DocuFind.ai: A Retrieval-Augmented LLM System for Intelligent Document and Web Question Answering

Orijeet Mukherjee
Northeastern University
mukherjee.o@northeastern.edu

Abstract

Extracting relevant information from diverse formats such as PDFs, DOCX, spreadsheets, presentations, and unstructured websites is a time-consuming task, particularly for researchers, analysts, and students. Traditional search techniques fail to leverage contextual understanding, resulting in inefficiencies. To address this, I propose DocuFind.ai, a document and website question-answering system that integrates state-of-the-art language models and vector-based search to automate information retrieval. Our system combines FireCrawl for structured web scraping, MiniLM-based sentence embeddings for semantic similarity, FAISS for efficient vector search, and an instruction-tuned LLM (meta-llama/Llama-3.2-1B-Instruct) orchestrated by a smolagent CodeAgent for intelligent response generation. Built on Streamlit for usability, DocuFind.ai provides a seamless interface for querying content-rich documents and websites. Empirical results demonstrate the system's effectiveness across formats and question types, showing promise as a general-purpose, real-time QA solution for knowledge extraction.

1 Introduction

Large volumes of unstructured information—in reports, manuals, articles, or web pages—pose a challenge when users want concise answers to specific questions. Traditional keyword-based search systems fail to capture semantic relevance or produce natural language answers. Recent advances in language models (LLMs) and retrieval-augmented generation (RAG) provide a promising alternative. This project introduces DocuFind.ai, a QA system that leverages these advances to process user-provided content (documents or URLs) and generate insightful answers to arbitrary queries.

2 Background

The DocuFind.ai system builds upon several foundational components in the field of natural language processing and information retrieval.

Sentence Embeddings: Sentence transformers, such as all-MiniLM-L6-v2, play a vital role in encoding unstructured text into dense, semantically rich vector representations. These embeddings enable the comparison of textual content based on meaning rather than simple keyword overlap, improving the accuracy of downstream retrieval tasks.

Vector Similarity Search: FAISS (Facebook AI Similarity Search) is a scalable solution for indexing and retrieving high-dimensional vectors. It allows for fast and efficient semantic search across millions of embeddings. In our system, FAISS is used to retrieve the top-k most relevant document or webpage chunks based on a user query embedding.

Retrieval-Augmented Generation (RAG): RAG architectures address limitations in standalone generative models by providing them with externally retrieved knowledge. This helps mitigate hallucinations and makes responses more grounded and factual. DocuFind.ai leverages this principle by feeding relevant retrieved chunks into the LLM before response generation.

Large Language Models (LLMs): The LLM used in our system is meta-llama/Llama-3.2-1B-Instruct, an instruction-tuned model capable of generating coherent and context-sensitive answers. The model is accessed via a smolagent CodeAgent, which encapsulates prompt construction and execution logic.

Smolagents and CodeAgent Abstraction: The agent paradigm allows for modular orchestration of LLM-based tasks. CodeAgent simplifies the execution flow by integrating retrieval, prompt formulation, and inference steps, which enhances maintainability and extensibility.

Integrating large language models with vector databases for enhanced QA has been studied in recent applied research, such as the system described by Huan and Zhou [Huan and Zhou(2024)]. Their work emphasizes real-time query retrieval in web environments using FAISS and LLMs. FAISS, a popular vector similarity search library, has been widely documented in both formal [Pinecone(2024b)] and tutorial formats [Pinecone(2024a)], supporting applications ranging from QA systems to semantic search engines. Smolagents, described in recent developer-centric literature [Codemaker(2023a)], offer a lightweight and modular interface for routing and managing prompt logic, which I incorporate via CodeAgent. Hugging Face's guide on agent orchestration [Face(2024)] provides useful abstractions for modular LLM workflows.

3 Related Work

The field of document and web-based question answering has seen major strides due to advances in large language models (LLMs), vector databases, and hybrid search techniques. Several commercial products such as ChatGPT Pro (OpenAI), Claude (Anthropic), and Perplexity.ai offer high-quality question answering interfaces that support document uploads and web citations. However, these services are either proprietary, costly, or lack transparency in their internal retrieval and ranking mechanisms. DocuFind.ai aims to offer an open, modular alternative built entirely using open-source technologies.

