Notes-Buddy: An Interactive Assistant

Project Overview

This project aims to provide an interactive platform for analyzing and extracting insights from lecture notes and PDF documents using advanced Language Models (LLMs) and the Gemini Pro API. The system allows users to upload notes, ask questions in natural language, and receive AI-generated summaries and insights.

Approach and Architecture

1. Document Processing

- **Upload**: Users upload lecture notes or PDFs through the Streamlit interface.
- **Extraction**: We use PDF processing libraries to extract raw text from the documents.
- **Chunking**: The extracted text is divided into smaller, manageable chunks to optimize processing and retrieval.s

2. Indexing and Embedding

- **Embedding Generation**: Each text chunk is converted into a high-dimensional vector representation using a pre-trained embedding model.
- **Vector Storage**: These embeddings are stored in an efficient vector database (e.g., FAISS or Pinecone) for fast similarity search.

3. Query Processing

- **User Input**: Natural language queries are received through the Streamlit UI.
- **Query Embedding**: The user's query is embedded into the same vector space as the document chunks.
- **Retrieval**: Relevant chunks are retrieved from the vector store based on similarity to the query embedding.

4. Response Generation

- **Context Preparation**: Retrieved chunks are concatenated to form a context for the LLM.
- **LLM Interaction**: The Gemini Pro API is used to generate a response based on the user's query and the provided context.
- **Output Rendering**: The generated response is displayed to the user in the Streamlit interface.

5. Summarization and Insight Generation

- **Document-level Embedding**: A global embedding is created for the entire document.

- **Prompt Engineering**: Carefully crafted prompts are sent to the Gemini Pro API along with the document embedding to generate summaries and key insights.
- **Interactive Exploration**: Users can ask follow-up questions, with the system maintaining context for a conversational experience.

Frameworks and Models

1. **Gemini Pro API**: Chosen for its state-of-the-art performance in natural language understanding and generation. It offers a good balance between quality and cost.
2. **LangChain**: This framework simplifies the integration of LLMs into our application. It provides utilities for document loading, text splitting, and prompt management.
3. **Streamlit**: Selected for rapid development of interactive web applications with minimal front-end coding. It's Python-friendly and has a low learning curve.
4. **Vector Databases**: Essential for efficient similarity search as the volume of documents grows. They enable quick retrieval of relevant context for user queries.

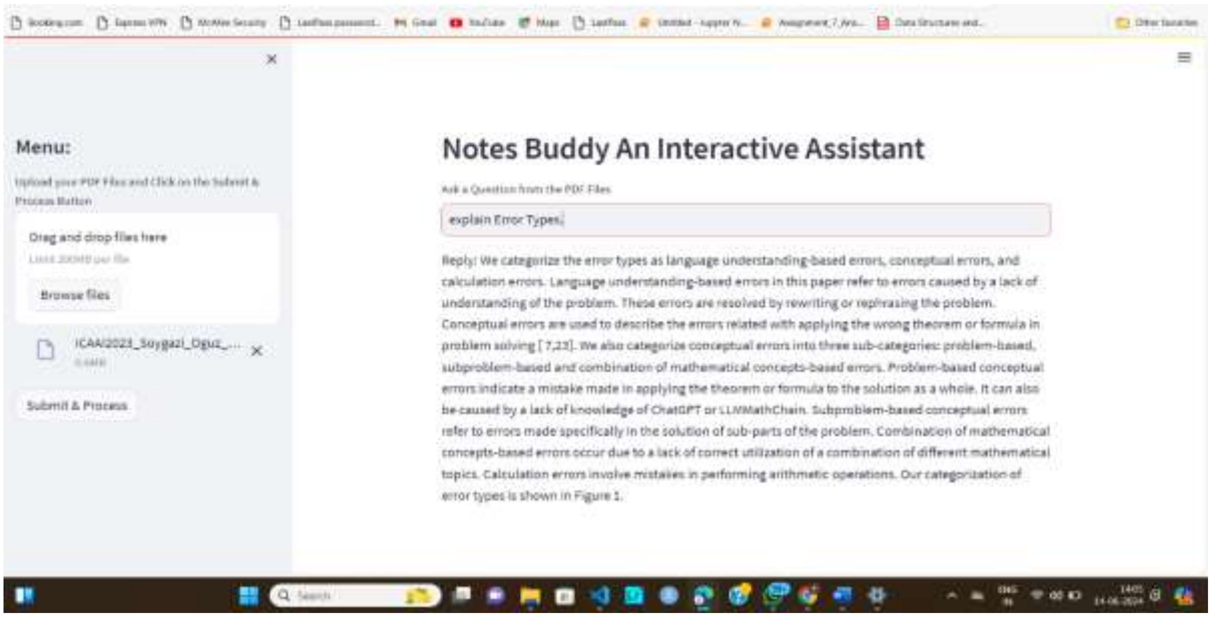
Areas for Improvement

1. **Fine-tuning**: Experiment with fine-tuning the LLM on a domain-specific dataset to improve response quality for specialized topics.
2. **User Feedback Loop**: Implement a mechanism for users to rate responses, using this feedback to improve retrieval and generation strategies.
3. **Security and Privacy**: Enhance document and query encryption to ensure user data privacy.

