

AI-POWERED PDF CHATBOT ASSISTANT

INTRODUCTION :

ABSTRACT :

This project presents an AI-powered chatbot assistant that allows users to upload a PDF document and ask questions related to its content. The chatbot uses Natural Language Processing (NLP) and Machine Learning (ML) techniques to analyse and extract relevant information from PDFs. It leverages Microsoft's Phi-2 model and Sentence Transformers for contextual understanding and similarity-based text retrieval.

OVERVIEW OF THE PROJECT :

The AI-Powered PDF Chatbot Assistant is a web-based application that allows users to upload PDF documents and ask questions related to their content. Instead of manually searching through large documents, the chatbot retrieves relevant information and provides AI-generated responses. The system uses Flask for the web API, pdfplumber for text extraction, Sentence Transformers for similarity-based retrieval, and Microsoft Phi-2 for generating accurate answers.

Key Components :

- 1. Document Processing** – Extracts text from PDFs using pdfplumber and splits it into smaller chunks for better retrieval.
- 2. Embedding & Retrieval** – Converts text and user queries into numerical embeddings using all-MiniLM-L6-v2, then finds the most relevant chunk based on similarity scores.
- 3. AI Response Generation** – Uses Phi-2 to generate context-aware answers based on the retrieved text.
- 4. Web API** – A Flask-based API manages document uploads and user interactions, allowing real-time responses.

PROBLEM DEFINITION :

The AI-Powered PDF Chatbot Assistant is designed to help users efficiently retrieve information from PDF documents by allowing them to upload a file and ask context-based questions. Instead of manually scanning through pages, the chatbot processes the document, extracts key information, and provides AI-generated answers. The system uses Natural Language Processing (NLP), Machine Learning (ML), and AI-driven response generation to enhance document comprehension and automate information retrieval.

Key Aspects of the Project :

- 1. AI-Powered Document Processing :** Extracts and processes text from PDFs using pdfplumber, ensuring efficient text retrieval.
- 2. Context-Based Information Retrieval :** Utilizes sentence-transformers to find the most relevant text chunks based on user queries.
- 3. NLP and AI-Driven Responses :** Leverages the Phi-2 model to generate accurate and context-aware answers.

- 4. Web-Based Interactive System :** Built with Flask, enabling users to upload documents and interact with the chatbot in real time.
- 5. Efficient and Scalable Architecture :** Processes large PDFs efficiently and retrieves information quickly using optimized embeddings.

OBJECTIVES OF THE PROJECT :

- ❖ Automate the process of reading and understanding PDFs.
- ❖ Enable users to ask queries related to uploaded documents and receive accurate responses.
- ❖ Improve information retrieval using NLP techniques.
- ❖ Develop an efficient and user-friendly web-based chatbot.

EXISTING SYSTEM :

The traditional method of retrieving information from PDF documents is manual search and keyword-based lookup, which has several inefficiencies. In organizations, research institutions, and educational settings, users often deal with lengthy and complex PDFs that require extensive time and effort to navigate.

Challenges in the Existing System :

1. Manual Search is Time-Consuming

- Users must manually scroll through pages to find relevant information.
- Finding specific details in large documents is inefficient.

2. Keyword-Based Searches Lack Context

- Simple search functions only find exact keyword matches but do not understand the context.
- Important details may be overlooked if different terminology is used.

3. No AI-Powered Assistance

- Existing systems do not provide intelligent, context-aware answers.
- Users need to interpret search results manually, leading to errors.

4. Inefficiency in Data Retrieval

- When multiple documents are involved, searching for relevant content becomes overwhelming.
- Users may have to compare multiple sections manually to extract meaningful insights.

PROPOSED SYSTEM :

To address these limitations, the AI-Powered PDF Chatbot Assistant introduces an intelligent, automated approach to document analysis and question-answering.

Key Features of the Proposed System :

1. Automated Document Processing

- Extracts text from PDFs using pdfplumber for structured and efficient text retrieval.
- Splits content into manageable chunks (500 characters with 100-character overlap) for better searchability.

2. Intelligent Information Retrieval

- Uses sentence-transformers to convert text into embeddings, allowing similarity-based matching.
- Finds the most relevant content based on semantic meaning, not just keywords.

