Intelligent Document Assistant A RAG Approach For Conversational Knowledge Access

Machine Learning Final Project Report

GROUP - 12

Ayaazuddin Mohammad , Mukesh Kumar Javvaji, Satvika Eda
Khoury College of Computer Sciences
Northeastern University
Boston, USA
mohammad.ay@northeastern.edu, javvaji.m@northeastern.edu,
eda.s@northeastern.edu

***Abstract*—In today's information-driven era, users often encounter challenges in efficiently accessing relevant knowledge from large textual documents, such as textbooks, research papers, and lecture notes. This project aims to develop an Intelligent Document Assistant using a Retrieval-Augmented Generation (RAG) approach, allowing users to upload lengthy documents and interact with them through natural language queries.**

The system leverages document parsing and preprocessing techniques to extract meaningful content from user-uploaded files. The extracted text is divided into manageable chunks and converted into dense vector embeddings using a custom embedding model. These embeddings are stored in a vector database, enabling efficient semantic similarity-based retrieval of relevant content. A chatbot interface facilitates seamless interaction by utilizing retrieved document chunks to generate contextually relevant responses via a generative language model.

The implementation involves integrating document preprocessing (e.g., PyPDF2), embedding generation, and vector database management (e.g., ChromaDB) to build a scalable and user-friendly solution. The chatbot provides real-time, context-aware responses, enhancing user engagement and enabling faster information access. The system's performance is evaluated through metrics such as Precision, Recall, and Mean Reciprocal Rank (MRR) to ensure its effectiveness and reliability.

Keywords: Retrieval-Augmented Generation, Large Language Models, Embedding Models

I. INTRODUCTION

As the volume of digital information continues to grow, the ability to quickly and effectively retrieve meaningful insights from large documents has become increasingly important. Educational materials such as textbooks, research papers, and lecture notes are rich sources of knowledge, yet their length and complexity often make it difficult for users to locate specific information or understand the content efficiently. Current search tools provide limited assistance, often returning results based on keyword matching rather than the actual semantic meaning of the query.

To address this challenge, this project introduces an Intelligent Document Assistant designed to enable conversational interactions with large documents. The system allows users to upload extensive files, such as PDFs or text documents, and engage with their content through natural language queries. Unlike traditional search engines, this assistant combines semantic search with generative language models to retrieve and summarize relevant information in real-time.

The project leverages advanced techniques for text parsing, embedding generation, and vector-based retrieval. Uploaded documents are pre-processed and split into manageable chunks, which are -then converted into dense vector representations. These representations are stored in a vector database, enabling efficient similarity-based retrieval. A generative language model is used to provide coherent, context-aware responses based on the retrieved information, offering an experience akin to interacting with a knowledgeable assistant.

The Intelligent Document Assistant is designed to enhance user learning and engagement by providing precise, contextually relevant answers to complex queries. By transforming large, static documents into dynamic, interactive resources, this system seeks to redefine how users interact with and consume information.

II. LITERATURE REVIEW

The integration of artificial intelligence into educational settings has introduced novel opportunities to enhance learning experiences and improve access to knowledge. Retrieval-Augmented Generation (RAG) systems have emerged as a particularly promising approach in higher education, leveraging the combined strengths of large language models (LLMs) and information retrieval techniques to provide precise and contextually relevant responses to student queries. These systems address several challenges traditionally associated with AI-powered educational tools, including mitigating hallucinations and inaccuracies in LLM outputs, providing clear source attribution for generated responses, and tailoring information to align with specific course content and materials.

One case study[1] conducted at a private university underscored the transformative potential of RAG systems in education. The prototype developed in this study demonstrated significant improvements in fostering interactive learning experiences, generating contextually accurate answers, and delivering efficient question-answering capabilities. This evidence highlights the capability of RAG systems to create a more dynamic and personalized educational environment, where students can engage deeply with subject matter and benefit from reliable AI assistance.

A crucial component of RAG systems is their reliance on efficient vector search mechanisms to operate on large datasets. The increasing use of vector embeddings necessitates advanced retrieval techniques capable of handling substantial data volumes. Recent advancements in vector search algorithms have prioritized both query efficiency and scalability. For instance, graph-based techniques and tree-based approaches have proven effective in optimizing indexing processes, while hybrid methods that combine multiple data structures have achieved significant performance gains. A notable development in this domain is the ELPIS[2] algorithm, which has demonstrated superior performance in latency-optimized settings, achieving up to twice the recall rates of other state-of-the-art methods for datasets containing up to one billion vectors.

