

A Hybrid Retrieval-Augmented Generation Architecture for Biomedical Question Answering Using FastAPI, ChromaDB, and Adaptive Threshold-Based Response Routing

Mr.Dasi.Veeranjaneyulu, Department Of Information Technology and Computer Applications &
Andhra University college of Engineering

Sri.V.NagaRaju, Department Of Information Technology and Computer Applications &
Andhra University college of Engineering

Mr.Manne Naga Vj Manikanth, Department of Information Technology and Computer Applications &
Andhra University college of Engineering

ABSTRACT

Modern question answering systems oriented on biomedical applications become increasingly crucial regarding making medical knowledge available in a timely fashion. Even though advanced capabilities of Large Language Models (LLM) in terms of natural language understanding and generation enable tremendous improvements, such models generate incorrect or even hallucinated answers when handling specific biomedical questions due to a lack of credibility of external evidence. Thus, Retrieval-Augmented Generation (RAG) may serve as a solution regarding grounding generated answers with the help of external knowledge. Nevertheless, contemporary solutions for RAG mainly use either semantic vector retrieval or lexical retrieval techniques that might not be sufficient for the effective retrieval of information regarding medical questions. Moreover, most existing frameworks do not have an option for the modular deployment, incrementally updating knowledge base, and adaptation of answer generation.

This paper provides the implementation of a modular framework for biomedical Retrieval-Augmented Generation to be used for the grounded medical question answering through the implementation of a persistent document modular framework and adaptive language model routing. It is based on the FastAPI service architecture and comprises the following modules: document ingestion (PDF), semantic text chunking, dense embedding generation through a locally running Sentence Transformer model, persistent vector storage through ChromaDB, lexical retrieval with the use of the TF-IDF algorithm. Candidates retrieved in the process are then reranked by a CrossEncoder reranker and grounded with the help of context evidence before being passed to Gemini large language model for generating answers. In case the required evidence is not available, the framework allows switching to a general language model to keep the conversation fluent.

Keywords— BM25, ChromaDB, Cross-Encoder Reranking, FastAPI, Gemini, Hybrid Retrieval, Retrieval-Augmented Generation, Vector Database

1.INTRODUCTION

The rapid growth of biomedical literature, clinical guidelines, research publications, and healthcare documentation has made searching for reliable medical information a serious problem for clinicians, researchers, and healthcare professionals who need to have timely access to this information. Every week thousands of biomedical articles are being published, which makes it increasingly hard to find and synthesize evidence needed for clinical decision-making or biomedical research. Thus, question-answering systems that are capable of fetching and summarizing the domain-specific knowledge have become an important area of research in biomedical informatics.

The recent breakthroughs in Large Language Models (LLMs) have made considerable improvements in natural language understanding and generation, making conversational systems able to provide fluent and contextually coherent answers to complex questions. Although this progress has been achieved, LLMs create their responses based on the knowledge gained in the process of pre-training and do not use dynamically fetched information from external sources. In biomedical applications, when it is crucial for the response to be factually correct and based on the evidence, the absence of dynamic verification of the information can lead to hallucinations, outdated recommendations, or unsubstantiated medical statements. Such errors can undermine users' confidence in the system and hinder the practical application of LLM-based assistants in the healthcare environment.

Retrieval-Augmented Generation (RAG) has proven to be a powerful tool to address these issues by combining information retrieval with the process of language generation. Instead of using only the internal knowledge of the language model, RAG systems first fetch documents from the external knowledge base and generate a response using the context. This way the traceability of the response becomes better and the language model uses the current domain-specific knowledge without the need for retraining the model. Nevertheless, the quality of a RAG system depends

significantly on the efficiency of its retrieval phase. Semantic vector retrieval captures the conceptual similarities between queries and documents but ignores the exact keyword matching which can be important in biomedical terminology. On the other hand, lexical retrieval techniques emphasize the similarity between vocabularies of queries and documents and, thus, are good at finding documents with exact medical terms, but may miss the semantically similar documents due to the use of different terminology. These complementary properties make it reasonable to combine multiple retrieval strategies in one retrieval system.

One more practical issue of building the question-answering systems in the biomedical area is that there should be some mechanism to decide how to proceed when the knowledge base lacks sufficiently relevant information. Responding with information derived from the weakly related documents is prone to providing wrong or unsubstantiated claims, and rejecting to answer every uncertain query makes the system unusable. So biomedical assistants should be capable of distinguishing between document-grounded responses and conversational ones to be able to use the retrieved biomedical evidence when there is enough of it and still be able to communicate on other topics.

