File Attachment Analysis in Chatbot

Report Created By: Anbhi Thakur
Date: 22 June , 2025
Title: *Implementation of Attachment Handling and Analysis in Chatbot*

---

1. Introduction

The purpose of this report is to document the implementation of a new feature in the chatbot that enables it to accept and analyze user attachments, including image files, PDF documents, and text files. The new functionality provides seamless interaction with users, extracting valuable information from uploaded files and performing necessary analysis using advanced algorithms.

This feature, when fully integrated, will allow the chatbot to handle various types of content, improve user experience, and automate the processing of documents.

---

2. Objectives

- To integrate file attachment handling in the chatbot.
- To implement algorithms that can interpret images, PDFs, and text files.
- To analyze and process uploaded files to extract meaningful information.
- To generate downloadable reports based on the analysis.

---

3. Algorithms and Techniques for File Processing

Image Processing (OCR)

- **Technique:** Optical Character Recognition (OCR) is used to extract text from images. The open-source tool **Tesseract OCR** is used to convert image data into machine-readable text.
- **Use Case:** Extracting text from scanned documents or screenshots.
- **Libraries Used:** `pytesseract`, `Pillow` (for image manipulation).

PDF File Processing

- **Techniques:** For PDF text extraction, **PyMuPDF** (fitz) and **pdfplumber** are used. These libraries help extract raw text from PDF pages, and also handle structured data such as tables and images.
- **Use Case:** Extracting both textual and tabular data from PDF files.

- **Libraries Used:** `pdfplumber`, `PyMuPDF`, `PyPDF2`.

Text File Processing

- **Techniques: spaCy** and **NLTK** are used for natural language processing (NLP) tasks such as named entity recognition (NER), sentiment analysis, and text summarization.
- **Use Case:** Processing plain text files to perform keyword extraction, sentiment analysis, or other NLP tasks.
- **Libraries Used:** `spaCy`, `NLTK`.

---

4. Detailed Methodology

The feature implementation involves several components, each handling different types of files. Below is a breakdown of the methodology used to process each file type:

1. **Attachment Upload and Handling:**
   - The chatbot frontend (using **Streamlit**) allows users to upload image files (JPG, PNG), PDF files, and text files using the `st.file_uploader()` component.
2. **Image Files (OCR Processing):**
   - Once an image file is uploaded, it is processed using **Tesseract OCR** to extract any text present within the image.
   - The extracted text is displayed within the chatbot interface for user review.
3. **PDF Files (Text and Data Extraction):**
   - When a PDF file is uploaded, **pdfplumber** is used to extract text from the document.
   - Any structured data, such as tables, are also identified and extracted for further analysis.
4. **Text Files (Text Processing with NLP):**
   - Text files are directly read and processed using **spaCy** for natural language tasks, such as entity recognition and summarization.
   - The text is analyzed, and named entities (such as person names, organizations, dates, etc.) are displayed.
5. **Report Generation:**
   - A detailed report is generated after processing the uploaded file. The report is available for download, containing a summary of the extracted information and insights.

