Github link: https://github.com/JTasnim

Q2 ==> Project: Customer Support System: Use ChatGPT to build a web-based system that can answer questions about a website.

Step 1.1: Command Line Solution

[1]:
```
import requests
import re
import urllib.request
from bs4 import BeautifulSoup
from collections import deque
from html.parser import HTMLParser
from urllib.parse import urlparse
import os

# Regex pattern to match a URL
HTTP_URL_PATTERN = r'^http[s]*://.+'

domain = "www.reddit.com/" # <- put your domain to be crawled
full_url = "https://www.reddit.com/" # <- put your domain to be crawled with https or http

# Create a class to parse the HTML and get the hyperlinks
class HyperlinkParser(HTMLParser):
    def __init__(self):
        super().__init__()
        # Create a list to store the hyperlinks
        self.hyperlinks = []

    # Override the HTMLParser's handle_starttag method to get the hyperlinks
    def handle_starttag(self, tag, attrs):
        attrs = dict(attrs)

        # If the tag is an anchor tag and it has an href attribute, add the href attribute to the list of hyperlinks
        if tag == "a" and "href" in attrs:
            self.hyperlinks.append(attrs["href"])
```

[2]:
```
# Function to get the hyperlinks from a URL
def get_hyperlinks(url):

    # Try to open the URL and read the HTML
```

[2]:
```python
# Function to get the hyperlinks from a URL
def get_hyperlinks(url):

    # Try to open the URL and read the HTML
    try:
        # Open the URL and read the HTML
        with urllib.request.urlopen(url) as response:

            # If the response is not HTML, return an empty list
            if not response.info().get('Content-Type').startswith("text/html"):
                return []

            # Decode the HTML
            html = response.read().decode('utf-8')
    except Exception as e:
        print(e)
        return []

    # Create the HTML Parser and then Parse the HTML to get hyperlinks
    parser = HyperlinkParser()
    parser.feed(html)

    return parser.hyperlinks
```

[3]:
```python
# Function to get the hyperlinks from a URL that are within the same domain
def get_domain_hyperlinks(local_domain, url):
    clean_links = []
    for link in set(get_hyperlinks(url)):
        clean_link = None

        # If the link is a URL, check if it is within the same domain
        if re.search(HTTP_URL_PATTERN, link):
            # Parse the URL and check if the domain is the same
            url_obj = urlparse(link)
            if url_obj.netloc == local_domain:
                clean link = link
```


```python
    return parser.hyperlinks
```

[3]:
```python
# Function to get the hyperlinks from a URL that are within the same domain
def get_domain_hyperlinks(local_domain, url):
    clean_links = []
    for link in set(get_hyperlinks(url)):
        clean_link = None

        # If the link is a URL, check if it is within the same domain
        if re.search(HTTP_URL_PATTERN, link):
            # Parse the URL and check if the domain is the same
            url_obj = urlparse(link)
            if url_obj.netloc == local_domain:
                clean_link = link

        # If the link is not a URL, check if it is a relative link
        else:
            if link.startswith("/"):
                link = link[1:]
            elif link.startswith("#") or link.startswith("mailto:"):
                continue
            clean_link = "https://" + local_domain + "/" + link

        if clean_link is not None:
            if clean_link.endswith("/"):
                clean_link = clean_link[:-1]
            clean_links.append(clean_link)

    # Return the list of hyperlinks that are within the same domain
    return list(set(clean_links))
```


[4]:
```python
def crawl(url):
    # Parse the URL and get the domain
    local_domain = urlparse(url).netloc

    # Create a queue to store the URLs to crawl
    queue = deque([url])
```