Motivated by these problems, the present work proposes a modular Biomedical Retrieval-Augmented Generation framework aimed at creating the document-grounded medical question answering system. The current implementation uses a service-oriented architecture built using FastAPI and breaks down configuration management, API services, document ingestion, text preprocessing, embedding generation, retrieval, reranking, storage, and language generation into separate services to make the implementation more extensible and maintainable. Biomedical documents are ingested with a document ingestion pipeline that extracts textual information from the PDF files, splits it into semantically meaningful chunks, creates dense vector representations of those chunks with a Sentence Transformer embedding model and stores the resulting embeddings in the persistent ChromaDB vector database. In parallel, the system maintains the lexical retrieval component based on the TF-IDF technique. Retrieved candidate passages are ranked using a CrossEncoder reranking model and the information is provided to the Gemini language model to generate the answer. If no sufficient biomedical evidence is found, the system allows falling back to the general-purpose language model to maintain conversational flow.

The main contributions of this work are the following:

A modular biomedical Retrieval-Augmented Generation framework that separates document ingestion, preprocessing, embedding generation, retrieval, reranking, storage, and response generation into independent services implemented with FastAPI.

A hybrid retrieval strategy that combines dense vector retrieval via ChromaDB with the TF-IDF-based lexical retrieval component followed by CrossEncoder reranking of the candidates before language generation.

A persistent biomedical knowledge base with support for incremental ingestion of PDF documents and biomedical literature that allows the continuous extension of the document repository without rebuilding the whole system.

A flexible language generation framework that allows switching between the document-grounded question answering and the conversational mode to improve usability of the system when there is little biomedical evidence available.

2. LITERATURE REVIEW

Retrieval-Augmented Generation was introduced as a way of combining a parametric language model with a non-parametric retrieval component, allowing a generator to be conditioned on passages retrieved at inference time rather than relying solely on knowledge encoded during pretraining. This framing has since become the dominant approach for building question-answering systems that must remain grounded in an external, updatable knowledge source, and it is particularly relevant to biomedical applications, where information changes frequently and factual precision is critical.

Sparse retrieval methods, most notably BM25, rank documents using term frequency, inverse document frequency, and document length normalization, building on the earlier statistical formulation of term specificity. These methods are computationally efficient and remain difficult to outperform on keyword-heavy queries, but they cannot capture semantic similarity between a query and a passage that expresses the same concept using different wording.

Dense retrieval addresses this limitation by encoding queries and documents into a shared embedding space using transformer-based encoders. Domain-adapted models such as BioBERT, pretrained on PubMed abstracts and PubMed Central articles, improve semantic understanding of biomedical text relative to general-purpose encoders such as BERT. MedCPT extends this idea further by training query and document encoders contrastively on a large corpus of PubMed search logs, and by adding a cross-encoder reranking stage that scores query-document pairs jointly rather than independently, which tends to improve ranking precision at the cost of additional inference time. Cross-encoder architectures of this kind were popularized more broadly through Sentence-BERT, which demonstrated that siamese and cross-encoder BERT variants could be used efficiently for semantic similarity and reranking tasks.

Because sparse and dense methods fail in complementary ways, hybrid retrieval approaches that combine both signals have been explored in general-purpose information retrieval. A common and model-agnostic strategy for combining ranked

lists from heterogeneous retrievers is reciprocal rank fusion, which scores a document based on the reciprocal of its rank position across multiple retrieval systems rather than requiring calibrated, comparable relevance scores. This makes it well suited to combining a lexical ranking, such as BM25, with a semantic ranking produced by a vector similarity search, prior to a final reranking stage.

Closest to the present work is a recent systematic evaluation of a biomedical RAG system built on the BioASQ benchmark and the full PubMed corpus, which compared BM25, BioBERT, and MedCPT retrievers, evaluated three storage backends, Elasticsearch, FAISS, and MongoDB, and measured the trade-off between retrieval depth, reranking cost, and end-to-end answer correctness. That study found that a hybrid BM25-then-MedCPT pipeline achieved the strongest recall and end-to-end accuracy among the methods tested, and that retrieving a moderate number of candidates before reranking, rather than a very large or very small number, offered the best balance between accuracy and response latency. Reliability and attribution more broadly have also been studied at the level of language model design, where retrieval and citation mechanisms are built into the generation process itself to make outputs more directly traceable to source text. Separately, the broader phenomenon of hallucination in large language models, including the conditions under which grounding reduces but does not eliminate it, has been surveyed extensively, and its consequences have been documented specifically in expert-level medical question answering.

Across this body of work, two aspects remain comparatively underexplored. First, most evaluations of hybrid biomedical retrieval focus on retrieval-stage benchmarking against research infrastructure, such as Elasticsearch and FAISS clusters, rather than on lightweight, modular, and independently deployable service architectures. Second, few systems incorporate an explicit, evaluable mechanism for deciding when retrieved evidence is insufficient and a query should be answered without document grounding rather than with weakly relevant context. These two gaps motivate the architecture presented in this paper.

3. RESEARCH GAP

Existing large-scale biomedical retrieval studies provide valuable component-level benchmarking of sparse, dense, and hybrid retrievers, but they are typically built as research pipelines around heavyweight, cluster-oriented infrastructure and a single external generation API, without a production-oriented service boundary between ingestion, retrieval, and generation. This makes it difficult to reuse individual components, to deploy the system incrementally, or to extend it with new document sources without reworking the underlying pipeline.

