



BotPDF: Assistant for Processing and Analysis of PDFs

Author: Wendy Zuloaga Victorio – 2212804

Advisor: Marco Antonio Casanova

Table of Contents

1.	<u>Introduction.....</u>	<u>3</u>
1.	<u>Context and Motivation.....</u>	<u>3</u>
2.	<u>Description and General Objectives of the Software.....</u>	<u>3</u>
2.	<u>Technologies used.....</u>	<u>4</u>
3.	<u>Requirements Specification.....</u>	<u>4</u>
4.	<u>Architecture.....</u>	<u>5</u>
5.	<u>Software Description.....</u>	<u>7</u>
6.	<u>Uses Cases.....</u>	<u>8</u>
1.	<u>Scenarios in which users successfully use the software.....</u>	<u>8</u>
2.	<u>Scenarios where users have problems using the software.....</u>	<u>9</u>
7.	<u>Bibliography.....</u>	<u>11</u>

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. BotPDF seamlessly integrates into the realm of "Artificial Intelligence," standing out for its ability to incorporate advanced natural language processing features. This approach enables the system to comprehend and respond to queries effectively.

BotPDF is crafted to meet the needs of individuals seeking to process documents and obtain answers efficiently without the necessity of thorough review. Its distinctive advantage lies in the ability to respond promptly to queries, even in different languages. Users can ask questions in the language they prefer, not limited to the language of the document

The longevity of this software hinges on various factors, encompassing the evolution of user needs, its adaptability to technological changes, and the quality of ongoing updates and maintenance.

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

Regarding the technical requirements of the development environment, it is recommended, first of all, the presence of a dual-core processor as a minimum for acceptable performance, although it is preferably advisable to opt for a quad-core processor in situations that involve more tasks. demanding. Regarding RAM, a minimum of 4GB is established, although it is strongly suggested to have 8GB or more to guarantee optimal performance during the development process. As for storage, it is essential to have at least 10GB of free hard drive space for installing the development environment and storing projects. Python, being compatible with various operating systems, offers flexibility, allowing you to choose between Windows, MacOS and Linux according to the developer's preferences and needs. In addition, it is essential to have an Internet connection to carry out the installation of dependencies and the necessary updates during software development.

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.

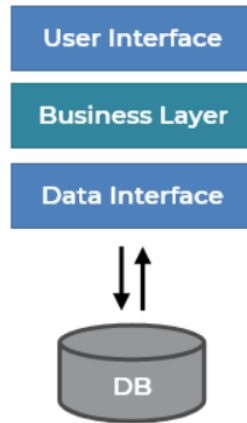


Fig.2 Monolithic Architecture

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.