[4]:
```
def crawl(url):
    # Parse the URL and get the domain
    local_domain = urlparse(url).netloc

    # Create a queue to store the URLs to crawl
    queue = deque([url])

    # Create a set to store the URLs that have already been seen (no duplicates)
    seen = set([url])

    # Create a directory to store the text files
    if not os.path.exists("text/"):
            os.mkdir("text/")

    if not os.path.exists("text/"+local_domain+"/"):
            os.mkdir("text/" + local_domain + "/")

    # Create a directory to store the csv files
    if not os.path.exists("processed"):
            os.mkdir("processed")

    # While the queue is not empty, continue crawling
    q_limit = 50
    count = 0
    while queue:
        if count==q_limit:
            break
        count += 1
        # Get the next URL from the queue
        url = queue.pop()
        print(url) # for debugging and to see the progress

        # Save text from the url to a <url>.txt file
        with open('text/'+local_domain+'/'+url[8:].replace("/", "_") + ".txt", "w", encoding="UTF-8") as f:

            # Get the text from the URL using BeautifulSoup
```

```
        with open('text/'+local_domain+'/'+url[8:].replace("/", "_") + ".txt", "w", encoding="UTF-8") as f:

            # Get the text from the URL using BeautifulSoup
            soup = BeautifulSoup(requests.get(url).text, "html.parser")

            # Get the text but remove the tags
            text = soup.get_text()

            # If the crawler gets to a page that requires JavaScript, it will stop the crawl
            if ("You need to enable JavaScript to run this app." in text):
                print("Unable to parse page " + url + " due to JavaScript being required")

            # Otherwise, write the text to the file in the text directory
            f.write(text)

        # Get the hyperlinks from the URL and add them to the queue
        for link in get_domain_hyperlinks(local_domain, url):
            if link not in seen:
                queue.append(link)
                seen.add(link)

crawl(full_url)
```

```
https://www.reddit.com/
https://www.reddit.com/r/popular/hot
https://www.reddit.com/r/fantasyfootball
https://www.reddit.com/r/fantasyfootball/hot
https://www.reddit.com/r/fantasyfootball/about
https://www.reddit.com/t/football
https://www.reddit.com/r/fantasyfootball/top
https://www.reddit.com/user/GreenDefinition5
https://www.reddit.com/r/fantasyfootball/comments/18amo4g/cj_stroud_leads_the_nfl_in_passing_yards_no
https://www.reddit.com/posts/2023
https://www.reddit.com/posts/2023/july/global
https://www.reddit.com/r/HumansBeingBros/comments/14x4gj6/great_success
https://www.reddit.com/best/communities/1/#t5_2xh58
https://www.reddit.com/best/communities/32
https://www.reddit.com/r/ChatGPTJailbreak
```

```
https://www.reddit.com/
https://www.reddit.com/r/popular/hot
https://www.reddit.com/r/fantasyfootball
https://www.reddit.com/r/fantasyfootball/hot
https://www.reddit.com/r/fantasyfootball/about
https://www.reddit.com/t/football
https://www.reddit.com/r/fantasyfootball/top
https://www.reddit.com/user/GreenDefinition5
https://www.reddit.com/r/fantasyfootball/comments/18amo4g/cj_stroud_leads_the_nfl_in_passing_yards_no
https://www.reddit.com/posts/2023
https://www.reddit.com/posts/2023/july/global
https://www.reddit.com/r/HumansBeingBros/comments/14x4gj6/great_success
https://www.reddit.com/best/communities/1/#t5_2xh58
https://www.reddit.com/best/communities/32
https://www.reddit.com/r/ChatGPTJailbreak
https://www.reddit.com/r/ChatGPTJailbreak/comments/1ft77je/video_guide_based_on_yell0wfever92_comp_doc
https://www.reddit.com/user/JustProPT
https://www.reddit.com/r/feedthebeast
https://www.reddit.com/r/feedthebeast/?f=flair_name%3A%22Alex's%20Caves%22
https://www.reddit.com/user/bababoongus
https://www.reddit.com/r/feedthebeast/comments/1dst05d/please_im_going_crazy_i_spent_so_many_hours
https://www.reddit.com/r/feedthebeast/?f=flair_name%3A%22Question%22
https://www.reddit.com/user/b52kl
https://www.reddit.com/user/b52kl/?sort=new
https://www.reddit.com/?sort=new&feedViewType=compactView
https://www.reddit.com/r/nursing/comments/1fu8isi/looking_to_make_a_switch_and_get_out_of_bedside
https://www.reddit.com/user/Goaltenderforlife
https://www.reddit.com/r/CrimeInTheD/comments/1ftw7s9/comment/lpxjod5
https://www.reddit.com/user/Main-Moment3089
https://www.reddit.com/user/Main-Moment3089/submitted
https://www.reddit.com/r/CrimeInTheD/comments/1evgdr6/i_love_lil_poodi_so_much_i_hold_the_dick_while_he
https://www.reddit.com/user/Main-Moment3089/submitted/?sort=new
https://www.reddit.com/user/Main-Moment3089/submitted/?sort=top
https://www.reddit.com/r/CrimeInTheD/comments/1c1vsmr/am_i_the_only_nigga_who_thoughts_get_real
https://www.reddit.com/?sort=top&feedViewType=compactView
https://www.reddit.com/search/?q=John+Amos&type=link&source=trending&cId=ecc11fe3-e35e-4315-985e-977f4f5f1e6d&iId=06d3ff4b-61eb-4fca-861b-a0b
9a7e860d0
https://www.reddit.com/r/80s/comments/1ftvj35/rip_john_amos_i_love_this_scene_from_coming_to
https://www.reddit.com/r/80s/?f=flair_name%3A%22Film%22
```

