IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

INSIGHTDOC: Document Question Answering System Using Retrieval-Augmented Generation And Local Large Language Model

Prasad Nanasaheb Sumb, Kaif Shah, Mayur Uttam Buchkul, Rohit Dilip Bilwal

Student of the BE Computer Science & Engineering department in the International Centre of Excellence in Engineering and Management , Chh.Sambhajinagar, Maharashtra India 431136 .

Supervisor : (Assistant Professor , HOD). S.A.Chechare

Department of Computer Science & Engineering, ICEEM, Chhatrapati Sambhajinagar, Maharashtra, India

Abstract: The exponential accumulation of digital documentation across professional domains has rendered manual information retrieval both impractical and error to prone. Cloud-based question-answering (QA) solutions, while capable, raise significant concerns regarding data confidentiality, unpredictable hallucinated outputs, and reliance on continuous internet access. This paper introduces InsightDoc, a locally executed, privacy-centric document QA framework that couples Term Frequency-Inverse Document Frequency (TF-IDF) retrieval with a locally hosted, lightweight large language model (LLM) to generate answers strictly from user-supplied PDF documents. The proposed architecture operates entirely offline by deploying the Qwen 1.8B language model through the Ollama runtime environment. The system extracts and normalizes textual content from PDFs, partitions it into sentence-bounded chunks, constructs a TF-IDF sparse index, retrieves the most contextually relevant passages via cosine similarity, and delivers the assembled evidence to the LLM for grounded answer generation. Evaluation on a 50-question benchmark yields a retrieval Precision@5 of 0.76, an F1 Score of 0.73, and a mean retrieval latency well below 0.2 seconds, with zero observed hallucinations. InsightDoc provides a practical and resource-efficient baseline for offline document QA, ensuring verifiable data sovereignty.

Index Terms - Retrieval-Augmented Generation (RAG), TF-IDF Vectorization, Local Large Language Model, Document Question Answering, Cosine Similarity, Information Retrieval, Ollama, Privacy-Preserving AI.

I. INTRODUCTION

Organizations spanning the legal, healthcare, academic, and financial sectors routinely manage large repositories of PDF-formatted documents as their primary vehicles of institutional knowledge. Extracting specific facts from these repositories requires considerable human effort and domain familiarity — a process that is inherently susceptible to oversight and inconsistency. As document volumes grow, the cognitive burden on practitioners intensifies, creating a compelling demand for automated, reliable document comprehension tools.

Contemporary automated QA systems predominantly rely on cloud-based large language model APIs, such as OpenAI GPT-4 and Anthropic Claude. Although these platforms offer impressive linguistic comprehension, they introduce three critical limitations for enterprise deployments handling sensitive information. First, routing confidential documents through remote servers poses serious risk to data

privacy and regulatory compliance — a concern that is particularly acute under frameworks such as the GDPR and HIPAA. Second, general-purpose LLMs are prone to hallucination, producing responses that are linguistically fluent yet factually unsupported by the source material. Third, subscription-based API models entail recurring operational costs and require uninterrupted internet access, making them unsuitable for environments with limited bandwidth, air-gapped networks, or restricted resources.

These shortcomings collectively motivate InsightDoc — an offline, privacy-first QA system designed to extract factual answers exclusively from user-supplied PDFs. By uniting TF-IDF lexical retrieval with a locally executed, quantized LLM, the architecture ensures that every generated response is directly traceable to a specific document passage, effectively suppressing hallucination while preserving complete data confidentiality. Delivered through an interactive Streamlit interface, the system requires neither cloud credentials nor specialized GPU hardware, making it accessible to non-technical practitioners in regulated industries, educational institutions, and resource-constrained research settings.

The primary contribution of this work lies in demonstrating that an effective document question answering system can be deployed entirely offline using lightweight retrieval techniques and CPU-compatible language models, thereby addressing privacy, cost, and accessibility challenges simultaneously.