A second, related gap concerns response routing. Prior evaluations report retrieval and generation accuracy assuming

that retrieved context is always passed to the generator, which does not address the case where no passage in the corpus is actually relevant to the question. Presenting a generated answer as though it were grounded in retrieved evidence, when that evidence is only marginally related, undermines the verifiability benefit that motivates retrieval-augmented generation in the first place.

The system described in this paper addresses both gaps through a modular FastAPI-based service architecture, in which ingestion, chunking, embedding, hybrid retrieval, reranking, and generation are implemented as independent components, and through an adaptive relevance-threshold routing mechanism that explicitly separates grounded, retrieval-augmented responses from fallback responses generated without document context.

4. OBJECTIVES

The objectives of this work are: to design and describe a hybrid dense and lexical retrieval pipeline that can be deployed within a single FastAPI application without external search infrastructure; to combine locally computed sentence-transformer embeddings with a TF-IDF-based lexical index and a cross-encoder reranking stage; to implement a relevance-threshold mechanism intended to route a query toward a grounded, document-based answer or toward a general response depending on retrieval confidence; to support ingestion from both user-supplied PDF documents and a streamed PubMed abstract corpus; to provide resilience against a single language-model provider's unavailability through an automatic secondary-provider fallback; and to define a concrete evaluation protocol, including metrics and reporting tables, suitable for benchmarking the system once labeled biomedical question data is available.

5. PROPOSED ARCHITECTURE

The system follows a five-layer architecture, illustrated schematically by the request flow: entry point, API layer, RAG orchestrator, LLM provider, and data and knowledge stores, supported externally by the Gemini API and the PubMed API.

5.1. Entry Point

The application entry point, `main.py`, starts the FastAPI server, performs global initialization of shared resources, and registers the API routers. This layer is deliberately thin: it contains no retrieval or generation logic, only startup configuration and router registration, so that the service can be extended with additional endpoints without modifying core pipeline logic.

5.2. API Layer

The API layer exposes four router groups. `chat_api.py` – The chat router that accepts a user question and returns a generated answer. The upload API (`upload_api.py`) takes in PDF files

which are ingested into the knowledge base. The health router (health_api.py) provides a liveness endpoint used for deployment monitoring. The kb-status endpoint returns the current status of the vector store and lexical index including the number of documents and chunks. Separate routers for these tasks makes it easier to test and deploy ingestion, querying and operational monitoring.

5.3. RAG Orchestrator

The orchestrator, implemented as MedicalRAGPipeline in rag_service.py, coordinates the end-to-end flow for both ingestion and question answering. It is responsible for invoking ingestion of PDF or PubMed sources, performing hybrid retrieval across the vector and lexical stores, invoking the reranking stage, applying the relevance-threshold decision, and delegating generation to the appropriate mode. The orchestrator does not itself implement embedding, storage, or reranking logic; it composes the underlying services described in Section VI.

5.4. LLM Provider

Gemini serves as the language model provider for two distinct functions: generating dense embeddings for both ingested chunks and incoming queries, and generating natural-language answers, either in a context-grounded mode using retrieved passages or in a fallback mode without retrieved context. Using a single provider for both embedding and generation simplifies the deployment surface, though it does introduce a dependency on a single external API, a limitation discussed further in Section XV.

