Report

May 26, 2023

1 May 22-26 Week

James León

1.1 Advances on the Udacity Summaries

- The course was completed up to 60% from the previous 20% To check the notes please refer to the following notion which will be soon uploaded to github as a PDF for more convenient access:

Notes on Udacity

Please refer to the next section of the report for Ney's text summarization using the OpenAI API

1.2 Chat GPT app to summarize text documents

1.2.1 Brief

138 conversations were summarized with this approach A tutorial was created for this project which is attached after the code. The complete code of the project can be found on the github repository

Here an explanation on the use:

1. The SimpleTokenizer.py receives a text file from accessapi.py and returns a processed result. This process is done once and creates the parsed_data.pkl structure to store the resulting objects to avoid repeating unnecesary steps.
2. When the parsed_data.pkl file doesn't exist, the SimpleTokenizer.py parses the received text to a dictionary made of a title and chunks of text smaller than 1700 words (To avoid surpassing the GPT models token limit).
3. Accessapi.py also stores a control value in the last_processed.txt which helps to re run the script in case of failure.
4. The accessapi.py connects to the OpenAI server and prompts the instructions with the chunks of text corresponding to a title from the generate_summary function.
5. The summarize_streams() joins all summarized chunks together to create a single summary for each title.
6. Resulting summaries are written to the summaries.txt file. Every time the script needs to be re run due to an error, the new summaries are appended to previous ones.

Caution: Some results may have inconsistent answers or typos. However, trial and error demonstrated that the best values to configure the model for this task are as set in accesapi.py

1.2.2 The Code

[ ]:
```
#SimpleTokenizer

import re

class Stream:
    def __init__(self, title):
        self.title = title
        self.parts = []
        self.current_part_words = 0
        self.current_part = []

    def add_words(self, words):
        for word in words:
            if self.current_part_words + len(word.split()) > 1700:
                self.parts.append(' '.join(self.current_part))
                self.current_part = [word]
                self.current_part_words = len(word.split())
            else:
                self.current_part.append(word)
                self.current_part_words += len(word.split())

    def finalize(self):
        if self.current_part:
            self.parts.append(' '.join(self.current_part))

def parse_file(filename):
    with open(filename, 'r', encoding='cp1252') as file: #Here you should␣
 ↪choose the right encoding for your text, this one worked for me
        content = file.readlines()

    streams = []
    current_stream = None

    for line in content:
        line = line.strip()  # remove leading/trailing whitespace
        if line.startswith("Stream name:"):
            if current_stream is not None:
                current_stream.finalize()
                streams.append(current_stream)
                #counter += 1
            current_stream = Stream(line[len("Stream name:"):].strip())
```

```
        elif current_stream is not None and line != '':  # ignore blank lines
            current_stream.add_words(re.split(r'\s+', line))

    if current_stream is not None:
        current_stream.finalize()
        streams.append(current_stream)

    return [{'title': stream.title, 'parts': stream.parts} for stream in␣
 ↪streams]
```

[ ]:
```
#access Api
import openai
import os
import pickle
from alive_progress import alive_bar
from local_settings import secret_key
from SimpleTokenizer import parse_file

openai.api_key = secret_key

def generate_summary(text):
    # Ensure text length is within GPT-3's maximum input size
    if len(text) > 2048:
        text = text[:2048]

    # Add specific instructions to the prompt
    prompt = f"The following text is a conversation from a trading transcript:␣
 ↪\"{text}\". Summarize the most important trading-related information in four␣
 ↪concise bullet points. After the bullet points, provide a brief paragraph␣
 ↪suggesting potential actions and key ideas based on the summarized points,␣
 ↪all in the style of Warren Buffet. Always penalize redundancy. Write only␣
 ↪sentences with full ideas"

    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=prompt,
        temperature=0.4,
        max_tokens=175
    )

    return response.choices[0].text.strip()

def summarize_stream(Sumstream):
    summarized_parts = [generate_summary(part) for part in Sumstream['parts']]
    return ' '.join(summarized_parts)

filename = 'transcripts.txt'  # or wherever your file is located
```

```
# Check if parsed data exists
if os.path.exists('parsed_data.pkl'):
    with open('parsed_data.pkl', 'rb') as f:
        parsed_data = pickle.load(f)
else:
    parsed_data = parse_file(filename)
    # Save parsed data
    with open('parsed_data.pkl', 'wb') as f:
        pickle.dump(parsed_data, f)

# Open a file for writing
with open('summaries.txt', 'a', encoding='UTF-8') as f:

    last_processed_title = ""
    if os.path.exists('last_processed.txt'):
        with open('last_processed.txt', 'r', encoding='UTF-8') as lp:
            last_processed_title = lp.read().strip()

    start_processing = False if last_processed_title else True

    with alive_bar(len(parsed_data)) as bar:
        for stream in parsed_data:
            if not start_processing:
                if stream['title'] == last_processed_title:
                    start_processing = True
                continue

            try:
                with open('last_processed.txt', 'w', encoding='UTF-8') as lp:
                    lp.write(stream['title'])
                summary = summarize_stream(stream)
                f.write(f"Title: {stream['title']}\n")
                f.write(f"Summary: {summary}\n\n")
                bar()
            except Exception as e:
                print(f"Error while processing title: {stream['title']}")
                print(str(e))
                break
```

