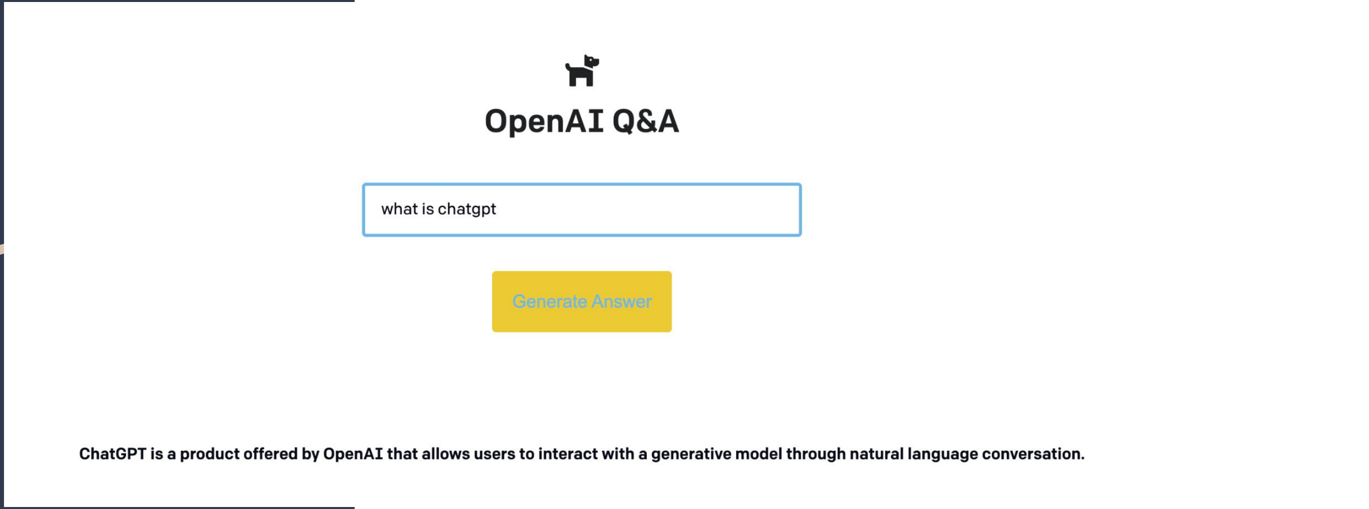
Customer Service Website

Interface: Command line & Web-based & Node.js

By Lianwei Deng(19874)

Overview

Use ChatGPT to build a web-based system that can answer questions about a website.



ChatGPT is a product offered by OpenAI that allows users to interact with a generative model through natural language conversation.