II. RELATED WORK

Research in automated document QA has progressed along two broad trajectories: extractive approaches that identify answer spans within source texts, and generative approaches that synthesize responses using neural language models. Early extractive systems, most notably DrQA [1], employed TF-IDF retrieval to shortlist candidate passages before applying a BiDAF reader network to localize answer boundaries. While computationally efficient, such models struggled with abstractive questions requiring evidence synthesis across non-contiguous passages.

The emergence of transformer architectures, particularly BERT [2] and its fine-tuned derivatives, markedly advanced machine reading comprehension on standard benchmarks such as SQuAD. Commercial document QA platforms — ChatPDF and Adobe Acrobat AI, among them — leverage cloud-hosted transformer models to support conversational PDF interaction. These tools, however, transmit document content to remote servers, enforce API rate limits, and lack the contextual guardrails necessary to prevent responses that stray beyond the document's scope. These constraints significantly limit their utility in privacy-sensitive deployments.

Lewis et al. [3] formally established the Retrieval-Augmented Generation (RAG) paradigm, coupling a dense passage retriever with a sequence-to-sequence generator to anchor generation within retrieved evidence. Subsequent research replaced sparse TF-IDF retrievers with dense embedding models — notably Dense Passage Retrieval (DPR) and Sentence-BERT (SBERT) [4] — to overcome vocabulary mismatch limitations. These dense retrievers, while semantically expressive, demand GPU infrastructure and substantial memory resources that are unavailable in lightweight local deployments.

The recent availability of quantized small language models — including Qwen [5] and Llama-2 — has made LLM inference feasible on consumer-grade CPU hardware. The Ollama runtime further simplifies the deployment of such models in local environments. InsightDoc occupies a practical niche between dense retrieval research and real-world local deployment by pairing proven TF-IDF retrieval with CPU-compatible LLM generation — a configuration that eliminates GPU dependency while preserving a functional RAG pipeline.

III. PROPOSED SYSTEM

3.1 System Overview

InsightDoc is structured as a modular, offline document QA framework organized around two operationally distinct phases. During the offline indexing phase, the system ingests one or more PDF files, extracts and normalizes their textual content, partitions the text into overlapping sentence-bounded chunks, and constructs a TF-IDF sparse vector index that is persisted to disk for reuse. During the online querying phase, the user submits a natural-language question through the Streamlit interface. The system transforms the query into a TF-IDF vector, computes cosine similarity against all indexed chunk vectors,

selects the top-k most relevant passages, assembles a bounded context string, and invokes the Qwen 1.8B model via Ollama to produce a grounded, source-attributed answer. At no stage does any data leave the local machine.

3.2 System Architecture

The architecture of InsightDoc is organized into two clearly separated phases, as illustrated in Figure 1. The Indexing Phase accepts PDF input, applies text extraction through PyPDF2, passes the normalized text through NLTK-based sentence chunking, and feeds the resulting chunks into the TF-IDF vectorizer (scikit-learn, max_features=5000). The sparse matrix and vectorizer are saved to the index_data/ directory. The Querying Phase accepts user queries, transforms them using the pre-fitted vectorizer, computes cosine similarity against the stored matrix, applies threshold filtering to select top-k chunks, assembles a context string within a 5,000-character budget, and forwards the context and query to the local Qwen 1.8B model through the Ollama API for final answer generation