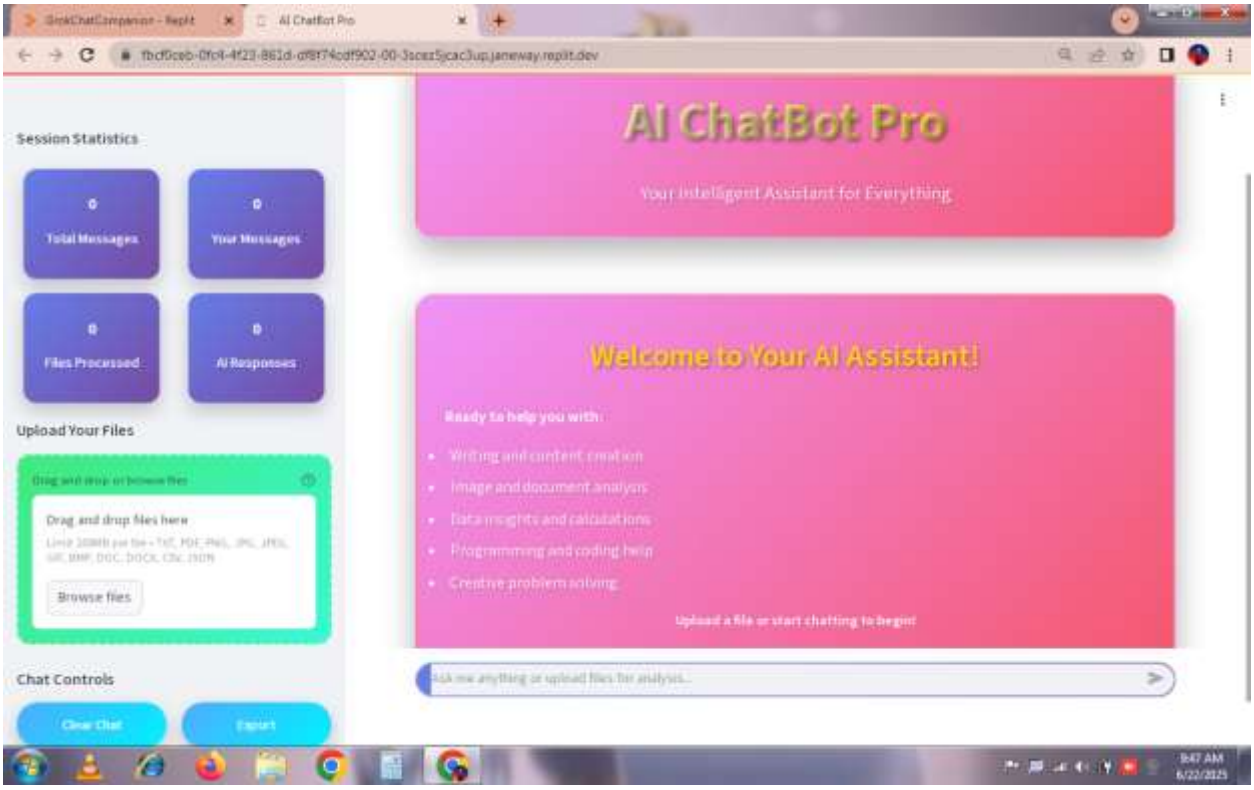
---

5. Implementation with Streamlit

The implementation was completed using **Streamlit**, which allows for quick web-based prototyping and visualization. The app is built to handle file uploads and display processed results in an interactive and user-friendly format. Below is the structure of the **Streamlit** app:

1. **File Upload Section:**
    - Users can upload image files (JPG, PNG), PDFs, or text files.
    - The app automatically detects the file type and routes it to the appropriate processing function.
2. **File Processing Functions:**
    - The `process_image()` function uses Tesseract OCR for image file processing.
    - The `process_pdf()` function uses pdfplumber to extract text from PDF files.
    - The `process_text_file()` function reads and processes text files.
3. **Output Results:**
    - Extracted text from the file is displayed for review.
    - For PDF files, text as well as any structured data (tables) are shown.
    - NLP-based analysis (e.g., named entity recognition) is performed on text files.
4. **Report Generation and Download:**
    - After processing, a downloadable report button is available for users to download their results.

5.Output



6. Conclusion

The feature to handle and analyze various file attachments (images, PDFs, text files) was successfully integrated into the chatbot using **Streamlit**. This feature not only improves the chatbot's interactivity but also enables it to process and extract data from user-provided documents.

Key technologies used include **Tesseract OCR** for image text extraction, **pdfplumber** for PDF parsing, and **spaCy** for natural language processing tasks. The integration of these technologies into a single interface ensures a seamless experience for users.

This feature is expected to be fully deployed by the end of the week, with real-time analysis and downloadable reports generated for users.

---

7. Future Improvements

- **Enhanced PDF Parsing:** Implementing more advanced parsing techniques for complex PDFs, such as handling embedded images and more structured content.
- **Real-time Analysis:** Adding real-time document analysis and feedback based on file content.
- **User Customization:** Allowing users to customize the type of analysis (e.g., sentiment analysis, summarization).

---

8. References

- Tesseract OCR: https://github.com/tesseract-ocr/tesseract
- PyMuPDF: https://pymupdf.readthedocs.io/
- pdfplumber: https://github.com/jsvine/pdfplumber
- spaCy: https://spacy.io/