



BotPDF: Assistant for Processing and Analysis of PDFs

Author: Wendy Zuloaga Victorio – 2212804

Advisor: Marco Antonio Casanova

Table of Contents

1.	Introduction.....	3
1.	Context and Motivation.....	3
2.	Description and General Objectives of the Software.....	3
2.	Technologies used.....	3
3.	Requirements Specification.....	4
4.	Architecture.....	4
5.	Software Description.....	6
6.	Bibliography.....	7

1. Introduction

1.1 Context and Motivation

In the realm of research, delving into documents can prove to be a challenging task. Thoroughly studying research requires a significant investment of time and dedication. While we may come across valuable documents, there are also many that contribute little or nothing to our work. Often, we access them based on names or references, only to find that they provide little to no benefit in the end. It would be highly intriguing to open a document and interact with it, gauging its usefulness immediately, thus saving us time. Currently, there are numerous technologies available that could streamline this process.

Feedback has been gathered from developers involved in the construction of vector databases, suggesting that the employment of GPT for the transformation of documents into varied formats has the potential to enhance the dependability of vector search in the development of RAG applications. A noteworthy illustration of this concept is the conversion of documents into question-and-answer pairs, followed by the indexing of documents based on vectors generated from these pairs. This approach intuitively appears to offer improved results for queries formatted as questions.

These studies revealed that in use cases where users are expected to pose questions rather than provide instructions to a Language Model, converting source documents into question and answer pairs is likely to yield more reliable results when performing vector retrieval. For instance, this preprocessing technique is likely more suitable for customer support or knowledge base search use cases, and less appropriate for agent-based workflows.

1.2 Description and General Objectives of the Software

The aim of this project is to utilize tools such as LangChain and FAISS to create a chatbot that enables users to upload PDFs and ask questions about the content therein.

2. Technologies used

The web application has been developed in Python, utilizing Flask for deployment. Flask is a lightweight web development microframework specifically designed for Python, known for its simplicity and user-friendly features essential for building web applications.

Regarding the frontend layer, it has been crafted using HTML, incorporating libraries such as Bootstrap and jQuery to enhance the interface and user experience.

PDF processing has been executed using LangChain (framework for developing applications powered by language models) and Faiss (library for efficient similarity search

and clustering of dense vectors) tools, leveraging their capabilities to efficiently perform specific operations on the content of PDF documents.

3. Requirements Specification

The primary requirement for the project is to have Python installed in a version equal to or greater than 3.10. Additionally, a specific set of essential libraries, such as Flask, LangChain, and OpenAI, is needed. To streamline this process, an "install.bat" file has been implemented, automating the creation of the necessary environment and the installation of corresponding dependencies.

A crucial aspect for the effective use of the project is obtaining an OpenAI key and adding the OPENAI_API_KEY variable to a ".env" configuration file. For detailed instructions on acquiring the API key, please refer to the following link: [API key](#).

All this information can be found in the project's README.



Fig.1 Requirements files within the project

4. Architecture

The architecture used for this project is monolithic. This choice is justified by the relatively simple nature of the project, where the interaction between components does not require significant complexity, and all elements are integrated in a holistic manner. The monolithic architecture provides an effective and easily manageable solution to meet the specific requirements of the project.

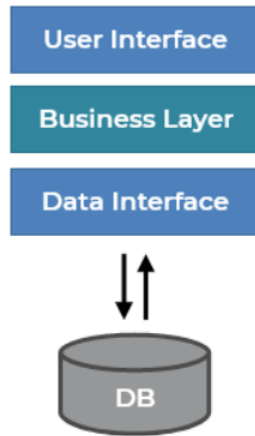


Fig.2 Monolithic Architecture

Below is the class diagram corresponding to this particular project. Key functions for the user are highlighted, such as loading documents, processing information, making queries through the chatbot, and the ability to export and clear the chat history.

