In-context_stance_detection

November 28, 2023

1 Stance Detection: Classification With In-context Learners (GPT-4)

This tutorial demonstrates stance detection using GPT-4. In-context learning is a fast moving field and this is a relatively simple implementation. Consider this a starting point rather than a comprehensive guide. This guide will use the openai library to gain access to GPT-4 and utilize their cloud computing services. OpenAI charges based on usage and this can be fairly expensive. However, new accounts are given a small stipend to play with. This should be enough to run through this tutorial. OpenAI Documentation

Requirements:

1. Basic python skills
2. An OpenAI account and API key

[ ]:
```
import pandas as pd
from openai import OpenAI
from sklearn.metrics import matthews_corrcoef
client = OpenAI(api_key = 'Insert your API key here')
```

For this example we will use a data set consists of tweets about President Trump that are manually labeled 1: approve, 0: not approve.

[2]:
```
test_df = pd.read_csv('https://raw.githubusercontent.com/XXXXX/
↪stance_detection_tutorials/main/data/gpt3_test.csv')
test_df.head()
```

[2]:

	text	stance	labels
0	I'd like #TedCruz to talk more policies that d...	none	0.0
1	"the past" and biden references obama years as...	none	0.0
2	Quando mi prende lo sconforto penso che qualcu...	none	0.0
3	@USER @USER think about it in ny where most a...	none	0.0
4	#GOPDebate think what #Trump2016 is saying tha...	none	0.0

Classification with in-context learning works by treating classification as a next word prediction task. We supply the model with a prompt that consists of a description of the task and the document to be labeled. The model then predicts what the label is.

Our first task is to generate a system message and prompt. The system message helps define the behavior of the language model and the prompt describes the task and presents it with the document to classify.

[ ]:
```
system_message = "You are a text classifier and are only allowed to respond with␣
 ↪1 or 0."

user_message = """You are a classifier that determines if the author of a text␣
 ↪supports Donald Trump.
I will post text about Trump and if the author is more likely to support Trump␣
 ↪than not return 1.
If it is not more likely that the author supports Trump return 0. Do not explain␣
 ↪the classification or say anything else.
You are only allowed to respond with 1 or 0. Here is the text.\nText: {}"""
```

To classify the data, we then pass each prompt to GPT-4 through the OpenAI API. We also set the temperature to zero for more reproducible results. Because I want the model to only respond with a 1 or 0, I also set the number of max tokens to 1. Finally, the logit-bias parameter can be used to bias the model towards only generating the requested labels. This is done by passing a dictionary with token IDs as keys and the bias as values. In this example, the token IDs 16 and 15 correspond to 1 and 0 respectively and I give them a bias of 100. You can look up token IDs online here: https://platform.openai.com/tokenizer

[ ]:
```
%%time
# I create an empty list that will hold document labels
labels = []
# Loop through the text passing each document to the model with the prompt and␣
 ↪system message.
for text in test_df['text']:
    # query API
    res = client.chat.completions.create(
        model = 'gpt-4',
        messages = [
            {'role':'system', 'content': system_message},
            {'role':'user', 'content': user_message.format(text)}
        ],
        max_tokens = 1,
        temperature = 0
        logit_bias = {16:100, 15:100}
    )
    labels.append(res.choices[0].message.content)
```

Now that we have our labels, we can simply add them to our data frame an analyze the results

[ ]:
```
test_df['GPT4_labs'] = labels

matthews_corrcoef(test_df['labels'], test_df['GPT4_labs'])
```