3.1 Foundational Literature

The Retrieval-Augmented Generation (RAG) architecture introduced in [Lewis et al.(2020)] combines dense vector retrieval with generative transformers and forms the backbone of our approach. Dense Passage Retrieval (DPR), introduced in [Karpukhin et al.(2020)], applied the use of BERT for encoding both queries and passages, resulting in vector-based retrieval that is efficient and scalable. I adopt a similar approach using MiniLM instead of BERT to reduce computational overhead.

The FAISS system [Johnson et al.(2017)], developed by Facebook AI, enables scalable similarity search over high-dimensional embeddings and is central to our indexing and retrieval step. HuggingFace Transformers [Wolf et al.(2020)] further streamline the use of pretrained models in production-level NLP pipelines. FireCrawl, while not an academic contribution, facilitates real-time extraction of structured website content for integration into QA pipelines.

3.2 Smolagents and Agentic RAG

Smolagents [Codemaker(2023b)] introduce a lightweight framework to structure LLM interactions via agents. I utilize its CodeAgent abstraction to handle context injection and prompt management for meta-llama, effectively modularizing the reasoning layer of our pipeline.

3.3 Alternatives Considered

Traditional lexical retrieval methods such as BM25, TF-IDF, and ElasticSearch were considered due to their simplicity and speed. However, these approaches rely heavily on keyword overlap and perform poorly when semantic meaning needs to be captured.

End-to-end generative systems like GPT-4 with plug-and-play RAG APIs (e.g., OpenAI Retrieval Plugin) offer convenience but are closed-source and expensive at scale. Similarly, LangChain-based systems offer chaining and memory but introduce considerable overhead and rely heavily on hosted APIs.

Extractive QA approaches based on span prediction models trained on datasets like SQuAD are precise but less flexible in open-domain or multi-document settings. Generative QA with retrieved grounding, as used in DocuFind.ai, is better suited for abstract, multi-hop, or synthesis questions.

In summary, while several alternatives exist, our method offers a scalable, interpretable, and cost-effective solution suitable for education, enterprise, and open research environments.

4 Project Description

DocuFind.ai is a hybrid and agentic retrieval-augmented generation (RAG) system designed to perform intelligent question answering over both uploaded documents and websites. The system architecture is modular, scalable, and highly extensible, supporting multiple file types and online sources while maintaining accuracy and speed. Figure 1 provides an overview of the complete pipeline, illustrating the interaction between document/web ingestion, vector embedding, retrieval, and large language model-based generation.

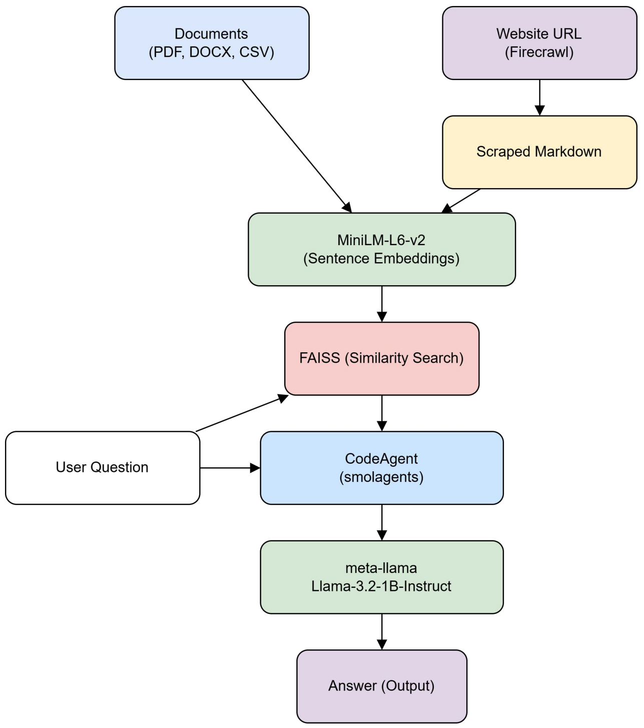


Figure 1: System Architecture of DocuFind.ai showing data flow from user input to final answer generation.