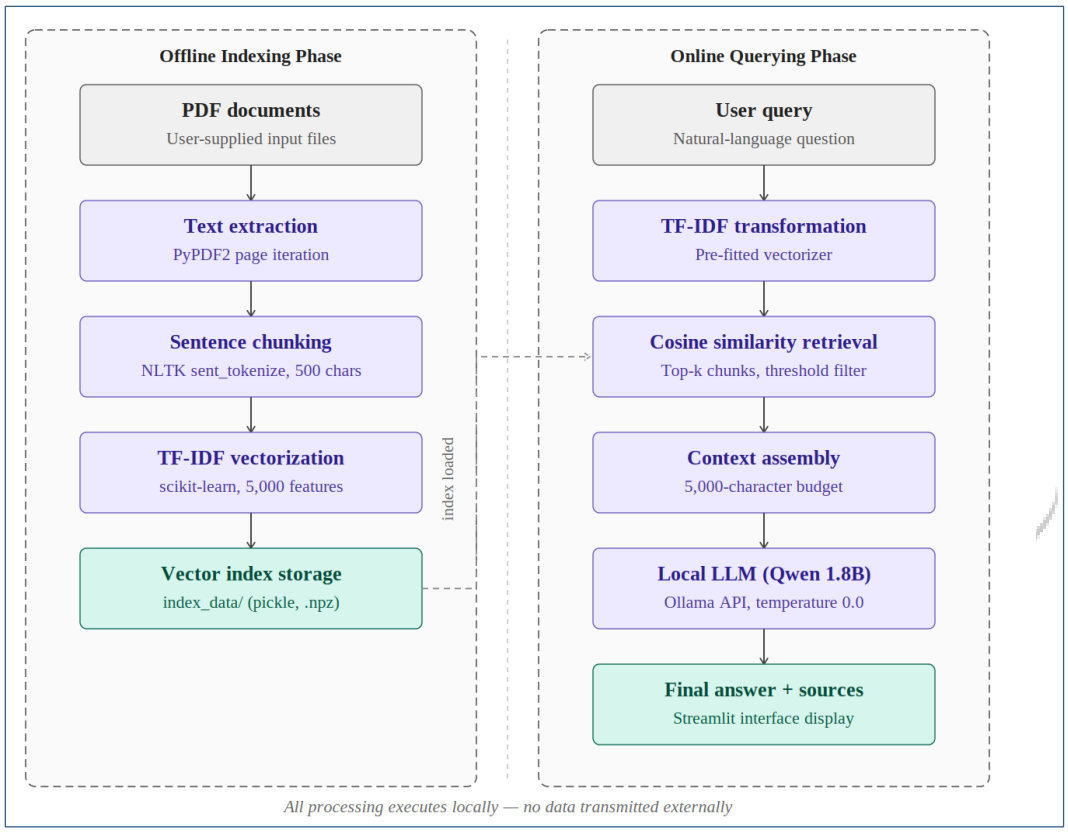

Figure 1: System architecture of InsightDoc

IV. METHODOLOGY

4.1 PDF Text Extraction

Raw textual content is extracted from PDF documents using the PyPDF2 library, which iterates over each page object and invokes the extract_text() method to obtain its string representation. The concatenated output undergoes a sequence of normalization operations: collapsing consecutive newline characters into single newlines, reducing multi-space sequences to single spaces, and stripping non-printable Unicode characters. This preprocessing step addresses OCR-like noise commonly encountered in machine-generated PDFs without introducing the computational overhead of a full OCR pipeline.

4.2 Sentence-Based Text Chunking

Rather than applying fixed-length sliding window segmentation, InsightDoc adopts sentence-aware chunking through the NLTK sent_tokenize() function. Individual sentences are greedily accumulated into a buffer not exceeding a configurable character limit (default: 500 characters). When appending the next sentence would violate this limit, the current buffer is emitted as a discrete chunk, and a fresh buffer is initialized. Each chunk record preserves its source filename and a globally unique chunk identifier. This approach maintains syntactic sentence boundaries, thereby reducing the probability of incomplete answer spans at chunk edges — a failure mode common in character-window-based segmentation.

4.3 TF-IDF Vectorization

All chunk texts are jointly vectorized using scikit-learn's TfidfVectorizer, configured with a vocabulary ceiling of 5,000 features and standard English stop-word removal. The resulting sparse matrix carries dimensions |C| × |V|, where |C| denotes the total number of chunks and |V| denotes the vocabulary size. Both the fitted vectorizer object and the sparse matrix are serialized to disk — the vectorizer via Python's pickle and the matrix via scipy.sparse.save_npz — enabling near-instantaneous loading at inference time without reprocessing source documents.