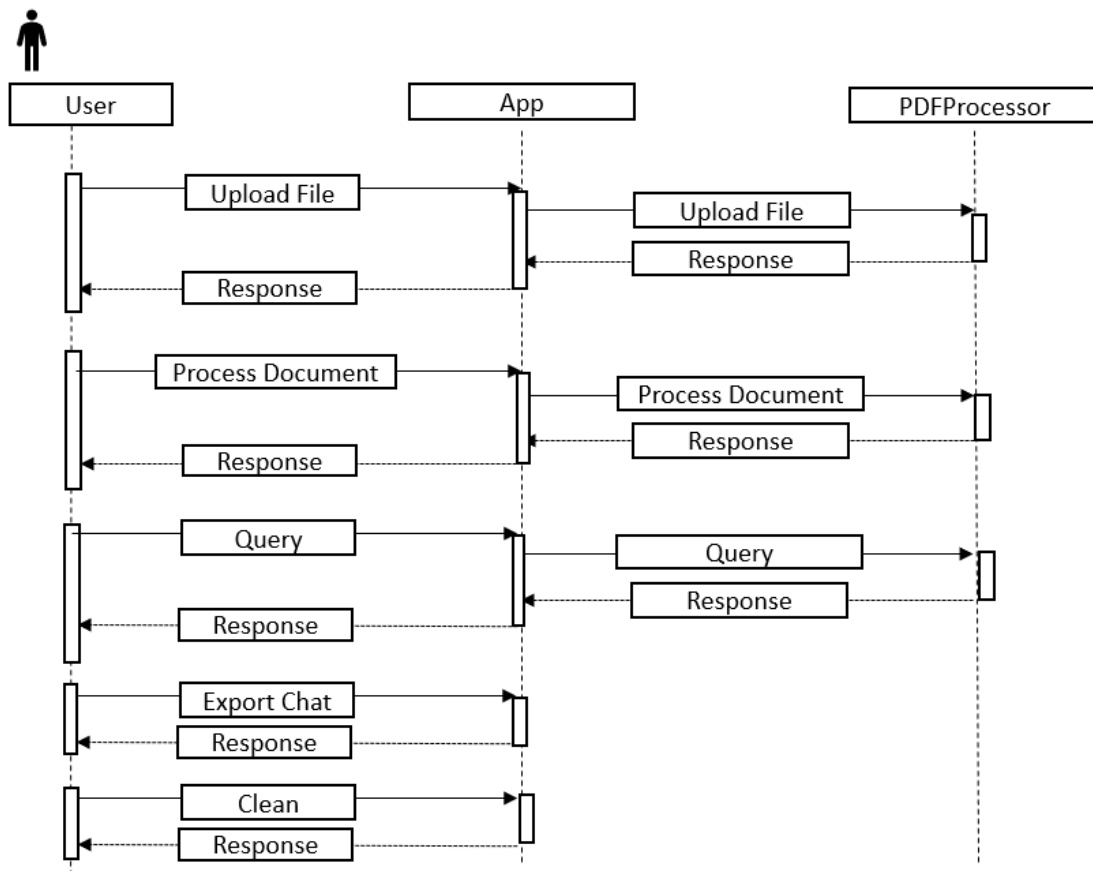


Fig.3 Sequence diagram

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 (Fig.4). Within the project, there is an install.bat and run.bat file that allow the user to install the necessary dependencies and deploy the application respectively.

```
> git clone https://github.com/wendyzv319/BotPDF.git
> cd /chatgpt_kws
Run the batch file:
> .\install.bat
Then, run:
> .\run.bat
```

Fig.4 Step sequences to start using BotPdf

The user interface of BootPdf is characterized by its simplicity. Once the user accesses the main page (Fig.5), a button is presented to facilitate the loading of PDF documents into the system.

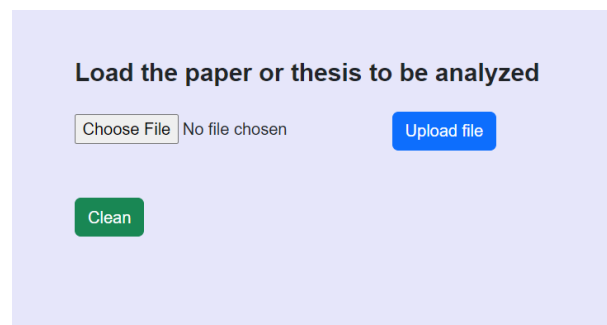


Fig.5 Main page

In the subsequent phase, the document analysis is initiated by the activation of a specific button (Process Document Fig.6). 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.

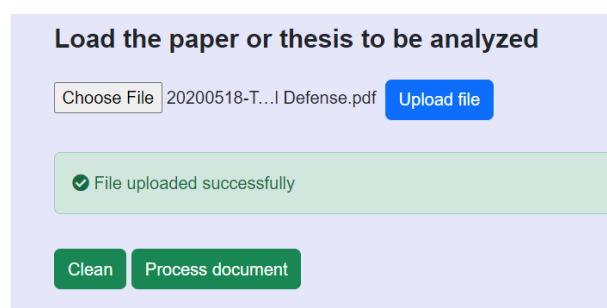


Fig.6 App after loading the document

If an error occurs during the document analysis process, a notification is displayed on the interface informing of the error (Fig. 7).