3. AI-Powered Answer Generation

- Uses Microsoft Phi-2, an advanced language model, to generate human-like, contextually accurate responses.
- Ensures concise, relevant answers rather than returning entire paragraphs.

4. Web-Based User Interaction

- Provides an easy-to-use Flask-based API, enabling users to upload PDFs and ask questions interactively.
- Eliminates the need for manual searches, allowing real-time, AI-assisted responses.

5. Efficiency and Scalability

- Processes large documents quickly without compromising accuracy.
- Can be expanded to support multiple document formats and enhanced AI models in the future.

By implementing this system, users can save time, improve accuracy, and access information effortlessly without navigating entire documents manually.

MODULE DESCRIPTION :

The AI-Powered PDF Chatbot Assistant consists of several modules working together to process PDFs, retrieve relevant content, and generate AI-driven responses. Below is a breakdown of each module with its functionality and technologies used.

1. Document Processing Module

Purpose

- Extracts and processes text from uploaded PDF documents for efficient retrieval.

Functionality

- Accepts PDF file uploads and extracts text using pdfplumber.
- Splits text into overlapping chunks (500 characters, 100-character overlap) to improve search accuracy.
- Stores processed text for similarity-based retrieval.

Technologies Used

- pdfplumber – Extracts text from PDFs.
- CharacterTextSplitter – Splits text into structured chunks.
- OS module – Handles file operations.

2. Embedding and Similarity Matching Module

Purpose

- Converts text and queries into numerical embeddings to find the most relevant content.

Functionality

- Uses Sentence Transformers (all-MiniLM-L6-v2) to convert text chunks and queries into vector embeddings.
- Computes cosine similarity between query and text embeddings to identify the most relevant chunk.
- Passes the best-matched chunk as context for AI-generated responses.

Technologies Used

- Sentence Transformers – Converts text into numerical embeddings.
- NumPy – Performs similarity calculations.
- Scikit-Learn – Computes cosine similarity.

3. AI Response Generation Module

Purpose

- Generates human-like answers using AI based on the retrieved text and user query.
- Functionality
- Uses Microsoft Phi-2, a pre-trained AI model, to generate responses.
- Structures input prompts to ensure relevant and concise answers.
- Limits output to 100 tokens, ensuring clear and direct responses.

Technologies Used

- Transformers (AutoTokenizer, AutoModelForCausalLM) – Loads and runs the Phi-2 model.
- Torch – Optimizes processing with GPU/CPU support.
- Hugging Face Pipeline – Manages AI-based text generation.

4. Web API Module

Purpose

- Provides an interface for users to interact with the chatbot through a web application.

Functionality

- Flask-based API manages file uploads and user queries.
- Handles requests to process PDFs (/process-document) and generate responses (/process-message).
- Supports real-time chatbot interactions via a web-based UI.

Technologies Used

- Flask – Web framework for handling API requests.
- Flask-CORS – Manages cross-origin requests for web interaction.
- UUID – Generates unique filenames for uploaded PDFs.

This modular structure ensures efficient document processing, intelligent retrieval, AI-driven responses, and seamless user interaction.

SOFTWARE USED :

- ❖ Python
- ❖ Flask (for API and web service)
- ❖ PyTorch (for AI model execution)
- ❖ pdfplumber (for PDF text extraction)

LIBRARIES USED :

- ❖ torch: For deep learning model execution.
- ❖ pdfplumber: Extracts text from PDFs.
- ❖ transformers: Provides pre-trained language models.
- ❖ sentence-transformers: Computes text embeddings for similarity comparisons.
- ❖ sklearn: Used for cosine similarity calculations.
- ❖ numpy: Numerical computations for embeddings and similarities.
- ❖ flask: Web framework for API implementation.

CONFIGURATION :

- ❖ **HARDWARE CONFIGURATION**
- ❖ **SOFTWARE CONFIGURATION**

HARDWARE CONFIGURATION :

Hardware Requirements

- **Processor:** Intel Core i5 or higher
- **RAM:** Minimum 8GB (16GB recommended for faster processing)
- **Storage:** Minimum 10GB free space
- **GPU** (optional but recommended for faster model inference)

SOFTWARE CONFIGURATION :

- **Operating System:** Windows/Linux/macOS
- **Python** 3.8 or higher
- Required **Python libraries** (installed via `pip install -r requirements.txt`)

SOFTWARE SPECIFICATION :

The AI-Powered PDF Chatbot Assistant is built using a combination of AI, NLP, and web technologies to ensure efficient document processing and interactive responses. Below are the key software specifications.