Deployment and Scaling Plan

1. **Containerization**: Package the application using Docker to ensure consistent environments across development and production.
2. **Serverless Deployment**: Utilize services like AWS Lambda or Google Cloud Functions for the backend, allowing automatic scaling based on demand.
3. **Content Delivery Network (CDN)**: Employ a CDN to serve static assets and improve global accessibility.
4. **Database Scaling**: As the number of documents grows:
   - Implement database sharding to distribute data across multiple servers.
   - Use a distributed vector database that supports horizontal scaling.
5. **Asynchronous Processing**: Implement a message queue (e.g., RabbitMQ, Apache Kafka) for handling document uploads and processing tasks asynchronously.
6. **Monitoring and Logging**: Integrate services like Prometheus and Grafana for real-time monitoring of system performance and user engagement metrics.

Screenshots: uploaded Research Paper and asked Questions to Notes-Buddy

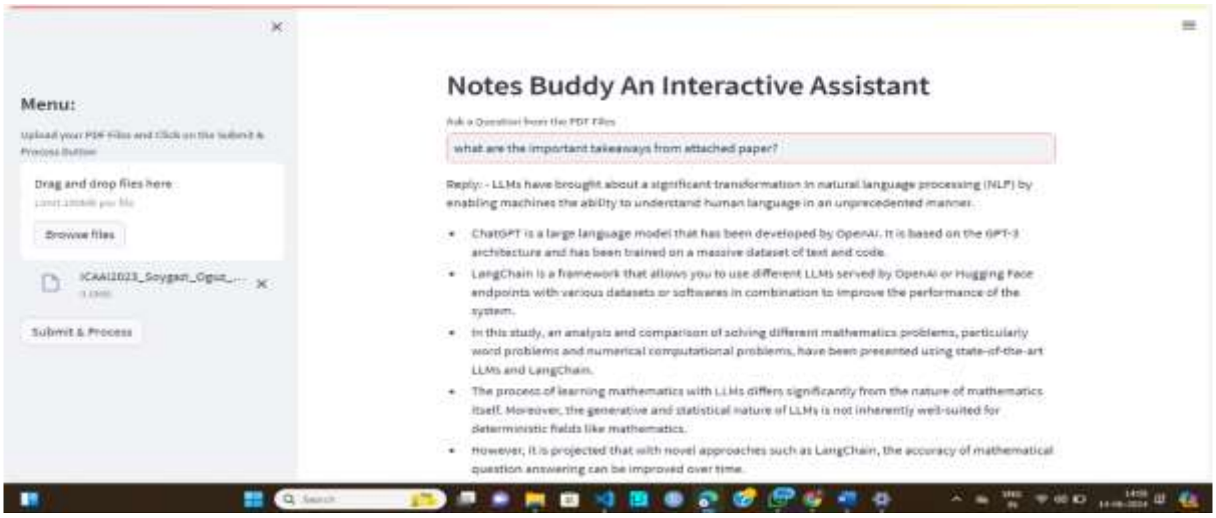

Menu:
Upload your PDF Files and Click on the Submit & Process Button
Drag and drop files here
Limit 200MB per file
Browse files
ICAAI2023_Soygazi_Oguz_...
Submit & Process

Notes Buddy An Interactive Assistant

Ask a Question from the PDF Files

explain Error Types.

Reply: We categorize the error types as language understanding-based errors, conceptual errors, and calculation errors. Language understanding-based errors in this paper refer to errors caused by a lack of understanding of the problem. These errors are resolved by rewriting or rephrasing the problem. Conceptual errors are used to describe the errors related with applying the wrong theorem or formula in problem solving [7,23]. We also categorize conceptual errors into three sub-categories: problem-based, subproblem-based and combination of mathematical concepts-based errors. Problem-based conceptual errors indicate a mistake made in applying the theorem or formula to the solution as a whole. It can also be caused by a lack of knowledge of ChatGPT or LLMMathChain. Subproblem-based conceptual errors refer to errors made specifically in the solution of sub-parts of the problem. Combination of mathematical concepts-based errors occur due to a lack of correct utilization of a combination of different mathematical topics. Calculation errors involve mistakes in performing arithmetic operations. Our categorization of error types is shown in Figure 1.


