Revolutionary Advances in Computing and Electronics (RACE): An International Journal
Volume 02, Issue 02, pp.17-32, 2026
ISSN (Online): 3107-8540

Received: 07/05/2026, Accepted: 14/07/2026

AI-Based Document Analysis and Question Answering System

Radhika Sharma, Devraj Gautam

Department of Electronics & Communication Engineering, Dr Akhilesh Das Gupta Institute of Professional Studies, GGSIPU, India

radhika.sharma200320@gmail.com, devrajgautam10@gmail.com

Abstract

The exponential growth of textual data, in the form of corporate documents, reports, and research papers, in the current digital era has increased the need for intelligent systems that can automatically comprehend documents. Analysing these documents by hand is ineffective and time-consuming. In order to extract valuable insights from documents, this study provides an AI- Based document analyzer with a question-answer system that makes use of Natural Language Processing approaches. Users can submit text or PDF files to the system, which then extracts content, generates succinct summaries, identifies keywords, and enables interactive question-answering. Python is used to build the architecture, which is then serverless deployed on Amazon Web Services (AWS) utilising Amazon S3 and Amazon EC2, and for the question-answering system, GEMINI is used. By reducing reading time and providing immediate access to pertinent information, the suggested solution increases productivity. It is affordable, scalable, and suitable for business, education, and research.

Keywords: *Artificial Intelligence, Natural Language Processing, Amazon Web Services, Simple Storage Service, Application Programming Interface, Question Answering, Portable Document Format.*

1. Introduction

Manual document analysis has become ineffective and time-consuming due to the sharp rise in digital documents across a variety of fields, including business, education, and research. Intelligent automation is necessary to extract valuable insights from massive amounts of text. An AI- Based document analyzer that uses NLP approaches to process documents is introduced in the proposed system. In addition to allowing users to upload files, it automatically extracts keywords, generates summaries, and responds to user inquiries based on document context. In contrast to conventional systems, our solution combines NLP and cloud computing to deliver a scalable, effective platform. AWS deployment guarantees affordability, dependability and accessibility.

1.1 Need for the System

There is a growing need for intelligent systems that can effectively analyze and comprehend massive amounts of textual data due to the exponential growth of digital information in the

*Corresponding author: Devraj Gautam, Department of Electronics & Communication Engineering, Dr Akhilesh Das Gupta Institute of Professional Studies, GGSIPU, India (devrajgautam10@gmail.com)

form of research papers, reports, resumes and corporate documents. In addition to taking a lot of time, manually reading and evaluating such texts is prone to human mistakes, particularly when prompt decision-making is needed. Systems that can automatically extract important information and display it succinctly are therefore desperately needed in order to minimize the effort required for manual reading. Furthermore, customers need immediate insights without having to read entire documents, underscoring the importance of automated summary methods. Enabling intelligent search within documents so users can quickly access specific information via queries is another necessity. In both academic and professional settings, where time and accuracy are critical, such capabilities greatly increase productivity. These requirements are met by the suggested system, which uses NLP to automate document comprehension and provides an interactive question-and-answer system that delivers pertinent information effectively and efficiently.

1.2 Applications

Through summaries and question-answering, academic research assistance helps researchers and students swiftly comprehend research papers, publications, and study materials. In recruitment, resume screening helps HR professionals extract important information from resumes, including experience, abilities and qualifications. Business document analysis allows firms to extract valuable insights for decision-making by analysing reports, documents, and internal papers. Legal document processing identifies key information and provisions in lengthy contracts, agreements and legal documents. Medical reports interpretation helps patients and healthcare providers comprehend and summarize complicated medical records. E-learning platforms assist students by streamlining study materials and offering prompt responses to their questions. Customer support systems enhance response times by evaluating manuals, support articles and FAQs to deliver precise responses to customer inquiries.

1.3 *Challenges*

There are several important obstacles in the way of creating an AI- Based document analyzer with a Q&A system. Managing massive amounts of unstructured textual data is one of the main challenges because documents can differ in terms of format, language, style, and complexity. Robust NLP approaches that comprehend context, semantics and relationships within the text are necessary to extract accurate and pertinent information from such data. Creating insightful summaries that preserve the text's main ideas without omitting crucial information or context is another significant challenge that requires a careful balancing act between completeness and conciseness. Furthermore, it is difficult to provide accurate, context-aware responses to user questions, as the system must determine the most pertinent information in the document and interpret the question’s intent. Maintaining performance in real-time settings and ensuring scalability are equally crucial, particularly when the system is expected to manage multiple users and large datasets simultaneously. Additionally, differences in document quality, linguistic ambiguity and the limitations of current NLP models might affect the system’s overall accuracy and dependability; it's critical to develop an effective and flexible solution.

2. Literature Review

Natural Language Processing forms the foundation of modern document analysis systems. The fundamental concepts of NLP, including text preprocessing, tokenisation, and syntactic analysis, are well explained in Natural Language Processing with Python by Bird et al., which provides essential techniques for processing and analysing textual data [1]. To support practical implementation of NLP tasks, tools such as the NLTK (Natural Language Toolkit) offer various functionalities, including tokenisation, stop-word removal, and text normalization, enabling efficient handling of textual data [2]. These tools are widely used for building NLP-based applications. In addition, information retrieval techniques play a crucial role in document analysis. Manning et al explain how text can be represented using vector space models, allowing efficient searching, indexing and ranking of documents based on relevance [3]. These methods form the basis for keyword extraction and document similarity analysis.

Further advancements in NLP include the development of word embedding techniques, such as Word2Vec, which capture semantic relationships between words by representing them as continuous vectors. This approach significantly improves the understanding of context and meaning in text processing systems [4]. More recently, deep learning-based models such as BERT have revolutionized NLP by enabling bidirectional context understanding. These models are capable of performing complex tasks such as text summarization and question answering with high accuracy by leveraging large-scale pertaining [5]. For handling document inputs, tools such as PyPDF2 are commonly used to extract textual data from PDFs, enabling efficient processing of unstructured documents [6]. This step is essential for converting raw documents into analysis-ready text.