PROGRAMMING LANGUAGE

- **Python 3.x** – The entire project is developed in Python due to its rich ecosystem of AI, NLP, and web development libraries.

FRAMEWORKS AND LIBRARIES

1. AI and NLP Libraries

- **Transformers** – Loads and runs the Phi-2 model for generating responses.
- **Sentence Transformers** – Converts text into embeddings for similarity-based retrieval.
- **Torch** – Provides GPU/CPU optimization for AI model execution.

2. Text Processing Libraries

- **pdfplumber** – Extracts text from PDFs.
- **LangChain CharacterTextSplitter** – Splits text into chunks for better retrieval.
- **NumPy & Scikit-Learn** – Computes similarity between user queries and document chunks.

3. Web Frameworks

- **Flask** – Handles API requests for document uploads and chatbot responses.
- **Flask-CORS** – Manages cross-origin requests for web interaction.

DEVELOPMENT AND EXECUTION ENVIRONMENT

- **Operating System** – Compatible with Windows, macOS, and Linux.
- **Development Tools** – Python IDEs such as VS Code, PyCharm, or Jupyter Notebook.
- **Package Manager** – Uses pip to install required dependencies.

MODEL AND API INTEGRATION

- **Microsoft Phi-2** – Pre-trained AI model for NLP-based response generation.
- **Hugging Face Pipeline** – Streamlines AI inference.

WEB API AND USER INTERACTION

- **Flask API Endpoints** – Handles document uploads (/process-document) and user queries (/process-message).
- **JSON Format** – Used for data exchange between frontend and backend.

This software stack ensures efficient processing, accurate information retrieval, and seamless AI-powered interactions for the chatbot.

LIBRARIES :

The AI-Powered PDF Chatbot Assistant utilizes various Python libraries for document processing, embedding generation, AI response handling, and web interaction. Below is a detailed explanation of the key libraries used in the project.

1. AI and Natural Language Processing (NLP) Libraries

Transformers

- Provided by Hugging Face, it is used to load and run the Microsoft Phi-2 model, which generates AI-driven responses.
- It includes AutoTokenizer and AutoModelForCausalLM to process text inputs and produce meaningful answers.

Sentence Transformers

- Used for embedding generation and semantic similarity matching.
- Converts both document chunks and user queries into numerical vectors, allowing the system to retrieve the most relevant text.

Torch

- A deep learning library used to efficiently run the AI model on CPU/GPU.
- Provides optimized execution for faster response times.

2. Text Processing and Document Handling

pdfplumber

- Extracts text from PDF documents while preserving structure.
- Enables multi-page document handling for accurate content extraction.

LangChain CharacterTextSplitter

- Splits long documents into smaller overlapping chunks (500 characters with 100-character overlap).
- Helps in improving retrieval accuracy when matching user queries.

NumPy

- Used for numerical operations in the similarity matching process.
- Helps in storing and manipulating embedding vectors efficiently.

Scikit-Learn

- Provides the cosine similarity function to measure how closely a text chunk matches a user query.
- Helps in retrieving the most relevant document section before AI response generation.

3. Web Development and API Handling

Flask

- A lightweight web framework used to build the backend API for handling user interactions.
- Manages requests such as file uploads (/process-document) and chat queries (/process-message).

Flask-CORS

- Enables Cross-Origin Resource Sharing (CORS), allowing web-based frontends to communicate with the Flask backend.

UUID

- Generates unique filenames for uploaded PDFs, preventing filename conflicts.

These libraries work together to create a fast, efficient, and intelligent chatbot that can extract, process, and answer queries from PDF documents.

SYSTEM IMPLEMENTATION :

IMPLEMENTATION DETAILS :

Worker.py:

- Loads the Phi-2 model and tokenizer.
- Processes PDFs using pdfplumber and splits text into chunks.
- Computes embeddings and retrieves the most relevant text section.
- Generates responses using the Phi-2 model.

Server.py:

- Provides API endpoints for document upload and query processing.
- Routes requests and integrates with worker.py.
- Handles user queries and returns chatbot responses.