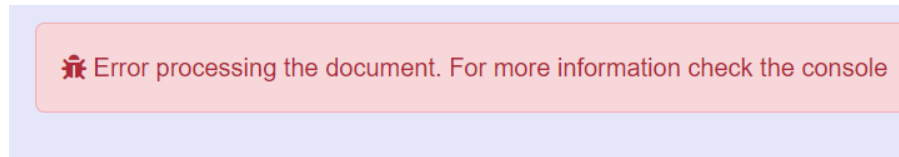


Fig.7 Notification when an error occurs processing the document

Once the document has been processed, a chat is displayed on the screen (Fig.8), allowing the user to make inquiries about the document. These questions and answers are stored (Fig.9) and can be exported as a .txt file. Additionally, there is a button available to reset the application.

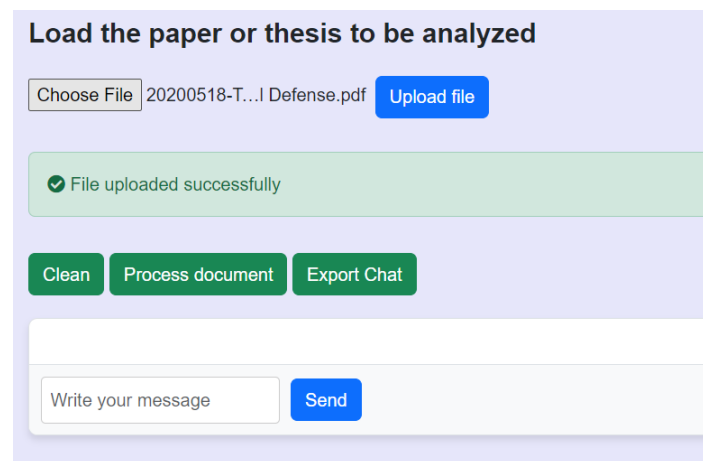


Fig.8 Input so that the user can make queries once the document has been processed

As depicted in Fig. 9, 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.

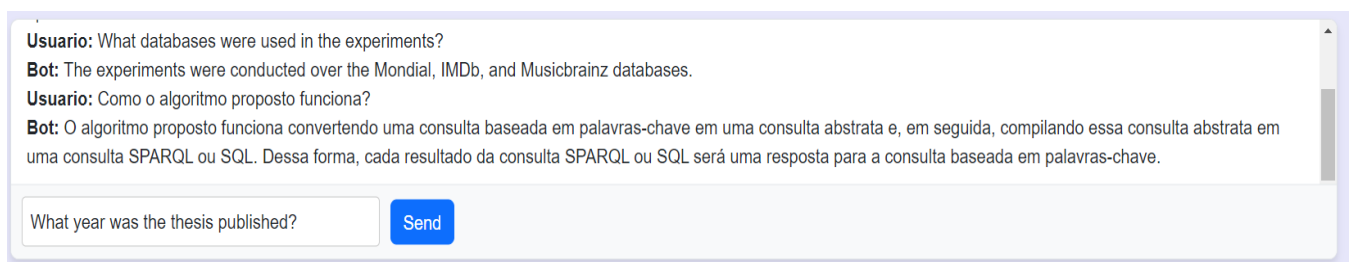


Fig.9 Example of how the chat would look

6. Uses cases

6.1 Scenarios in which users successfully use the software

Scenario 1: Academic Researcher during an investigation process

User: An academic researcher.

Context and Activity: The researcher is involved in extensive research and needs to quickly analyze large sets of documents in multiple languages to extract relevant information.

Narrative: The researcher downloads and installs BotPDF in their research environment. Configures the software to recognize and process documents in different languages. Uses the document processing function to automatically analyze the content of PDFs. Triggers the export function to save the extracted information in a format useful for your research.

Evaluation and Utilization: The researcher obtains accurate results and quickly processes large volumes of data, saving significant time compared to manual methods.