com/r/feedthebeast/comments/1dst05d/please_im_going_crazy_i_spent_so_many_hours

```
'ascii' codec can't encode character '\xe1' in position 39: ordinal not in range(128)
https://www.reddit.com/r/AliExpressBR
https://www.reddit.com/r/AliExpressBR/comments/1fu7xeo/ja_começou_as_prisões_do_amor
'ascii' codec can't encode character '\xe7' in position 44: ordinal not in range(128)
```

[5]:
```python
def remove_newlines(serie):
    serie = serie.str.replace('\n', ' ')
    serie = serie.str.replace('\\n', ' ')
    serie = serie.str.replace('  ', ' ')
    serie = serie.str.replace('  ', ' ')
    return serie
```

[6]:
```python
import pandas as pd

# Create a list to store the text files
texts=[]

# Get all the text files in the text directory
for file in os.listdir("text/" + domain + "/"):

    # Open the file and read the text
    with open("text/" + domain + "/" + file, "r", encoding="UTF-8") as f:
        text = f.read()

        # Omit the first 11 lines and the last 4 lines, then replace -, _, and #update with spaces.
        texts.append((file[11:-4].replace('-',' ').replace('_', ' ').replace('#update',''), text))

# Create a dataframe from the list of texts
df = pd.DataFrame(texts, columns = ['fname', 'text'])

# Set the text column to be the raw text with the newlines removed
df['text'] = df.fname + ". " + remove_newlines(df.text)
df.to_csv('processed/scraped.csv')
df.head()
```

[6]:

	fname	text
0	com r 80s ?f=flair name%3A%22Film%22	com r 80s ?f=flair name%3A%22Film%22. Reddit ...

```
# Set the text column to be the raw text with the newlines removed
df['text'] = df.fname + ". " + remove_newlines(df.text)
df.to_csv('processed/scraped.csv')
df.head()
```

[6]:

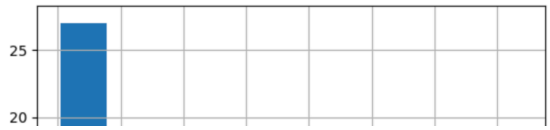
	fname	text
0	com r 80s ?f=flair name%3A%22Film%22	com r 80s ?f=flair name%3A%22Film%22. Reddit ...
1	com r fantasyfootball comments 18amo4g cj stro...	com r fantasyfootball comments 18amo4g cj stro...
2	com r CrimeInTheD comments 1evgdr6 i love lil ...	com r CrimeInTheD comments 1evgdr6 i love lil ...
3	com t football	com t football. Reddit - Dive into anything ...
4	com best communities 32	com best communities 32. Reddit - Dive into a...

[7]:

```
import tiktoken

# Load the cl100k_base tokenizer which is designed to work with the ada-002 model
tokenizer = tiktoken.get_encoding("cl100k_base")

df = pd.read_csv('processed/scraped.csv', index_col=0)
df.columns = ['title', 'text']

# Tokenize the text and save the number of tokens to a new column
df['n_tokens'] = df.text.apply(lambda x: len(tokenizer.encode(x)))

# Visualize the distribution of the number of tokens per row using a histogram
df.n_tokens.hist()
```