1.2.3 The Results

Let's take the first conversation from the script —which turned out to be a really long one.

`Stream name: $1.3M gain on $TVIX.`

> Morning, guys. Can you hear me? Okay, great, great, great. Yes, you bet I did. You bet I did. Oh, yeah. I caught some tweak short yesterday. Oh, yeah. Oh, yeah. I did.

> Oh, it's some big size. Oh, yeah. I'm going to talk about it later. Oh, you bet I caught. Oh, yes. Oh, yes, I did. I did. Big time. Big time. Oh, yeah. I got some tweaks short yesterday. Yes, I did. I was close to that. Yes, I was close to that before it bounds, yes, at 1.3 I was up 1.51 0.6 at 1 point But then it had a bounce later I held some overnight and then it cap down and now it's kind of like grinding higher in premarket, I'm going to say. Yes. It was about 1.3 – well, it was 1.3 when I took in Charlotte I was up at 1.51 0.1 Yes. Oh, yes. Oh, yes. And I could see the decoupling. You could clearly see when it started decoupling from the indices. Like here, this is where I will short some like before, but here is when I started really, really adding because I saw like here 12:45, like the Spice, they were nowhere near their highs or 12 5 here is well 45 They were nowhere near its highs, but TVIX started taking out the lows of the day or was very close to lows of the day. And here, when it built these lower highs, I added some here. Here too, where is it? 145 145, where is it here? Like the Spice, again, Spice, we're building lower highs on the 5 minute chart, and they were nowhere near at the highs of the day, and Twicks was also building lower highs, and that had a range break This being multi day range break, that's where I added some size. It went down over 30 percent in like an hour or 2 That was just if you clearly see the relative weakness on Twigg.

(4920 words later…)

> I am not on a radar, there's a lot today, probably mostly going to focus on TVA. It looks like it's about to take out loads of the day again. I don't know if I'm going at this time or not, because now I have pretty decent size. And I don't want to get my average too low, but you never know. I think like it looks like the size should – the size have a big grip is like there's an air pocket here from mid-200s to like a 02:47 to 02:56 or something like a 02:57 big air pump Okay Yes. Thanks, guys. I hope you had a great trading week. And if not, I hope you have a great trading week next week. And have a great weekend. Thanks all to you on Monday..

The total of words is 5395. Which may result in 4 chunks of text for the first title. Now, the prompt used for every summary is as follows:

```
[ ]: prompt = f"The following text is a conversation from a trading transcript:␣
      ↪\"{text}\". Summarize the most important trading-related information in four␣
      ↪concise bullet points. After the bullet points, provide a brief paragraph␣
      ↪suggesting potential actions and key ideas based on the summarized points,␣
      ↪all in the style of Warren Buffet. Always penalize redundancy. Write only␣
      ↪sentences with full ideas"
```

Then, the results will be written to the .txt as 4 bullet points followed by a conclusion for every chunk of processed text:

> Title: $1.3M gain on $TVIX.
>
> - The trader shorted Twick yesterday and held some overnight.
> - The trader noticed Twick decoupling from the indices and built lower highs on the 5 minute chart.
> - The trader added size when Twick broke its multi-day range and went down over 30% in an hour or two.
> - The trader covered some of the position in the late fee market.

