Document-Based Question Answering

Riya Tendulkar
rtend@illinois.edu

June 3, 2025

1 Introduction

Recent advances in document-based question answering have focused on combining strong retrieval methods with large language models to improve answer accuracy on complex documents. Several studies show that using document structure—such as sections, tables, and figures—alongside semantic retrieval leads to better context selection and more reliable answers.

In this work, I review the main approaches from recent literature and implement a system that brings together these ideas. The methodology integrates retrieval-augmented generation (RAG), semantic chunking based on document structure, and large language models (LLMs). Documents are divided into meaningful chunks, and both chunks and user queries are embedded using transformer models. Metadata such as page numbers and section titles guide the retrieval process. Relevant chunks are selected using embedding similarity, and an LLM generates answers grounded in the retrieved context. This approach, inspired by recent research, enables more accurate and context-aware question answering across a variety of document types.

2 Literature Survey

In recent years, document-based question answering (QA) has advanced rapidly, enabling automatic extraction of information from diverse sources such as research articles, manuals, and emails Lee et al. [2023], Saad-Falcon et al. [2024], Muludi et al. [2024]. The latest systems combine advanced retrieval methods with large language models (LLMs) to locate relevant passages in complex documents and generate accurate, natural language answers Muludi et al. [2024], Rasool et al. [2023]. Moving beyond simple fixed-size chunking, modern approaches leverage document structure—such as sections, tables, and figures—and semantic cues to improve retrieval and context selection Saad-Falcon et al. [2024], Shaier et al. [2024]. Some systems now handle multimodal content and reason across multiple documents, providing more comprehensive, context-aware responses Lee et al. [2023], Li et al. [2018]. BM25, a probabilistic ranking model, remains widely used for effectively scoring and retrieving relevant documents based on term frequency, document length, and inverse document frequency Zaeem et al. [2023].These innovations have significantly improved machine reading and understanding, bringing us closer to automated knowledge extraction.

To understand the current state of the art, I reviewed several research papers in the field. Table 1 provides a summary of these works, highlighting each paper's main objective and the novel approach it introduces. This overview captures the key directions and innovations that are driving progress in automatic knowledge extraction from complex documents.

Reference	Objective	Novel Approach
Saad-Falcon et al. [2024]	QA over long, structured documents.	Structure-aware retrieval leveraging document elements for better context selection and answer extraction.
Muludi et al. [2024]	Improve document QA with LLMs.	Retrieval-Augmented Generation combining dense retrieval and LLM-based answer generation.
Rasool et al. [2023]	Evaluate LLMs on exact and numerical QA.	Introduces CogTale dataset; benchmarks LLMs on span and numerical reasoning.
Lee et al. [2023]	QA on scientific articles with complex queries.	Framework integrating evidence tracking and reasoning over scientific content.
Shaier et al. [2024]	Analyze context use in QA systems.	Proposes desiderata for improved context integration in QA.
Li et al. [2018]	Model human-like reading for QA.	Neural network inspired by human reading strategies, tested on WikiQA.
Zaeem et al. [2023]	Knowledge graph construction from articles.	Uses the BM25 probabilistic retrieval model to extract and rank relevant article content.

Table 1: Literature survey of recent document-based question answering research.

Focusing on two key approaches Retrieval-Augmented Retrieval-Augmented Generation (RAG) pipelines encode queries and document passages into vector embeddings to efficiently retrieve relevant segments, which are then combined with the user's question and passed to a large language model for context-aware answer generation Muludi et al. [2024]. PDFTriage Saad-Falcon et al. [2024] further improves performance by leveraging detailed document structure—such as sections, tables, figures, and captions—and using an LLM-based triage mechanism to identify and select the most pertinent sections or pages for each query. This targeted approach significantly boosts accuracy, especially for complex or multi-part questions involving structured and multimodal content. I chose these two papers because they best depicted the requirements of the task, showcasing state-of-the-art approaches for combining advanced retrieval techniques with large language models and for leveraging document structure to improve question answering performance.