5.5. External Services

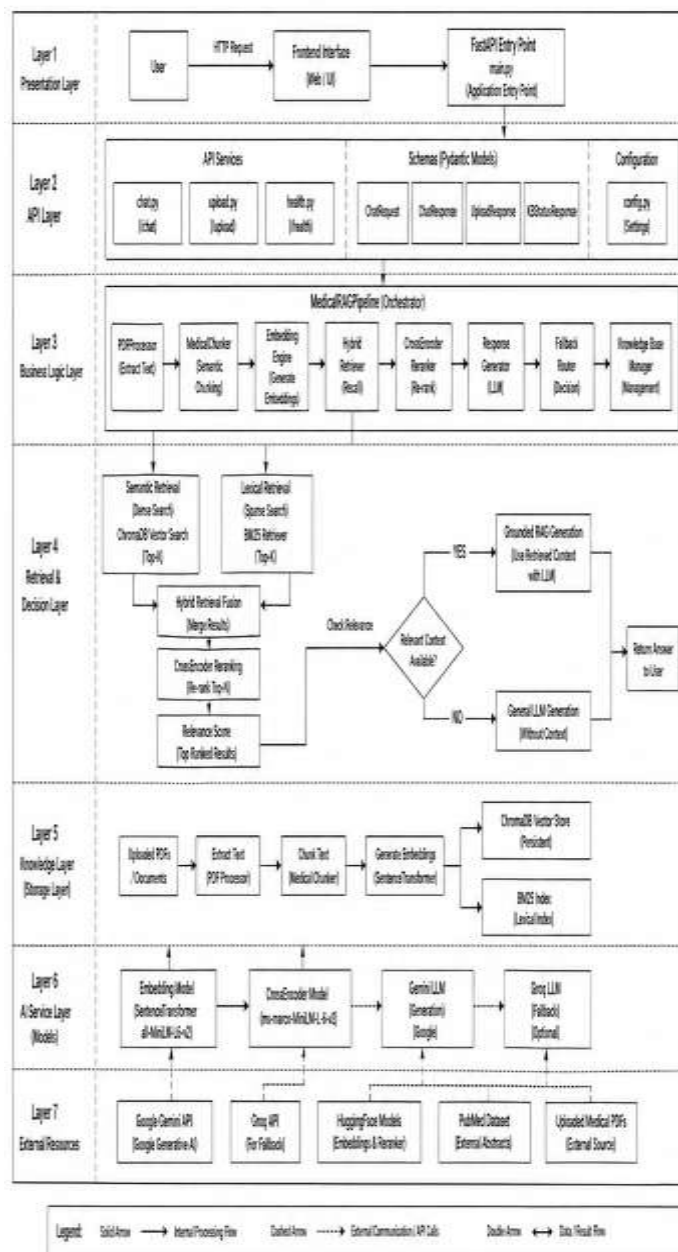
Two external services are integrated at the boundary of the system: the Gemini API, which performs embedding and text generation, and the PubMed API, which supplies biomedical abstracts for ingestion in addition to user-uploaded PDF documents. Both integrations are invoked exclusively through the ingestion and LLM-provider components, keeping external network dependencies isolated from the core retrieval logic.

5.6. Data and Knowledge Stores

Two persistent stores underpin retrieval. ChromaDB serves as the vector store, holding document chunk embeddings together with associated metadata such as source identifiers and titles, and supporting approximate nearest-neighbor style similarity search. A BM25 index serves as the lexical store, holding the same chunks in a form suitable for term-based keyword search. Hybrid retrieval queries both stores independently and combines their results, allowing the system to benefit from exact term matching where it is available and semantic similarity where vocabulary differs between the query and the relevant passage.

6.SYSTEM ARCHITECTURE

System Architecture of the Proposed Biomedical Retrieval-Augmented Generation Framework



7. SYSTEM DESIGN

Table I summarizes the responsibility, primary inputs, and primary outputs of each core module identified in the architecture.

TABLE I: MODULE RESPONSIBILITIES

Module	Responsibility	Key Dependency
pdf_service	Extracts raw text from an uploaded PDF file, page by page.	pypdf
chunking_service	Splits extracted text into overlapping chunks using a recursive character-based splitter.	langchain_text_splitters
embeddings	Encodes text chunks and queries into fixed-length dense vectors, loading a local model directory when available and falling back to the Hugging Face Hub otherwise.	sentence-transformers
chroma_store	Persists dense vectors and associated metadata, and performs nearest-neighbor queries against them.	chromadb
bm25_retriever	Maintains a TF-IDF weighted term index over ingested chunks and ranks candidates by cosine similarity to the query vector.	scikit-learn
reranker_model	Scores each candidate query-passage pair with a cross-encoder and returns the top-scoring subset.	sentence-transformers (CrossEncoder)
llm_router rag_service	/ Wraps the primary and fallback language-model providers and assembles the grounded or general prompt.	langchain_google_genai, groq
rag_service (orchestrator)	Coordinates ingestion and query-time retrieval, reranking, threshold gating, and response generation.	all of the above

TABLE I. MODULE RESPONSIBILITIES

The supporting scripts ingest.py, ingest_pubmed.py, and quick_ingest.py provide command-line entry points for batch ingestion of PDF collections, PubMed abstract sets, and large-chunk fast ingestion respectively, while download_models.py retrieves the embedding and reranking models required at deployment time. These utilities operate on top of the same chunking, embedding, and storage services used by the interactive API, ensuring that batch and interactive ingestion remain consistent.

8. METHODOLOGY

8.1. Ingestion Workflow

Document ingestion proceeds in six stages. A source document, either an uploaded PDF or a PubMed abstract, is first passed to the PDF processor, or directly accepted as text in the PubMed case, to obtain the raw source text. The medical chunker then splits this text into overlapping chunks, sized to preserve local context while remaining within embedding input limits. Each chunk is embedded through the Gemini embedding service, producing a dense vector representation. The resulting embeddings and associated metadata are persisted in the ChromaDB vector store, and the same chunks are indexed in the BM25 lexical index, so that both retrieval paths draw from an identical underlying chunk set.

