Document Genie

Overview

Document Genie is an AI-powered chatbot that enables users to extract insights from PDF documents effortlessly. Using Google Generative AI for embeddings and responses, the chatbot processes uploaded PDF files, converts them into vector embeddings, and allows users to query the content in natural language. The system employs **FAISS (Facebook AI Similarity Search)** for efficient retrieval of document chunks relevant to user queries.

Features

- **PDF Processing**: Extracts text from uploaded PDF files.
- **Text Chunking**: Splits large text into smaller, manageable chunks.
- **Vector Embeddings**: Converts text into embeddings using Google Generative AI.
- **FAISS Indexing**: Stores embeddings for fast similarity search.
- **Conversational AI**: Answers user queries based on document content.
- **Streamlit UI**: Provides an easy-to-use web interface.

Installation

To run Document Genie, follow these steps:

1. Clone the repository:
2. git clone https://github.com/your-repo/document-genie.git
3. cd document-genie
4. Install the required dependencies:
5. pip install streamlit PyPDF2 langchain langchain-google-genai google-generativeai faiss-cpu
6. Run the Streamlit app:
7. streamlit run app.py

Usage

1. **Enter your Google API Key** in the provided input field.
2. **Upload one or multiple PDF files.**
3. Click on **Submit & Process** to extract and store document content.
4. Enter a **query** to ask questions related to the documents.
5. View AI-generated responses based on the document context.

Code Explanation

1. Streamlit Page Configuration

st.set_page_config(page_title='Document Genie', layout='wide')

- Sets the Streamlit app title and layout.

2. API Key Input

```
api_key = st.text_input('Enter your Google API key:', type="password", key="api_key_input")
```

- Users must input their **Google Generative AI API key** for processing documents.

3. PDF Text Extraction

```
def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text
```

- Uses **PyPDF2** to extract text from uploaded PDF files.

4. Splitting Text into Chunks

```
def get_text_chunks(text):
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=10000, chunk_overlap=1000)
    chunks = text_splitter.split_text(text)
    return chunks
```

- Splits long text into smaller, overlapping chunks for efficient retrieval.

5. Generating Vector Embeddings and FAISS Indexing

```
def get_vector_store(text_chunks, api_key):
    embeddings = GoogleGenerativeAIEmbeddings(model="model/embedding-001",
google_api_key=api_key)
    vector_store = FAISS.from_texts(text_chunks, embedding=embeddings)
    vector_store.save_local("faiss_index")
```

- Converts text chunks into vector embeddings using **Google Generative AI**.
- Stores embeddings in a **FAISS index** for fast similarity search.

6. Retrieving Relevant Chunks and Generating Responses

```
def user_input(user_question, api_key):
    embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001",
google_api_key=api_key)
    new_db = FAISS.load_local("faiss_index", embeddings)
```

```
    docs = new_db.similarity_search(user_question)

    chain = get_conversational_chain()

    response = chain({"input_documents": docs, "question": user_question}, return_only_outputs=True)

    st.write("Reply: ", response["output_text"])
```

- Loads stored embeddings from FAISS.
- Searches for relevant text chunks based on **user queries**.
- Uses **LangChain's question-answering model** to generate responses.

7. Streamlit UI

```
def main():

    st.header("AI Clone Chatbot 🤦 ")

    user_question = st.text_input("Ask a Question from the PDF Files", key="user_question")


    if user_question and api_key:

        user_input(user_question, api_key)


    with st.sidebar:

        st.title("Menu:")

        pdf_docs = st.file_uploader("Upload your PDF Files", accept_multiple_files=True, key="pdf_uploader")

        if st.button("Submit & Process", key="process_button") and api_key:

            with st.spinner("Processing..."):

                raw_text = get_pdf_text(pdf_docs)

                text_chunks = get_text_chunks(raw_text)

                get_vector_store(text_chunks, api_key)

                st.success("Done")


if __name__ == "__main__":

    main()
```

- Creates a **Streamlit-based UI** where users:
    - Upload PDFs.
    - Ask questions about the content.

    - View AI-generated responses.

Technologies Used

- **Python** (Main programming language)
- **Streamlit** (UI framework)
- **LangChain** (Conversational AI framework)
- **Google Generative AI** (Embeddings and AI responses)
- **PyPDF2** (PDF text extraction)
- **FAISS** (Efficient vector search)

Future Enhancements

- Support for **TXT, CSV, and Excel files**.
- **Multi-language support** for document processing.
- **Advanced AI models** for improved answers.
- **Cloud storage integration** for document uploads.

License

This project is licensed under the MIT License.

Contact

For any issues or contributions, feel free to reach out!

This README provides a **detailed explanation** of your chatbot's functionality. Let me know if you want modifications!