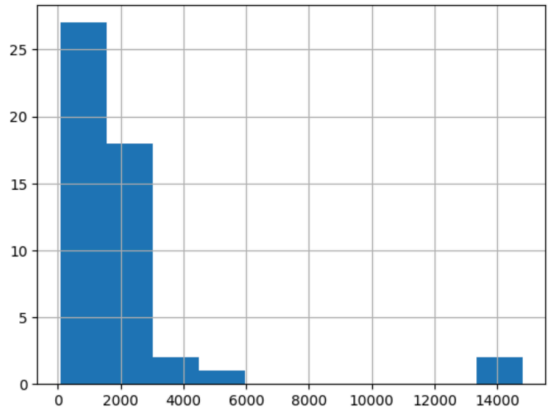
[7]:

```
<Axes: >
```



[7]:

```
<Axes: >
```



[8]:

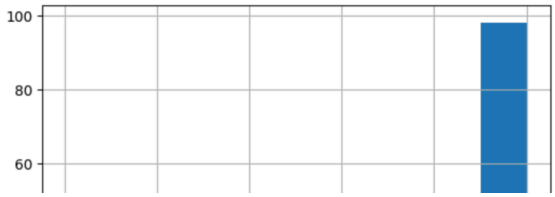
```
max_tokens = 500

# Function to split the text into chunks of a maximum number of tokens
def split_into_many(text, max_tokens = max_tokens):

    # Split the text into sentences
    sentences = text.split('. ')

    # Get the number of tokens for each sentence
    n_tokens = [len(tokenizer.encode(" " + sentence)) for sentence in sentences]
```

[8]:
```python
max_tokens = 500

# Function to split the text into chunks of a maximum number of tokens
def split_into_many(text, max_tokens = max_tokens):

    # Split the text into sentences
    sentences = text.split('. ')

    # Get the number of tokens for each sentence
    n_tokens = [len(tokenizer.encode(" " + sentence)) for sentence in sentences]

    chunks = []
    tokens_so_far = 0
    chunk = []

    # Loop through the sentences and tokens joined together in a tuple
    for sentence, token in zip(sentences, n_tokens):

        # If the number of tokens so far plus the number of tokens in the current sentence is greater
        # than the max number of tokens, then add the chunk to the list of chunks and reset
        # the chunk and tokens so far
        if tokens_so_far + token > max_tokens:
            chunks.append(". ".join(chunk) + ".")
            chunk = []
            tokens_so_far = 0

        # If the number of tokens in the current sentence is greater than the max number of
        # tokens, go to the next sentence
        if token > max_tokens:
            continue

        # Otherwise, add the sentence to the chunk and add the number of tokens to the total
        chunk.append(sentence)
        tokens_so_far += token + 1

    return chunks
```

```python
    return chunks


shortened = []

# Loop through the dataframe
for row in df.iterrows():

    # If the text is None, go to the next row
    if row[1]['text'] is None:
        continue

    # If the number of tokens is greater than the max number of tokens, split the text into chunks
    if row[1]['n_tokens'] > max_tokens:
        shortened += split_into_many(row[1]['text'])

    # Otherwise, add the text to the list of shortened texts
    else:
        shortened.append( row[1]['text'] )
```

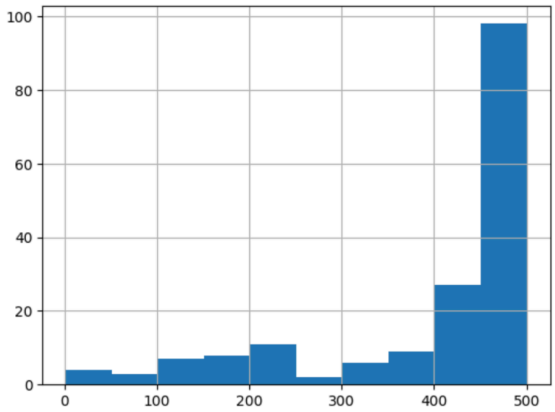
[9]:
```python
df = pd.DataFrame(shortened, columns = ['text'])
df['n_tokens'] = df.text.apply(lambda x: len(tokenizer.encode(x)))
df.n_tokens.hist()
```

[9]:
```
<Axes: >
```



[9]:
```
df = pd.DataFrame(shortened, columns = ['text'])
df['n_tokens'] = df.text.apply(lambda x: len(tokenizer.encode(x)))
df.n_tokens.hist()
```

[9]:
```
<Axes: >
```



[10]:
```
print (df)
```

```
                                                text  n_tokens
0    com r 80s ?f=flair name%3A%22Film%22.  Reddit ...       316
1    com r fantasyfootball comments 18amo4g cj stro...       214
2    com r CrimeInTheD comments 1evgdr6 i love lil ...       197
3    com t football.  Reddit - Dive into anything  ...        87
4                            com best communities 32.         6
```