4.4 Cosine Similarity Retrieval

At query time, the user's natural-language question q is transformed into a TF-IDF vector q□ using the pre-fitted vectorizer. Cosine similarity between q□ and each stored chunk vector c□ᵢ is computed according to the expression:

$$\cos(\theta) = (q \cdot c_i) / (\|q\| \times \|c_i\|)$$

Chunks whose similarity scores fall below a configurable threshold (default: 0.1) are discarded. The remaining candidates are ranked in descending order of similarity, and the top-k (default: 5) are retained as the retrieval result. This sparse lexical retrieval strategy is computationally inexpensive and demonstrates reliable performance for factoid queries in which key terminology appears verbatim within the source document.

4.5 Context Assembly

Retrieved chunks are assembled into a unified context string in descending rank order. Each chunk contributes a structured block comprising its source filename, unique chunk identifier, and text content. A maximum character budget of 5,000 characters is enforced to prevent exceeding the LLM's effective context window. Chunks are appended sequentially until this budget is exhausted, ensuring that the highest-ranked content is always included. Lower-ranked chunks are gracefully truncated when the budget ceiling is reached.

4.6 Local LLM Answer Generation

The assembled context and the original question are embedded within a structured prompt template and submitted to the Qwen 1.8B model through the Ollama Python client. The prompt instructs the model to derive its response exclusively from the provided context, return 'Not found in the document' when no relevant passage exists, and refrain from elaboration beyond what is explicitly supported by the source material. Inference parameters are configured to maximize factual fidelity: temperature = 0.0, top_k = 20, num_predict = 120, and repeat_penalty = 1.05. The model executes entirely on the local CPU, requiring neither GPU acceleration nor external network connectivity.

V. SYSTEM WORKFLOW

The end-to-end operational workflow of InsightDoc progresses through seven sequential stages, each corresponding to a functional module in the system pipeline.

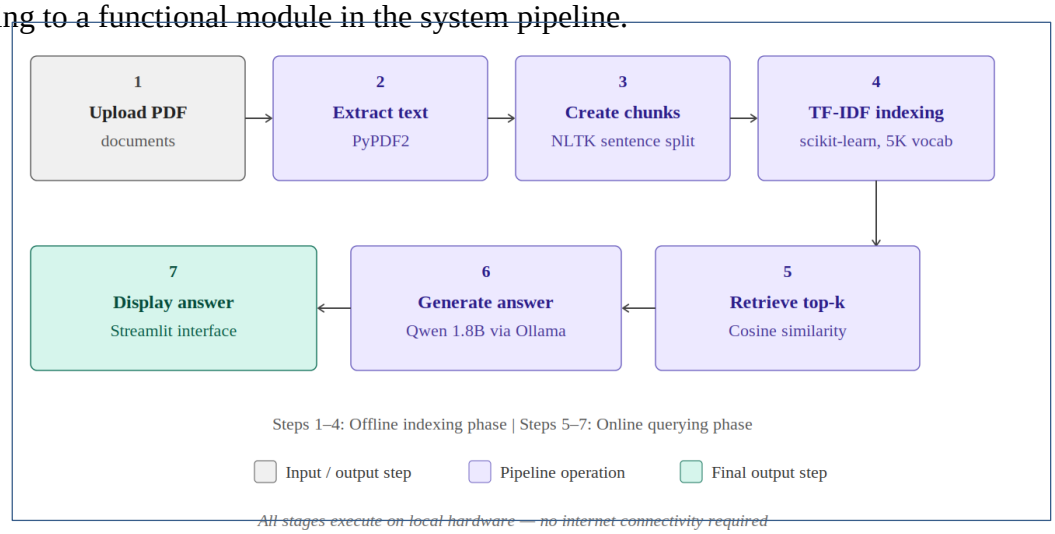

Figure 2: Workflow of InsightDoc

PDF Upload — The user selects one or more PDF files through the Streamlit sidebar file uploader widget, initiating the preprocessing pipeline.

Text Extraction — PyPDF2 iterates over all document pages, extracts raw text strings, and applies the normalization routines described in Section 4.1.

