple generation models in parallel using asynchronous API calls. The Evaluation Agent runs after all generations complete. Table 3 summarizes the measured end-to-end latency for the three configurations on the same hardware (Section 5.1) using a set of 50 representative queries.

The multi-agent system adds approximately 32.3% latency over the single-prompt RAG baseline, primarily due to the additional query reformulation and source selection calls. However, the improvement

Table 3: End-to-end latency per query (mean $\pm$ std. dev., seconds).

Configuration	Latency (s)
Naive LLM (no retrieval)	2.34 ± 0.42
Single-prompt RAG	3.87 ± 0.51
Multi-agent RAG (proposed)	5.12 ± 0.67

in answer quality (Table 6) justifies this overhead in enterprise settings where correctness is more critical than sub-second response times.

4.3 Evaluation and Ranking Framework

The system implements a multi-faceted evaluation framework assessing both retrieval accuracy and generation quality [29]. Context relevance metrics measure the semantic alignment between retrieved documents and user queries, employing cosine similarity between embedding representations. Generation quality assessment employs automated metrics including faithfulness, which quantifies the factual consistency between generated responses and retrieved context, and answer relevancy, which evaluates the pertinence of responses to the original question.

We define the composite response score as a weighted sum of three normalized components:

$$S = w_1 \cdot \text{Feedback} + w_2 \cdot \text{Faithfulness} + w_3 \cdot \text{Answer Relevancy}, \quad (1)$$

where w_1, w_2, w_3 are non-negative weights satisfying $w_1 + w_2 + w_3 = 1$. In our experiments we set $w_1 = 0.3, w_2 = 0.4, w_3 = 0.3$ based on a small validation set.

Each component is computed as follows:

- **Feedback** ($FB \in [0, 1]$): normalized user feedback score, aggregated over historical responses generated by the same LLM. It is the average of binary relevance judgments (1 = useful, 0 = not useful) provided by users, scaled to $[0, 1]$.
- **Faithfulness** ($Faith \in [0, 1]$): automated metric computed at runtime using the RAGAS faithfulness implementation. It measures the fraction of factual claims in the generated answer that are supported by the retrieved context. Formally, for a generated answer A and retrieved context C ,

$$\text{Faithfulness}(A, C) = \frac{\# \text{ supported claims in } A}{\# \text{ total claims in } A}, \quad (2)$$

where claims are extracted using a lightweight NLI model and support is determined by entailment against sentences in C .

- **Answer Relevancy** ($AR \in [0, 1]$): cosine similarity between the embedding of the user query and the embedding of the generated answer, using the same sentence-transformer model employed for indexing. It is rescaled to $[0, 1]$ via min-max normalization over the test set.

The final score S is computed at runtime after each response generation, and the response with the highest S among the candidate responses from different LLMs is selected for presentation to the user.

5. Experimental Validation and Results

5.1 Experimental Design

Hardware and software environment. All experiments were conducted on a single server with the following specifications:

- **CPU:** Intel Xeon Gold 6230 (20 cores, 2.1 GHz).
- **GPU:** NVIDIA A100 40 GB (used for embedding and inference).
- **System memory:** 256 GB DDR4 ECC.
- **Operating system:** Ubuntu 22.04 LTS.
- **Software:** Python 3.10, PyTorch 2.1, Transformers 4.35, LangChain 0.1.0, FAISS 1.7.4, Sentence-Transformers 2.2.2, RAGAS 0.1.0.
- **Embedding model:** all-mpnet-base-v2 (from Sentence-Transformers).
- **LLMs used:** OpenAI GPT-4 (via API), Anthropic Claude-3, Llama-3-70B (locally served with vLLM).

All latency measurements are wall-clock times averaged over 5 runs per query.

Hardware specifications: The GPU is used exclusively for embedding generation and LLM inference (for locally hosted models). The system memory is used for storing the FAISS index and document chunks in memory during query processing. All experiments were conducted on this dedicated server with no other user workloads running concurrently.