[10]:
```
print (df)
```

```
                                                text  n_tokens
0    com r 80s ?f=flair name%3A%22Film%22.  Reddit ...       316
1    com r fantasyfootball comments 18amo4g cj stro...       214
2    com r CrimeInTheD comments 1evgdr6 i love lil ...       197
3    com t football.  Reddit - Dive into anything  ...        87
4                            com best communities 32.         6
..                                                 ...       ...
170  We want to keep things fun, but not at the exp...       498
171  com user Main Moment3089 submitted ?sort=top. ...       461
172     I love lil poodi so much I hold the dick wh...        90
173  com user Main Moment3089 submitted.  Reddit - ...       377
174  Created by u/RealQuickTD, (No Affiliation with...       140

[175 rows x 2 columns]
```

[6]:
```
import openai
import os

# Load environment variables from the .env file
with open("/Users/jubaidatasnim/GenAi/.env") as env:
    for line in env:
        key, value = line.strip().split("=")
        os.environ[key] = value

# Set the OpenAI API key from environment variables
api_key = "sk-proj-vDYnHgUdiN5z2i7sP6dx-FqJW6sXWGLmlTxS7btP8PUzbfPjZOHD3duQT1qgTTYye1zYcttnJ5T3BlbkFJBocU_fV3-POHUf0MeoF8r8dZu_yUoxszMuVaE8uN5
openai.api_key = api_key
# Example usage of embeddings (assuming you have a DataFrame called df)
df['embeddings'] = df.text.apply(lambda x: openai.Embedding.create(input=x, engine='text-embedding-ada-002')['data'][0]['embedding'])

# Save the embeddings to a CSV file
df.to_csv('processed/embeddings.csv')

# Display the first few rows
df.head()
```

jupyter **Customer Support System** Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

```
[486 rows x 2 columns]
```

[59]:
```
import openai
import os
import requests

# Load environment variables from the .env file
with open("/Users/jubaidatasnim/GenAi/.env") as env:
    for line in env:
        key, value = line.strip().split("=")
        os.environ[key] = value

# Set the OpenAI API key from environment variables

openai.api_key = os.getenv('API_KEY')
openai.api_key = api_key
# Example usage of embeddings (assuming you have a DataFrame called df)
df['embeddings'] = df.text.apply(lambda x: openai.Embedding.create(input=x, engine='text-embedding-ada-002')['data'][0]['embedding'])

# Save the embeddings to a CSV file
df.to_csv('processed/embeddings.csv')

# Display the first few rows
df.head()
```

[59]:

	text	n_tokens	embeddings
0	com r 80s ?f=flair name%3A%22Film%22. Reddit ...	316	[0.0023201224394142628, -0.02078080736100673...
1	com user TinolangEsophagus. Reddit - Dive int...	463	[0.014561430551111698, 0.0049392483197152615, ...
2	Members Online • What's the most unusua...	463	[-0.013696198351681232, -0.016805235296487808,...
3	Members Online • obey traffic laws at a...	390	[0.007803275249898434, -0.003955921158194542, ...
4	com r olympics comments 1fcmak5 australian bre...	211	[-0.025677474215626717, 0.008606936782598495, ...

jupyter **Customer Support System** Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

```
# Save the embeddings to a CSV file
df.to_csv('processed/embeddings.csv')

# Display the first few rows
df.head()
```

[59]:

	text	n_tokens	embeddings
0	com r 80s ?f=flair name%3A%22Film%22. Reddit ...	316	[0.0023201224394142628, -0.02078080736100673...
1	com user TinolangEsophagus. Reddit - Dive int...	463	[0.014561430551111698, 0.0049392483197152615, ...
2	Members Online • What's the most unusua...	463	[-0.013696198351681232, -0.016805235296487808,...
3	Members Online • obey traffic laws at a...	390	[0.007803275249898434, -0.003955921158194542, ...
4	com r olympics comments 1fcmak5 australian bre...	211	[-0.025677474215626717, 0.008606936782598495, ...