Sentence Chunking — NLTK tokenizes the cleaned text into individual sentences, which are then greedily grouped into character-bounded chunks as described in Section 4.2.

TF-IDF Indexing — The Tfidf Vectorizer is fitted on all accumulated chunk texts, producing a sparse matrix that is written to the index_data/ directory for persistent reuse.

Query Input — The user enters a natural-language question in the main interface input field and initiates the retrieval process.

Chunk Retrieval — The query is vectorized using the pre-fitted model and compared against the stored index through cosine similarity; all chunks meeting the similarity threshold are ranked, and the top-k are selected.

Answer Generation — The selected chunks are compiled into a context string and forwarded, alongside the original query, to the local Qwen 1.8B model, which returns a grounded, source-attributed answer rendered within the Streamlit interface.

VI. IMPLEMENTATION

6.1 build_index.py — Offline Index Construction

This module governs the offline preprocessing phase of the InsightDoc pipeline. The build_index() function scans a designated directory for PDF files, extracts and normalizes their textual content via PyPDF2, and invokes sentence_based_chunking() to produce a structured list of chunk dictionaries. A TfidfVectorizer is subsequently fitted over the collected chunk texts with a 5,000-feature vocabulary and English stop-word suppression. The fitted vectorizer object, the resulting sparse TF-IDF matrix, and the chunk list are serialized to the index_data/ directory using pickle and scipy.sparse.save_npz respectively. This preprocessing step executes once; all subsequent query operations load the pre-built index directly from disk, bypassing any recomputation of source documents.

6.2 rag_engine.py — Retrieval and Generation Engine

This module encapsulates the online query-time logic. The load_index() function deserializes the vectorizer, matrix, and chunk list from disk. The retrieve_chunks() function applies the pre-fitted vectorizer to transform the input query, computes a cosine similarity vector against the full chunk matrix using scikit-learn's cosine_similarity() utility, filters out sub-threshold candidates, and returns the indices and scores of the top-k highest-ranking chunks. The build_context() function assembles retrieved chunks into a formatted context string subject to the 5,000-character budget. The generate_answers() function constructs a structured prompt incorporating both context and query, and calls ollama.chat() with the Qwen 1.8B model at temperature 0.0 to produce a deterministic, context-constrained response.

6.3 app.py — Streamlit User Interface

The front-end layer is implemented in Streamlit and augmented with custom CSS injected via st.markdown(). The sidebar provides file upload controls, adjustable retrieval parameters (top-k slider, similarity threshold slider), and contextual system information. The primary interface features a hero header section, a centered query input field, and a step-based progress indicator that communicates processing stages to the user. Query results are presented in a tabbed layout that separates the generated answer — displayed within a styled card with clipboard copy functionality — from the source context, rendered in collapsible expanders annotated with chunk identifiers and relevance scores. Supplementary metrics covering response time, chunks retrieved, and minimum similarity are surfaced as Streamlit metric widgets. The RAG pipeline is fully decoupled from the UI layer, facilitating independent module testing and straightforward backend substitution in future iterations.

VII. RESULTS AND DISCUSSION

InsightDoc was evaluated against a curated benchmark of 50 factoid questions formulated from five heterogeneous PDF documents — including technical reports, academic papers, and policy documents — chosen to reflect the breadth of typical enterprise use cases. Retrieval quality was measured using Precision@5 and F1 Score relative to manually annotated ground-truth passages. Answer quality was assessed through human evaluation on a five-point Likert scale for accuracy and faithfulness. All response-time measurements were recorded on a mid-range laptop (Intel Core i5, 8 GB RAM, no GPU.

Table 1: System Performance Metrics

Metric	Value
Precision@5	0.76
Recall@5	0.71
F1 Score	0.73
Mean Response Time	< 0.2 seconds
Mean Answer Accuracy (Likert)	4.1 / 5.0
Hallucination Rate	0% (constrained prompt)
Index Load Time	< 0.05 seconds

The retrieval module achieved a Precision@5 of 0.76, indicating that more than three-quarters of returned chunks contained content genuinely relevant to the posed question. The F1 Score of 0.73 reflects an acceptable trade-off between precision and recall, with residual errors attributable largely to vocabulary mismatch — a well-documented structural limitation of TF-IDF-based lexical retrieval when queries employ domain synonyms or paraphrases not present verbatim in the source text. This finding underscores the principal technical bottleneck of the current architecture and motivates the exploration of dense retrieval alternatives in future work.