PROGRAM CODE AND SCREENSHOTS OF AI-POWERED PDF CHATBOT ASSISTANT :

PROGRAM CODE :

```
import os
import torch
import pdfplumber
import numpy as np
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
from langchain.text_splitter import CharacterTextSplitter
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity

# Initialize global variables
qa_pipeline = None
embeddings_model = None
text_chunks = []
tokenizer = None

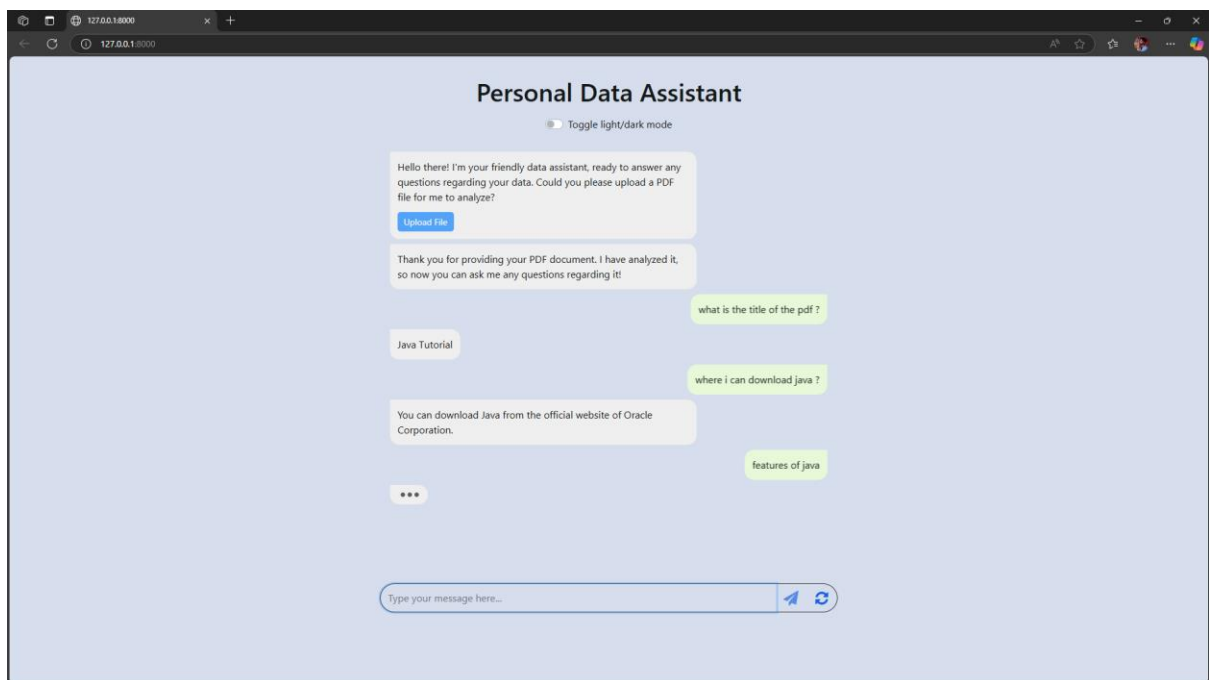
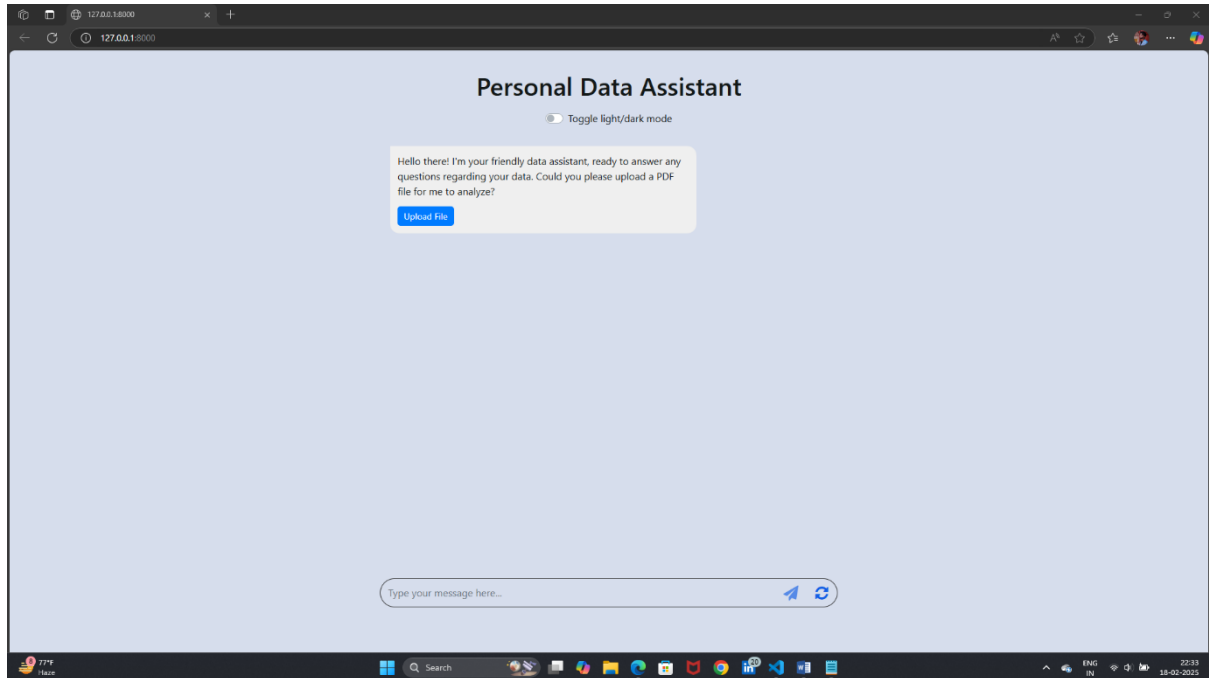
# Initialize the optimized model (Phi-2)
def init_llm():
    global qa_pipeline, embeddings_model, tokenizer

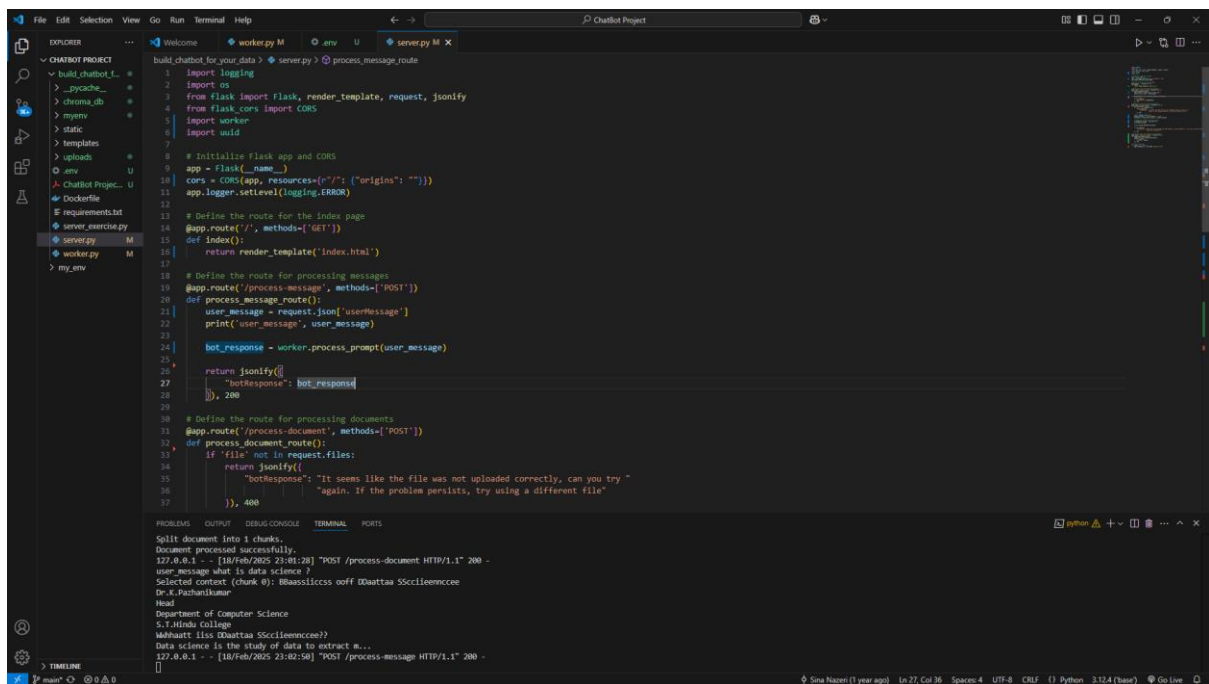