Design

1. Set up a web crawler
2. Building an embeddings index
3. Building a question answering system

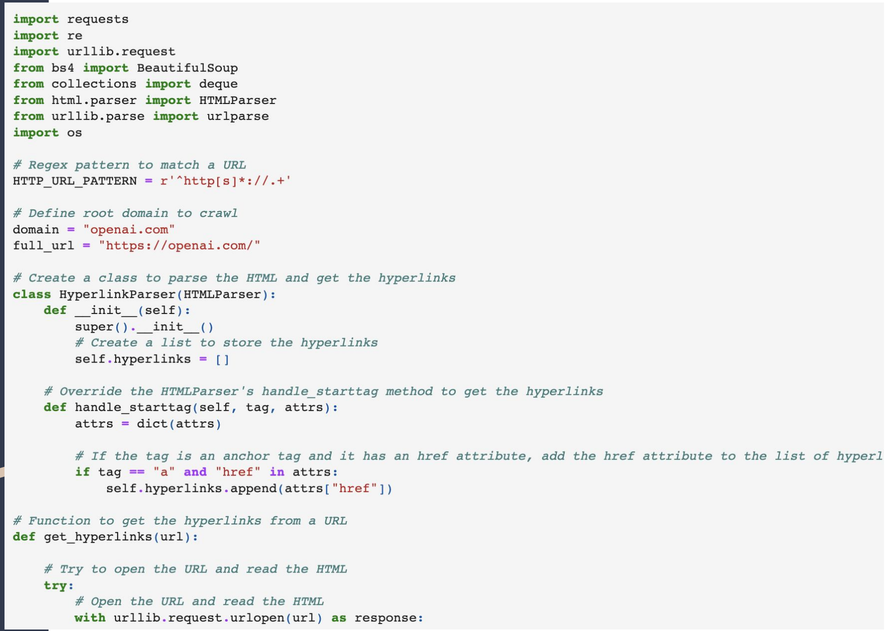
Command line / Web-based Solution

- mac(unix operating system)
- Using OPENAI-Cookbook to build an AI that can answer questions about a website. (https://platform.openai.com/docs/tutorials/web-qa-embeddings)

- Use python to create a web-based user interface
- Integrate command line with python code to create a web-based interface

Command line

- Step1: setup variable environment with your API key (openai.api_key = 'your-api-key')
- Step2: install packages
  - Pip install -r requirements.txt
- Step3: Create Interaction Script. (openai has the source code(web-qa.ipynb or web-qa.py)
  - Build a Python script for customer interactions.
  - Use the OpenAI API to generate responses to user input.
  - Handle user queries and provide AI-driven responses.



```python
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]*://.+'

# Define root domain to crawl
domain = "openai.com"
full_url = "https://openai.com/"

# 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 hyperl
        if tag == "a" and "href" in attrs:
            self.hyperlinks.append(attrs["href"])

# 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:
```

Web-based(python -flask)

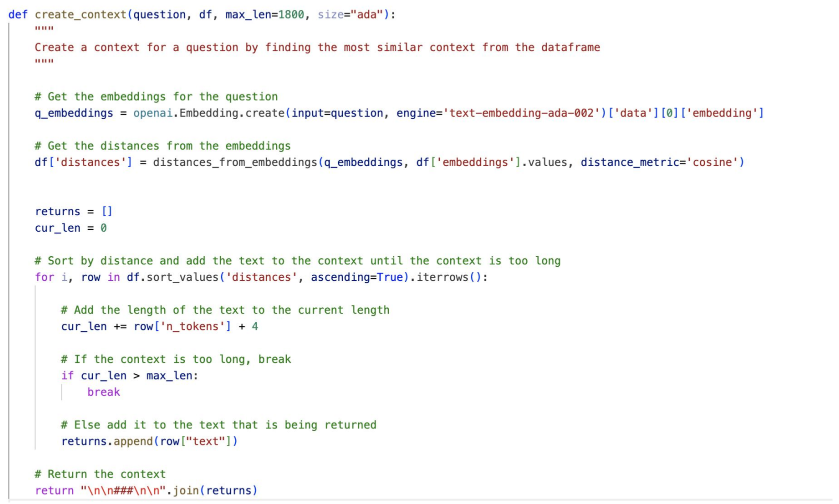
- Step1: Creating a working directory
  - Mkdir quickstart_python ; cd quickstart_python
- Step2: Download the code to the working directory
  - git clone https://github.com/openai/openai-quickstart-python.git
- Step3: Add the API Key
  - cd openai-quickstart-python
  - cp .env.example .env
  - vi .env
- Step4: Run the app in the Server(in this directory: quickstart_python/openai-quickstart-python)
  - Vi run
    - # python -m venv venv
    - python3 -m venv venv
    - . venv/bin/activate
    - pip install -r requirements.txt
    - flask run
  - Chmod 755 run
  - Run
  - After running the python code, open a browser to access http://localhost:5000

cont’d

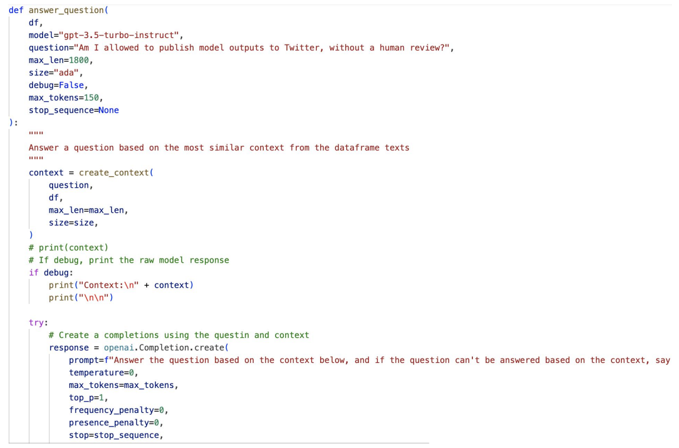
Step5: integrate command line and python flask

