Explore tokenization

July 7, 2023

Explore tokenization

In the notebook 'Iterative Prompt using Chat-GPT' the model 'text-davinci-003' was selected for the sentiment clasification of the spanish sentences.

The model used defines that requests can use up to 4000 tokens shared between prompt and completion.

Let's explore how tokens can be counted programmatically count tokens.

Setup

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

Read the CSV file with all 'versos al paso'

Load CSV file, show columns and other stats

[2]:
```
import pandas as pd
versos_al_paso_file_path = './input/versosalpaso.csv'
versos_al_paso = pd.read_csv(versos_al_paso_file_path,sep="|")
spanish_sentences = versos_al_paso.verso.tolist()
```

OpenAI tokenization

The following prompt was defined in the 'Iterative Prompt using Chat-GPT' notebook,

```
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: '''{spanish_sentences}'''
"""
```

as well as the following request

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

The model used defines that requests can use up to 4000 tokens shared between prompt and completion. It is necessary that the question does not exceed a maximum of 50% of the available tokens for which tokens must be counted.

Using OpenAI's interactive Tokenizer tool, the following prompt with an empty list to review has **85 tokens** and 343 characters.

[3]:
```
def get_prompt(spanish_sentences: list = []) -> str:
    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: '''{spanish_sentences}'''
    """
    return prompt

print(get_prompt())
```

```
    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: '''[]'''
```

Let's take a few sentences to test tokenisation.

[4]:
```
split_no = 25
some_spanish_sentences = spanish_sentences[0:split_no]
print(some_spanish_sentences)
```

```
['Quizá el secreto de la vida tan solo consista En tener un lugar al que
regresar', 'Cuando andamos cabeza abajo entonces es cuando empezamos a vivir de
nuevo.', 'El bosque es el primigenio nudo iniciático, el hogar de los hombres
perdidos.', 'Mi garganta es una gruta deslizante de acentos. La forma musical
para sanar.', 'Hoy somos más viejos que nunca, piensas; pero nunca serás ya tan
joven como hoy.', 'Dudarás: ¿Qué razón en tocar sus lunares? Pensará: Si lloro,
diré \\', 'Quien olvida abandona, mata, entierra la memoria. Por ello, nunca
olvido', 'No puede reparase con las manos una tela de araña. No hay dedos tan
exquisitos', 'La política tiene colgado el cartel de rebajas', 'Queda de ti un
vestido de aguas a la deriva, queda de ti tu nombre que no digo', 'Yo seré
mimoso, pero tú serás mi musa', 'Soy origen y descendencia, soy luz, sombra y
herencia, soy mujer soy resiliencia', 'En memoria de las auténticas cebras, las
que en el Mioceno galopaban por Madrid', 'Nacer fue algo oblicuo. No se incendia
el bosque sino el interior de cada árbol.', 'Peladitos ven TV y quieren ser así,
están aprendiendo a matar antes que a vivir', 'Donde la distancia entre la
tierra y nuestras manos era la medida de todas las cosas', 'La voz que tripula
este barco navega sin olas.', 'Por ti, que eres cuerpo sostenido en un alma
bemol sale Sol, brilla La, sale Sol', 'Amor es llegar en el momento oportuno y
ser del tamaño del hueco vacío del otro.', 'Sola, inmersa en el ruido de la
ciudad, oigo el silencio de la vida arrostrándose', 'Somos préstamos de calor',
'Te espero en la orilla del sueño de siempre', 'Nuestros castillos de arena
están a salvo: se ha secado el mar de dudas', 'Y tú, ¿a cuál de todas tus
cicatrices le escribes?', 'Que el teu camí sigui llarg, i ben aviat entenguis
que la ciutat és casa.']
```

, **688 tokens** and 1815 characters for the above set of sentences. The full request would have **772 tokens** and 2156 characters.

Let's use OpenAI's Tiktoken library to count tokens as shown in OpenAI Cookbook.

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

[3]:
```
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
```

[8]:
```
empty_request = get_prompt()
sentences_str = f"{some_spanish_sentences}"
```

```
full_request = get_prompt(some_spanish_sentences)
print(f"""
    empty request {num_tokens_from_string(empty_request)} tokens␣
 ↪{len(empty_request)} characters
    sentences: {num_tokens_from_string(sentences_str)} tokens␣
 ↪{len(sentences_str)} characters
    request: {num_tokens_from_string(full_request)} tokens {len(full_request)}␣
 ↪characters
""")
```

```
    empty request 90 tokens 373 characters
    sentences: 688 tokens 1815 characters
    request: 777 tokens 2186 characters
```

The expected completion will be a JSON object as a string as following

`{"positive": [], "neutral": [], "negative": []}`

Let's count its tokens

[9]:
```
expected_completion = '{"positive": [], "neutral": [], "negative": []}'
print(f"""
    {num_tokens_from_string(expected_completion)} tokens␣
 ↪{len(expected_completion)} characters
""")
```

```
    15 tokens 47 characters
```

One empty request and one empty completion are 105 tokens.

Let's do some maths

tokens available: 4,000 - 105 = 3,895

maximum number of tokens to be reviewed: 3,895 / 2 = 1,947.5

percent: 1,947.5 / 4,000 = 48,69%

Based on these results, this method will allow us to group the sentences in a way that does not exceed 48% of the tokens allowed in the request. A **more conservative 45% or 1,800 tokens** limit may be a better idea.

Split verses to request sentiment

[7]:
```
import unittest
import numpy

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


splitted_list = split_list(spanish_sentences)
requested_sentences = list(numpy.concatenate(splitted_list).flat)

tc = unittest.TestCase()
tc.assertListEqual(requested_sentences, spanish_sentences)
```