Beyond retrieval mechanisms, the broader application of AI in education has expanded to include the development of AI-powered learning environments. These systems, which incorporate advanced techniques such as LLMs, RAG, and vector embedding, aim to assist students in understanding complex scientific texts and fostering critical thinking skills. For example, the OwlMentor system was specifically designed to support university students in engaging with scientific literature. It features document-based chat functionality, automatic question generation to enhance comprehension, and AI-driven quiz creation with feedback mechanisms. By providing these interactive and adaptive learning tools, OwlMentor[5] exemplifies how AI systems can empower students to navigate and assimilate complex information more effectively.

The adoption and effectiveness of AI-powered learning tools also depend significantly on student acceptance and engagement. The Technology Acceptance Model (TAM) has been widely utilized to assess the factors influencing the adoption of new educational technologies. Key determinants such as perceived ease of use, perceived usefulness, and intention to use have been shown to play a critical role in the actual usage and effectiveness of these tools. Studies suggest that when students perceive AI systems as intuitive and beneficial, their engagement with these tools increases, ultimately enhancing learning outcomes.

In summary, the convergence of RAG systems, efficient vector search technologies, and AI-powered learning environments presents a compelling vision for the future of education. These innovations not only address existing limitations in AI-based educational tools but also provide pathways for creating highly interactive, contextually relevant, and user-friendly learning experiences. Furthermore, understanding and addressing factors influencing technology acceptance is essential to maximizing the impact of these systems in improving educational outcomes.

III. METHODOLOGY

To achieve the objectives of the Intelligent Document Assistant, the project employs a structured methodology comprising the following steps:

A. Document Parsing and Preprocessing:

The first step in the system pipeline involves extracting meaningful content from user-uploaded documents. This process includes:

- File Handling: Supporting document uploads in formats such as PDF and text files.
- Text Extraction: Using tools like PyPDF2 to extract raw text from PDFs or directly processing plain text files.
- Preprocessing: Cleaning and normalizing the extracted text, which includes removing unnecessary metadata, handling special characters, and converting text to lowercase.

B. Document Chunking and Vectorization

To ensure efficient storage and retrieval, the parsed text is split into smaller, manageable chunks and converted into dense vector representations:

- Text Splitting: Implementing chunking techniques with overlapping contexts using a RecursiveCharacterTextSplitter to preserve the semantic flow across sections.
- Vector Embedding: Generating dense vector representations for each chunk using a custom embedding model that captures the semantic meaning of the text. These embeddings form the core of the knowledge base.

C. Knowledge Base Creation

The processed document chunks and their corresponding embeddings are stored in a vector database for semantic search and retrieval:

- Vector Database Setup: Using ChromaDB to store and manage embeddings efficiently.
- Persistence: Ensuring embeddings are persisted for reuse across multiple sessions, allowing users to query documents uploaded earlier.

D. Semantic Retrieval

When a user submits a query, the system identifies the most relevant sections of the document based on semantic similarity:

- Query Embedding: The user's query is embedded into the same vector space as the document chunks.
- Similarity Search: Using algorithms like cosine similarity to retrieve the top-k most relevant chunks from the vector database.

E. Response Generation

To provide meaningful answers, the system integrates the retrieved text chunks with a generative language model:

- Prompt Construction: Combining the user query with retrieved chunks into a structured prompt.
- Language Model Interaction: Leveraging a pre-trained model, such as ChatGPT, to generate contextually coherent responses based on the prompt.
- Error Handling: Managing situations where no relevant chunks are retrieved by returning a fallback message.

F. Chatbot Interface Development

The system includes an intuitive and user-friendly interface for interaction:

- Frontend Development: Built using Streamlit to allow users to upload documents, submit queries, and view responses in real-time.
- Session Management: Storing user messages and system responses to maintain conversational context across queries.
- File Upload Validation: Ensuring compatibility with supported formats and restricting large files to maintain performance.

