Customer Support System

Using ChatGPT and openAI API.

Prepared by: Kruti Dhyani

Table of content

1. Introduction
2. Design
3. Implementation
4. Test
5. Enhancement Ideas
6. Conclusion
7. Bibliography / References

Introduction

- To satisfy need of the customer using Leveraging Artificial Intelligence for enhanced support the Application is developed
- Used openAI API_key and Org_ID to use openAI API. For that account is created and set the key and org_id then credit the balance to use its API
- The Objective is develop web based and command based application for website related queries. If Information is mentioned in that website It prints the ans otherwise prints Don’t know
- Key Technology: ChatGPT by OpenAI, python Flask and Node.js.

Project Phases

Three phases

1. Phase 1: Data Collation-Gathering information from various sources

   Data can be in any form: Webpages, Local Files, Database, Videos, Drives

   For this first project Web Pages are used

2. Phase 2: Training-preparing ChatGPT project for effective response

   Training can be done using files, API and Fine-tuning

   For this first project ChatGPT API are used

3. Phase 3: User Interface - output method.

   Text message or Speech. For this project **Text message is used**

Design

To achieve fully we have to do the following in sequence. This steps are used for creating NLP model that collect data and model training for large dataset or LLM

1. **Crawler [Data collection]:** In this step, First, the given website or page link is passed in as domain and then it crawl the website using this steps. Generate .csv file which has all the website crawling information and stored with name “Scraped.csv”
2. **Embedding [Model Training]:**Once you have gathered a substantial dataset, you can proceed to the embedding phase, which involves training an NLP model like GPT (Generative Pre-trained Transformer) on this data. The model learns to understand the patterns, relationships, and semantics present in the text data through a process called "embedding." Here “embeddings.csv” file is generated at this step.
3. **Testing:** output gives by combining above 2 methods using Question - Answers model, either by command line argument or GUI based method.

Implementation Error and Solution

1. Do not require to create a new virtual environment every time. If you created venv successfully in homework-1 then Directly write command

workon <your venv name > For Example, workon chatgpt

2. Just run your crawler.py, embedding.py and app.py file like normal executing python file

```
python3 crawler.py
```

3. It gives suggestions to install supported packages. and installed it using

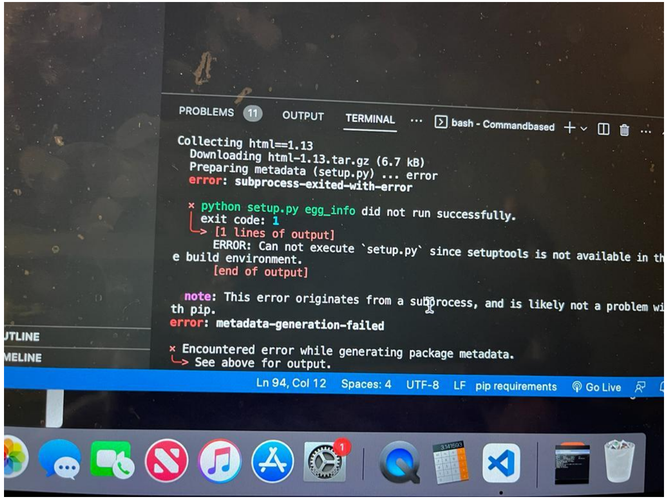
```
pip3 install <suggested package_name>.
```

Repeat step 2 & 3 until it runs successfully and try for embedding.py as well

Cont..

