GPT Research Assistant
User Documentation

Created by Alex Clayton, AJ Waizmann, Henry Hebner, and Quentin Sharier.

April 16, 2023
Version 0.5

Table of Contents

Introduction

This project is a chatbot that uses GPT-3 to answer questions based on a repository of research documents. This document will explain how to set up the project on a local machine, how the code works, where specific functions are written, and how to fix some bugs we have encountered while developing.

User's Guide

How to use the chatbot as a user of the product.

Feature 1: Asking the bot a question

To ask the bot a question, simply type in your question in the big text box after navigating to localhost:5000 (or the server address) and press submit. The website will start processing the question and will find relevant sources to pull information from. Our bot determines what is close to the prompt by multiplying vectors that make up the question with every vector that makes up all the documents. This is a lengthy process but is pretty accurate. Please refer to the troubleshooting guide when there are bugs.

Feature 2: Document parsing

To add new documents to the index, go to the google drive and upload pdfs into the pdf folder. The next time the website receives a request for the documents it will get converted into an index and added to the document pool.

Developer Guide

Tech Stack

This application is built in Python and uses Flask to make the web server. The front end is simple HTML and CSS, and the "database" we are using to store the files is Google Drive. The server will receive an HTML form from the user, and will take that and input it to GPT-3 along with the 20 closest vectors from the documents. The output of these queries will be shown on the homepage. GPT will summarize all of the outputs together, but all of the individual outputs are shown below that. Before the requests for the summaries are made, the indexes of the documents are checked to see if the folder is up to date. This program does keep logs of everything that is sent to GPT.

Setup

Three things must be present in order for this project to work. These are:
1. Prerequisite Libraries
2. Google drive credentials file
3. .env file

Prerequisite Libraries

The initial requirement for this project is to pull the github repository. Once you have the repository downloaded onto your local system, install the prerequisite libraries.

- This can be done by running the command "**pip install -r requirements.txt**". This will recursively install every requirement for the project.

Google Drive Credentials File

Currently, every research paper supported by this program is hosted within a Google Drive folder. This folder is accessed via a library which scans the entire folder to see if there is a new PDF file which does not have a .txt file equivalent yet. To run the program successfully, you must give it access to this google drive account.

- This can be done by putting the "credentials.json" file into the root file directory of "\GPT-VectorSimilarity".
- Follow the directions on this link to get your credential.json file. https://developers.google.com/drive/api/quickstart/python

.Env File

The program works by accessing your OpenAI API key which is stored in a .env file. This file must be created by the user and must contain the following line of code:

- APIKEY="enter-your-api-key"

Place this .env file within the root file directory of "\GPT-VectorSimilarity".

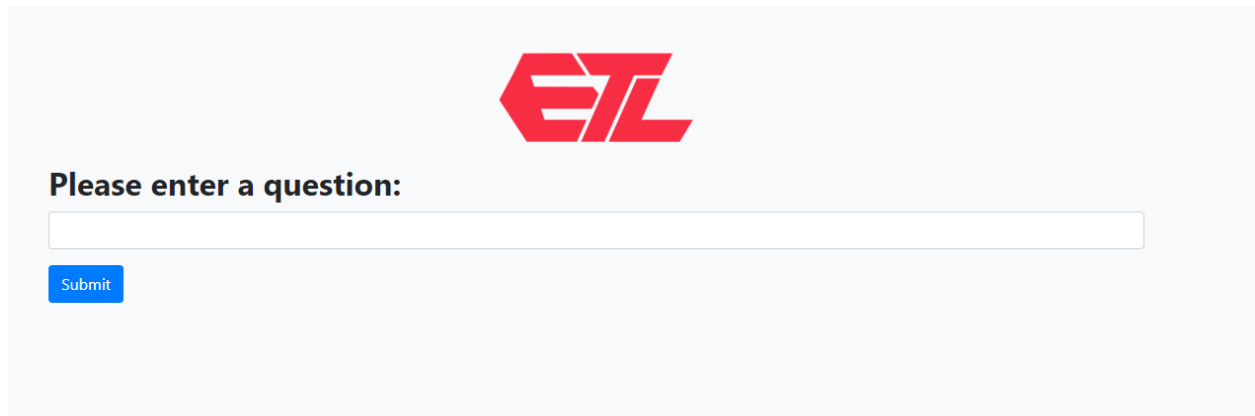Once these three things have been done, you will be able to proceed with the rest of the project.

In order to get the API key from OpenAI, log into OpenAI and navigate to the profile. From there find view API keys. Be careful as this program does use a lot of tokens when running a single request. Do not forget to set a payment limit on the API. As of writing there is no way to pay for infinite requests and it is a pay as you go system.

Front-end

To run the program, type "py -3 website.py". This will start the webserver.

`\GPT-VectorSimilarity>py -3 website.py`

Then, open your web browser and navigate to **"localhost:5000".**

From there, you can enter your question into the textbox and click "submit". This will update the home webpage with the answers displayed and links to the documents GPT references. This may take a minute.