Change some part of code in python flask(app.py)

Use command line (asking question part to put in app.py)



```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)
```



```python
def answer_question(
    df,
    model="gpt-3.5-turbo-instruct",
    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,
    )
    # print(context)
    # If debug, print the raw model response
    if debug:
        print("Context:\n" + context)
        print("\n\n")

    try:
        # Create a completions using the questin and context
        response = openai.Completion.create(
            prompt=f"Answer the question based on the context below, and if the question can't be answered based on the context, say
            temperature=0,
            max_tokens=max_tokens,
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0,
            stop=stop_sequence,
```

Node.js

If we use mac to run node.js, we need to update package list first:

1. $brew update
2. $brew upgrade
3. Verify installation: node -v; npm -v
4. Install Node.js $brew install node

Node.js

Step1 is the same like python flask, the only different is the directory name:

Mkdir quickstart_node; cd quickstart_node

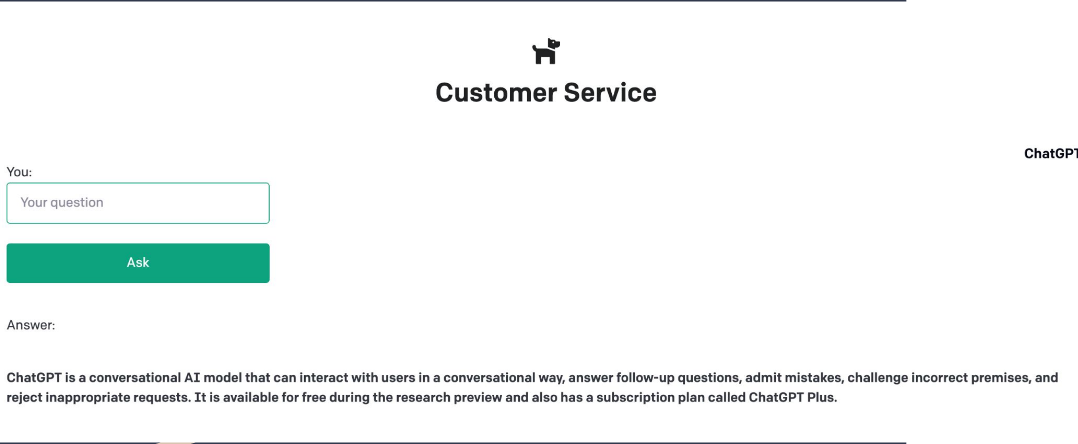
Step2: git clone https://github.com/openai/openai-quickstart-node.git

Step3: Add API key(same as step3 in python flask)