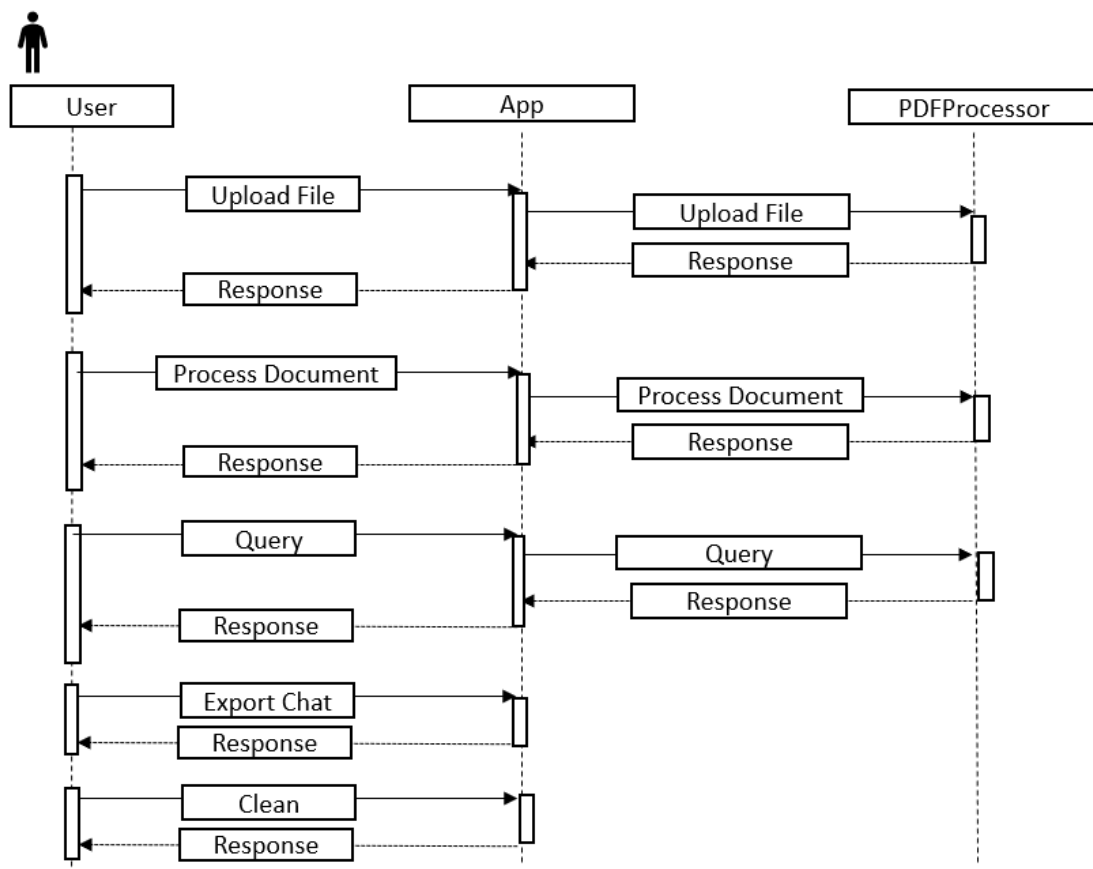


Fig.3 Sequence diagram

5. Software description

The project can be found in the following GitHub repository: <https://github.com/wendyzv319/BotPDF.git>. Simply clone it to access all of its content effortlessly. Within the project, there is a run.bat file that enables the user to deploy the application and run it in any browser effortlessly. The user interface of BotPdf is characterized by its simplicity. Once the user accesses the main page, a button is presented to facilitate the loading of PDF documents into the system.

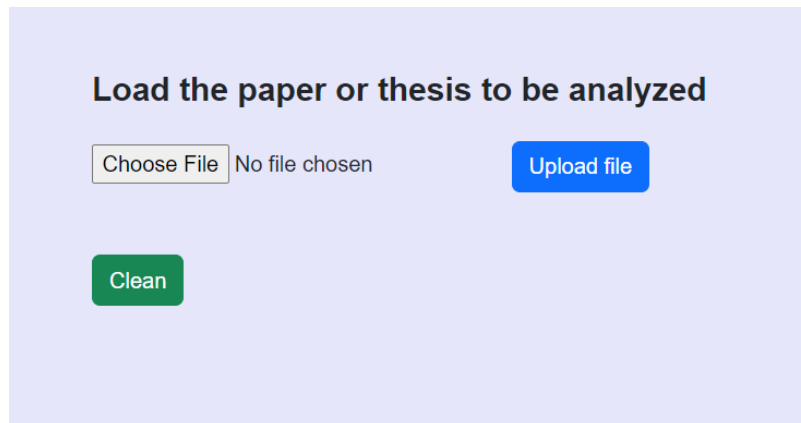


Fig.4 Main page

In the subsequent phase, the document analysis is initiated by the activation of a specific button. During this process, guided by the interface, an estimated duration is considered, which may extend for a few moments, possibly reaching up to a minute, during which the system conducts a detailed evaluation of the document's content.