4.1 Pipeline Overview

The complete end-to-end system comprises five major stages:

1. **Input Interface:** Users can upload files (PDF, DOCX, PPTX, Excel, CSV, TXT) or enter a URL for web-based content extraction.
2. **Text Extraction:** Using file-type-specific parsers and FireCrawl for websites, unstructured text is extracted and pre-processed.
3. **Semantic Embedding:** Extracted text is segmented into chunks and converted into semantic vector representations using all-MiniLM-L6-v2[Karpukhin et al.(2020)].

4. **Similarity Search:** FAISS is used to build a vector index of the document chunks and to perform top-k nearest neighbor search to retrieve the most relevant contexts based on a user's query.
5. **Answer Generation:** The retrieved context and user query are passed to a CodeAgent, which builds the prompt and queries meta-llama/Llama-3.2-1B-Instruct for a grounded natural language answer.

4.2 Text Chunking and Embedding

Documents and website content are broken into fixed-size overlapping chunks (e.g., 150 words, 50-word stride). Let $T = \{t_1, t_2, ..., t_n\}$ denote the set of text chunks. Each chunk t_i is encoded into a dense vector $v_i \in \mathbb{R}^d$:

$$v_i = f_{\text{MiniLM}}(t_i) \tag{1}$$

where f_{MiniLM} is the embedding function provided by the all-MiniLM-L6-v2 transformer model.

4.3 Query Embedding and Retrieval

Given a query q, it is similarly embedded as $v_q \in \mathbb{R}^d$. The top-k relevant document chunks are retrieved by minimizing Euclidean distance in the FAISS index:

$$\text{TopK}(q) = \text{argmin}_{t_i \in T} \|v_q - v_i\|^2 \tag{2}$$

This retrieval step is critical in grounding the final answer in semantically related context.

4.4 Prompt Construction and Answer Generation

The CodeAgent module is responsible for taking the top-k retrieved document chunks $C = \{c_1, ..., c_k\}$ and formatting them alongside the query into a structured prompt P. This prompt P is then passed to the LLaMA model for generation:

$$\hat{a} = \text{LLM}_{\text{LLaMA}}(P) \tag{3}$$

The output $\hat{a}$ is a fluent, context-grounded answer that addresses the user's query based on the retrieved evidence.

4.5 Frontend Integration

The interface is built using Streamlit, allowing users to:

- Upload files or provide a URL.
- View previews of parsed content.
- Select example questions or type custom ones.
- Receive contextual answers in real-time.

This setup supports reproducible evaluation and real-time responsiveness suitable for research, education, and professional domains.

5 Empirical Results

5.1 Experiment Settings

I evaluated the system using a diverse set of documents and tested both the single and multi-document QA settings. The document types included:

- Academic PDFs (research papers)
- Personal documents (resumes)
- Web articles (news reports, blog posts)

For each document type, I tested DocuFind.ai by uploading one document at a time and posing related queries. I also tested scenarios where multiple documents (typically 3–5 at once) were uploaded, and the system was asked to retrieve and answer questions based on a specific one.

5.2 Multi-Document Answer Accuracy

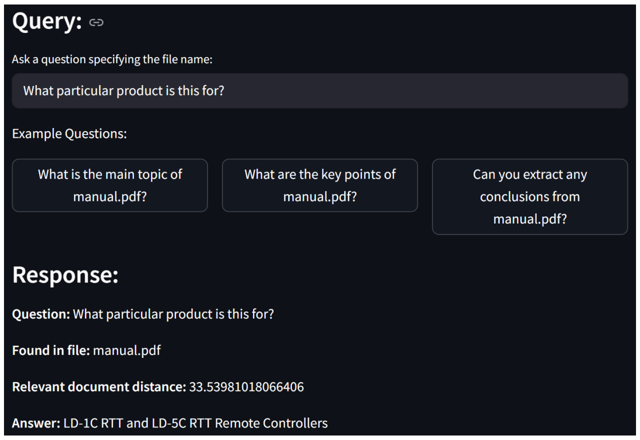
In a focused evaluation over 12 multi-document query runs, the system correctly identified and answered questions using the appropriate document in 10 out of 12 attempts. This yields an accuracy of:

$$\text{Accuracy} = \frac{10}{12} = 83.33\% \tag{4}$$

The model demonstrated strong performance in document classification and relevant context extraction even when several documents were simultaneously ingested.