Cloud computing platforms, particularly AWS, provide scalable infrastructure for deploying such systems. AWS documentation outlines how cloud services can be used to build efficient and reliable applications [7]. Among these services, Amazon S3 is used for secure storage of documents and processed outputs, ensuring high availability and scalability [8]. AWS Lambda enables serverless execution of code, allowing automatic processing of documents without managing infrastructure [9]. Additionally, API Gateway facilitates communication between users and the system, enabling users to send queries and receive responses seamlessly [10]. These services collectively support real- time document analysis systems. Furthermore, modern AI platforms such as Napkin AI demonstrate the growing trend of intelligent tools that assist in visualising and organising information, highlighting the increasing demand for automated document understanding solutions [11].

Despite these advancements, challenges such as language ambiguity, maintaining contextual accuracy, and ensuring real-time performance remain. The proposed system builds upon these existing technologies to develop an efficient and scalable AI-based document analyser with a question-answer system.

3. Research Methodology

3.1 Data Collection

The calibre and diversity of input documents significantly impact the document analyser system's efficiency. Research papers, reports, resumes, legal documents and company records are just a few of the many document kinds that the system can manage, mostly PDF and text files. A straightforward upload interface is used to gather these papers from users, and Amazon S3 is used to store them safely in the cloud.

Figure 3.1 shows the data processing funnel used in the system. It begins with document upload, where users upload PDF or text files. Then the documents are stored in cloud storage securely. The system then processes the content with NLP techniques to extract meaningful information.

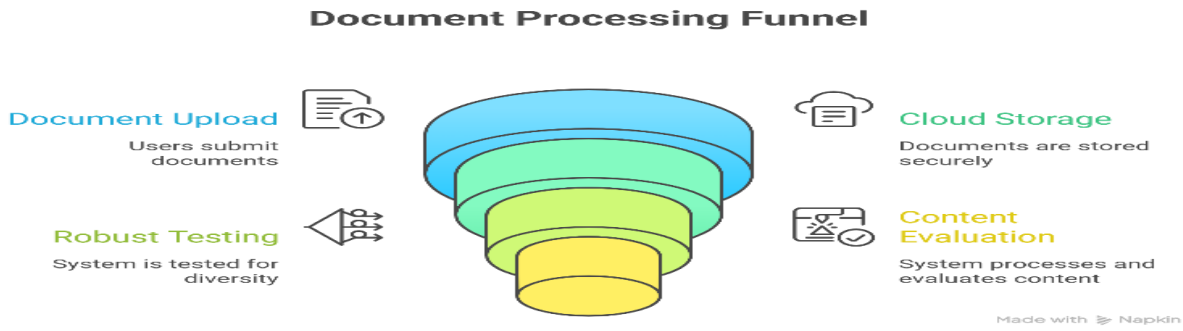

Figure 3.1: Data Processing Funnel [11].

The system is tested on documents with different durations, formats and levels of complexity to guarantee robustness. This covers both unstructured text with irregular layout and structured papers with clear formatting. This diversity enhances the system's capacity to generalize across various document kinds and aids in its adaptation to real-world situations. Documents may also contain casual writing, domain-specific terminology, or technical language, which makes it more difficult for the system to correctly process and evaluate content. A figure of data collection is shown below.

3.2 Data Preprocessing

An essential step in getting raw text ready for analysis is data preparation. Noise like special characters, punctuation, unnecessary symbols, and inconsistent formatting are frequently seen in the retrieved text from documents. The system uses a number of natural language processing methods to clean and standardise the data in order to address this. Tokenisation, which divides the text into smaller pieces like words or sentences, is the first step in the preprocessing pipeline. The next step is stop-word removal, which gets rid of frequently used words like "the," "is," and "and" that don't really add much to the text's meaning. To preserve consistency, normalization strategies like changing text to lowercase and eliminating superfluous spaces are also used. Additionally, words can be reduced to their base or root form using stemming or lemmatization, which increases the effectiveness of subsequent analysis. These pretreatment procedures guarantee that the text is clean, organized, and appropriate for activities like question answering, keyword extraction, and summarization.

Below is a visual representation of data preprocessing. It starts with tokenization, which is the process of breaking texts into words or sentences. Then stop-word removal removes common words that do not add any meaningful information. Normalization puts the text into a standard format. Finally, stemming or lemmatization reduces words down to their root form. The process cleanses and readies the text for efficient analysis and accurate results.

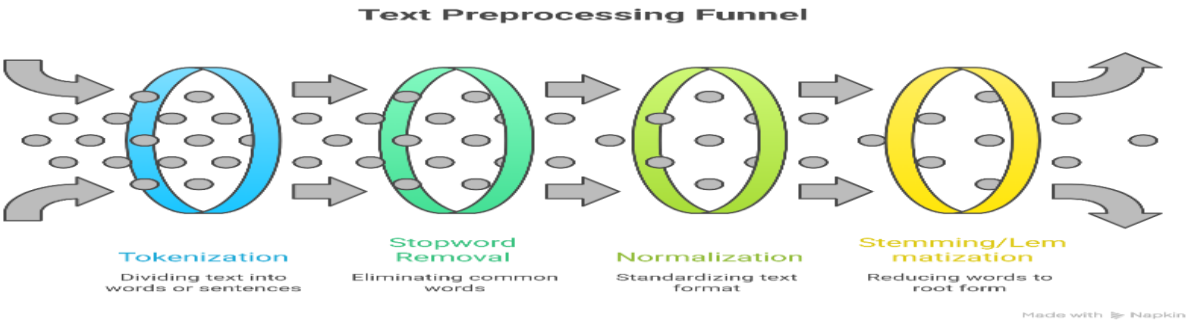

Figure 3.2: Text Preprocessing Funnel [11].

Figure 3.2 illustrates the text preprocessing pipeline, where raw text is cleaned through tokenization, stop-word removal**,** normalization, and stemming/lemmatization. These steps convert unstructured text into a clean and standardized format, making it suitable for accurate analysis and machine learning tasks.

3.3 Summary Generation

To produce succinct document summaries, the system uses an extractive summarization technique. Extractive summarization chooses the most significant sentences directly from the source text, in contrast to abstract techniques that create new phrases. To ascertain the significance of every word in the document, the first step in the procedure is to compute word frequencies. The frequency of these crucial terms is then used to assess sentences. Sentences with higher scores are chosen to make up the summary since they are thought to be more pertinent.