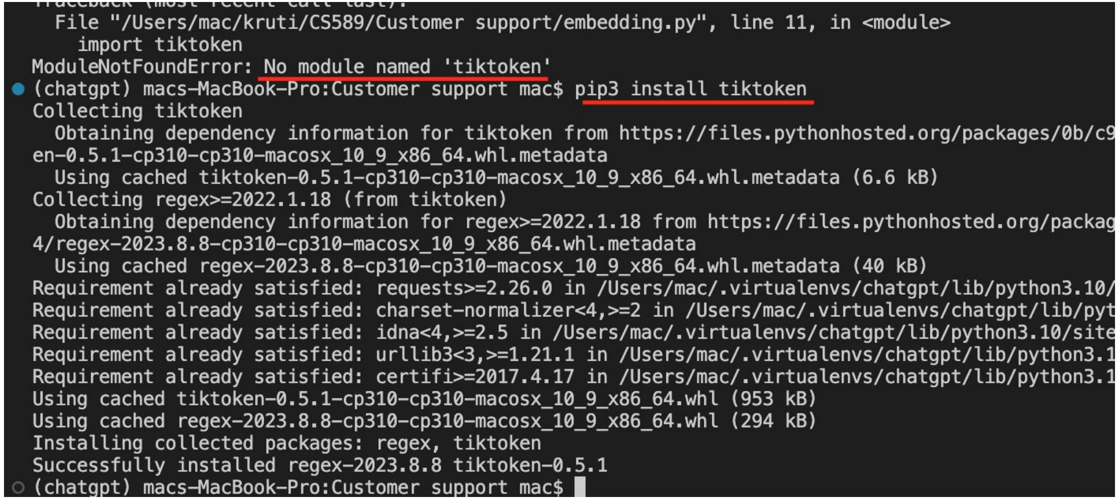
Screenshot Error and Solution is attached

Error screenshot



```
Collecting html==1.13
  Downloading html-1.13.tar.gz (6.7 kB)
  Preparing metadata (setup.py) ... error
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
  │ exit code: 1
  ╰─> [1 lines of output]
      ERROR: Can not execute `setup.py` since setuptools is not available in th
e build environment.
      [end of output]

  note: This error originates from a subprocess, and is likely not a problem wi
th pip.
error: metadata-generation-failed

× Encountered error while generating package metadata.
╰─> See above for output.
```

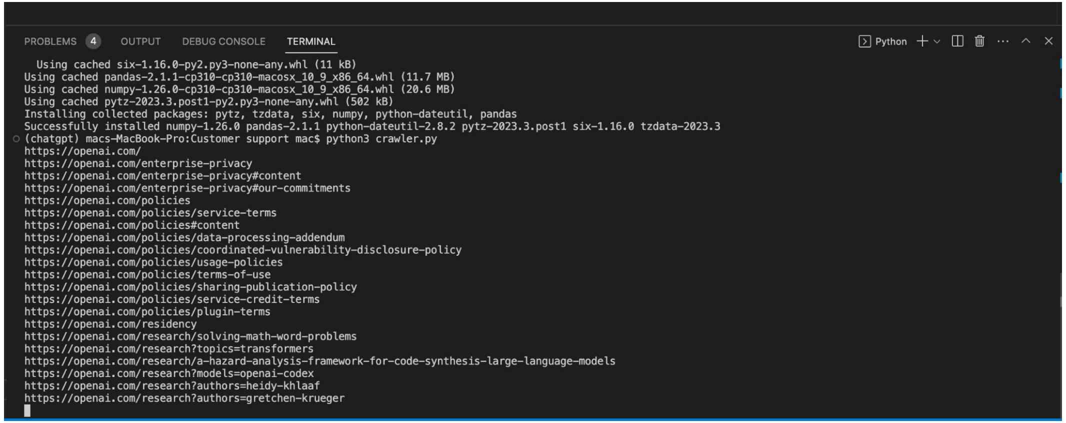
Solution Screenshot



```
Traceback (most recent call last):
  File "/Users/mac/kruti/CS589/Customer support/embedding.py", line 11, in <module>
    import tiktoken
ModuleNotFoundError: No module named 'tiktoken'
(chatgpt) macs-MacBook-Pro:Customer support mac$ pip3 install tiktoken
Collecting tiktoken
  Obtaining dependency information for tiktoken from https://files.pythonhosted.org/packages/0b/c9
en-0.5.1-cp310-cp310-macosx_10_9_x86_64.whl.metadata
  Using cached tiktoken-0.5.1-cp310-cp310-macosx_10_9_x86_64.whl.metadata (6.6 kB)
Collecting regex>=2022.1.18 (from tiktoken)
  Obtaining dependency information for regex>=2022.1.18 from https://files.pythonhosted.org/packag
4/regex-2023.8.8-cp310-cp310-macosx_10_9_x86_64.whl.metadata
  Using cached regex-2023.8.8-cp310-cp310-macosx_10_9_x86_64.whl.metadata (40 kB)
Requirement already satisfied: requests>=2.26.0 in /Users/mac/.virtualenvs/chatgpt/lib/python3.10/
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/mac/.virtualenvs/chatgpt/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /Users/mac/.virtualenvs/chatgpt/lib/python3.10/site
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/mac/.virtualenvs/chatgpt/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /Users/mac/.virtualenvs/chatgpt/lib/python3.1
Using cached tiktoken-0.5.1-cp310-cp310-macosx_10_9_x86_64.whl (953 kB)
Using cached regex-2023.8.8-cp310-cp310-macosx_10_9_x86_64.whl (294 kB)
Installing collected packages: regex, tiktoken
Successfully installed regex-2023.8.8 tiktoken-0.5.1
(chatgpt) macs-MacBook-Pro:Customer support mac$
```