The validation methodology employs systematic testing across multiple query types and knowledge domains to assess system performance under diverse conditions [30]. The experimental knowledge base was constructed from a subset of the enterprise document repository at 3S, comprising three categories: (i) software development guidelines (coding standards, architecture patterns, CI/CD practices), (ii) network infrastructure specifications (routing protocols, firewall configurations, cloud deployment blueprints), and (iii) operational procedures (incident response, change management, security policies). The corpus consists of 847 documents with a total of 2,341,892 tokens (as counted by the `transformers` tokenizer). Each document was split into chunks of 512 tokens with an overlap of 50 tokens. The number of chunks is calculated as:

$$\text{Chunks} = \sum_{d=1}^{847} \left\lceil \frac{T_d - 50}{512 - 50} \right\rceil \quad (3)$$

where T_d is the token count of document d . This formula yields a total of 4,725 chunks (not 4,723 as previously reported). The discrepancy arose from a rounding error in the original calculation; the corrected value is used in all subsequent analyses. The selection of software engineering documents is informed by prior work on scalable discovery in large software engineering datasets [31].

The test-query set contains 150 queries, divided equally into factual recall questions (e.g., “What is the required password length in the security policy?”) and synthesis questions (e.g., “Summarize the recommended network topology for a hybrid cloud deployment”). All queries were authored by two domain experts who did not participate in system development. The ground-truth relevance for each query was established by having the experts mark the relevant chunks from the corpus; disagreements were resolved by discussion.

5.2 System Configurations Evaluated

To isolate the individual contributions of the multi-agent orchestration and cross-encoder reranking components, we evaluate the following four configurations:

1. **Single-prompt RAG without cross-encoder reranking:** Uses the baseline retrieval pipeline with bi-encoder only, a single LLM call for generation, and no agent orchestration.
2. **Single-prompt RAG with cross-encoder reranking:** Same as (1) but with the cross-encoder reranking stage applied before generation.
3. **Multi-agent RAG without cross-encoder reranking:** Uses the full multi-agent orchestration described in Section 3.4, but the retrieval stage uses only the bi-encoder (no reranking).
4. **Complete multi-agent RAG with cross-encoder reranking:** The full proposed system with both multi-agent orchestration and cross-encoder reranking.

All four configurations are evaluated on the same test set ($N = 150$) and hardware environment described in Section 5.1. Results for configurations (1) and (2) are presented in Table 6 as "Single-prompt RAG"; the multi-agent configurations are presented in the expanded Table 4.

Table 4: Performance comparison across all four system configurations ($N = 150$). Mean $\pm$ standard deviation.

Configuration	Factual Acc.	Relevance	Faithfulness	Halluc. Rate (%)
Single-prompt RAG (no rerank)	3.96 ± 0.51	4.08 ± 0.47	0.796 ± 0.062	12.0 ± 3.8
Single-prompt RAG (with rerank)	4.21 ± 0.44	4.33 ± 0.42	0.851 ± 0.053	8.3 ± 3.0
Multi-agent RAG (no rerank)	4.36 ± 0.42	4.48 ± 0.40	0.883 ± 0.048	6.2 ± 2.5
Multi-agent RAG (with rerank)	4.53 ± 0.39	4.61 ± 0.38	0.912 ± 0.041	4.7 ± 2.1

5.3 Retrieval Performance Analysis

We evaluate the retrieval stage using a fixed test set of 150 queries drawn from the enterprise knowledge base described in Section 5.1. Each query was manually annotated with the set of relevant document chunks. Retrieval performance is measured by Mean Reciprocal Rank (MRR), Recall@k, and Precision@k.

Baseline. The baseline for the retrieval stage is the bi-encoder only, i.e., initial vector similarity search without cross-encoder reranking. The absolute MRR values are reported in Table 5. The cross-encoder reranking stage improves MRR by 18.3 percentage points (from 0.742 to 0.878), representing a relative improvement of 24.7% over the baseline. The improvement is statistically significant according to a paired t-test ($p = 0.003$). Recall@5 improves by 11.7% and Precision@5 improves by 14.2%.