3 Methodology

To implement the system, I have combined the methodologies from Saad-Falcon et al. [2024] and Muludi et al. [2024]. The methodology is demonstrated below -

- **Text Extraction:** The system uses the PyMuPDF (fitz) library to read the PDF document. For each page, it performs chunking—where each chunk typically corresponds to a paragraph—to maintain the semantic meaning of the content. It also identifies section titles using regular expressions. Each chunk is then associated with its respective page number and section title.
- **Metadata Construction:** The metadata is created by collecting all unique page numbers and

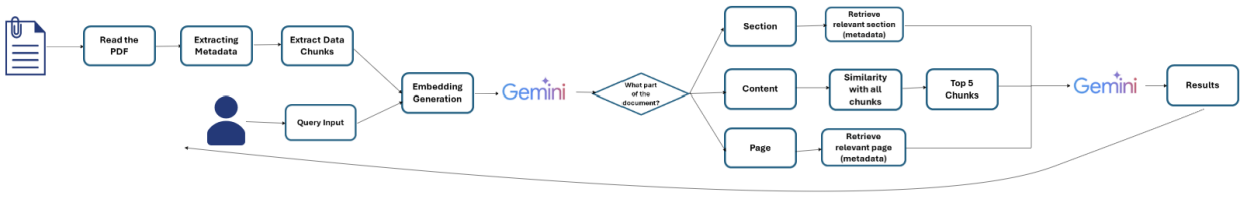

Figure 1: Methodology Implemented

section titles from the extracted chunks. This metadata helps in categorizing and retrieving relevant information based on user queries.

- **Semantic Embedding Generation:** The sentence-transformers library (specifically, the 'all-MiniLM-L6-v2' model) is used to convert each text chunk into a vector embedding.
- **User Query Processing:** When a user submits a query, it is also converted into an embedding using the same model.
- **Query Classification:** A large language model (LLM) classifies the user's query as referring to a page number, section, or a general concept, using document metadata and a prompt template. The LLM analyzes both the content and structure of the query in the context of the document's metadata to determine the most appropriate retrieval strategy.
- **Context Retrieval via 'get' Functions:** Based on the function call initiated by the LLM, the following functions will be called:
  - *get top chunk*: The cosine similarity between the query embedding and each chunk embedding is calculated to find the most relevant content.The top-N most chunks are returned.
  - *get page chunks*: Retrieves and concatenates all chunks from the specified page number.
  - *get section chunks*: Retrieves and concatenates all chunks from the specified section title.
- **Answer Generation:** The large language model (LLM) receives the retrieved context and the user's question, and generates an answer strictly based on the provided context.

Technology Stack The code uses the *all-MiniLM-L6-v2* transformer model from Sentence Transformers to generate semantic embeddings for text chunks and user queries. For large language model (LLM) tasks, the *Google Generative AI (Gemini) API via LangChain* is utilized. This LLM API was chosen because it was free, however it does not support direct function calling or tool use from within the model and hence the complete methdology of PDFTriage was not used. The frontend is developed using Flask.

4 System Evaluation

Accuracy To evaluate the performance of the system, I have used the dataset created by Muludi et al. [2024]. The dataset has 20 documents, five questions for each document and the ground truth.

The document types are - research papers, news articles, technical articles and recipes (5 documents for each type) with a total of 100 questions. To evaluate the performance of my system, I generated outputs for each document and question using the system and then compared them to the ground truth. Out of the 100 questions, the system was able to accurately generate answers for 92 questions, making the **accuracy as 92%**.

BERT Score The BERT scores were calculated using the `bert-score` Python package, which compares the embeddings of generated answers to those of the ground truth references. The BERT score results demonstrate that the system's generated answers are highly semantically similar to the reference answers. Most scores fall between 0.8 and 1.0, with a large proportion above 0.9, indicating strong alignment in meaning even when the wording is not identical. Only a few low or zero scores showing rare cases of mismatch.

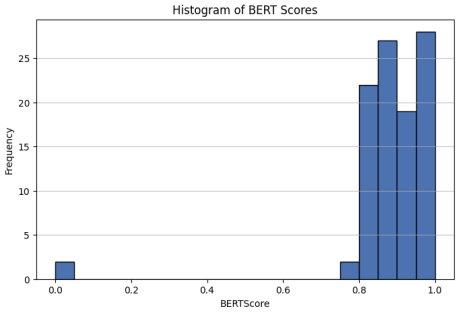

Figure 2: Results: BERT Score

5 Limitations and Ideas

To better understand the source of the 8% inaccuracy in my system's answers, I conducted a deep dive into the retrieval and generation pipeline.