5.3 Functionality Examples

In addition to simple QA, DocuFind.ai supports tasks like document summarization, named entity extraction, and thematic analysis. Examples of these capabilities are visualized in Figure 2 and Figure 3.



Query:

Ask a question specifying the file name:

What particular product is this for?

Example Questions:

What is the main topic of manual.pdf?
What are the key points of manual.pdf?
Can you extract any conclusions from manual.pdf?

Response:

Question: What particular product is this for?

Found in file: manual.pdf

Relevant document distance: 33.53981018066406

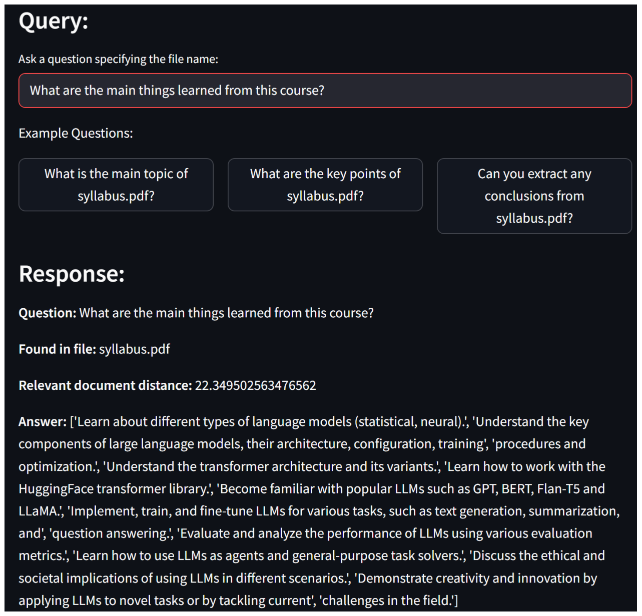
Answer: LD-1C RTT and LD-5C RTT Remote Controllers

Figure 2: Answering a query based on Remote Controllers when a user manual is uploaded

5.4 Failure Cases

- **Timeout Errors:** If model inference or file parsing exceeds allowed time, especially for large PDFs, the session state is lost and the document must be re-uploaded.
- **Short PDFs:** Some short or minimal-content PDFs lead to nonsensical or generic answers due to insufficient retrievable content.

These cases highlight the importance of robust error handling and better session memory for future iterations.


Query:
Ask a question specifying the file name:
What are the main things learned from this course?
Example Questions:
What is the main topic of syllabus.pdf?
What are the key points of syllabus.pdf?
Can you extract any conclusions from syllabus.pdf?
Response:

Question: What are the main things learned from this course?

Found in file: syllabus.pdf

Relevant document distance: 22.349502563476562

Answer: ['Learn about different types of language models (statistical, neural).', 'Understand the key components of large language models, their architecture, configuration, training', 'procedures and optimization.', 'Understand the transformer architecture and its variants.', 'Learn how to work with the HuggingFace transformer library.', 'Become familiar with popular LLMs such as GPT, BERT, Flan-T5 and LLaMA.', 'Implement, train, and fine-tune LLMs for various tasks, such as text generation, summarization, and', 'question answering.', 'Evaluate and analyze the performance of LLMs using various evaluation metrics.', 'Learn how to use LLMs as agents and general-purpose task solvers.', 'Discuss the ethical and societal implications of using LLMs in different scenarios.', 'Demonstrate creativity and innovation by applying LLMs to novel tasks or by tackling current', 'challenges in the field.']

Figure 3: Information extraction from a document describing course

5.5 Website Evaluation

The website QA performance heavily depends on the quality and richness of textual content on the site. On well-structured, text-rich websites (e.g., blogs, academic articles, product documentation), the system performs well, returning accurate and informative responses. There is an example in Figure 4.