Based on the summarized points, it is clear that the trader was able to identify a potential opportunity in Twick and capitalize on it. The trader was able to recognize the relative weakness in Twick and take advantage of the multi-day range break. This suggests that traders should always be on the lookout for potential opportunities and be willing to take calculated risks in order to maximize their profits. Warren Buffet's famous quote, "Risk comes .

- Walter experienced a million dollar loss over the weekend.
- The VIX was showing relative strength to the size.
- Walter noticed a change in the VIX the day before, which was the first time in two weeks.
- Walter shorted and added to his position as the VIX continued to decline.

Based on the summarized points, it is clear that Walter was able to take advantage of a changing market to make a profit. He was able to identify a potential opportunity and act on it quickly. This is a key idea that Warren Buffet has always been known for; being able to recognize a good opportunity and taking action quickly. By doing this, Walter was able to make a profit despite the losses he experienced over the weekend. This is a great example of how being aware of the market and taking advantage of changes can be .

- Veeva is struggling and may present a big short opportunity.
- Tilray, TVI, and BLPH are all stocks to watch.
- Many stocks may have 50-200% bounce potential.
- PLPH is building higher lows, so it may be a good idea to have a tight automated stop.

Based on the summarized points, it appears that there are some stocks that may be worth watching, particularly Veeva and Tilray, which may present a big short opportunity. Additionally, there are many stocks that may have a 50-200% bounce potential, so it may be a good idea to keep an eye on those as well. Finally, PLPH is building higher lows, so it may be a good idea to have a tight automated stop in order to protect against any potential losses. Warren Buffet would .

- BLPH is intriguing and has the potential to go to 50% from the lows.
- The speaker is considering shorting it instead of buying it.
- The speaker is focusing on TVA and has a decent size position.
- The speaker is not looking to lower their average price.

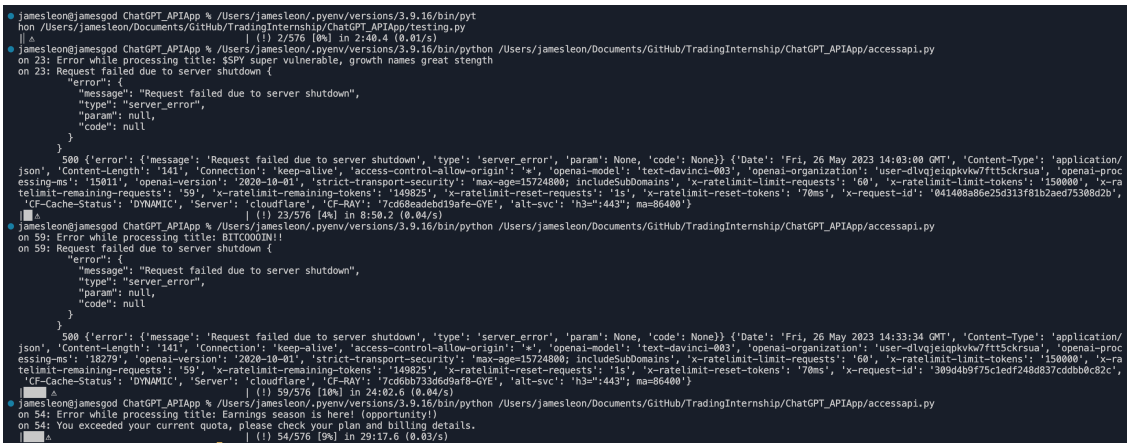
Based on the summarized points, it appears that the speaker is looking to make a profit off of BLPH by shorting it instead of buying it. This is a risky move, but one that could potentially pay off if the stock goes to 50% from the lows. The speaker is also focusing on TVA, which appears to have a big grip and could be a good opportunity for a profit. The speaker also has a decent size position and is not looking to lower their average price. Overall, the speaker appears to be taking

Just as expected the resulting text supposes four chunks of text resulting from the first conversation.

However, the conclusions provided are mostly incomplete. The solutions to these are discussed in the following *What's Next* section.

1.2.4 What's Next

Update the alive_bar The alive_bar is always taking the whole dataset for its progress indications instead of the remaining values. This is a problem as it difficults to know the true remaining conversations to be processed



Get a paid API_key Though the Models are not expensive to use ($0.02 per token~word). This code was created using the free $5 provided by OpenAI. That's the reason why the last error is not because of a connection interruption but an out of quota error.