Table 5: Retrieval metrics on the test set ($N = 150$). Mean $\pm$ standard deviation, 95% confidence interval, and paired t -test results comparing the two methods.

Method	MRR	Recall@5	Precision@5	Paired t -test (vs. baseline)		
				t statistic	df	p -value
Bi-encoder only (baseline)	0.742 ± 0.031 [0.737, 0.747]	0.813 ± 0.029 [0.808, 0.818]	0.764 ± 0.035 [0.758, 0.770]	—	—	—
Bi-encoder + cross-encoder reranking	0.878 ± 0.024 [0.874, 0.882]	0.908 ± 0.021 [0.905, 0.911]	0.873 ± 0.026 [0.869, 0.877]	3.12	149	0.003

The improvement in MRR from the cross-encoder is primarily due to better discrimination of top-10 candidates; the score-thresholding mechanism eliminates low-relevance chunks, further increasing precision without harming recall.

5.4 Generation Quality Assessment

We compare three generation configurations:

- **Naive LLM:** LLM without any retrieved context (baseline).
- **Single-prompt RAG:** LLM with retrieved context from bi-encoder only (no cross-encoder reranking, no multi-agent).
- **Proposed multi-agent RAG:** full system with cross-encoder reranking and multi-agent orchestration.

For each query in the test set ($N = 150$), responses are generated by all configurations. Human evaluators (three independent raters) score each response on a 1–5 Likert scale for factual accuracy, relevance, and completeness. Inter-rater agreement (Krippendorff’s α) was 0.84. In addition, automated faithfulness is measured using the RAGAS faithfulness metric.

Table 6 reports mean scores and standard deviations. The proposed multi-agent RAG achieves a statistically significant improvement over both baselines (paired t -test, $p < 0.01$ for all pairwise comparisons). The reduction in hallucination rate (percentage of responses containing at least one unsupported factual claim) is 30.0 percentage points (from 34.7% to 4.7%) compared to the naive LLM, representing a relative reduction of 86.5%. Compared to the single-prompt RAG, the hallucination rate is reduced by 7.3 percentage points (from 12.0% to 4.7%), representing a relative reduction of 60.8%.

Table 6: Generation quality metrics ($N = 150$). Mean $\pm$ standard deviation, 95% confidence interval, and statistical significance tests.

Configuration	Factual Acc.	Relevance	Faithfulness	Halluc. Rate (%)	Paired t -test (vs. baseline)		
					t stat.	df	p -value
Naive LLM (baseline)	2.81 $\pm$ 0.64 [2.71, 2.91]	3.12 $\pm$ 0.58 [3.03, 3.21]	0.523 $\pm$ 0.087 [0.509, 0.537]	34.7 $\pm$ 5.2 [33.9, 35.5]	—	—	—
Single-prompt RAG	3.96 $\pm$ 0.51 [3.88, 4.04]	4.08 $\pm$ 0.47 [4.00, 4.16]	0.796 $\pm$ 0.062 [0.786, 0.806]	12.0 $\pm$ 3.8 [11.4, 12.6]	4.56	149	< 0.001
Multi-agent RAG (proposed)	4.53 $\pm$ 0.39 [4.47, 4.59]	4.61 $\pm$ 0.38 [4.55, 4.67]	0.912 $\pm$ 0.041 [0.905, 0.919]	4.7 $\pm$ 2.1 [4.4, 5.0]	6.23	149	< 0.001

The gains from multi-agent orchestration over single-prompt RAG are attributable to better query interpretation, source selection, and answer synthesis; the cross-encoder reranking contributes to higher retrieval precision, which directly improves faithfulness and factual accuracy.

6. Conclusion and Future Directions

This paper has presented a data-driven AI-based solution for enterprise knowledge retrieval, grounded in natural language processing and retrieval-augmented generation techniques. The proposed system addresses the challenge of preliminary research bottlenecks during software development and infrastructure deployment phases. Through experimental validation (Section 5), we demonstrate that retrieval-augmented generation with multi-agent orchestration yields accurate and relevant responses from proprietary knowledge bases. Our approach achieves significant improvements over naive LLM inference in factual accuracy, response relevance, and hallucination prevention.