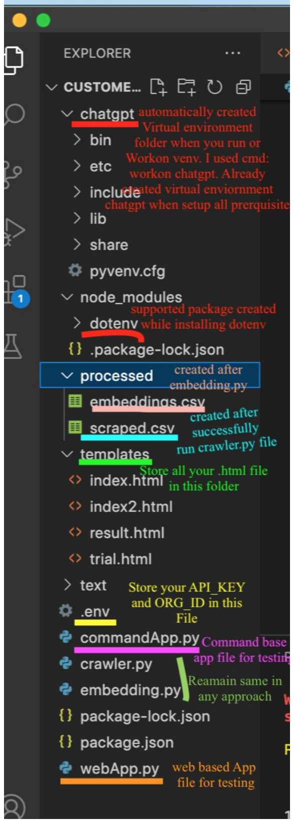
Implementation

Crawler.py file output



```
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Using cached pandas-2.1.1-cp310-cp310-macosx_10_9_x86_64.whl (11.7 MB)
Using cached numpy-1.26.0-cp310-cp310-macosx_10_9_x86_64.whl (20.6 MB)
Using cached pytz-2023.3.post1-py2.py3-none-any.whl (502 kB)
Installing collected packages: pytz, tzdata, six, numpy, python-dateutil, pandas
Successfully installed numpy-1.26.0 pandas-2.1.1 python-dateutil-2.8.2 pytz-2023.3.post1 six-1.16.0 tzdata-2023.3
(chatgpt) macs-MacBook-Pro:Customer support mac$ python3 crawler.py
https://openai.com/
https://openai.com/enterprise-privacy
https://openai.com/enterprise-privacy#content
https://openai.com/enterprise-privacy#our-commitments
https://openai.com/policies
https://openai.com/policies/service-terms
https://openai.com/policies#content
https://openai.com/policies/data-processing-addendum
https://openai.com/policies/coordinated-vulnerability-disclosure-policy
https://openai.com/policies/usage-policies
https://openai.com/policies/terms-of-use
https://openai.com/policies/sharing-publication-policy
https://openai.com/policies/service-credit-terms
https://openai.com/policies/plugin-terms
https://openai.com/residency
https://openai.com/research/solving-math-word-problems
https://openai.com/research?topics=transformers
https://openai.com/research/a-hazard-analysis-framework-for-code-synthesis-large-language-models
https://openai.com/research?models=openai-codex
https://openai.com/research?authors=heidy-khlaaf
https://openai.com/research?authors=gretchen-krueger
```