The summary generation diagram is given below. It starts by counting word frequencies to identify important terms in the document. Then these frequencies are used to score and rank sentences. Then the system selects the key sentences with higher scores to construct the summary. The process generates short, meaningful summaries that are extracted from the original.

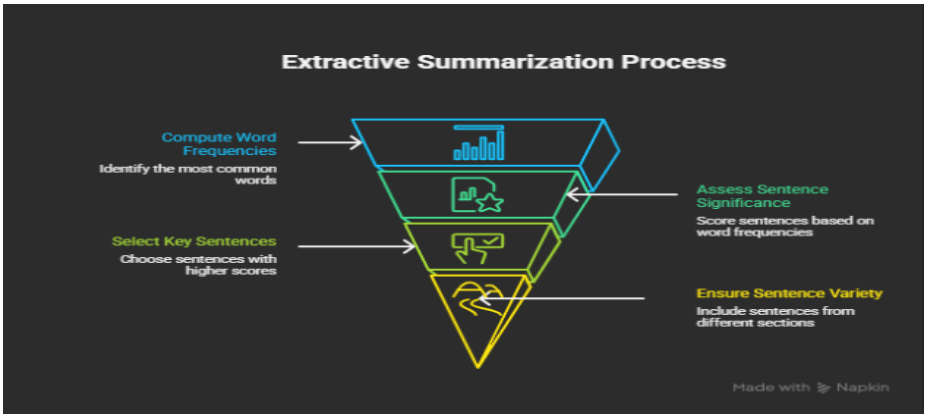

Figure 3.3: Extractive Summarization Process [11].

Figure 3.3 illustrates the extractive summarization process, where word frequencies are computed to identify important terms and sentences are scored based on their significance. The highest-ranked and most diverse sentences are then selected to generate a concise and informative summary while preserving the original content.

The system selects a variety of sentences that cover different sections of the document in order to prevent redundancy and guarantee quality. Users may easily understand the key points without reading the entire text thanks to the final summary, which offers a clear and succinct depiction of the original content. It greatly increases productivity, particularly when working with long documents. A diagram is given below of summary generation.

3.4 Keyword Extraction

The most significant terms that best capture a document's main ideas are found through keyword extraction. Words that appear frequently and have significant context are given more weight by the system, which uses frequency-based algorithms to estimate word significance. Following preprocessing, unnecessary words are eliminated, and the remaining words are examined to find important phrases. In addition to improving search and retrieval within the system, these keywords aid consumers in rapidly comprehending the topic of the text. By taking into account both the local and global significance of terms, methods like (Term Frequency–Inverse Document Frequency) can sometimes be used to improve keyword relevance. The retrieved keywords serve as a fast reference and provide light on the document's primary subjects.

The keyword extraction process diagram is given below for better understanding. From this keyword extraction process important words are selected. Keywords are the main ideas, they help to better understand and retrieve the content of document.

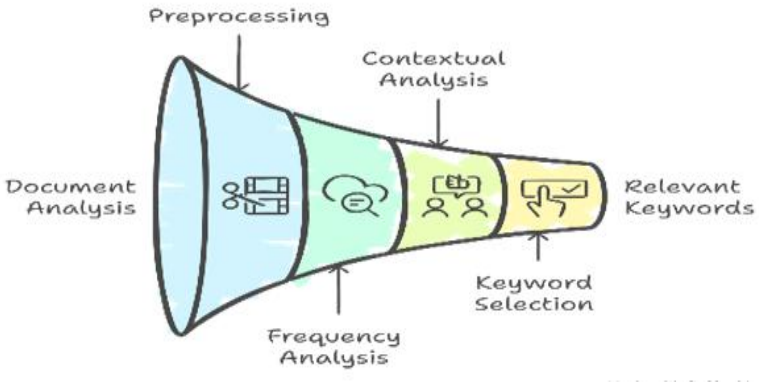

Figure 3.4: Keyword Extraction Process [11].

Figure 3.4 illustrates the keyword extraction process, where a document undergoes preprocessing**,** frequency analysis**,** contextual analysis, and keyword selection. These steps help identify the most relevant keywords, improving information retrieval and text analysis accuracy.

3.5 Question–Answer System

Interactive communication between the user and the system is made possible by the question-answer module. Users can ask questions about the uploaded document, and the system will respond with the most pertinent details. The procedure entails evaluating the user's query, identifying key terms, and comparing them to the content of the processed document. The most pertinent sentence is then returned as the response once the system determines which sentences best match the inquiry. With this method, users can swiftly access particular information without having to go through the full document by hand. The system can be improved using sophisticated NLP models to increase contextual comprehension and accuracy, even though keyword matching is its primary method.

The question-answering system is shown by a diagram below. It starts with query assessment, where the system looks at what question the user is asking. Then, key term identification is employed to extract important keywords from the query. Then the system compares the content by matching these keywords with the content of the document. This match is used to retrieve relevant information. Finally, the system gives accurate and context-aware answers for the user.

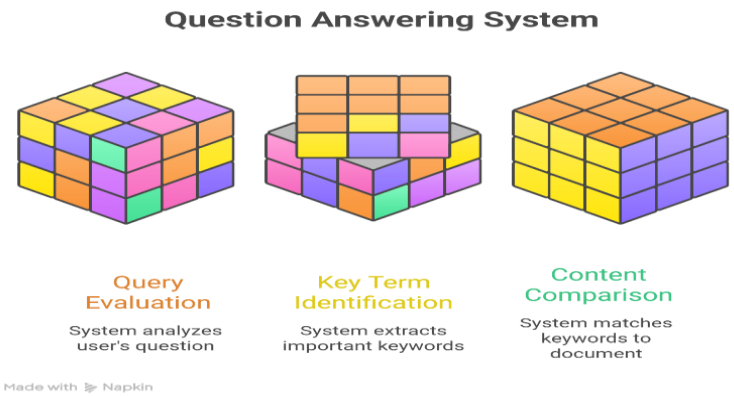

Figure 3.5: Question Answering System [11].