[60]:
```
import pandas as pd
import numpy as np
from openai.embeddings_utils import distances_from_embeddings

df=pd.read_csv('processed/embeddings.csv', index_col=0)
df['embeddings'] = df['embeddings'].apply(eval).apply(np.array)

df.head()
```

[60]:

	text	n_tokens	embeddings
0	com r 80s ?f=flair name%3A%22Film%22. Reddit ...	316	[0.0023201224394142628, -0.02078080736100673...
1	com user TinolangEsophagus. Reddit - Dive int...	463	[0.014561430551111698, 0.0049392483197152615, ...
2	Members Online • What's the most unusua...	463	[-0.013696198351681232, -0.016805235296487808,...
3	Members Online • obey traffic laws at a...	390	[0.007803275249898434, -0.003955921158194542, ...
4	com r olympics comments 1fcmak5 australian bre...	211	[-0.025677474215626717, 0.008606936782598495, ...

4	com best communities 32.	6	[0.03258850425481796, -0.0011461181566119194, ...

[65]:
```python
def create_context(
    question, df, max_len=1800, size="ada"
):
    """
    Create a context for a question by finding the most similar context from the dataframe
    """
    # Get the embeddings for the question
    q_embeddings = openai.Embedding.create(input=question, engine='text-embedding-ada-002')['data'][0]['embedding']
    # Get the distances from the embeddings
    df['distances'] = distances_from_embeddings(q_embeddings, df['embeddings'].values, distance_metric='cosine')

    returns = []
    cur_len = 0

    # Sort by distance and add the text to the context until the context is too long
    for i, row in df.sort_values('distances', ascending=True).iterrows():

        # Add the length of the text to the current length
        cur_len += row['n_tokens'] + 4

        # If the context is too long, break
        if cur_len > max_len:
            break

        # Else add it to the text that is being returned
        returns.append(row["text"])

    # Return the context
    return "\n\n###\n\n".join(returns)
```

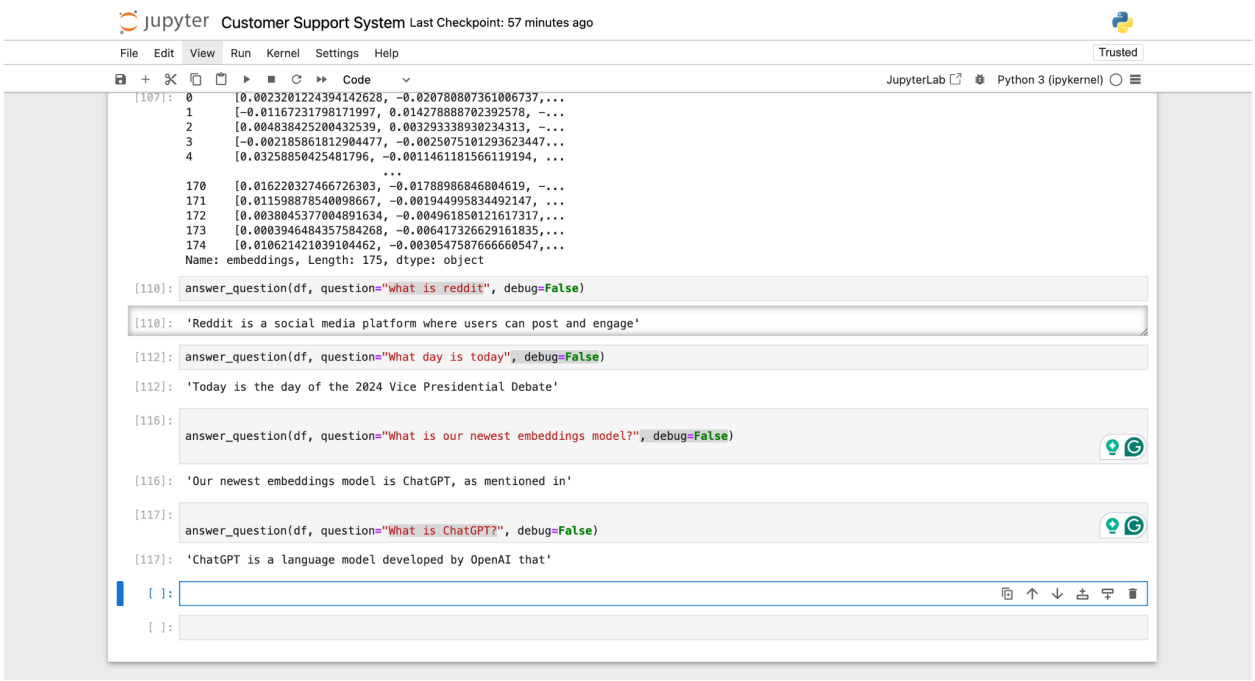
[106]:
```python
def answer_question(
    df,
    model="gpt-3.5-turbo",
    question="Am I allowed to publish model outputs to Twitter, without a human review?",
```