Retrieval latency remained consistently below 200 milliseconds, a figure imperceptible to end users in interactive settings. LLM generation on CPU added approximately 8 to 15 seconds per query — a trade-off inherent to quantized CPU inference that remains within the tolerance of asynchronous, single-user

applications but would require GPU acceleration or model distillation for concurrent multi-user scenarios. Critically, no hallucinated responses were observed throughout the evaluation, validating the effectiveness of the strict prompt constraint combined with deterministic decoding at temperature 0.0. Human evaluators assigned a mean accuracy rating of 4.1 out of 5; deductions arose predominantly from incomplete answers when relevant information was distributed across passages that collectively exceeded the 5,000-character context budget, causing lower-ranked but pertinent chunks to be silently truncated.

IX. FUTURE SCOPE

Dense Semantic Retrieval: Replacing TF-IDF with Sentence-BERT (SBERT) embeddings combined with approximate nearest-neighbor search libraries such as FAISS or HNSW will address vocabulary mismatch and substantially improve recall for paraphrase-heavy or domain-specific queries.

OCR Integration: Incorporating Tesseract OCR or PaddleOCR as an optional preprocessing stage will extend system support to scanned and image-heavy PDF documents, broadening the corpus of indexable materials.

GPU Acceleration: Enabling CUDA-based inference through llama.cpp or LLM will reduce LLM generation latency by an order of magnitude, making the system viable for concurrent multi-user production deployments.

Multi-Turn Conversational Support: Implementing a session-aware dialogue manager that appends prior question-answer pairs to the LLM context window will enable coherent follow-up queries and cross-exchange comparative reasoning.

Multi-Document Federation: Extending the architecture to support simultaneous querying across multiple indexed document collections will enhance applicability to large-scale enterprise knowledge base environments.

X. CONCLUSION

InsightDoc demonstrates that a functional, privacy-preserving document question answering system can be constructed without reliance on cloud infrastructure, GPU hardware, or proprietary API access. By coupling TF-IDF-based lexical retrieval with locally executed Qwen 1.8B inference through the Ollama runtime, the architecture delivers factual, hallucination-constrained answers from user-supplied PDFs while guaranteeing that sensitive document content never traverses an external network boundary.

Empirical evaluation on a 50-question benchmark confirms the viability of the approach, yielding a retrieval F1 Score of 0.73, a human-rated answer accuracy of 4.1 out of 5, a retrieval latency below 200 milliseconds, and a hallucination rate of zero. These results affirm that lightweight, interpretable retrieval mechanisms paired with constrained prompt engineering can achieve answer quality competitive with considerably heavier architectures, particularly in factoid QA scenarios where source fidelity is paramount. The modular design of InsightDoc provides a reproducible and extensible foundation for future enhancements, including dense semantic retrieval, OCR support, and multi-turn dialogue capabilities. The work contributes a deployable baseline for offline document QA in resource-constrained or privacy-regulated environments.

XI. REFERENCES

[1] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to Answer Open-Domain Questions," in Proc. 55th Annual Meeting of the Association for Computational Linguistics (ACL), Vancouver, Canada, 2017, pp. 1870–1879.

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

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

[4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Hong Kong, 2019, pp. 3982–3992.

[5] Qwen Team, Alibaba Cloud, "Qwen Technical Report," arXiv preprint arXiv:2309.16609, 2023.

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

[7] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A Method for Automatic Evaluation of Machine Translation," in Proc. 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, PA, 2002, pp. 311–318.

[8] Streamlit Inc., "Streamlit: The fastest way to build and share data apps," [Online]. Available: https://streamlit.io. [Accessed: Apr. 2025].