Git

Here is the link to our github with all our code. We recommend forking this repo and making your own. https://github.com/sharier2/GPT-VectorSimilarity

Troubleshooting Guide

When running website.py waiting for GPT to respond.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    COMMENTS

127.0.0.1 - - [12/Apr/2023 20:54:20] "POST / HTTP/1.1" 500 -
127.0.0.1 - - [12/Apr/2023 20:54:20] "POST / HTTP/1.1" 500 -
Traceback (most recent call last):
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask\app.py", line 2551, in __call__
    return self.wsgi_app(environ, start_response)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask\app.py", line 2531, in wsgi_app
    response = self.handle_exception(e)
               ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask\app.py", line 2528, in wsgi_app
    response = self.full_dispatch_request()
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask\app.py", line 1825, in full_dispatch_request
    rv = self.handle_user_exception(e)
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask\app.py", line 1823, in full_dispatch_request
    rv = self.dispatch_request()
         ^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask\app.py", line 1799, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "e:\CSE\GPT-Vector\GPT-VectorSimilarity\website.py", line 16, in result
    summary, answers = GPTfromWebsite(text)
                       ^^^^^^^^^^^^^^^^^^^^^
  File "e:\CSE\GPT-Vector\GPT-VectorSimilarity\website.py", line 22, in GPTfromWebsite
    return answer_questions.queryGPT(text)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "e:\CSE\GPT-Vector\GPT-VectorSimilarity\answer_questions.py", line 81, in queryGPT
    update_google_drive_folders()
  File "e:\CSE\GPT-Vector\GPT-VectorSimilarity\pdf_to_txt_to_index.py", line 105, in update_google_drive_folders
    pdf_files = get_files_from_drive_folder(PDF_FOLDER_ID)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "e:\CSE\GPT-Vector\GPT-VectorSimilarity\pdf_to_txt_to_index.py", line 97, in get_files_from_drive_folder
    results = SERVICE.files().list(q=query, fields="nextPageToken, files(id, name, mimeType)").execute()
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\googleapiclient\_helpers.py", line 130, in positional_wrapper
    return wrapped(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\googleapiclient\http.py", line 923, in execute
    resp, content = _retry_request(
                    ^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\googleapiclient\http.py", line 222, in _retry_request
```

```
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\googleapiclient\http.py", line 222, in _retry_request
    raise exception
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\googleapiclient\http.py", line 191, in _retry_request
    resp, content = http.request(uri, method, *args, **kwargs)
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\google_auth_httplib2.py", line 218, in request
    response, content = self.http.request(
                        ^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\httplib2\__init__.py", line 1720, in request
    (response, content) = self._request(
                          ^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\httplib2\__init__.py", line 1440, in _request
    (response, content) = self._conn_request(conn, request_uri, method, body, headers)
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\site-packages\httplib2\__init__.py", line 1392, in _conn_request
    response = conn.getresponse()
               ^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 1374, in getresponse
    response.begin()
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 318, in begin
    version, status, reason = self._read_status()
                              ^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 279, in _read_status
    line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\socket.py", line 706, in readinto
    return self._sock.recv_into(b)
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\ssl.py", line 1278, in recv_into
    return self.read(nbytes, buffer)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\user4\AppData\Local\Programs\Python\Python311\Lib\ssl.py", line 1134, in read
    return self._sslobj.read(len, buffer)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ssl.SSLError: [SSL: WRONG_VERSION_NUMBER] wrong version number (_ssl.c:2546)
127.0.0.1 - - [12/Apr/2023 20:54:20] "GET /?__debugger__=yes&cmd=resource&f=style.css HTTP/1.1" 200 -
127.0.0.1 - - [12/Apr/2023 20:54:20] "GET /?__debugger__=yes&cmd=resource&f=debugger.js HTTP/1.1" 200 -
127.0.0.1 - - [12/Apr/2023 20:54:20] "GET /?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 200 -
127.0.0.1 - - [12/Apr/2023 20:54:20] "GET /?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
```

Fix: Restart the server?

GPT3 error for too many tokens used. Reduce question length or change the query file. The process we are using does not allow for a large question.

Summary:

GPT3 error: This model's maximum context length is 4097 tokens, however you requested 4117 tokens (2117 in your prompt; 2000 for the completion). Please reduce your prompt; or completion length.

Deployment Guide

Run website.py and if there are no errors it should give back a response when prompted. The response is slow.

Project Plan

Setup Trello (2/2/23-2/2/23)
● Add initial tasks to Trello
● Make sure everyone has permissions

Sprint 0 (2/2/23-2/9/23)
● Excel sheet of use case and measurable outcomes
● Design 20 use cases in Cucumber format
● Email Brad 10 use cases by Monday @12pm EST
● Design initial system architecture and email to Brad by EOD Sunday
● Email Brad 1 page reports on initial use cases

Sprint 1 (2/9/23-2/16/23)
- Explore and document Zapier possibilities / Limitations (Feb 15)
- Develop Presentation for in class Tuesday 2/14
- Further whiteboard architecture

Sprint 2 (2/16/23-2/23/23)
- Whiteboard architecture
- Get access to backend systems (Stalled+dropped as a requirement)

Sprint 3 (2/24/23-3/23/23)
- Build presentation for Tuesday 3/7
- Unify everyone under one GPT account + get credit card so we can pay for credits
- Combine all sources into single document (3/21 created, 3/23 finished)
- Research PDF parsing technology and develop POC
- Moscow Priorities
- Train GPT model based on multiple pdfs

Sprint 4 (3/23/23-4/06/23)
- Train GPT model based on multiple pdfs (continued from last)
- Chose between vector similarity and generating questions using GPT
- Systems documentation initial drafts
- Develop testing procedures for the code

Sprint 5 (4/06/23-4/20/23)
- Create front end
- Put project on server
- Improve UI and results from GPT
- Test cases for quality