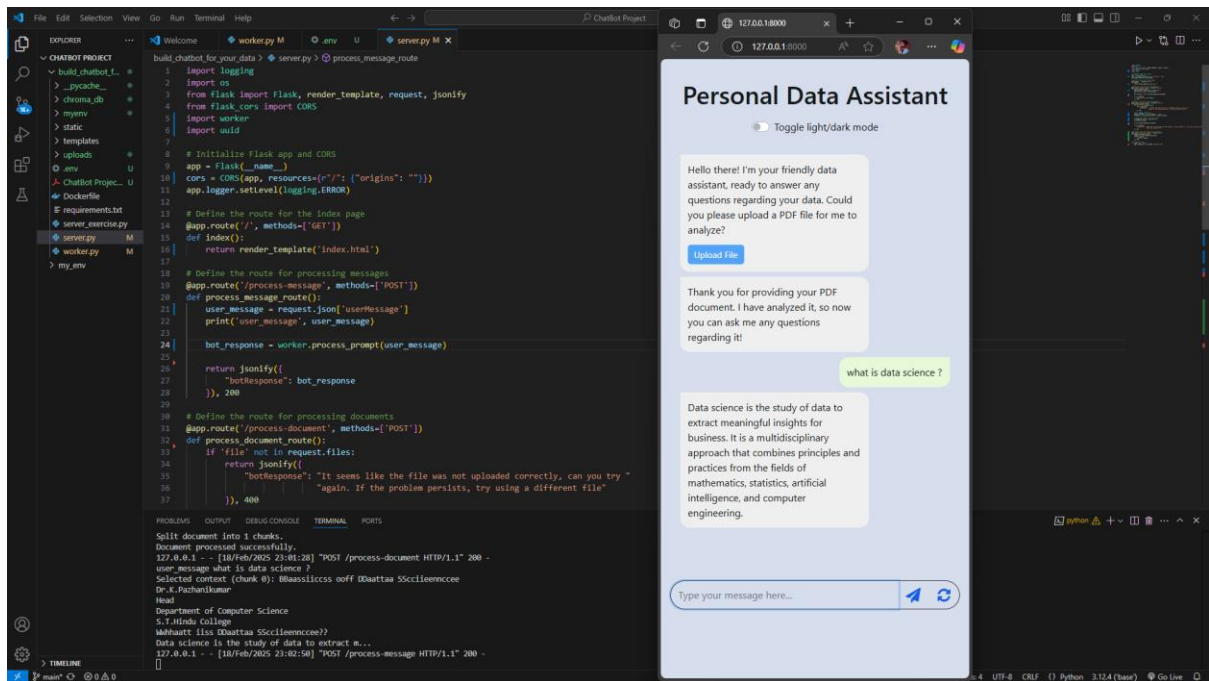
    model_name = "microsoft/phi-2" # Open-source model

    tokenizer = AutoTokenizer.from_pretrained(model_name)

    model = AutoModelForCausalLM.from_pretrained(
        model_name,
        torch_dtype=torch.float16,
        device_map="auto"
    )
```

SCREENSHOTS :









CONCLUSION AND FURTHER ENHANCEMENTS :

CONCLUSION :

The AI-powered PDF chatbot assistant automates document understanding and query resolution, making information retrieval faster and more efficient. By leveraging NLP and deep learning models, the system provides contextually relevant answers to user questions.

SCOPE FOR FURTHER DEVELOPMENT :

-  **Multi-Document Processing:** Enhance the chatbot to analyze multiple PDFs simultaneously.
-  **Improved Response Generation:** Fine-tune the AI model for better accuracy.
-  **Integration with Speech Recognition:** Allow users to ask questions via voice input.
-  **Mobile Application:** Develop a mobile-friendly version of the chatbot.

BIBLIOGRAPHY :

Flask - <https://flask.palletsprojects.com/>

Transformers Library - <https://huggingface.co/docs/transformers/>

Sentence Transformers - <https://www.sbert.net/index.html/>

Microsoft Phi-2 Model - <https://huggingface.co/microsoft/phi-2>

Pdfplumber Documentation - <https://github.com/jsvine/pdfplumber>