The significance of this work extends beyond its specific corpus, with implications for generative AI adoption in enterprise domains. The modular architecture and extensible design offer a blueprint for organizations to leverage AI capabilities while maintaining control over proprietary information and computational resources. The multi-agent coordination framework provides a viable solution to the increasing complexity of AI-enabled information systems.

Future research directions present promising opportunities for enhancement. Investigating graph-based retrieval techniques, such as Microsoft's GraphRAG, may better capture relationships between knowledge entities compared to current vector-based methods [32]. Reinforcement learning from human feedback could enable interactive improvement and user preference adaptation. Additionally, experimenting with specialized document-answering models available in the Hugging Face ecosystem may improve both retrieval and response quality.

As large language models and embedding technologies continue to evolve, RAG-powered systems will benefit from expanding options for knowledge updates, multimodal content integration, and advanced reasoning capabilities. As these technologies mature, the framework developed in this research provides the flexibility necessary to incorporate advances while maintaining security, reliability, and customizability essential for enterprise deployment.

References

- [1] M. P. Singh Shanmugam. Ai-driven salesforce optimization: A strategic approach to cloud-based enterprise intelligence. In *Proc. 2025 5th Int. Conf. on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, pages 1–6, Mandya, India, 2025. doi: 10.1109/ICERECT65215.2025.11375924.
- [2] K. Shah. Data tier architectures for enterprise saas: A comparative performance study of tenant isolation models. *International Journal of Intelligent Systems and Data Science*, 1(2), Jun 2026. doi: 10.67231/x7z7ys59.
- [3] C. Li, J. Gong, J. Luo, and Z. Qiu. The impact of technology convergence on the sustainable innovation of china's modern manufacturing enterprises: The mediating role of the knowledge base. *Sustainability*, 16(13), 2024. Art. no. 5307.
- [4] Y. S. Kamma. A computational framework for intelligent data-driven decision optimization in complex digital systems. *Research Square*, Jan 2026. doi: 10.21203/rs.3.rs-8662357/v1. preprint.
- [5] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- [6] L. Yan, H. Cai, and Y. Pei. The impact of the variety of enterprises' knowledge base on the complementary and balance effect between independent innovation and collaborative innovation. *Science & Technology Progress and Policy*, 39(2):111–120, 2022.
- [7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

- [8] S. H. Kolla. Deep learning-driven retrieval-augmented generation for enterprise itsm automation: A governance-aligned large language model architecture. *Journal of Computational Analysis and Applications*, 31(4):612–628, 2023.
- [9] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024.
- [10] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. Active retrieval augmented generation. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pages 7969–7992, 2023.
- [11] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. RAGAS: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.
- [12] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 3982–3992, 2019.
- [13] S. Bagde. Ontology-driven integration of urban planning data for enhanced civil engineering applications. In S. Mondal, S. S. Kumar, A. Gupta, and A. d. Barros, editors, *Innovation and Sustainability in Civil Engineering: Shaping Tomorrow’s Infrastructure, Volume 2*, volume 808 of *Lecture Notes in Civil Engineering*, Singapore, 2026. Springer. doi: 10.1007/978-981-95-6354-8_14.
- [14] P. Panwar. Generalized signature method for multivariate time series: A data-driven framework for feature extraction. In *Proc. 2025 Int. Conf. on Advances in Modern Age Technologies for Health and Engineering Science (AMATHE)*, pages 1–8, Shivamogga, India, 2025. doi: 10.1109/AMATHE65477.2025.11081364.
- [15] G. Bachu. Empowering data-driven decision making: The rise of self-service data wrangling in the age of automation. In *Proc. 2025 5th Asian Conf. on Innovation in Technology (ASIANCON)*, pages 1–5, PIMPRI, India, 2025. doi: 10.1109/ASIANCON66527.2025.11280864.
- [16] O. Ladapo. Revolutionizing data preparation and access for visual and multi-modal business analytics. *International Journal of Science and Applied Technology (IJSAT)*, 16(2), 2025.
- [17] S. Vangala. Computational mapping of interdisciplinary intellectual linkages in network science using bibliometric metadata. *International Journal of Intelligent Systems and Data Science*, 1(4), Aug 2026. doi: 10.67231/gtv6nz43.
- [18] M. P. S. Shanmugam. Ai-driven data cleansing and representation learning for crm optimization: A webly-supervised approach. In *Proc. 2025 IEEE 4th Int. Conf. for Advancement in Technology (ICONAT)*, pages 1–6, Goa, India, 2025. doi: 10.1109/ICONAT66879.2025.11362638.
- [19] P. Raichur. Data-driven identification of irregular patterns. In *Proc. 2025 Global Conf. on Information Technology and Communication Networks (GITCON)*, pages 1–5, Belagavi, India, 2025. doi: 10.1109/GITCON65266.2025.11377690.
- [20] S. R. Gantla. Medgenie: Advancing clinical ai through multi-modal medical data fusion and diagnostic-conversational benchmarking. In *Proc. 2025 Int. Conf. on Computational Engineering, Sensing Technology and Management (ICCETM)*, pages 1–6, Sydney, Australia, 2025. doi: 10.1109/ICCETM66557.2025.11557857.