Chunking Issues One limitation I observed is that inconsistent chunk sizes—caused by splitting text without regard to semantic boundaries—lead to uneven information distribution across chunks. This inconsistency can cause the similarity calculation between the user query and document chunks to be less effective, as relevant information may be split or diluted. To address this, I suggest implementing a dynamic, semantic-aware chunking strategy (Pinecone [2023]), which uses embeddings to identify topic shifts and creates more coherent, contextually meaningful chunks. This should improve both retrieval accuracy and answer quality.

PDF Reading and Title Detection Another limitation is the system's difficulty in accurately reading PDFs and detecting section titles, especially when documents have complex formatting or inconsistent layouts. This can result in missed or misidentified sections, negatively impacting context selection for question answering. To overcome this, integrating PDF parsing tools—such as the Adobe PDF Extract API can help reliably extract text and metadata, accurately identify section headings.

References

Yoonjoo Lee, Kyungjae Lee, Sunghyun Park, Dasol Hwang, Jaehyeon Kim, Hong in Lee, and Moontae Lee. Qasa: Advanced question answering on scientific articles. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19036–19052, 2023. URL https://proceedings.mlr.press/v202/lee23n/lee23n.pdf.

Weikang Li, Wei Li, and Yunfang Wu. Based on human-like reading strategy. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 604–611, 2018. URL https://cdn.aaai.org/ojs/11316/11316-13-14844-1-2-20201228.pdf.

Kurnia Muludi, Kaira Milani Fitria, Joko Triloka, and Sutedi. Retrieval-augmented generation approach: Document question answering using large language model. *International Journal of Advanced Computer Science and Applications*, 15(3), 2024. ISSN 2156-5570. Informatics Engineering Graduate Program, Darmajaya Informatics and Business Institute, Bandar Lampung, Indonesia.

Pinecone. Chunking strategies for llm applications, 2023. URL https://www.pinecone.io/learn/chunking-strategies/. Accessed: 2025-06-02.

Zafaryab Rasool, Stefanus Kurniawan, Sherwin Balugo, Scott Barnett, Rajesh Vasa, Courtney Chesser, Benjamin M. Hampstead, Sylvie Belleville, Kon Mouzakis, and Alex Bahar-Fuchs. Evaluating llms on document-based qa: Exact answer selection and numerical extraction using cogtale dataset. *arXiv preprint arXiv:2311.07878*, 2023.

Jon Saad-Falcon, Joe Barrow, Alexa Siu, Ani Nenkova, David Seunghyun Yoon, Ryan A. Rossi, and Franck Dernoncourt. Pdftriage: Question answering over long, structured documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2024. URL https://aclanthology.org/2024.emnlp-industry.13/.

Sagi Shaier, Lawrence Hunter, and Katharina von der Wense. Desiderata for the context use of question answering systems. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 777–792, St. Julian's, Malta, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.47.pdf.

Jasir Mohammad Zaeem, Vibhor Garg, Kirti Aggarwal, and Anuja Arora. An intelligent article knowledge graph formation framework using bm25 probabilistic retrieval model. In *Knowledge Graphs and Semantic Web. KGSWC 2023. Lecture Notes in Computer Science*, volume 14344, pages 32–44. Springer, 2023. doi: 10.1007/978-3-031-47745-4_3. URL https://doi.org/10.1007/978-3-031-47745-4_3.