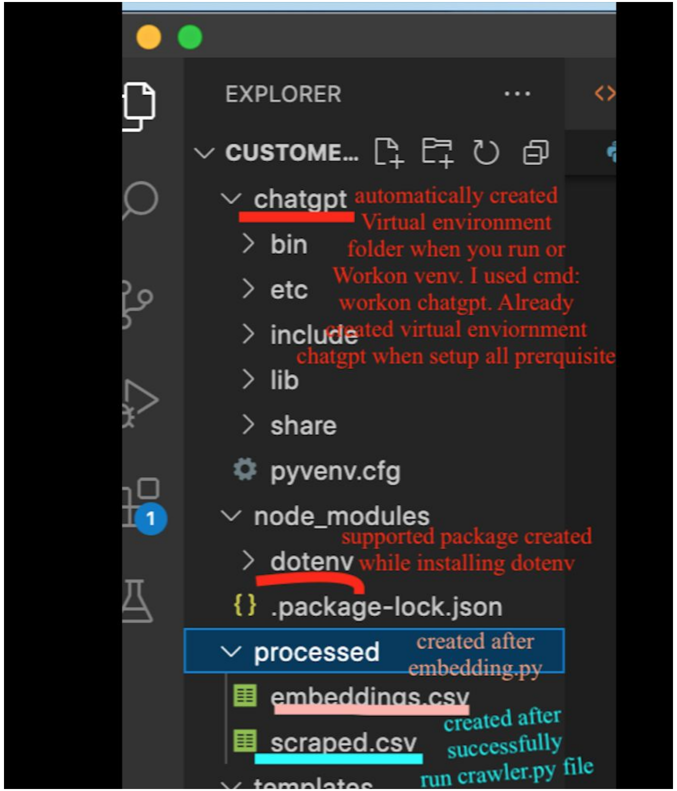
FILE HIERARCHY & ITS MEANING [MATCH THE COLOR LINE AND TEXT FOR MEANING]



EXPLORER ...

CUSTOME...

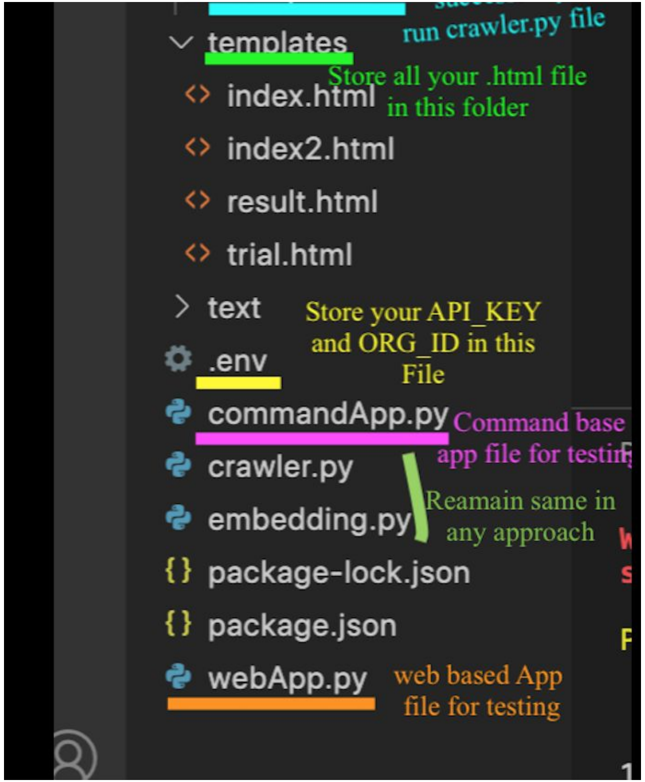
- chatgpt — automatically created Virtual environment folder when you run or Workon venv. I used cmd: workon chatgpt. Already created virtual enviornment chatgpt when setup all prerquisite
  - bin
  - etc
  - include
  - lib
  - share
  - pyvenv.cfg
- node_modules — supported package created while installing dotenv
  - dotenv
  - .package-lock.json
- processed — created after embedding.py
  - embeddings.csv
  - scraped.csv — created after successfully run crawler.py file
- templates — Store all your .html file in this folder
  - index.html
  - index2.html
  - result.html
  - trial.html
- text
- .env — Store your API_KEY and ORG_ID in this File
- commandApp.py — Command base app file for testin
- crawler.py
- embedding.py — Reamain same in any approach
- package-lock.json
- package.json
- webApp.py — web based App file for testing