[106]:
```python
def answer_question(
    df,
    model="gpt-3.5-turbo",
    question="Am I allowed to publish model outputs to Twitter, without a human review?",
    max_len=1800,
    size="ada",
    debug=False,
    max_tokens=150,
    stop_sequence=None
):
    """
    Answer a question based on the most similar context from the dataframe texts
    """
    context = create_context(
        question,
        df,
        max_len=max_len,
        size=size,
    )
    # If debug, print the raw model response
    content = "Context: " + context + "\n\n---\n\nQuestion: " + question + "\nAnswer:"
    if debug:
        print("Context:\n" + content)
        print("\n\n")
    try:
        # Create a chat completion using the question and context
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "user", "content": content}
            ],
            temperature=0,
            max_tokens=12,
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0,
```

```
    content = "Context: " + context + "\n\n---\n\nQuestion: " + question + "\nAnswer:"
    if debug:
        print("Context:\n" + content)
        print("\n\n")
    try:
        # Create a chat completion using the question and context
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "user", "content": content}
            ],
            temperature=0,
            max_tokens=12,
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0,
            stop=stop_sequence,
        )
        return response.choices[0].message.content
    except Exception as e:
        print(e)
        return ""
```

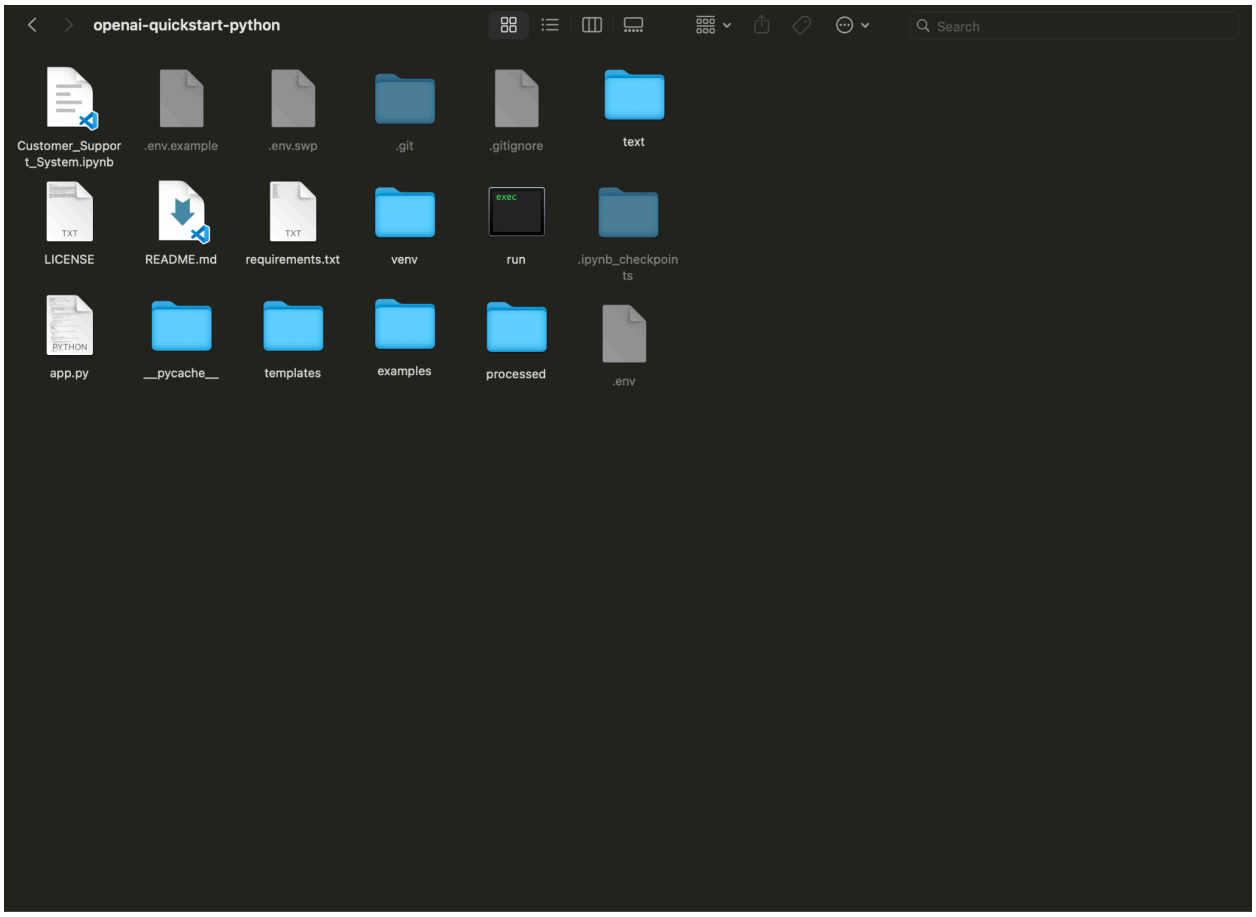
```
[107]: df['embeddings']
```

```
[107]: 0      [0.0023201224394142628, -0.020780807361006737,...
       1      [-0.01167231798171997, 0.014278888702392578, -...
       2      [0.004838425200432539, 0.003293338930234313, -...
       3      [-0.002185861812904477, -0.0025075101293623447...
       4      [0.03258850425481796, -0.0011461181566119194, ...
                                    ...
       170    [0.016220327466726303, -0.01788986846804619, -...
       171    [0.011598878540098667, -0.0019944995834492147, ...
       172    [0.0038045377004891634, -0.004961850121617317,...
       173    [0.0003946484357584268, -0.006417326629161835,...
       174    [0.010621421039104462, -0.0030547587666660547,...
       Name: embeddings, Length: 175, dtype: object
```