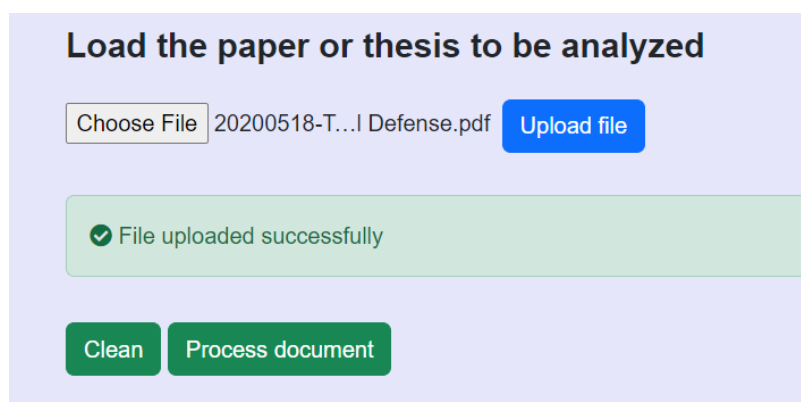
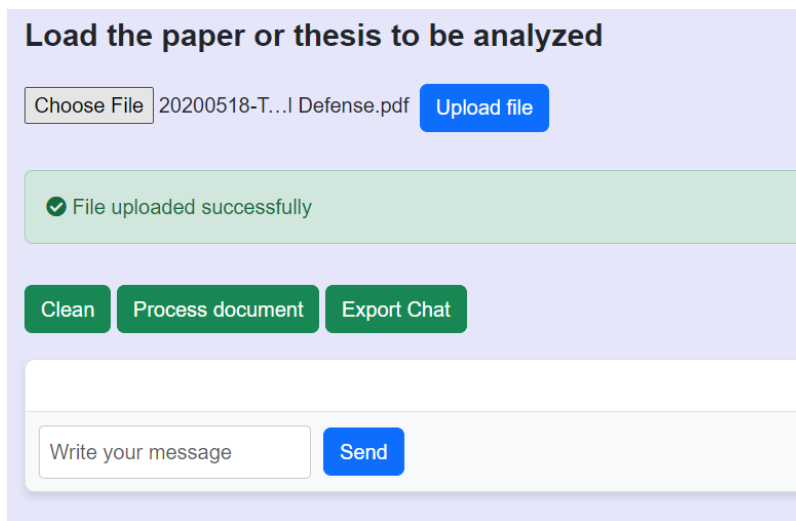


Fig.5 App after loading the document

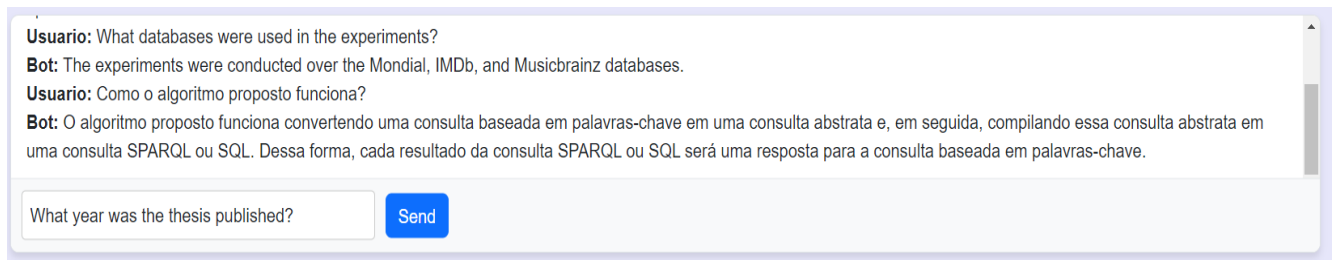
Once the document has been processed, a chat is displayed on the screen (Fig.6), allowing the user to make inquiries about the document. These questions and answers are stored

(Fig.7) and can be exported as a .txt file. Additionally, there is a button available to reset the application.



The screenshot shows a web interface with a light purple background. At the top, a heading reads "Load the paper or thesis to be analyzed". Below this, there is a file selection area with a "Choose File" button, a text input containing "20200518-T...I Defense.pdf", and an "Upload file" button. A green notification bar below the upload area states "File uploaded successfully" with a checkmark icon. Underneath, there are three green buttons: "Clean", "Process document", and "Export Chat". At the bottom, there is a white text input field with the placeholder "Write your message" and a blue "Send" button.

Fig.6 Input so that the user can make queries once the document has been processed



The screenshot displays a chat window with a light purple border. The chat history shows three messages: a user question in Portuguese, a bot response in Portuguese, and another user question in Portuguese. The bot's response explains the algorithm's function in Portuguese. At the bottom, there is a white text input field containing the question "What year was the thesis published?" and a blue "Send" button.

Fig.7 Example of how the chat would look

As depicted in Fig. 7, the chat responds in the language chosen by the user, regardless of whether the document is in English, Portuguese, or Spanish. BotPdf will consistently analyze the document and provide a response related to the information provided by the user. This represents another advantage of the software, as it enables the analysis of documents in any language.

6. Bibliography

- [Jason Fan \(2023\) - Improving vector search by converting documents to question/answer pairs](#)
- [Lucas Soares \(2023\) - Summarizing and Querying Multiple Research Papers with LangChain](#)
- https://python.langchain.com/docs/get_started
- <https://github.com/facebookresearch/faiss/wiki>