Figure 3.5 illustrates the workflow of a Question Answering System, where the user's query is first evaluated, important keywords are identified, and the query is then compared with available content. This process enables the system to retrieve and provide the most relevant and accurate answer to the user's question.

3.6 System Deployment

Frontend layer:
Streamlit was used to develop the user interface, which enables users to interact with the system via a web-based interface and upload documents. For ease of access, it can be set up locally or on a cloud server.

Backend layer:
FastAPI, which manages document processing, query management, and communication with AI models, is used to implement the backend. It effectively handles user inquiries and provides answers.

AI processing layer:
Vector database operations, embedding creation and NLP models all come under this layer. A large model is used to generate replies, FAISS is used for storage, and Sentence Transformers are used for embeddings.

Cloud deployment:
Amazon EC2 is used to host the backend and frontend. Amazon S3 is used for document storage. Gemini is used for API Services.

4. Results and Discussion

4.1 Experimental Configuration

A varied dataset of .pdf and .txt documents, comprising scholarly articles, reports and descriptive text files, was used to assess the suggested GenAI-based document comprehension system. Ten documents in all, each with fifteen to thirty pages, were used. 300 factual, inferential, and logic-based queries were created to evaluate the system’s performance. The system pipeline used Sentence Transformers to create embeddings, FAISS for semantic retrieval and PyMuPDF for text extraction, all of which were connected by LangChain.

4.2 System Output Demonstration

The document analyser system allows users to upload documents; then the system processes it using libraries like PyMuPDF and LangChain and offers a question-answer system also. It generates context-aware answers along with supporting document references. The figure below shows the different steps of the system.