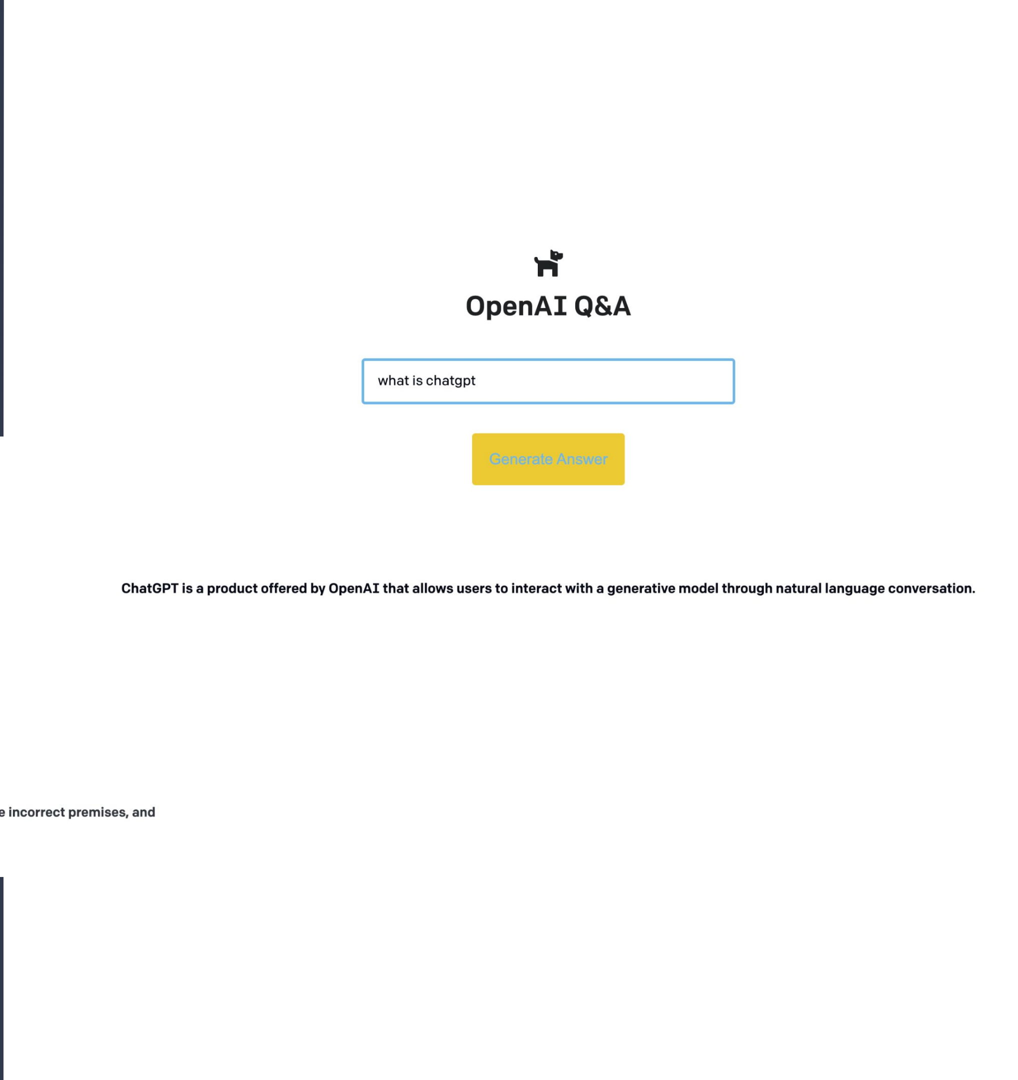
Step4: run the app

1. Npm install
2. Npm run dev
3. After running this code, open a browser to access http://localhost:3000

Test



ChatGPT is a conversational AI model that can interact with users in a conversational way, answer follow-up questions, admit mistakes, challenge incorrect premises, and reject inappropriate requests. It is available for free during the research preview and also has a subscription plan called ChatGPT Plus.



ChatGPT is a product offered by OpenAI that allows users to interact with a generative model through natural language conversation.

Conclusion

This project is using two ways to implement

- Customers need to download my code
    - Command line
- Customers do not need to download the code
    - Web-based(flask & Node.js)

The system now provides flexibility with both command-line and web-based interfaces for user convenience.

The goal is to offer users various ways to interact with ChatGPT, catering to different preferences and requirements.

Enhance

- Future steps involve exploring web-based solutions using Node.js (Step 1.3) for a broader range of implementation options.
- Continue refining and optimizing the user interface for a seamless experience.
- Consider user feedback and potential improvements to enhance system usability.

References

https://hc.labnet.sfbu.edu/~henry/sfbu/course/machine_learning/chatgpt/slide/exercise_chatgpt.html

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

https://github.com/openai/openai-cookbook/tree/main/apps/web-crawl-q-and-a