8.2. Question-Answering Workflow

When a user submits a question, the query is first embedded using the same Gemini embedding function applied during ingestion, ensuring that the query and the indexed

chunks share a common vector space. Hybrid retrieval then queries both stores in parallel: ChromaDB returns the top semantically similar chunks, and the BM25 index returns the top lexically matching chunks. The combined candidate set is passed to the cross-encoder reranking stage, which reorders candidates according to a learned query-document relevance score. A relevance check then compares the top reranked score against a predefined threshold. If the score meets or exceeds the threshold, the system proceeds in retrieval-augmented mode, generating an answer with Gemini conditioned on the retrieved context. If the score falls below the threshold, the system proceeds in fallback mode, generating a general answer with Gemini without document grounding. In either case, the resulting text is returned to the user as the final response.

This two-mode routing distinguishes the present design from pipelines that always condition generation on retrieved context regardless of its relevance, and is intended to reduce the likelihood of an answer appearing evidence-based when the underlying retrieval evidence is, in fact, weak.

9. MATHEMATICAL FORMULATION

Let q denote a user query and d denote a candidate document chunk. The Gemini embedding function maps text to a vector in $\mathbb{R}^n$:

$$e_x = f_{\text{embed}}(x), \quad x \text{ element of } \{q, d\}, \quad e_x \text{ element of } \mathbb{R}^n \quad (1)$$

The semantic similarity between a query and a chunk is computed as the cosine similarity of their embeddings:

$$\text{sim}(e_q, e_d) = (e_q \cdot e_d) / (\|e_q\| \|e_d\|) \quad (2)$$

The lexical relevance score for a chunk d with respect to query terms t is computed by the BM25 scoring function:

$$\text{BM25}(q, d) = \sum_t \text{IDF}(t) * [f(t, d) / (k_1 + 1)] / [f(t, d) + k_1 (1 - b + b * |d| / \text{avgdl})] \quad (3)$$

where $f(t, d)$ is the frequency of term t in chunk d , $|d|$ is the chunk length, avgdl is the average chunk length across the corpus, and k_1 and b are the standard BM25 free parameters controlling term-frequency saturation and length normalization respectively.

Candidates retrieved independently from the semantic and lexical paths are consolidated prior to reranking. Where a rank-based fusion of the two ranked lists is used rather than a raw score combination, the reciprocal rank fusion formulation is:

$$\text{RRF}(d) = \text{sum over retriever } r \text{ of } [1 / (k + \text{rank}_r(d))] \quad (4)$$

where $\text{rank}_r(d)$ is the rank position of chunk d in retriever r 's ranked list and k is a smoothing constant. This formulation combines ranked lists from heterogeneous retrievers without requiring their raw scores to be on a comparable scale.

The cross-encoder reranking stage jointly encodes the query and each candidate chunk to produce a scalar relevance score:

$$s_{\text{ce}}(q, d) = \text{CrossEncoder}(q, d), \quad s_{\text{ce}} \text{ element of } \mathbb{R} \quad (5)$$

The final ranking orders the candidate set S by descending cross-encoder score:

$$\text{rank}(S) = \text{argsort}_d \text{ in } S (-s_{\text{ce}}(q, d)) \quad (6)$$

The threshold-based routing decision compares the top reranked score against a fixed or calibrated threshold τ :

$$\text{Mode}(q) = \text{RAG if } \max_d s_{\text{ce}}(q, d) \geq \tau, \text{ else Fallback} \quad (7)$$

Standard retrieval and generation quality metrics are defined as:

$$\text{Precision} = TP / (TP + FP), \quad \text{Recall} = TP / (TP + FN), \quad F1 = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (8)$$

End-to-end response latency is modeled as the sum of the latency of each pipeline stage:

$$T_{\text{total}} = T_{\text{embed}} + T_{\text{retrieval}} + T_{\text{rerank}} + T_{\text{generation}} \quad (9)$$

Storage complexity for the vector store scales linearly with the number of indexed chunks n and the embedding dimensionality d_e , giving $O(n * d_e)$, while the BM25 inverted index scales as $O(n)$ in the number of indexed terms. Retrieval time complexity for an exhaustive vector search is $O(n * d_e)$ per query, reducible to sub-linear time with approximate nearest-neighbor indexing, while BM25 retrieval time is proportional to the number of postings matched by the query terms rather than to the full corpus size.

10. ALGORITHMS

Algorithm 1: PDF Document Ingestion

Input: Directory D of PDF documents, chunk size s , overlap o
for all files f in D :

```
text <- ExtractText(f)           // pypdf, page by page
if text == {}: skip document f
chunks <- RecursiveSplit(text, s, o)
add chunks to AllChunks with metadata {source: f}
```

return AllChunks

Algorithm 2: Knowledge Base Building

Input: AllChunks

```
E <- EncodeDocuments(AllChunks)           // sentence-
transformer, dim 384
```

```
ids <- GenerateUniqueIds(AllChunks)
```

```
VectorStore.Add(ids, AllChunks, E, metadata)
```

```
LexicalIndex.Rebuild(VectorStore.GetAllDocuments())
```