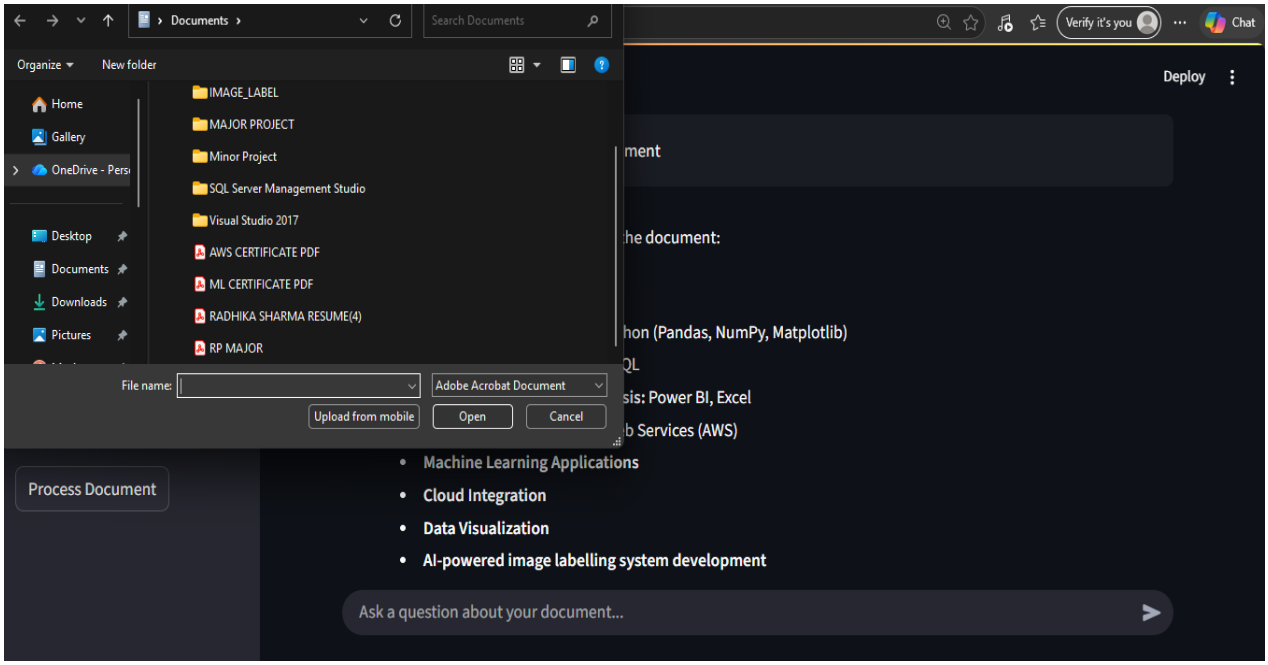

Figure 4.2 (a): User Interface showing document uploading

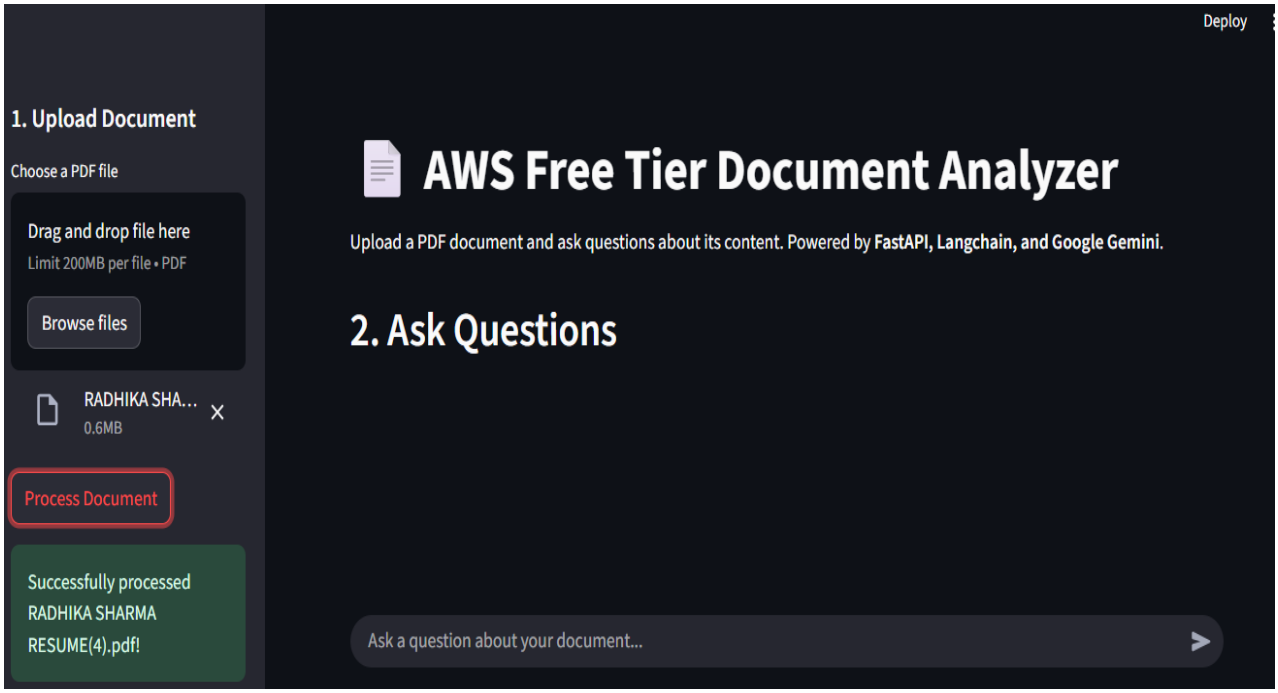

Figure 4.2 (b): User interface showing document is being processed

Figure 4.2 (a) shows the document uploading interface, and Figure 4.2 (b) shows that after uploading the document, it is being processed by the system. After processing, users can ask questions related to the uploaded document. The system retrieves relevant chunks (LangChain is used for text extraction and chunking) and generates answers grounded in document context. The images below show the question-answer system that is answering the query related to the uploaded document. The documents that are uploaded in the given images are a 50-page report and a 50-page history book.

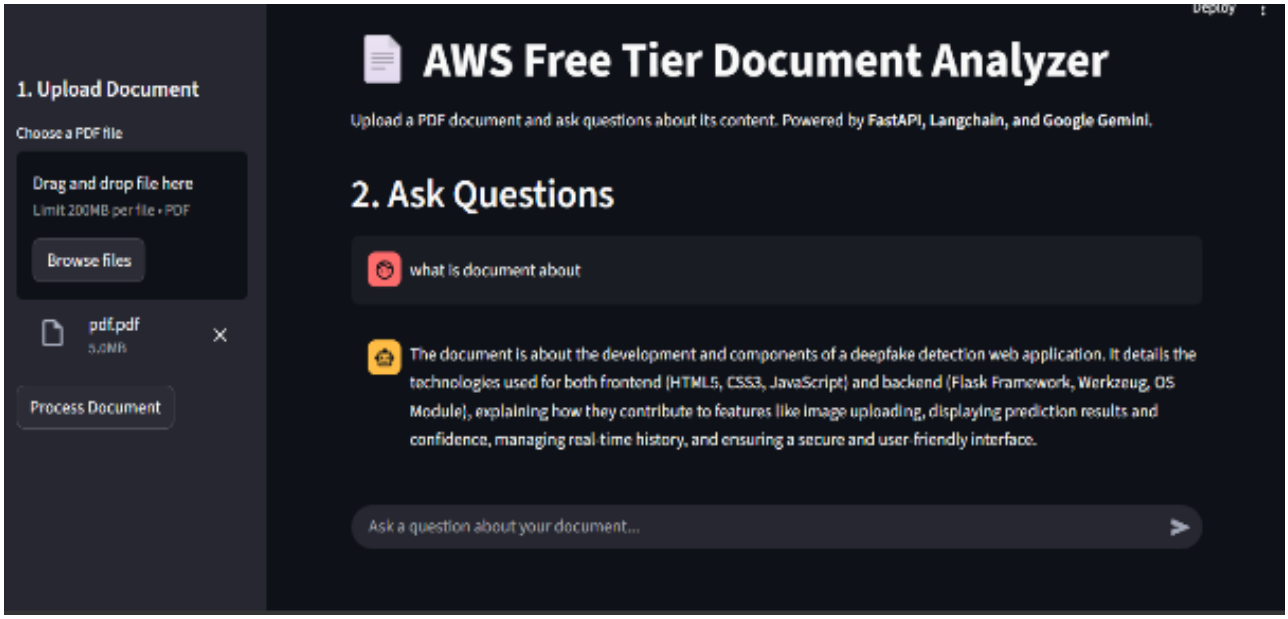

Figure 4.2 (c): System answers a query

Figure 4.2.(d) shows the system's question answering capabilities. The user uploads a document and asks a question on the content of the document. The system processes the document using NLP techniques and generates a relevant context aware response.

::