However, websites dominated by graphical elements (e.g., infographics, image-heavy landing pages) or minimal text content lead to degraded performance. In such cases, the scraped markdown contains limited retrievable value. Hence, DocuFind.ai is best applied to websites with semantically rich, text-dense content.

5.6 Evaluation Metrics and Datasets

To assess the effectiveness of our retrieval-augmented generation (RAG) pipeline, we used standard text generation metrics such as ROUGE. ROUGE-1, ROUGE-2, and ROUGE-L measure the lexical overlap between the generated answer and the reference answer, while BLEU focuses on n-gram precision and is commonly used in machine translation and summarization tasks so we won't use BLEU.

We selected two well-established datasets for evaluation: SQuAD and MuSiQue. The SQuAD dataset provides clean, single-hop question-answer pairs with known answer spans in their corresponding context paragraphs. This makes it ideal for controlled evaluation of baseline QA capabilities. MuSiQue, on the other hand, is a multihop dataset that includes multiple paragraphs per question, where the answer must be inferred by combining information across documents. This allows us to evaluate the benefits of retrieval more meaningfully.

We report ROUGE scores for both datasets in Tables 1 and 2. As expected, the RAG approach significantly outperforms the non-RAG baseline, especially on MuSiQue, where document context plays a more crucial role in accurate answer generation.

5.7 Validation on SQuAD with ROUGE Score

To further validate our retrieval-augmented generation (RAG) approach, we used a small subset from the SQuAD dataset (Stanford Question Answering Dataset). In the RAG setting, each question was paired with its corresponding context paragraph during generation. In contrast, the non-RAG setting provided only the question to the model. We compared the generated answers against ground-truth references using ROUGE and BLEU metrics. The table 1 presents a performance comparison. Note: smol agents is being left out for this validation since the free credit limit was exceeded but rest everything was the same.

Table 1: ROUGE and BLEU Comparison on SQuAD (Subset)

Setting	ROUGE-1	ROUGE-2	ROUGE-L
RAG	0.5326	0.2698	0.5326
Non-RAG	0.2021	0.1569	0.2021

5.8 Validation on MuSiQue with ROUGE-1 Score

To evaluate our approach in a multihop question answering context, we used a subset of the MuSiQue dataset, which includes complex questions and multiple supporting paragraphs. We compared the performance of a retrieval-augmented generation (RAG) system — which retrieves top-k paragraphs using FAISS and MiniLM — against a non-RAG baseline that provides the model with the same question but no external knowledge context. Table 2 shows the ROUGE-1 scores obtained from both approaches, demonstrating the effectiveness of retrieval-augmented methods in multihop settings.

Table 2: ROUGE-1 Scores on MuSiQue (Subset)

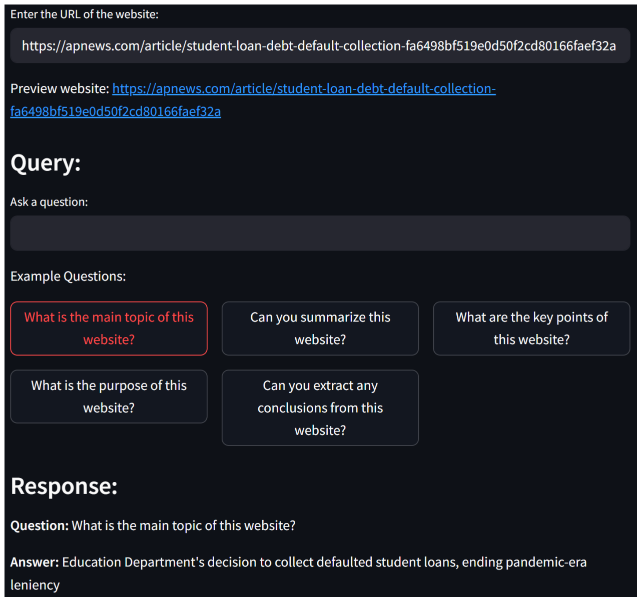
Setting	ROUGE-1 Score
RAG	0.2000
Non-RAG	0.0218

6 Broader Implications

DocuFind.ai has potential across:

- **Education:** Students can ask questions about lecture slides or academic papers.
- **Enterprise:** Employees can query manuals, policies, or reports instantly.
- **Journalism / Research:** Analysts can quickly extract insights from source documents.