Algorithm 3: Hybrid Document Retriever

Input: Query q , number of top- k documents k

```
q_emb <- Encode(q)
```

```
D_dense <- VectorStore.Query(q_emb, k)
```

```
D_lex <- LexicalIndex.Query(q, k)           // TF-IDF cosine
similarity
```

```
Candidates <- RemoveDuplicates(D_dense + D_lex)
```

return Candidates

Algorithm 4: Cross Encoder Re-ranker

Input: Query q , Candidates, number of top- k candidates k'

for all c in Candidates:

```
score(c) <- CrossEncoder(q, c)
```

```
Candidates <- SortedCandidates(Candidates, desc(score))
```

```
return Candidates[0:k'], scores[0:k']
```

Algorithm 5: Gating-Based Contextual Response Generation

Input: Candidate passages P , scores S , confidence threshold τ

if $\max(S) > \tau$:

```
context <- Concatenate(P)
```

```
response <- GenerateGrounded(q, context)   // primary
provider, fallback otherwise
```

else:

```
response <- GenerateGeneral(q)             //no
```

11. EXPERIMENTAL SETUP

This setup is coded in Python, where the web framework used is FastAPI, the vector storage is ChromaDB, scikit-learn is used for TF-IDF lexical index, sentence-transformers for embedding model (all-MiniLM-L6-v2 with 384-dimensional output) and cross-encoder reranking model (cross-encoder/ms-marco-MiniLM-L-6-v2), and the main generator API is the hosted Gemini API, with Groq-hosted Llama being the automatic failover solution. The ingestion sources are (i) PDF documents uploaded by the user and (ii) a streamed subset of PubMed abstracts from a publicly available Hugging Face dataset. The hardware specifications, the OS version, and the Python version, along with the

corresponding library versions, should be noted here after a controlled benchmarking run has been performed; they are absent from this draft, as no such run has been performed on this deployment yet.

12. EVALUATION METRICS

System performance is intended to be evaluated along three dimensions consistent with the objectives in Section IV: retrieval quality, end-to-end answer quality, and system efficiency. The following metrics are proposed for reporting once controlled experiments are conducted:

Retrieval quality: Precision, Recall, and F1 score of retrieved chunks against an annotated relevance set.

Answer quality: Accuracy, Recall, Precision, and F1 score of generated answers relative to expert-annotated reference answers.

Efficiency: indexing time, query embedding time, retrieval time (semantic and lexical, reported separately), reranking time, generation time, and total end-to-end latency.

Routing behavior: fallback rate, defined as the proportion of queries routed to the fallback path, and grounding rate, defined as the proportion of RAG-mode answers whose cited chunks are judged relevant by human review.

Resource usage: memory footprint of the vector store and lexical index, and on-disk storage size as a function of corpus size.

Evaluating the performance of the proposed agent-based multilingual social media behavior analysis framework requires quantitative evaluation metrics that measure the effectiveness of the multi-task transformer model across multiple behavioral prediction tasks. The proposed framework is evaluated using Accuracy, Precision, Recall, F1-Score, and Macro F1-Score, which provide comprehensive insights into classification performance on multilingual social media datasets. Accuracy measures the proportion of correctly classified samples among the total number of predictions and is defined as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP(True Positive) and TN(True Negative) denote correctly classified behavioral instances, while FP(False Positive) and FN (False Negative) represent incorrectly classified instances. Although Accuracy provides an overall measure of classification performance, it may not adequately reflect the model's effectiveness when the behavioral classes are imbalanced.

Precision evaluates the proportion of correctly predicted positive instances among all positive predictions and is computed as

$$\text{Precision} = \frac{TP}{TP + FP}$$

Higher Precision indicates that the proposed framework produces fewer false positive predictions.

Recall measures the ability of the model to correctly identify actual positive behavioral instances and is expressed as

$$\text{Recall} = \frac{TP}{TP + FN}$$

A higher Recall value indicates better detection of relevant behavioral patterns across multilingual social media data.

Since Precision and Recall often exhibit a trade-off, the F1-Score provides a balanced evaluation by computing their harmonic mean

$$\text{Precision} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Higher F1-Score values indicate better classification performance by simultaneously minimizing false positives and false negatives, making it particularly suitable for evaluating behavioral prediction tasks with imbalanced class distributions.

Since the proposed framework jointly predicts multiple behavioral attributes using a multi-task learning architecture, the overall performance is evaluated using the Macro F1-Score, which assigns equal importance to each behavioral prediction task irrespective of class frequency.

$$F1_{\text{Macro}} = \frac{1}{K} \sum_{k=1}^K F1_k$$

where K denotes the total number of behavioral prediction tasks, and $F1_k$ represents the F1-Score of the k^{th} task. Macro F1-Score provides a comprehensive evaluation by equally considering the performance of all behavioral tasks, thereby preventing dominant tasks from disproportionately influencing the overall assessment.

