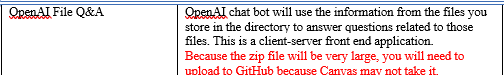
Girija Rijal

Part 1



YouTube link for Part 1<https://youtu.be/OHsM4i9clhk>

Github link <https://github.com/grijal776/Client-server-chatbot>



YouTube link for Part 2 <https://youtu.be/Y72qROpCKhk>

Github link <https://github.com/grijal776/Local-file-based-Q-and-A-chatbot>

Part 1

Technical Document for File Question and Answer App

Introduction

The File Question and Answer App is a web application that allows users to upload files and search for specific content within them. The application uses OpenAI's GPT-3.5 language model to generate answers to user queries related to the uploaded files.

Architecture

The application consists of a Flask server and a Next.js client. The server is responsible for handling user requests, processing the uploaded files, and generating answers to user queries. The client provides a user interface for uploading files and entering search queries.

The server uses Flask, a lightweight Python web framework, to handle incoming requests. When a user uploads a file, the server reads the contents of the file and sends it to Pinecone, a similarity search service, to create an index. The index is then used to generate answers to user queries.

The server uses OpenAI's GPT-3.5 language model to generate answers to user queries. When a user enters a search query, the server sends the query to the language model, which generates a response based on the content of the uploaded files.

The client is built using Next.js, a React-based framework for building web applications. The client provides a user interface for uploading files and entering search queries. When a user uploads a file, the client sends the file to the server using HTTP POST. When a user enters a search query, the client sends the query to the server using HTTP GET and displays the response.

Prerequisites

Before setting up and running the application, make sure you have the following:

Python 3 (version 7 to version 10) installed on your computer

An OpenAI API key

A Pinecone API key and index name

Node.js and npm installed on your computer

Installation

To install and run the application, follow these steps:

Download the source code from the GitHub repository: https://github.com/openai/openai-cookbook

Unzip the downloaded file to a directory on your computer.

Open the server directory in a PowerShell window and fill out the config.yaml file with your Pinecone API key, index name, and environment.

Install the required Python and Node.js dependencies using the following commands:

pip install openai

npm install openai

Create a virtual environment for the server by running the following commands:

python -m venv venv

.\venv\Scripts\activate

pip install -r .\requirements.txt

pip install python-dotenv

Start the Flask server by running the following command:

python .\app.py

Open a new PowerShell window and navigate to the client directory.

Install the required Node.js dependencies by running the following command:

npm install

Start the Next.js client by running the following command:

npm run dev

Open a web browser and navigate to http://localhost:3000 to access the application.

Usage

To use the application, follow these steps:

Upload a file using the "Choose File" button.

Once the file is uploaded, enter a question related to the content of the file in the search bar.

Click on the "Search" button to generate an answer to your question.

Conclusion

The File Question and Answer App is a web application that allows users to upload files and search for specific content within them. The application uses OpenAI's GPT-3.5 language model to generate answers to user queries related to the files.

Part 2

Objective

The objective of this Python script is to crawl a website and retrieve the text from each page. The text is then processed and tokenized, and the resulting tokens are used to create embeddings for each page. These embeddings are then used to create a context for answering questions about the website.

Dependencies

This script relies on several Python libraries, including:

requests for making HTTP requests

BeautifulSoup for parsing HTML

pandas for data manipulation and storage

numpy for numerical operations

openai for language processing and embeddings

tiktoken for tokenization

dotenv for loading environment variables

Workflow

Import the necessary libraries

Define a class to parse HTML and retrieve hyperlinks

Define a function to get hyperlinks from a given URL

Define a function to crawl a website, save the text of each page, and add any new URLs to the queue for crawling

Define a function to remove newlines from text data

Retrieve the text from each page on the website and save it to a Pandas DataFrame

Tokenize the text and save the number of tokens to a new column in the DataFrame

Define a function to split the text into chunks of a maximum number of tokens

Apply the split\_into\_many function to the DataFrame to create a new DataFrame with shorter texts

Create embeddings for each text in the new DataFrame using OpenAI's text-embedding-ada-002 model

Load the embeddings into a new DataFrame

Define a function to create a context for a given question using the most similar text embeddings from the DataFrame, and another function to use this context to answer a question

Test the question-answering functionality with several example questions

Limitations

The script does not account for websites with dynamic content, such as those that use JavaScript to load data

The embeddings may not be accurate enough for more complex questions, as they are generated using a pre-trained language model rather than a model trained specifically on the website's content

The question-answering functionality relies on OpenAI's language processing API, which is subject to rate limits and may require an API key to access.