Step 1.2: Web-based Solution (Python Flask webserver)

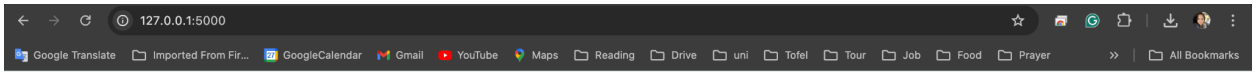
Step 1.2.1: use Python to create a web-based interface to ChatGPT



Step 1.2.2: Integrate the Python code created in Step 1.1 and Step 1.2.1 to a create a web-based interface to let the users ask ChatGPT questions about the website using a browser.

flask run

```
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
Before
q_embeddings
after
ERROR
127.0.0.1 - - [01/Oct/2024 21:43:38] "POST / HTTP/1.1" 200 -
^CJubaidas-MacBook-Pro:openai-quickstart-python jubaidatasnim$ flask run
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
Before
q_embeddings
after
ERROR
127.0.0.1 - - [01/Oct/2024 21:50:14] "POST / HTTP/1.1" 200 -
Before
q_embeddings
after
ERROR
127.0.0.1 - - [01/Oct/2024 21:50:15] "POST / HTTP/1.1" 200 -
^CJubaidas-MacBook-Pro:openai-quickstart-python jubaidatasnim$ flask run
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
Before
q_embeddings
after
ERROR
127.0.0.1 - - [01/Oct/2024 21:51:13] "POST / HTTP/1.1" 200 -
Before
q_embeddings
after
ERROR
127.0.0.1 - - [01/Oct/2024 21:51:22] "POST / HTTP/1.1" 200 -
^CJubaidas-MacBook-Pro:openai-quickstart-python jubaidatasnim$ flask run
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
strip
127.0.0.1 - - [01/Oct/2024 22:42:01] "POST / HTTP/1.1" 200 -
strip
127.0.0.1 - - [01/Oct/2024 22:42:07] "POST / HTTP/1.1" 200 -
^CJubaidas-MacBook-Pro:openai-quickstart-python jubaidatasnim$ flask run
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [01/Oct/2024 22:42:55] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [01/Oct/2024 22:42:59] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [01/Oct/2024 22:43:09] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [01/Oct/2024 22:59:17] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [01/Oct/2024 22:59:24] "POST / HTTP/1.1" 200 -
```



Text Input Form

Enter some text: What is ChatGPT? Submit

You entered: ChatGPT is an AI-powered chatbot that can engage