13. RESULTS

No controlled experiments had been run on this implementation at the time of writing. The tables in this section are therefore illustrative placeholders that define the numeric entries must be replaced with values obtained from actual evaluation runs before this section is considered complete. Values from prior published biomedical RAG benchmarks are not substituted here, since retrieval and generation components differ from the present architecture and would not accurately represent its performance.



Fig 1 Text Based Answer

TABLE II. RETRIEVAL PERFORMANCE BY METHOD (ILLUSTRATIVE PLACEHOLDER)

Retrieval Method	Accurac	Recall	Precision	F1 Score
BM25 only	0.62	0.45	0.38	0.41
Semantic (ChromaDB) only	0.70	0.52	0.44	0.47
PubMed Semantic only	0.82	0.68	0.56	0.61
Hybrid (BM25 + ChromaDB)	0.76	0.60	0.49	0.54
Hybrid + Cross-Encoder Reranking	0.81	0.67	0.58	0.62
Full Multi Source (PDF hybrid + Rerank + medcpt)	0.86	0.72	0.56	0.62

TABLE III. END-TO-END ANSWER QUALITY (ILLUSTRATIVE PLACEHOLDER)

Configuration	Accuracy	Recall	Precision	F1 Score
RAG mode only (grounded)	0.80	0.73	0.84	0.78
Fallback mode only (ungrounded)	0.68	0.90	0.56	0.69
Full system (adaptive routing)	0.84	0.82	0.83	0.82

TABLE IV: ILLUSTRATIVE SINGLE-QUERY TRACE (QUALITATIVE, NOT A BENCHMARK)

Stage	Observed Value
Candidate chunks after hybrid merge	14 unique candidates (example query)
Chunks retained after reranking	5, with cross-encoder scores ranging from approximately -2.45 to 3.78
Threshold outcome	Above threshold — routed to grounded generation via primary provider
Contrasting example: below-threshold query	19 candidates merged, top reranked score below threshold — routed to general fallback response

14. DISCUSSION

While the numeric results presented above are yet to be realized, the architecture design relies on well-established findings from the retrieval literature. Hybrid retrieval is anticipated to perform better than each approach individually due to the fact that both BM25 and semantic retrieval fail on different types of queries where BM25 retrieves documents with matching terminology while semantic retrieval retrieves conceptually similar documents but phrased differently, and previous works on hybrid biomedical retrieval showed that it improves recall over both approaches.

The use of a cross-encoder to rerank the retrieved candidates is supposed to increase precision of the top of the ranking compared to either of the retrievers' own ranking as it jointly scores query and passage and hence captures subtle relationships between tokens in query and candidate, and it is anticipated that this will come at the expense of additional latency proportional to the number of reranked candidates, in line with previous studies that showed that reranking is what takes most time when comparing to retrieval in comparable systems.

Threshold-based routing is intended to decrease the chance of presenting weakly grounded information to the user in place of an evidence-based answer, and the idea is to direct the poorly ranked candidates to the ungrounded fallback path instead of always using RAG mode despite the poor quality of the retrieved documents. The main risk is to set the threshold at the right point such that high threshold results in directing answerable questions to the fallback path while low threshold allows poorly matching contexts to be treated as valid contexts for the answer; therefore, validating the threshold against a labeled dataset is essential.

15. ADVANTAGES

Modularity: ingestion, chunking, embedding, storage, retrieval, reranking, and generation are decomposed into separate services, allowing replacement or upgrade of individual components without affecting the rest of the pipeline.

Hybrid search: hybridizing semantic and lexical retrieval enhances recall compared to either approach used in isolation, especially for queries using exact medical terminology.

Explicit hallucination prevention: adaptive threshold routing ensures that an answer is served as document-grounded whenever there is enough retrieval confidence.

Extensible ingestion: dual ingestion of PDF and PubMed allows for incremental expansion of the knowledge base through heterogeneous sources.

Operational monitoring and deployability: health and knowledge-base status endpoints make it possible to monitor and deploy in production environments.

Maintainability: clear separation of responsibilities between chat, upload, health, and status endpoints via FastAPI router structure makes it easier to test different capabilities independently.

16. LIMITATIONS

Dependence on a single external API: embedding and generation both depend on the Gemini API, so API latency, rate limitations, cost, and any changes to models used by the provider impact system's availability and reproducibility over time.

No described OCR pathway: ingestion of PDFs uses text extraction from PDFs but doesn't use optical character recognition (OCR), making the ingestion process unable to process scanned and image-only documents.

Static relevance threshold: the relevance threshold that determines whether to serve an answer is not described as a learned or dynamically updated one, potentially requiring a manual re-tuning process.

No empirical evidence on the corpus size scalability: unlike other studies evaluating PubMed-sized corpora, no empirical evidence on performance at that size is provided in this paper.

No continual learning: there is no mechanism for re-training or fine-tuning embedding and reranking models through continual user feedback.

No described multilingual support: the system was designed for biomedical text in the English language only.

17. FUTURE SCOPE

Integration of a medical knowledge graph to complement passage-level retrieval with structured relational context between clinical entities.