Ethical considerations include ensuring LLM responses are grounded and minimizing hallucinations.


Enter the URL of the website:

https://apnews.com/article/student-loan-debt-default-collection-fa6498bf519e0d50f2cd80166faef32a

Preview website: https://apnews.com/article/student-loan-debt-default-collection-fa6498bf519e0d50f2cd80166faef32a

Query:

Ask a question:

Example Questions:

What is the main topic of this website?
Can you summarize this website?
What are the key points of this website?
What is the purpose of this website?
Can you extract any conclusions from this website?

Response:

Question: What is the main topic of this website?

Answer: Education Department's decision to collect defaulted student loans, ending pandemic-era leniency

Figure 4: Information extraction from a website on student loans

7 Conclusion and Future Work

DocuFind.ai demonstrates that intelligent document and web-based question answering systems can be built with open, modular components and still deliver practical value. By integrating semantic search (via FAISS), lightweight transformers (MiniLM), and instruction-tuned LLMs (meta-llama), the system provides meaningful responses across varied content types.

The Streamlit-based interface makes the system easily accessible for both technical and non-technical users, while smolagents' CodeAgent abstraction introduces a clean separation between retrieval and generation logic.

Looking forward, several enhancements could significantly elevate the system's impact:

- **OCR Support:** Extending functionality to image-based documents or scanned PDFs would unlock additional content domains (e.g., printed books, scanned archives).
- **Incorporating Guardrails:** Incorporating this would prevent hallucination in the LLM.
- **User Profiling:** Adaptive responses tailored to user expertise, context, or domain could improve clarity and reduce irrelevant verbosity (e.g., legal vs. general users).
- **Retrieval Optimization:** Incorporating hybrid retrieval (dense + sparse) may improve recall in certain edge cases.

Advice to future DS 5983 students:

- Start simple.
- Don't train large models or long pipelines without validating your code thoroughly—check everything at least three times.
- Keep iterating. An AI system that works most of the time is not enough; polish is what makes it truly usable.

Project Repository

GitHub: `https://github.com/orijeet100/LLM_final_project`

Hugging Face: `https://huggingface.co/spaces/orijeetmukherjee/final_project`

Please Make sure that when copying the URL there are no blank spaces or "%" characters in the link address, if there are then remove it. Latex adds a blank space at the line break in the middle, so be careful. Thanks,

References

[Codemaker(2023a)] Codemaker. 2023a. Building Smarter AI: Introduction to Smolagents and Agentic RAG.

[Codemaker(2023b)] Codemaker. 2023b. Building Smarter AI: Introduction to Smolagents and Agentic RAG. *Medium*. Accessed April 2024.

[Face(2024)] Face, H. 2024. Agents.

[Huan and Zhou(2024)] Huan, X.; and Zhou, H. 2024. Integrating Advanced Language Models and Vector Database for Enhanced AI Query Retrieval in Web Development. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 15(6).

[Johnson et al.(2017)] Johnson, J.; et al. 2017. Billion-scale similarity search with GPUs. arXiv preprint arXiv:1702.08734. FAISS whitepaper.

[Karpukhin et al.(2020)] Karpukhin, V.; et al. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6769–6781. Online: Association for Computational Linguistics.

[Lewis et al.(2020)] Lewis, P.; et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 9459–9474. Vancouver, Canada: Curran Associates, Inc.

[Pinecone(2024a)] Pinecone. 2024a. FAISS Tutorial.

[Pinecone(2024b)] Pinecone. 2024b. Vector DB Implementation Using FAISS. *SQL Server Central*.

[Wolf et al.(2020)] Wolf, T.; et al. 2020. Transformers: State-of-the-Art Natural Language Processing. arXiv preprint arXiv:1910.03771. HuggingFace Transformers Library.