Reformat the results The results are being saved in the same format as they were received. Thus, for better visualization and comprehension, improvements can be made on the output

1.2.5 The tutorial

Dependencies

- Python 3.9.16
- pip install openai
- optional: npm install openai
- a valid OpenAI API key
- other libraries

Notes on this version of the code

> The present version of the code won't need more help than the provided to this point. The present tutorial has been made for Ney use on how to create a full app using the OpenAI API.

2 Roadmap

- Crash course on OpenAI and ChatGPT → Environment setup → Use the serverless API → Integrate with platforms → Create full real solution apps

3 Tech Stack of a ChatGPT app

- Front-end (Like .bubble): The visuals
- Back-end (Like .bubble): The code
- Wrapper API (Azure funcs): Connection to Open AI
- OpenAI API (API): Communication with the server
- GPT-X Model (API): Response to the prompt, then goes all the way up again.

From the wrapper down it will be always the same, but the back and the front depend on the environment.

4 Crash Course on OpenAI and ChatGPT

4.0.1 Using ChatGPT to answer problems

- The basic structure of an interaction with ChatGPT is prompt → response and back and forth.
- The prompt: Needs an instruction, tone (politely, to a certain person on a certain situation, etc), context, detail, etc.

4.0.2 OpenAI

- Research and deployment company that manage models using APIs that you can deploy and integrate in your application (for a very cheap value to cost proportion).

4.0.3 OpenAI Models

- GPT: Natural Language tasks with prompt
  - DaVinci-003: The most capable and updated model. The highest quality with 4000 token size. The relative cost versus other models is high. Use it for everything.
  - Curie-001: Great for nuanced tasks like sentiment analysis, language translation, and complex classification (i.e. right vs wrong equations). Use it for specific tasks.
  - Babbage-001: For straightforward tasks like simple classification and semantic searching.
  - Ada-001: For simple tasks like parsing text, address correction, and certain classification tasks.
  - The takeaway on selecting a model: Use DaVinci for most business solutions.
- Codex: Natural Language to code
- DALL-E: Create and edit original images

4.0.4 DaVinci Capabilities

- Classification: Statements into categories
- Generation: Create something from an instruction
- Conversation: Keep the context to follow up
- Transformation: From a text
- Completion: Fill the blanks

4.0.5 Prompt Optimization

- Concepts

Prompt: What you give the model || **Response**: What you get back from the model

- Models are heavily depended on reacting to good prompts: Garbage in = Garbage out

Tokens: the currency of ChatGPT: The model process text by breaking it down into smaller units called tokens