EXPLORER ...

CUSTOME...

- chatgpt — automatically created Virtual environment folder when you run or Workon venv. I used cmd: workon chatgpt. Already created virtual enviornment chatgpt when setup all prerquisite
  - bin
  - etc
  - include
  - lib
  - share
  - pyvenv.cfg
- node_modules — supported package created while installing dotenv
  - dotenv
  - .package-lock.json
- processed — created after embedding.py
  - embeddings.csv
  - scraped.csv — created after successfully run crawler.py file
- templates

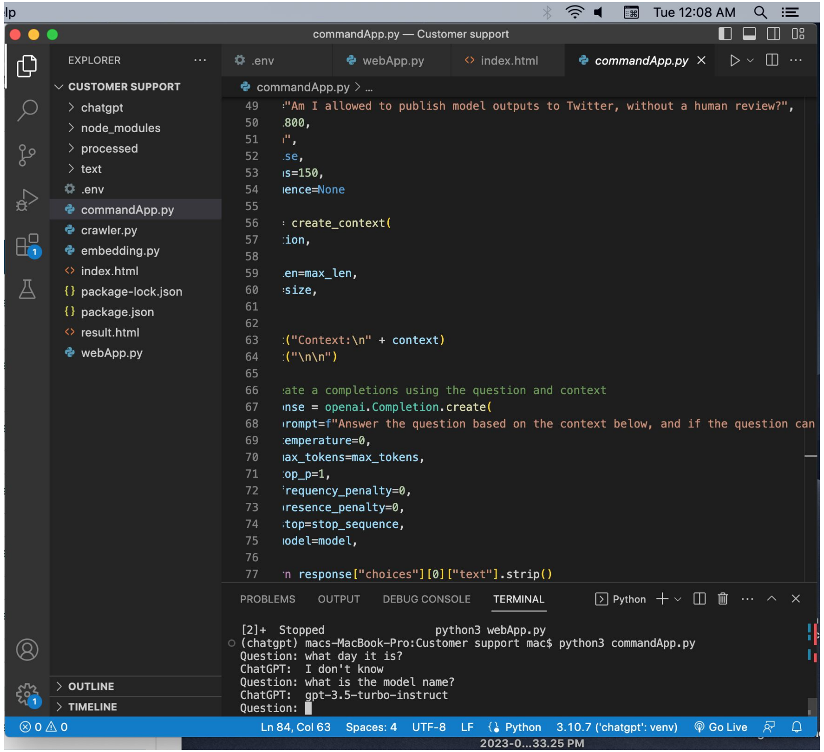


run crawler.py file

- templates — Store all your .html file in this folder
  - index.html
  - index2.html
  - result.html
  - trial.html
- text
- .env — Store your API_KEY and ORG_ID in this File
- commandApp.py — Command base app file for testin
- crawler.py
- embedding.py — Reamain same in any approach
- package-lock.json
- package.json
- webApp.py — web based App file for testing

Test

- Command based Output:

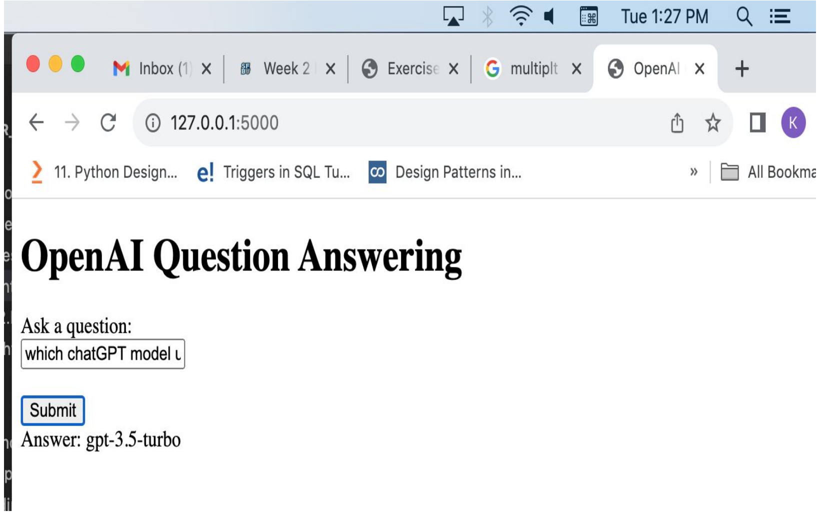


```
:"Am I allowed to publish model outputs to Twitter, without a human review?",
.800,
ı",
.se,
ıs=150,
ıence=None

: create_context(
:ion,

.en=max_len,
:size,


:("Context:\n" + context)
:("\n\n")

:ate a completions using the question and context
ınse = openai.Completion.create(
ırompt=f"Answer the question based on the context below, and if the question can
:emperature=0,
ıax_tokens=max_tokens,
:op_p=1,
'requency_penalty=0,
ıresence_penalty=0,
:top=stop_sequence,
ıodel=model,

'n response["choices"][0]["text"].strip()
```

```
[2]+  Stopped                 python3 webApp.py
(chatgpt) macs-MacBook-Pro:Customer support mac$ python3 commandApp.py
Question: what day it is?
ChatGPT:  I don't know
Question: what is the model name?
ChatGPT:  gpt-3.5-turbo-instruct
Question: 
```

Test continue

- web based Output: using python Flask



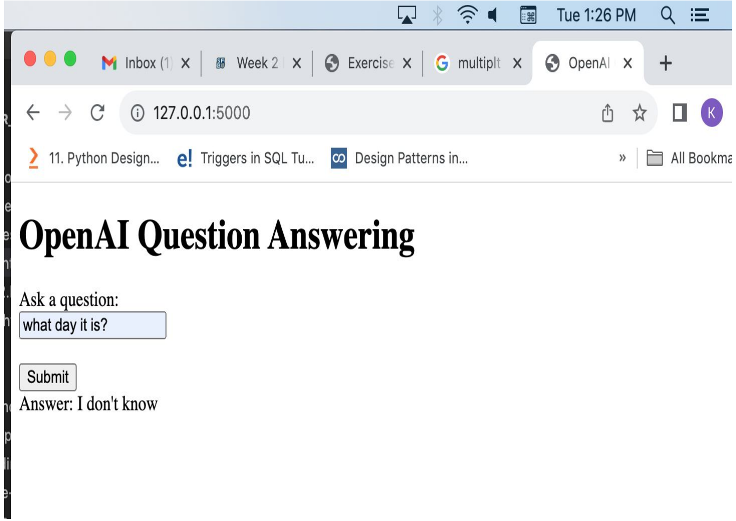
OpenAI Question Answering

Ask a question:

which chatGPT model u

Submit

Answer: gpt-3.5-turbo



OpenAI Question Answering

Ask a question:

what day it is?

Submit

Answer: I don't know

Conclusion

AI contributed to generate NLP projects using API based on large dataset model very quick compare to traditional approach. It require some basic knowledge and how to connect the phases. Used 3 steps crawling, embedding and testing in sequence to implement. Perform both Command based and web based solution.

Github link

Github project repository link:

https://github.com/DKruti/Customer_support_openAI

References

- https://platform.openai.com/docs/tutorials/web-qa-embeddings