Documentation for QA from PDF using Large Language Models

Welcome to the documentation for the Question-Answering (QA) application using Large Language Models (LLMs), such as OpenAI. This documentation aims to provide a comprehensive overview of the QA application, detailing how to use it, what each component does, and how the application overall works. Let's get started!

Table of Contents

Overview

This Python application enables users to select a document and ask questions based on the content of that document. The application uses Large Language Models (LLMs), such as OpenAI, to answer these questions intelligently and accurately.

Installation & Setup

To get started with this application, you'll need to have Python installed, along with a few additional dependencies. You'll also need to have a directory of PDF documents that the application can process.

Usage

The application is built with a user-friendly web interface powered by Streamlit.

Interface

Upon launching the application, you're greeted with a title and an overview of what Large Language Models (LLMs) can do. This intro sets the stage for the use of LLMs in improving documentation quality and the overall user experience.

Selecting a Document

To select a document, look for the dropdown box labeled "Please select the article:". The dropdown is populated with the names of available documents. The selected document's name is stored and used to construct the full path to the PDF file.

Question Prompting

Once a document is selected, you're prompted to ask a question about the content of that document. The question is entered in a text box labeled "Ask your question here".

Answering

Once a question is asked, the application performs a similarity search with the question as the query. The results of the similarity search are used by the LLM to generate an answer to the question. If the LLM cannot answer the question, it will respond with "I need more information or the requested information is not available to me!". Otherwise, the LLM will provide an answer based on the content of the selected document.

Technical Details

The QA application is powered by several libraries and dependencies.

Dependencies

These include:

- streamlit for the web interface.
- PyPDF2 to read the PDF files.
- pdf2image to convert PDF pages to images.
- dotenv to load environment variables.
- langchain to leverage language models and vector search services.

Langchain Loading

The application uses the load_qa_chain function from the langchain library to load a QA chain of the 'stuff' type. This chain uses the OpenAI() class object to build the model.

PDF Processing

The script reads the PDF file using the PdfReader from the PyPDF2 library. It iterates through each page of the PDF, extracting the text and appending it to a raw_text variable. After the entire document has been read, this raw_text is split into chunks to prevent the text from being too large to process.

Text Embedding

The split text is then embedded using the OpenAIEmbeddings class. These embeddings are used by FAISS to create a vector space representation of the text for efficient similarity searching.

LLM Creativity

An instance of OpenAI is created with a temperature of 0.5. This parameter is used to control the randomness of the output from the LLM.

Conclusion

This application provides a straightforward and efficient way to ask questions about the content of PDF documents and get answers generated by Large Language Models (LLMs). It's a versatile tool that can be used across a wide range of scenarios, from reading technical documents to educational books.