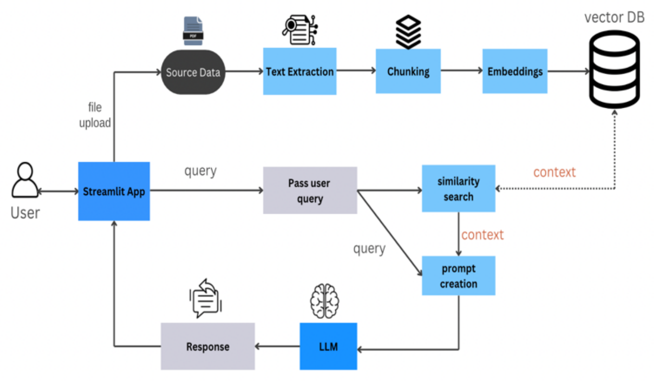

Figure 1 : Architecture

By integrating these steps into a cohesive pipeline, the Intelligent Document Assistant facilitates efficient and accurate retrieval of information from large documents, transforming them into interactive, conversational knowledge resources.

IV. IMPLEMENTATION

A. Text Splitting

To achieve effective text segmentation, we employ the RecursiveCharacterTextSplitter utility. This method divides the text based on a predefined character limit, while respecting semantic structures such as sentences or paragraphs. Overlapping regions between consecutive chunks ensure smooth transitions and preserve relationships between adjacent text segments.

The text-splitting process proceeds as follows:

1. Input:
   - A document's text and its associated metadata are provided as input.
2. Chunk Generation:
   - The RecursiveCharacterTextSplitter divides the text into chunks, ensuring that each chunk adheres to the specified size and overlap.
3. Output:
   - The output is a list of Document objects, where each object contains a chunk of text and its associated metadata.

B. Custom Embeddings Model

The embedding model is a critical component of our system, designed to transform discrete token indices into dense vector representations. These embeddings capture the semantic meaning of words or text chunks, enabling effective downstream tasks such as similarity search and context-aware predictions. Below, we outline the architecture and functionality of the embedding model, implemented using PyTorch.

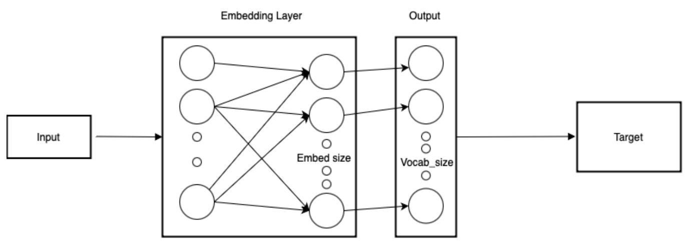

Figure 2: Embedding Model

The model generates vector embeddings that:

- Represent the semantic meaning of text chunks and user queries.

- Enable comparison between text chunks and queries in a shared vector space.
- Form the basis for efficient similarity-based retrieval in the vector database.

i. Layers:

The embedding model comprises two primary components: the embedding layer and the output layer. The former maps tokens to a continuous vector space, while the latter projects these embeddings back into the vocabulary space for target token prediction. The architecture is lightweight yet powerful, making it well-suited for both training and inference scenarios.

1. Embedding Layer:
    - This layer maps token indices to dense vector representations using a trainable embedding matrix of size V×DV times DV×D, where VVV is the vocabulary size and DDD is the embedding dimension.
    - The embedding matrix is initialized randomly and updated during training to capture semantic relationships between tokens.
2. Output Layer:
    - A fully connected layer projects embeddings into the vocabulary space, producing logits for each token in the vocabulary. This layer includes a weight matrix of size D×VD times VD×V and a bias vector of size VVV.

The forward pass involves retrieving embeddings for input tokens, transforming them through the output layer, and producing logits for all vocabulary tokens.

ii. Training Process:

The training process optimizes the embedding model to predict context words given a target word, using a dataset of (target, context) word pairs. The following steps outline the training workflow:

1. Data Preparation:
    - The input comprises a list of (target, context) pairs, where each pair is converted to numerical indices based on the vocabulary.
2. Loss Function:
    - Cross-Entropy Loss is used to measure the discrepancy between the predicted logits and the true context word. It encourages the model to assign higher probabilities to the correct context word.
3. Optimizer:
    - The Adam optimizer is employed to update the embedding matrix and output layer weights. Adam is chosen for its adaptive learning rate, which ensures efficient convergence.
4. Training Loop:
    - The model iterates over the dataset for a specified number of epochs, minimizing the loss through backpropagation.