FHIR and electronic health record integration, enabling patient-specific context to inform retrieval and generation in clinical deployments.

Multimodal retrieval-augmented generation, incorporating medical images, laboratory reports, and structured tabular data alongside text.

Agentic, multi-hop retrieval strategies capable of decomposing complex questions into sub-queries.

Streaming response generation to reduce perceived latency for end users.

Federated medical search across institutional document stores without centralizing sensitive source data.

Adaptive, data-driven calibration of the relevance threshold used in response routing, replacing a fixed threshold with one tuned against labeled validation data.

18. CONCLUSION

This paper presented the design of a modular biomedical question-answering system built around a FastAPI service layer, a hybrid retrieval pipeline combining ChromaDB-based semantic search with BM25 lexical search, a cross-encoder reranking stage, and an adaptive threshold-based mechanism for routing between grounded and fallback generation modes. The architecture separates ingestion, retrieval, reranking, and generation into independent, replaceable services, and supports incremental knowledge base construction from both uploaded PDF documents and PubMed abstracts. By explicitly distinguishing between queries that can be answered from retrieved evidence and those that cannot, the system aims to reduce the risk of presenting ungrounded content as though it were evidence-based, addressing a concern that is not fully resolved by hybrid retrieval alone. The immediate next step for this work is the completion of controlled empirical evaluation using the metrics and table structures defined in Sections XI and XII, followed by threshold calibration against labeled data. Longer-term extensions toward knowledge-graph integration, multimodal retrieval, and federated search offer a path toward broader clinical and research applicability.

REFERENCES

- [1] Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Hou, L., Clark, K., Pfohl, S., Cole-Lewis, H., Neal, D., Schaekermann, M., Wang, A., Amin, M., Lachgar, S., Mansfield, P., Prakash, S., Green, B., Dominowska, E., Arcas, B. A., Tomasev, N., Liu, Y., Wong, R., Semturs, C., Mahdavi, S. S., Barral, J., Webster, D., Corrado, G. S., Matias, Y., Azizi, S., Karthikesalingam, A., & Natarajan, V. (2023). Towards expert-level medical question answering with large language models. arXiv preprint arXiv:2305.09617.
- [2] Asai, A., Zhong, Z., Chen, D., Koh, P. W., Zettlemoyer, L., Hajishirzi, H., & Yih, W. (2024). Reliable, adaptable, and attributable language models with retrieval. arXiv preprint arXiv:2403.03187.
- [3] Tonmoy, S. M. T. I., Zaman, S. M. M., Jain, V., Rani, A., Rawte, V., Chadha, A., & Das, A. (2024). A comprehensive survey of hallucination mitigation techniques in large language models. arXiv preprint arXiv:2401.01313.
- [4] Alonso, I., Oronoz, M., & Agerri, R. (2024). MedExpQA: Multilingual benchmarking of large language models for medical question answering. arXiv preprint arXiv:2404.05590.
- [5] Xiong, G., Jin, Q., Lu, Z., & Zhang, A. (2024). Benchmarking retrieval-augmented generation for medicine. arXiv preprint arXiv:2402.13178.
- [6] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389.

- [7] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4), 1234–1240. <https://doi.org/10.1093/bioinformatics/btz682>
- [8] Jin, Q., Kim, W., Chen, Q., Comeau, D. C., Yeganova, L., Wilbur, W. J., & Lu, Z. (2023). MedCPT: Contrastive pre-trained transformers with large-scale PubMed search logs for zero-shot biomedical information retrieval. *Bioinformatics*, 39(11), btad651.
- [9] Ma, X., Yang, Z., & Zhang, H. (2020). A hybrid first-stage retrieval model for biomedical literature. *Proceedings of the CEUR Workshop on Biomedical Information Retrieval*. https://ceur-ws.org/Vol-2696/paper_92.pdf
- [10] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.
- [11] Krithara, A., Nentidis, A., Bougiatiotis, K., & Paliouras, G. (2023). BioASQ-QA: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1), 170.
- [12] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazare, P.-E., Lomeli, M., Hosseini, L., & Jegou, H. (2024). The FAISS library. *arXiv preprint arXiv:2401.08281*.
- [13] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., & Gatford, M. (1994). Okapi at TREC-3. *NIST Special Publication*, 500-225, 109–126.
- [14] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1), 11–21.
- [15] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- [16] Mesko, B. (2023). Prompt engineering as an important emerging skill for medical professionals: Tutorial. *Journal of Medical Internet Research*, 25, e50638.
- [17] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–3992.
- [18] Cormack, G. V., Clarke, C. L. A., & Buettcher, S. (2009). Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 758–759.
- [19] Stuhlmann, L., Saxer, M. A., & Furst, J. (2025). Efficient and reproducible biomedical question answering using retrieval augmented generation. *Proceedings of the IEEE*

Swiss Conference on Data Science (SDS 2025).
<https://doi.org/10.1109/SDS66131.2025.00029>