A System Outputs

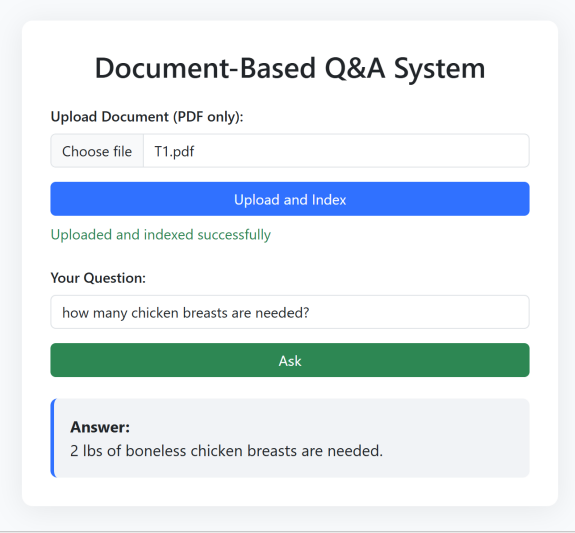

Figure 3: Output: Q&A for a Recipe

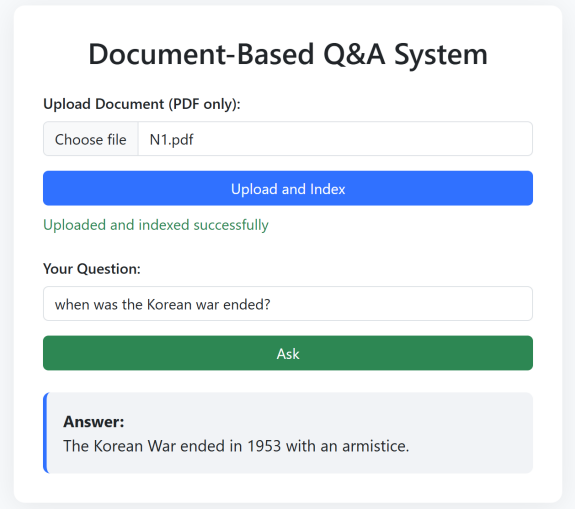

Figure 4: Output: Q&A for a News Article

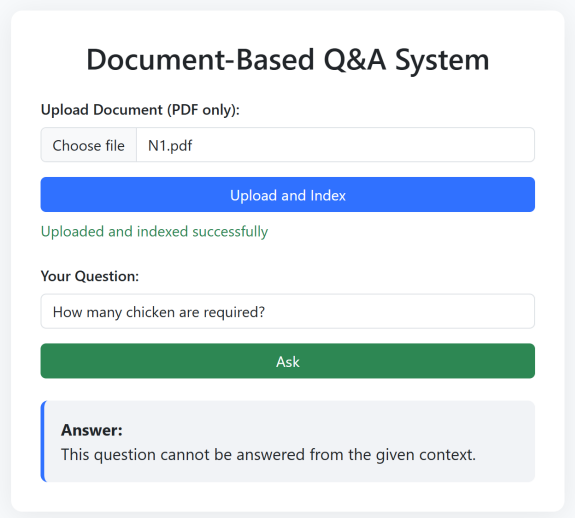

Figure 5: Output: Incorrect question output for a News Article

	A	B	C
1	Files	Question	Ground Truth
2	J1	what is the command for install the FLAIR library?	pip install flair
3	J1	what is the proposed solution from this paper?	FLAIR framework
4	J1	what github links of this framework?	https://github.com/zalandoresearch/flair
5	J1	What are the presented framework used for?	ramework designed to facilitate experimentation with different embedding types as well as training and distributing sequence labeling and text
6	J1	what is the minimum python version to setup FLAIR in the environment?	at least version 3.6 of Python
7	J2	The experiment are based on what method?	Edge Probing
8	J2	Mention two metrics that used in this experiment?	Scalar Mixing Weights and Cumulative Scoring
9	J2	what kind of model that introduced scalar mixing technique?	ELMo model
10	J2	what language model used in this paper?	BERT model
11	J2	What is the main purpose of edge probing?	to measure how well information about linguistic structure can be extracted from a pre-trained encoder
12	J3	nducted in this research to quantify the computational and environmental costs of training deep neur	popular off the-shelf NLP models as well as a case study of the complete sum of resources required to develop LISA a state-of-the-art NLP mod
13	J3	w long does it take to train Transformer base model and Transformer big model with NVIDIA P100 GPU	Transformer base model was trained for 12 hours and Transformer big model was trained for 3.5 days
14	J3	is TPUs more cost-efficient than GPUs based on Table 3 analysis?	Yes Of note is that TPUs are more cost-efficient than GPUs on workloads that make sense for that hardware
15	J3	what kind of GPU and CPU used for train all models in this experiment?	NVIDIA Titan X GPU and Intelâ€™s Running Average Power Limit interface CPU
16	J3	List the four models used in this experiment?	Transformers ELMo BERT GPT-2
17	J4	what is BRAT?	brat rapid annotation tool is an intuitive web-based tool for text annotation supported by Natural Language Processing (NLP) technology
18	J4	How many annotations has BRAT created in its used at academic institutions?	well-over 50.000 annotations
19	J4	what is the homepage links for the free open-source BRAT?	http://brat.nlplab.org
20	J4	what kind of architecture used for implementing BRAT?	client-server architecture with communication over HTTP using JavaScript Object Notation (JSON)
21	J4	How much is the reduction percentage in annotation time as a result of this experiment?	15.40%
22	J5	What kind of framework proposed in this research paper?	TextAttack

Figure 6: Sample of the Testing Document