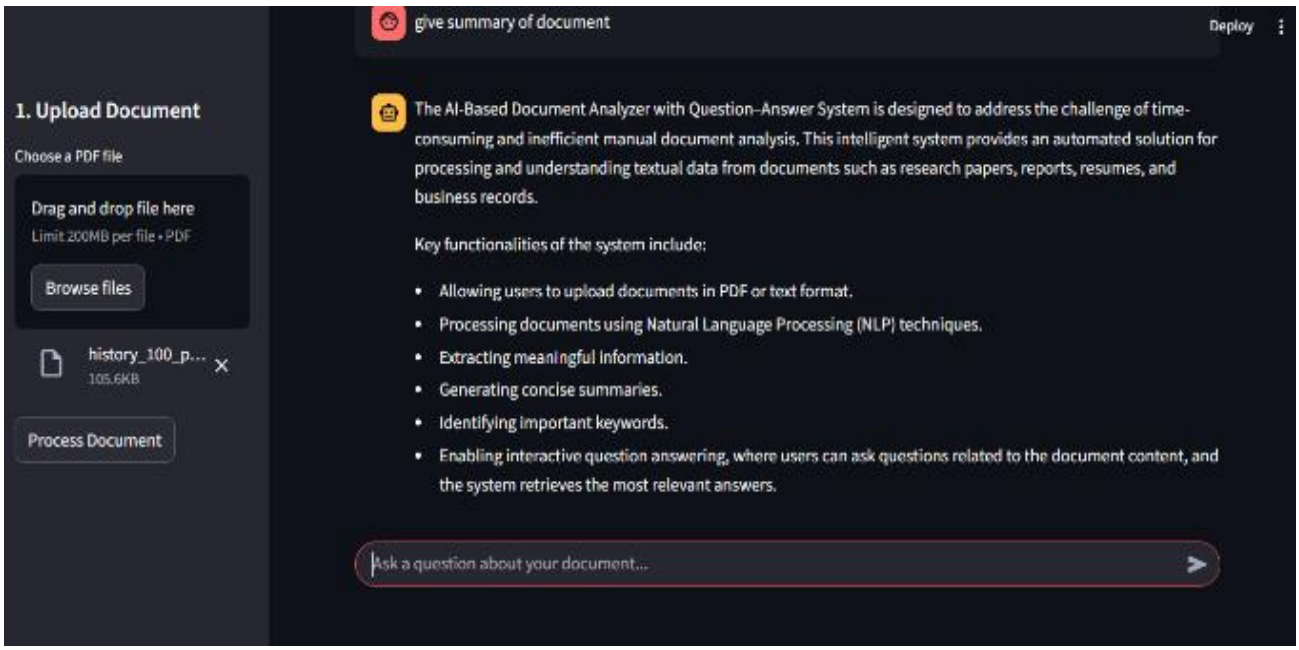

Figure 4.2 (d): Summary given by system

Figure 4.2 (e) shows question answer ability of the system with contextual understanding. After a document has been uploaded, the user asks a question, and the system finds relevant information and provides a detailed answer. This shows the system's capability to understand context, analyze content and provide informative, structured answers.

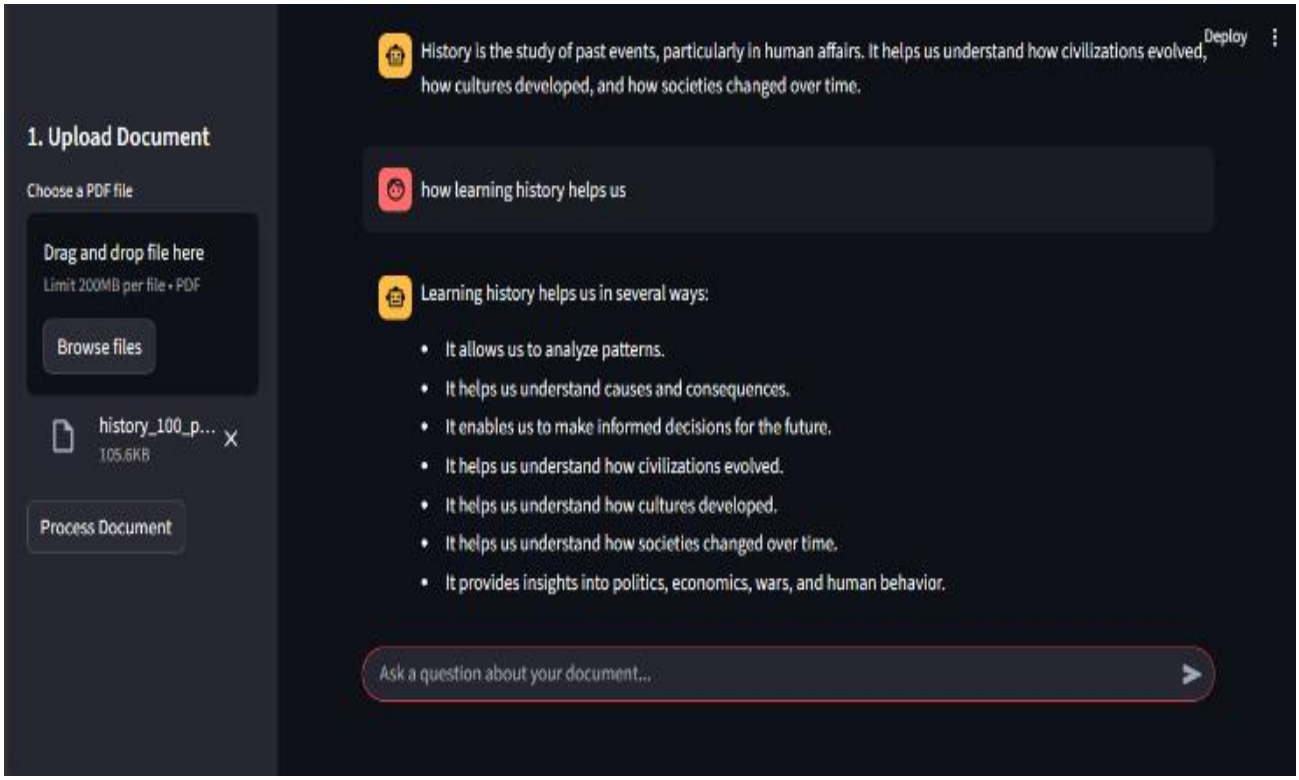

Figure 4.2 (e): Example of question answering with retrieved contextual information

4.3 Metrics for Evaluation

The following measures were used to assess the system's performance:

1. Precision: Accuracy is the number of questions the system got right divided by the total number of questions asked. This is the primary metric for assessing the overall accuracy of the system's responses. Here, accuracy was calculated for 250 questions across four categories (factual, summary, logic-based and inferential).

2. Precision (precision@3): Precision measures the relevance of the top-k document chunks returned by the system for a given query. A precision@3 score of 0.83 means that about 2.5 out of 3 retrieved chunks were relevant to the question. This metric is particularly relevant for retrieval-augmented generation (RAG) systems, where the quality of the answers directly relates to the quality of retrieved contexts.

3. Semantic similarity score: semantic similarity is the measures is the system-generated answer and the ground–truth answer. It is computed by cosine similarity on sentence embeddings produced by sentence transformers. Semantic similarity is better suited for open-ended question answering since it takes into account paraphrased or contextually equivalent responses as opposed to exact match metrics.

4. Mean response time: The response time is the average time the system takes to process a query and generate an answer. This shows the system's practical usability in real-world scenarios.