- [21] A. Gausen, B. Mitra, and S. Lindley. A framework for exploring the consequences of ai-mediated enterprise knowledge access and identifying risks to workers. In *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAccT)*, pages 207–220, 2024.
- [22] J. Wang, Y. Hong, J. Wang, J. Xu, Y. Tang, Q. L. Han, and J. Kurths. Cooperative and competitive multi-agent systems: From optimization to games. *IEEE/CAA Journal of Automatica Sinica*, 9(5): 763–783, 2022.
- [23] H. Cai, Y. Su, and J. Huang. *Cooperative Control of Multi-Agent Systems*. Springer, Cham, Switzerland, 2022.
- [24] V. Galuzin, A. Galitskaya, S. Grachev, V. Larukhin, D. Novichkov, P. Skobelev, and A. Zhilyaev. Autonomous digital twin of enterprise: Method and toolset for knowledge-based multi-agent adaptive management of tasks and resources in real time. *Mathematics*, 10(10), 2022. Art. no. 1662.
- [25] H. Du, S. Thudumu, R. Vasa, and K. Mouzakis. A survey on context-aware multi-agent systems: Techniques, challenges and future directions. *arXiv preprint arXiv:2402.01968*, 2024.
- [26] N. Krishnan. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *arXiv preprint arXiv:2504.21030*, 2025.
- [27] C. R. Pamnani. Ai-driven automation for digital hardware design: A multi-agent generative approach. In *Proc. 2025 4th Int. Conf. on Frontiers of Artificial Intelligence and Machine Learning (FAIML)*, pages 26–30, New York, NY, USA, 2025. Association for Computing Machinery. doi: 10.1145/3748382.3748388.
- [28] S. Han, Q. Zhang, W. Jin, and Z. Xu. Llm multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578*, 2024.
- [29] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu. Evaluation of retrieval-augmented generation: A survey. In *CCF Conf. Big Data*, pages 102–120, Singapore, 2024. Springer Nature.
- [30] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation. In *Proc. AAAI Conf. Artificial Intelligence*, volume 38, pages 17754–17762, 2024.
- [31] P. K. C. Nimmalapudi. Bubble: A scalable and efficient bellwether discovery method for large-scale software engineering datasets. In *Proc. 2025 Int. Conf. on Computer, Electrical & Communication Engineering (ICCECE)*, pages 1–9, Kolkata, India, 2025. doi: 10.1109/ICCECE61355.2025.10940624.
- [32] B. Yang and Y. M. Liao. Research on enterprise risk knowledge graph based on multi-source data fusion. *Neural Computing and Applications*, 34(4):2569–2582, 2022.