1 token ~= 4 characters

75 words ~= 100 tokens

Total Tokens = prompt text + completed text

Max for DaVinci = 4000 tokens ~= 3000 words

$0.02 per 1k tokens

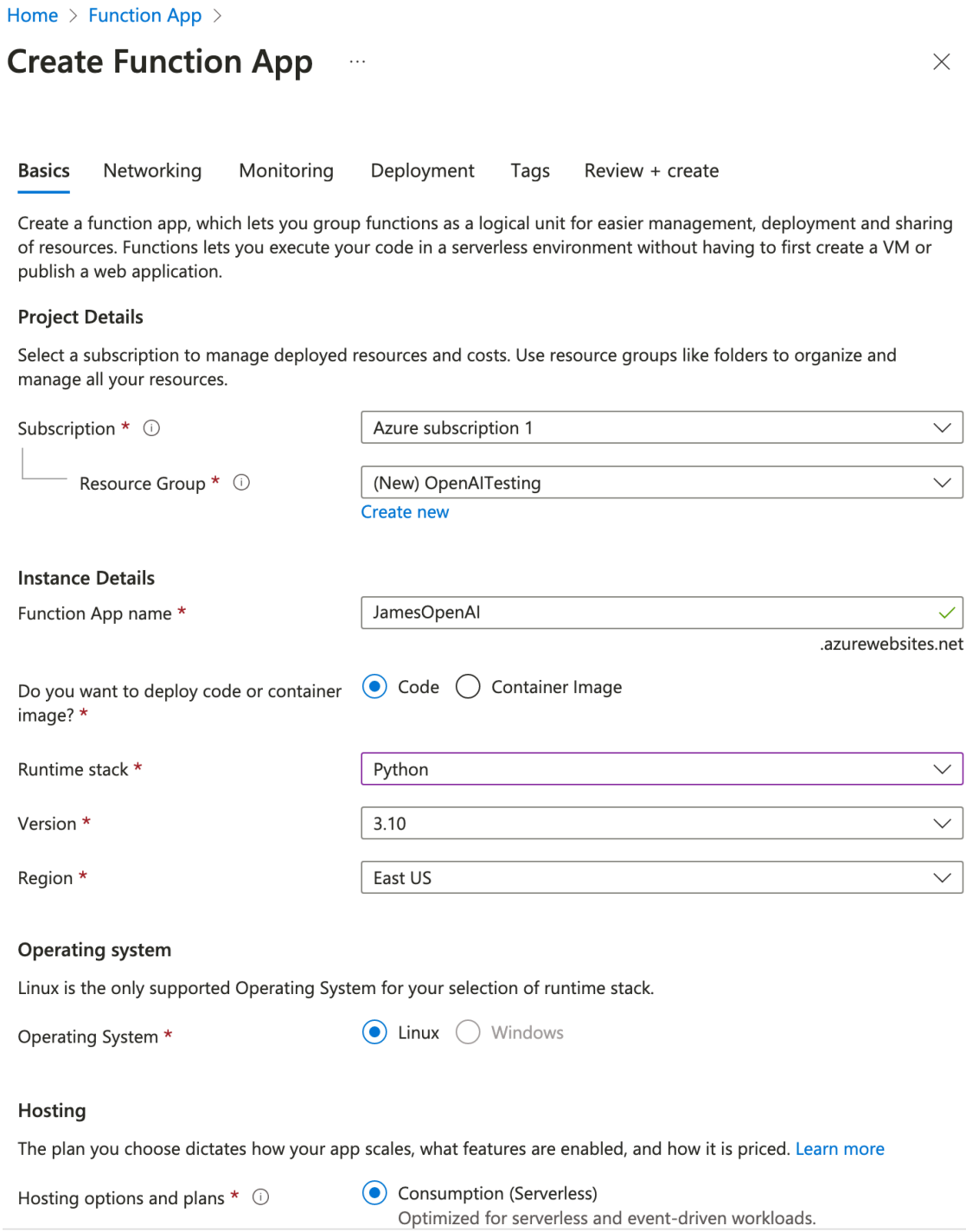
[Free trial of $18 ~= 900k tokens ~= 675k words] for developing and testing

Temperature: From 0 to 1, sets the risk that the model uses for the response. 1 means more creative, 0 means a well-defined answer. [For a business solution you should go for 0]

- Be very instructional in what you want, and as specific as you want
- Provide examples with good quality data: Show and tell the model how to respond
- Don't rely on factual responses

5 Environment Setup

- **Get the API from**: API keys - OpenAI API
- **Get an Azure account from**: # Shorten the text to fit into one assistant response "Create Your Azure Free Account Today | Microsoft Azure
  - As a recommendation, for developing and personal use 'Pay as you go' is the right match. However, some organizations and universities provide a free $200 credit for Microsoft Azure services.
- On the Microsoft Azure workspace, search for Function App on the search bar → Create. For categorization purposes, create a resource group named 'OpenAITesting'.



- On Networking make sure it is set to public. Everything else can be left as it is. However, you can connect the service to a github account for deployment.
- Finally, go to Review+create and create.

Home > Function App >

Create Function App ...

Basics | Networking | Monitoring | Deployment | Tags | **Review + create**

Summary

Function App
by Microsoft

Details

Subscription	0d3bbd21-035f-461d-beed-11603fb080ba
Resource Group	OpenAITesting
Name	JamesOpenAI
Runtime stack	Python 3.10

Hosting

Plan (New)

Hosting options and plans	Consumption (Serverless)
Name	ASP-OpenAITesting-98e9
Operating System	Linux
Region	East US
SKU	Dynamic

Monitoring (New)

Application Insights	Enabled
Name	JamesOpenAI
Region	East US

Deployment

Continuous deployment	Not enabled / Set up after app creation

- The azure function would ping the OpenAI server.
- **Make sure Python is installed and updated to 3.9.16 on your PC or virtual env (pyenv does the work, other versions of python can't handle Azure functions).**
- Install VSCode and set a folder for the work environment.
  - Check the interpreter is 3.9.16
  - Install the azure functions extension (ms-azuretools.vscode-azurefunctions)
  - Sign In to azure with the same account as before
- For connection with power apps and power automate, connect your microsoft 365 account. (Using a temporary email you can access to the free trials on most of these).
- All set!"