C. Prompt Engineering

The engineered prompt includes several critical details to ensure the chatbot's responses are accurate, context-aware, and user-focused. It begins with a role definition, specifying that the chatbot is designed to answer questions related to the provided document. The purpose and scope are clarified, emphasizing that responses must rely exclusively on the given context, avoiding external knowledge or unsupported assumptions.

The prompt includes answering guidelines to provide factual and relevant answers, ensuring adherence to the document's content. To handle situations where information is unavailable, the prompt specifies a fallback strategy, instructing the chatbot to inform the user and suggest query rephrasing. It also details prioritization rules, focusing on accuracy, relevance, and demonstrating understanding of user queries while gracefully indicating when answers are not found.

Additionally, formatting instructions ensure uniform presentation of responses, maintaining professionalism and consistency. The prompt also includes instructions for handling conversational queries, enabling the chatbot to provide generic yet engaging responses when the interaction deviates from document-specific questions. These details collectively guide the language model to generate coherent, context-appropriate, and user-friendly outputs.

V. RESULTS

The embedding model performed as expected, effectively capturing semantic relationships between textual inputs. To validate its performance, cosine similarity scores were computed for pairs of sentences to evaluate their semantic alignment. The results demonstrated a consistent correlation between high similarity scores and semantically related sentences, confirming the reliability of the embedding model. This alignment indicates that the model successfully encodes textual information in a meaningful way, which is critical for downstream tasks such as context retrieval in the RAG system.

For each query, both precision and recall were calculated, and an average was taken across multiple queries to assess the overall performance. The model's ability to maintain high recall, ensuring that relevant information is not overlooked, alongside high precision, which limits the retrieval of irrelevant content, was key to its effectiveness.

These metrics were calculated based on the assumption that only content explicitly found in the uploaded document is relevant to the query. Given the constraints of the system, the performance of the embedding model was highly consistent, showing that the embedding process and retrieval mechanism were functioning as intended. The evaluation strategy, which utilized k-fold cross-validation, helped address potential issues of overfitting and underfitting, ensuring that the model generalizes well across various query types and document structures.

The precision and recall values highlight the model's capacity to accurately retrieve context-aware responses from the document, providing reliable insights and contributing to an enhanced user experience. Furthermore, an ablation study conducted to evaluate the impact of different configurations confirmed the robustness of the embedding model in diverse settings, reinforcing its suitability for use in real-time document interaction applications.

VII. FUTURE FOCUS

1. OCR Integration: Implement support for image uploads, enabling seamless text extraction from images to enhance data accessibility and usability.
2. Multi-file Upload: Develop functionality for uploading multiple files simultaneously, allowing richer context generation for user queries.
3. Feedback Mechanism: Introduce a response rating system to collect user feedback, driving continuous improvement in system performance and accuracy.
4. Context Persistence: Implement mechanisms to store and retrieve chat sessions, ensuring continuity and personalized user experiences over time.
5. Advanced Query Handling: Integrate re-ranking algorithms to optimize the relevance and quality of responses for complex or nuanced queries.

VIII. CONCLUSION

The Intelligent Document Assistant successfully integrates retrieval-augmented generation (RAG) with prompt engineering to create an effective conversational interface for exploring large documents. By combining document chunking, semantic embeddings, and carefully structured prompts, the system delivers accurate, context-aware responses while gracefully handling missing information. It enhances user engagement by transforming static documents into interactive knowledge resources. This project demonstrates the scalability and adaptability of the RAG framework, making it a valuable tool for education, research, and information access.

IX. REFERENCES

1. *Triwicaksana, M. B., & Oktavia, T. (2023). Building a Retrieval-Augmented Generation System for Enhanced Student Learning: Case Study at Private University. Journal of Theoretical and Applied Information Technology, 101(22), 7381-7386*

2. *Azizi, I. (2024). Vector Search on Billion-Scale Data Collections. VLDB 2024 Ph.D. Workshop.*

3. *Vidra, N. (2024). Improving Retrieval for RAG based Question Answering Models on Financial Documents. arXiv preprint arXiv:2404.07221.*

4. *Štula, M., Šimić, D., & Radovan, A. (2024). Integrating a Virtual Assistant by Using the RAG Method and Google Vertex AI Palm-2 Model. Applied Sciences, 14(22), 10748.*

5. *Mayer, R. E., Fiorella, L., & Stull, A. (2024). Exploring generative AI in higher education: a RAG system to support students' comprehension of scientific texts. Frontiers in Psychology, 15, 1474892.*