Scenario 2: Student with Poor Foreign Language Skills

User: A university student with basic knowledge of technology and limited foreign language proficiency.

Context and Activity: The student embarks on an academic investigation that involves the study of documents in several foreign languages, despite having limited linguistic skills in *foreign languages*.

Narrative: The student installs BotPDF, configures the software, uploads the documents, processes them and can ask questions about them.

Evaluation and Resulting Utility: The student experiences a significant improvement when using BotPDF. The tool allows you to address documents in foreign languages efficiently, overcoming linguistic limitations, thus achieving a more accessible understanding of the material studied.

6.2 Scenarios where users have problems using the software

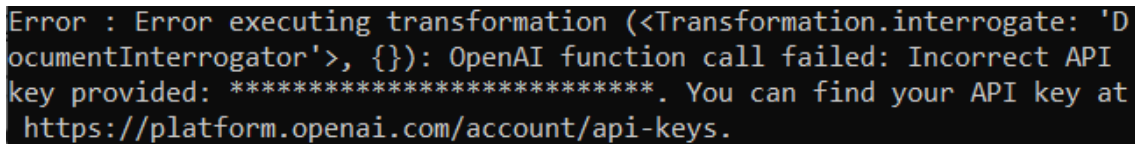
Scenario 1: Student or Academic Researcher without OpenAI API Key

User: A student or researcher with basic technical knowledge.

Context and Activity: The user is using BotPDF to process documents in an academic research project. You have not configured the OpenAI API key correctly in the software settings.

Narrative: The user installs BotPDF. During setup, the user does not provide the required OpenAI API key, attempts to process a document, and receives error messages indicating that authentication with OpenAI has failed (Fig.10).

Diagnosis and Recovery: The user could have avoided the problem by reading the setup instructions and providing the necessary API key. To correct this, a key must be placed in the .env file contained in the project; Otherwise, you will not be able to use the software.

A screenshot of a terminal window with a black background and white text. The text displays an error message from the BotPDF application. It states that an error occurred while executing a transformation, specifically mentioning 'Transformation.interrogate' and 'DocumentInterrogator'. The core of the error is that the OpenAI function call failed due to an 'Incorrect API key provided'. The message includes a placeholder for the API key (a series of asterisks) and provides a URL to find the API key: https://platform.openai.com/account/api-keys.

```
Error : Error executing transformation (<Transformation.interrogate: 'DocumentInterrogator'>, {}): OpenAI function call failed: Incorrect API key provided: *****. You can find your API key at https://platform.openai.com/account/api-keys.
```

Fig.10 Error message indicating that authentication with OpenAI has failed

This scenario highlights the importance of following configuration instructions and providing the information necessary for the software to function properly, especially when API keys are required from external services such as OpenAI.

Scenario 2: User without Basic Technical Knowledge

User: A person without knowledge of technology.

Context and Activity: The user attempts to use BotPDF to process documents without prior software configuration experience or basic technology knowledge.

Narrative: The user downloads and installs BotPDF with the intention of processing documents for an academic assignment. When trying to open the software, you find a series of files and instructions that you would not be able to understand. Lack of technical knowledge prevents the user from performing the necessary configuration, and cannot progress in using the software.

Diagnosis and Recovery: Due to the lack of technical knowledge, the user is unable to perform the initial configuration or use the software functions. The only solution for this user would be to seek help from someone with minimal technical knowledge. Assistance from a technology-savvy person could guide the user through basic setup and use of the software.

In this scenario, the limitation lies in the user's lack of technical knowledge. The solution is to rely on the assistance of someone with minimal technical skills to overcome the initial barriers and use the software effectively.

7. Bibliography

- [Jason Fan \(2023\) - Improving vector search by converting documents to question/answer pairs](#)
- <https://medium.com/@murtuza753/using-llama-2-0-faiss-and-langchain-for-question-answering-on-your-own-data-682241488476>
- https://python.langchain.com/docs/get_started
- <https://github.com/facebookresearch/faiss/wiki>