Classify verses by sentiment

July 24, 2023

What is the sentiment for the verses of the crosswalks in Madrid using OpenAI?

Classify verses by the associated sentiment using OpenAI.

Setup

[ ]:
```
!pip install openai
```

[ ]:
```
!pip install python-dotenv
```

[ ]:
```
!pip install pandas
```

[ ]:
```
!pip install tiktoken
```

Read the CSV file with all 'versos al paso'

Load CSV file, show columns and other stats

[5]:
```
import pandas as pd

versos_al_paso_file_path = './input/versosalpaso.csv'
versos_al_paso = pd.read_csv(versos_al_paso_file_path, sep="|",␣
 ↪encoding='utf-8')
versos_al_paso.columns
```

[5]:
```
Index(['id', 'latitud', 'longitud', 'autor', 'barrio', 'verso', 'direccion'],
dtype='object')
```

[6]:
```
import numpy as np
memory_in_use = np.sum(versos_al_paso.memory_usage(True, True).values)

no_of_rows = len(versos_al_paso.index)
a_verse = versos_al_paso.loc[0: 0].verso.values[0]

[f'{memory_in_use} bytes', f'{no_of_rows} rows', a_verse]
```

[6]:
```
['581256 bytes',
 '1100 rows',
 'Quizá el secreto de la vida tan solo consista En tener un lugar al que
regresar']
```

[7]:
```
spanish_sentences = versos_al_paso.verso.tolist()
```

Classifying with OpenAI

Read the secrect key

[8]:
```
import openai
import os

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv())

openai.api_key = os.getenv('OPENAI_API_KEY')
```

The Completions API and the '**text-davinci-003**' model are used as can see in the Jupyter Notebook 'Iterative Prompt using Chat-GPT'.

Let's copy the required elements

[9]:
```
def get_completion(prompt: str, model: str = "text-davinci-003") -> str:
    response = openai.Completion.create(
        model=model,
        prompt=prompt,
        temperature=0,
        max_tokens=2000
    )
    return response.choices[0].text
```

The model used has a rate limit as free trial user.

In this case rate limits are:

Model	RPM	TPM
TEXT		
text-davinci-003	60	150.000

Under these limits there should be no problem in making requests for the 1100 phrases that make up the 'versos al paso'.

Let's copy the code for batch processing according to the token limit in each request as explored in Jupyter Notebook 'Explore tokenization' and adding the prompt to use from the Jupyter Notebook above.

[10]:
```
import tiktoken

def num_tokens_from_string(string: str, model_name: str = "text-davinci-003")␣
 ↪-> int:
    enc = tiktoken.encoding_for_model(model_name)
    assert enc.decode(enc.encode(string)) == string
    num_tokens = len(enc.encode(string))
    return num_tokens
```

[11]:
```
def split_list(sentences: list, tokens_limit: int = 1800) -> list:
    no_of_sentences = len(sentences)

    from_pos = 0
    splited_list = []
    while True:
        sentence = sentences[from_pos:from_pos+1][0]

        a_token_intent = num_tokens_from_string(f'{sentence}')
        sentences_for_request = (tokens_limit // a_token_intent) + 1

        to_pos= from_pos + sentences_for_request

        while True:
            sentences_to_request = sentences[from_pos:to_pos]
            no_of_tokens = num_tokens_from_string(f'{sentences_to_request}')

            if tokens_limit < no_of_tokens:
                while tokens_limit < no_of_tokens:
                    no_of_tokens -=␣
↪num_tokens_from_string(sentences_to_request[-1])
                    sentences_to_request.pop()
                break
            else:
                to_pos+= 1
                if to_pos >= no_of_sentences:
                    break

        splited_list.append(sentences_to_request)

        from_pos += len(sentences_to_request)
        if from_pos >= no_of_sentences:
            break

    return splited_list
```

[12]:
```
import json
import re

def fix_xinvalid(m):
    return chr(int(m.group(1), 16))

def fix_escape(str):
    xinvalid = re.compile(r'\\x([0-9a-fA-F]{2})')
    return xinvalid.sub(fix_xinvalid, str)

def get_sentiments(splitted_list: list) -> json:
    sentiments_json = {}
    for sentences in splitted_list:
        prompt = f"""
        What is the sentiment of the following Spanish sentences,
        which is delimited with triple backticks?

        Classify sentences according to their sentiment.

        The sentiment will be as a single word, \
        either "positive" or "neutral" or "negative".

        Give your answer as JSON, where the key is the sentiment and the value␣
 ↪is the list.

        Review text: '''{sentences}'''
        """
        response = get_completion(prompt)

        try:
            json_sentiments = json.loads(fix_escape(response))
            for key, value in json_sentiments.items():
                if key in sentiments_json:
                    sentiments_json[key] += value
                else:
                    sentiments_json[key] = value

        except Exception as e:
            if 'error' in sentiments_json:
                sentiments_json['error'] += sentences
            else:
                sentiments_json['error'] = sentences

    return sentiments_json
```

Save the result as a CSV file

[13]:
```
versos_al_paso_sentiment = versos_al_paso.copy()
versos_al_paso_sentiment['sentiment'] = ''
```

[14]:
```
def update_sentiment(sentiments_json: json, versos_pd: pd.core.frame.DataFrame):
    for key, value in sentiments_json.items():
        if 'error' != key:
            for sentence in value:
                idx = versos_pd.verso.eq(sentence)
                versos_pd.loc[idx, 'sentiment'] = key
```

[ ]:
```
splitted_list = split_list(spanish_sentences)
sentiments_json = get_sentiments(splitted_list)
update_sentiment(sentiments_json, versos_al_paso_sentiment)

versos_al_paso_sentiment_file_path = './output/
 ↪versosalpaso_sentiment_text-davinci-003.csv'
versos_al_paso_sentiment.to_csv(versos_al_paso_sentiment_file_path, sep=';',␣
 ↪encoding='utf-8')

if 'error' in sentiments_json:
    assert [] == sentiments_json['error']
```