



ChatWithPdf

Interactive PDF Document Analysis and Question Answering System

06-10-2023

Ritesh Tambe

CSMSS Chh Shahu College of Engineering

Artificial Intelligence and Data Science

Aurangabad

Overview

Project Overview: Interactive PDF Document Analysis and Question Answering System

Objective:

The objective of this project is to create a user-friendly system that allows users to interact with a PDF document, extract relevant information from it, and obtain answers to their questions based on the content of the PDF.

Goals

Efficient Information Retrieval:

The project aims to create a tool that allows users to quickly find specific information within PDF documents, which can be challenging when dealing with lengthy or complex documents.

Context-Aware Responses:

When users ask questions, the system should provide answers that are relevant to the content of the PDF. This means the responses should fit within the context of the document.

User-Friendly Interaction:

The system is designed to offer a user-friendly experience, making it accessible to a wider range of users. It uses a conversational approach, making it feel like you're having a natural conversation with an assistant.

Advanced Document Analysis:

The system incorporates natural language processing techniques to understand the content of PDF documents. This enables it to extract meaningful insights and facts from the documents, making it easier to answer questions accurately.

Support for Education and Research:

The project serves as a valuable tool for students, researchers, and academics. It assists in quickly accessing relevant information from academic papers, research reports, and other similar documents.

Specifications

Input Handling:

Accept PDF documents as input for analysis.

Support both user-provided PDFs and a library of preloaded PDFs for testing.

Validate input PDFs to ensure they are readable and in a supported format.

Text Extraction:

Utilize the PyPDF2 library to extract text from PDF documents.

Handle various types of PDF content, including text, images, tables, and annotations.

User Interaction:

Provide a user-friendly interface for users to input questions or queries related to the content of the PDF.

Enable users to interact with the system through text-based or voice-based interfaces, as appropriate.

Semantic Similarity Calculation:

Calculate semantic similarity between user queries and sentences or sections of the PDF.

Use cosine similarity or other suitable metrics to measure similarity.

Context-Aware Responses:

Generate responses that are contextually relevant to the content of the PDF document.

Ensure that answers are coherent within the context provided by the document.

Accuracy and Relevance:

Ensure that the system's responses are accurate and directly address user queries.

Implement mechanisms to validate the accuracy of answers, potentially using external knowledge sources.

Technologies and Programming Used

Programming Language – Python

PDF Document Handling (PyPDF2):

PyPDF2 is a Python library used for handling PDF documents.

In the code, it is used to open a PDF file, extract text from its pages, and store the extracted text in a list.

Text Extraction from PDFs:

The code extracts text from each page of the PDF document and collects it for further processing.

Sentence Embeddings (SentenceTransformer):

SentenceTransformer is a Python library for generating sentence embeddings.

It uses pre-trained models to convert sentences into numerical representations that capture their semantic meaning.

In the code, it loads a pre-trained sentence embedding model ('paraphrase-MiniLM-L6-v2').

Cosine Similarity Calculation (PyTorch and Hugging Face Transformers):

The code uses PyTorch, a deep learning framework, to perform vector operations.

It calculates the cosine similarity between user input embeddings and embeddings of sentences from the PDF document.

Hugging Face Transformers library (util.pytorch_cos_sim) is used to compute the cosine similarity.

User Interaction (Conversational AI - PALM):

PALM is a conversational AI framework or platform that facilitates natural language interactions between users and AI systems.


In the code, PALM is used to create a conversational experience for users, allowing them to ask questions and receive responses in a chat-like format.

Context Management:

The code maintains context by storing user messages and conversation history in the 'messages' list.

It keeps track of the ongoing conversation and reference context.

AI Configuration (palm.configure):



The code configures the AI system with specific settings, including the choice of AI model and parameters.

An API key may be provided to authenticate and access the AI service.

Contextual Examples (Examples for AI Training):

The code defines contextual examples that help guide the AI in understanding and responding to user queries within a specific context.

Examples include user inputs and corresponding AI responses.

Text Preprocessing (String Replacement):

A custom 'replace' function is defined to remove unwanted characters (e.g., newline, carriage return, spaces) from the AI response.

This preprocessing step ensures cleaner and more readable responses.

User Query Processing and Response Generation (lang_model):

The 'lang_model' function processes user queries, appends them to the conversation history, and sends them to the AI for response.

It retrieves the AI's response, preprocesses it, and returns the final response to the user.