Menu:
Upload your PDF Files and Click on the Submit & Process Button
Drag and drop files here
Limit 200MB per file
Browse files
ICAAI2023_Soygazi_Oguz_...
Submit & Process

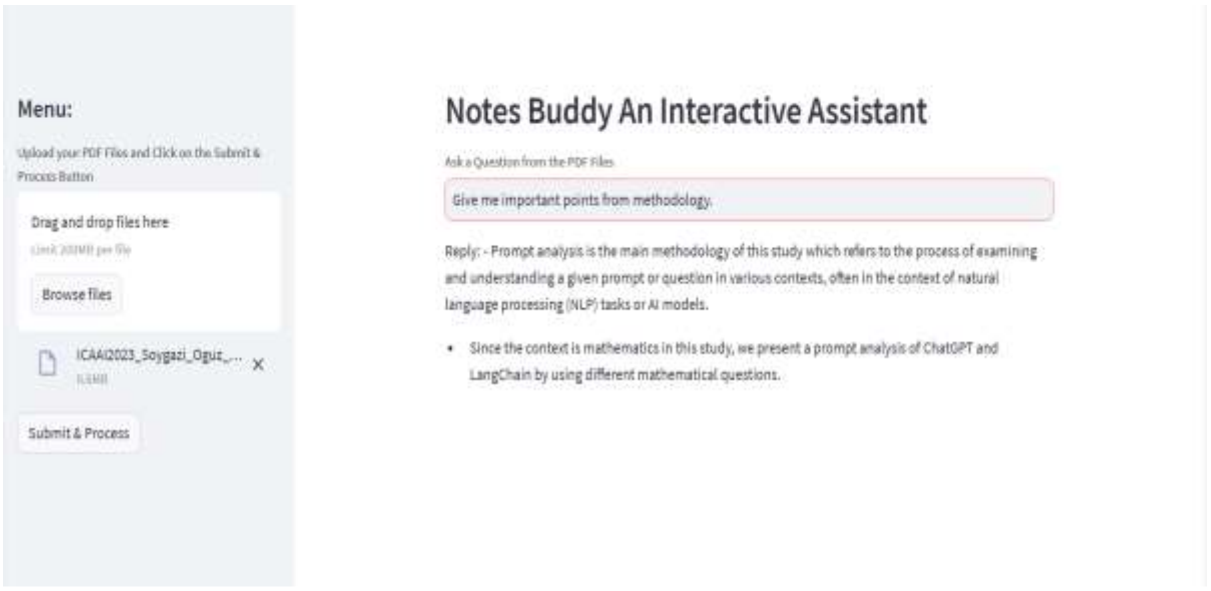
Notes Buddy An Interactive Assistant

Ask a Question from the PDF Files

what are the important takeaways from attached paper?

Reply: - LLMs have brought about a significant transformation in natural language processing (NLP) by enabling machines the ability to understand human language in an unprecedented manner.

- ChatGPT is a large language model that has been developed by OpenAI. It is based on the GPT-3 architecture and has been trained on a massive dataset of text and code.
- LangChain is a framework that allows you to use different LLMs served by OpenAI or Hugging Face endpoints with various datasets or softwares in combination to improve the performance of the system.
- In this study, an analysis and comparison of solving different mathematics problems, particularly word problems and numerical computational problems, have been presented using state-of-the-art LLMs and LangChain.
- The process of learning mathematics with LLMs differs significantly from the nature of mathematics itself. Moreover, the generative and statistical nature of LLMs is not inherently well-suited for deterministic fields like mathematics.
- However, it is projected that with novel approaches such as LangChain, the accuracy of mathematical question answering can be improved over time.


Menu:
Upload your PDF Files and Click on the Submit & Process Button
Drag and drop files here
Limit 200MB per file
Browse files
ICAAI2023_Soygazi_Oguz_...
Submit & Process

Notes Buddy An Interactive Assistant

Ask a Question from the PDF Files

Give me important points from methodology.

Reply: - Prompt analysis is the main methodology of this study which refers to the process of examining and understanding a given prompt or question in various contexts, often in the context of natural language processing (NLP) tasks or AI models.

- Since the context is mathematics in this study, we present a prompt analysis of ChatGPT and LangChain by using different mathematical questions.