4.4 Quantitative Results

Table 4.4: Performance evaluation table

Question Type	Counts	Accuracy
Factual	80	95%
Summary	50	98%
Logic Based	60	82%
Inferential	60	76%
Overall	250	87.92%

Table 4.4 is a performance evaluation table of the proposed document analyzer system for different types of questions. The table shows how many questions were tested in each category and the accuracy that the system has achieved.

The system achieves the best performance for summary-based questions with 98% accuracy, which indicates that the system is very capable of understanding the document content. On factual questions, the system gets high accuracy of 95%, which means it can get direct information from the document efficiently. For logic-based questions, the system achieves an accuracy of 82%, which reflects the ability of the system to perform reasoning and understanding of context to a good extent. However, the performance is relatively lower for inferential questions, with an accuracy of 76%, as these questions require deeper interpretation.

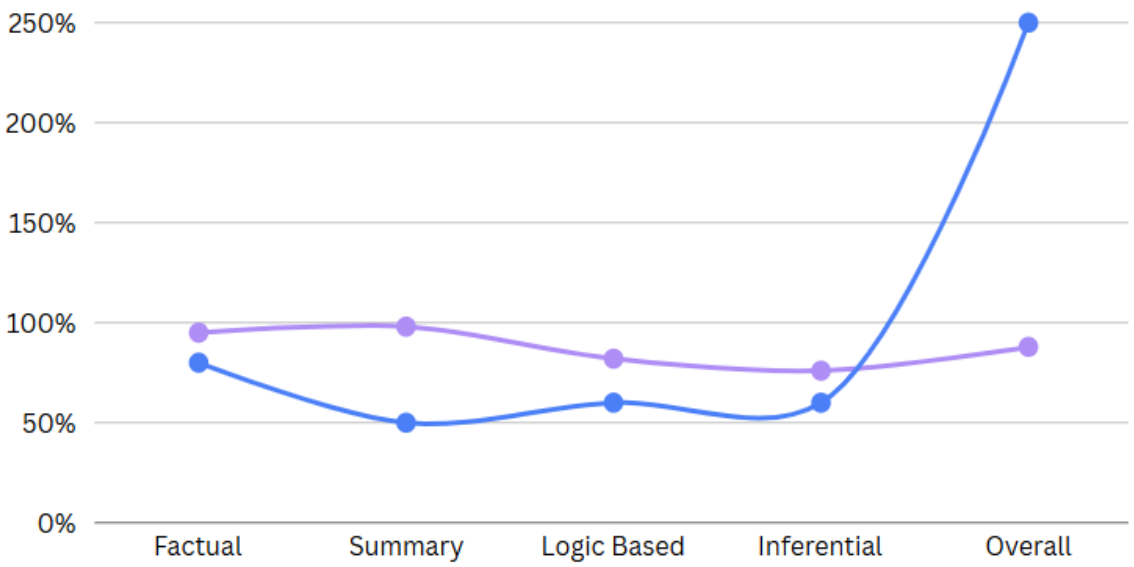

Figure 4.4 (a): Graph representation

The graph represents a quantitative analysis of the system. In this, the blue line represents Counts and the violet line represents Accuracy. The graph shows the number of questions and accuracy for different question types.

We observe that the total number of questions varies by category, with overall category having the highest total count. The performance is good for retrieving direct and summary information, as evidenced by consistently high accuracy for factual and summary questions. However, very poor accuracy is observed for logic based and inferential questions, indicating higher complexity for such queries.

As shown in the graph the system is efficient for simple tasks and summary-based tasks. The system is accurate and shows effectiveness for document analysis and question answering.

4.5 Comparative analysis

Table 4.5: *Comparative analysis*

Model	**Accuracy**	**Time**
TF-IDF + Cosine similarity	63.4%	1 sec
LLM only	69.8%	4 sec
BM25 + LLM	78.5%	2 sec
FAISS + LLM	87%	2 sec

The results of the document analyzer with the question-answer system show how well it processes the document and provides answers and a summary as well. With high accuracy and effectiveness, semantic search, text extraction and answer creation are accomplished. The system offers helpful features such as summary generation and a question-answer system. The system can be successfully used in several fields, which include business for report analysis, education for learning reports, and research for massive document analysis. The system exhibits practical applications. Some applications are given below

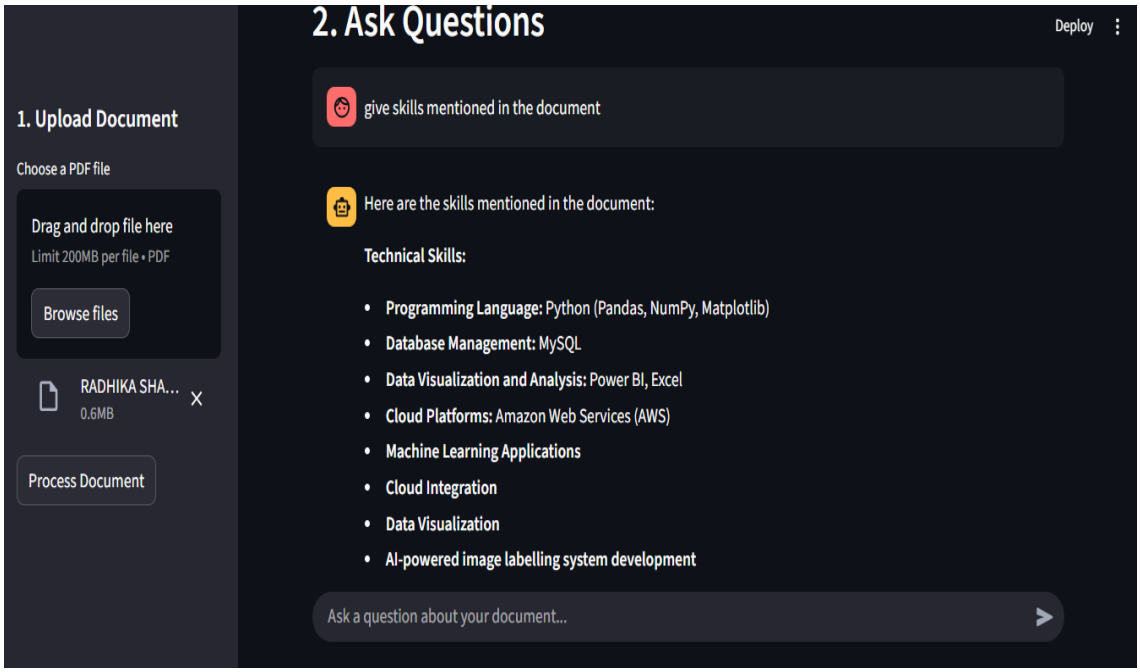

Figure 4.5 (a): Resume analysis

Figure 4.5 (a) shows resume analysis of the system. For example, the user uploads resume as a PDF and the system pulls ley information from it. It answers questions like “skills mentioned”. This shows accuracy in document understanding and response generation.

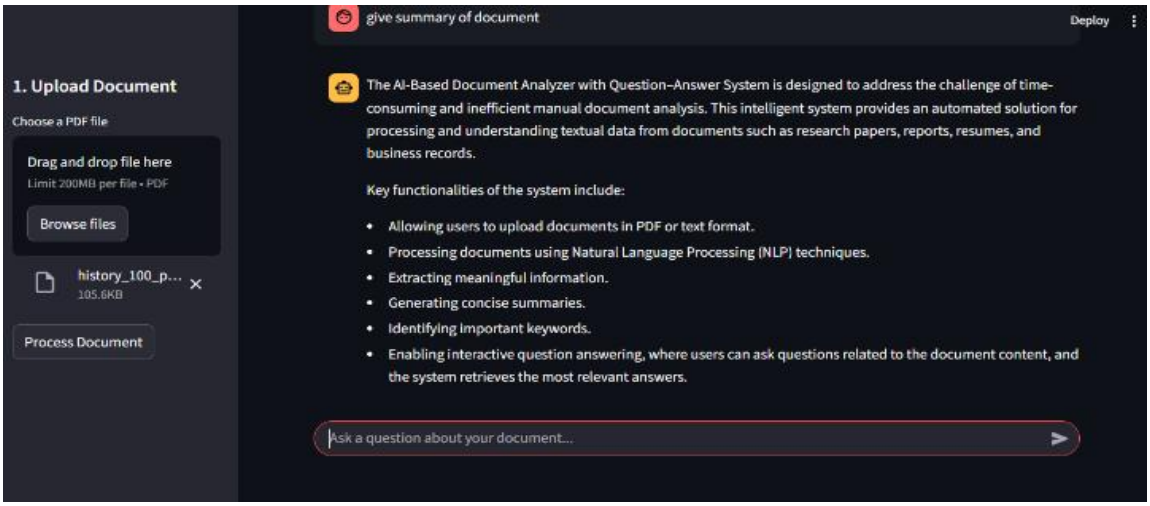

Figure 4.5 (b): Massive data analysis

Figure 4.5 (b) shows the summary generation feature of the system. You upload a document, the system reads the document, does chunking and text extraction and then gives a summary. It shows important features such as Keyword identification and an interactive question-answering system.

5. Conclusions

In order to handle and analyze user uploaded documents in the .pdf and .txt formats, the proposed project offers a GenAI-based document understanding and question- answering system. Retrieval-augmented generation (RAG) is used by the system to deliver precise, context-aware, comprehensible replies. The procedure starts with document intake, when PyMuPDF is used to extract text and preprocess it into digestible pieces. Sentence transformers are used to convert these segments into vector embeddings, which are then saved in a vector database driven by FAISS. To ensure responses are grounded in the source material, the system uses a broad language model, integrated via LangChain, to retrieve the most pertinent document segments when a user query is received. Furthermore, the approach improves transparency and dependability by justifying the return of supporting text references.

The system is appropriate for educational and knowledge management applications since it has capabilities like automatic document summarisation, question development and answer evaluation in addition to question answering. FastAPI is used to create the backend, and Streamlit is used to develop the frontend, allowing for an interactive user experience. According to experiment results, the system maintains low response latency while achieving excellent accuracy and effective retrieval performance. The system’s scalability and practical usefulness are highlighted by its noteworthy ability to handle huge texts of up to 50 pages without noticeably degrading performance. All things considered, the project effectively combines generative AI with retrieval techniques to provide an intelligent, scalable and comprehensible document helper.

6. Future Scope

The proposed document analyzer with a question-answering system can be further improved in several ways to enhance its performance and usability. The system can be extended to support other file formats, such as Word documents, images, and scanned PDFs, using OCR techniques. More advanced, fine-tuned models can improve accuracy, especially for complex and inferential queries. The system could be extended to reach more people by adding multilingual support in the future. The system can also be developed as a mobile app for more convenience and ease of use. Also, improving its data security and privacy capabilities will make it fit for processing sensitive documents. Also, real-time collaboration features can be added to allow multiple users to work on the same document. The system can also be customised for specific domains, such as healthcare, legal, and finance, to provide more specialised and accurate results. In conclusion, these improvements will make the system more robust, scalable, and adaptable to real-world applications.

References

[1] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2009.

[2] NLTK, "NLTK Documentation," [Online]. Available: https://www.nltk.org/. [Accessed: Mar. 19, 2026].

[3] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge University Press, 2008.

[4] T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," arXiv preprint arXiv:1301.3781, 2013

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019.

[6] PyPDF2, "PyPDF2 Documentation," [Online]. Available: https://pypdf2.readthedocs.io/. [Accessed: Mar. 19, 2026].

[7] Amazon Web Services, "AWS Documentation," [Online]. Available: https://docs.aws.amazon.com/. [Accessed: Mar. 19, 2026].

[8] Amazon Web Services, "Amazon S3 User Guide," [Online]. Available: https://docs.aws.amazon.com/s3/. [Accessed: Mar. 19, 2026].

[9] Amazon Web Services, "AWS Lambda Developer Guide," [Online]. Available: https://docs.aws.amazon.com/lambda/. [Accessed: Mar. 19, 2026].

[10] Amazon Web Services, "Amazon API Gateway Documentation," [Online]. Available: https://docs.aws.amazon.com/apigateway/. [Accessed: Mar. 19, 2026].

[11] Napkin AI, “Napkin AI.” [Online]. Available: https://www.napkin.ai. [Accessed: Apr. 10, 2026].