Maestría en Inteligencia Artificial Aplicada

Curso: Procesamiento de Lenguaje Natural (NLP)

Tecnológico de Monterrey

Prof Luis Eduardo Falcón Morales

Adtividad de la Semana 02

Introducción al procesamiento de texto.

En esta actividad deberás utilizar los datos del siguiente archivo que se encuentra en Canvas:

MNA_NLP_semana_02_Actividad_datos.txt

El archivo contiene comentarios en inglés sobre servicios de comida de la página de Yelp: https://www.yelp.com/ .

Son mil comentarios y forman parte del conjunto de datos que se encuentra en el Machine Learning Repository de la UCI, llamado "Sentiment Labelled Sentences": https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences#

Parte 1. Cargamos los datos.

Cargar los datos del archivo indicado y obtener una lista de longitud de 1000 strings/comentarios.

Por el momento solamente requerimos las bibliotecas de Numpy y re, para el manejo de los arreglos y de las expresiones regulares en Python.

En particular, no necesitarás en esta actividad la biblioteca de Pandas.

NOTA: En esta actividad no debes importar nada más, con estas dos bibliotecas será *suficiente*.

```
import numpy as np    # importamos Numpy para el manejo de los arreglos.
import re             # importamos re para el manejo de las expresiones regulares.
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Ejecuta las siguientes instrucciones para cargar la información del achivo dado:

with open('/content/drive/MyDrive/Natural Language Processing/Modulo1 - Corpus y expresiones regulares/2.3 Actividad 1  Procesamiento de texto/MNA_NLP_sem
          mode='r',     # abrimos el archivo en modo lectura.
          ) as f:
    docs = f.readlines()    # separamos cada comentario por líneas

f.close()  # ya que tenemos la información en la variable docs, cerramos el archivo
```

```
type(docs) == list   # Verifica que tu variable "docs" es una lista
```

```
True
```

```
len(docs)==1000  # verifica que la longitud de "docs" es de mil comentarios.
```

```
True
```

```
docs[0:10]     # observa algunos de los primeros comentarios
```

```
['Wow... Loved this place.\n',
 'Crust is not good.\n',
 'Not tasty and the texture was just nasty.\n',
 'Stopped by during the late May bank holiday off Rick Steve recommendation and loved it.\n',
 'The selection on the menu was great and so were the prices.\n',
 'Now I am getting angry and I want my damn pho.\n',
 "Honeslty it didn't taste THAT fresh.)\n",
 'The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer.\n',
 'The fries were great too.\n',
 'A great touch.\n']
```

Parte 2: sección de preguntas (regex).

Instrucciones:

A continuación deberás contestar cada una de las preguntas que te piden usando expresiones regulares (regex).

Por el momento no hay restricción en cuanto al número de líneas de código que agregues, pero trata de incluir las mínimas posibles.

- **Pregunta 1.**

Busca y elimina todos los saltos de línea '\n' que se encuentran al final de cada comentario.

Una vez finalizado, imprime los primeros 10 comentarios del resultado obtenido.

```
deleted_row_jump= [row.replace('\n', '') for row in docs]
deleted_row_jump [0:10]
```

```
['Wow... Loved this place.',
 'Crust is not good.',
 'Not tasty and the texture was just nasty.',
 'Stopped by during the late May bank holiday off Rick Steve recommendation and loved it.',
 'The selection on the menu was great and so were the prices.',
 'Now I am getting angry and I want my damn pho.',
 "Honeslty it didn't taste THAT fresh.)",
 'The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer.',
 'The fries were great too.',
 'A great touch.']
```

- **Pregunta 2.**

Busca e imprime todas las palabras que terminan con dos o más signos de admiración seguidos, por ejemplo "!!!".

Debes imprimir tanto la palabra como la totalidad de signos de admiración que le siguen.

Indica cuántos resultados obtuviste.

```
pattern= r'\b\w+\b!{2,}'
palabras1=[]
for row in docs:
  found1=re.findall(pattern,row)
  palabras1.extend(found1)

print(palabras1)
print (len(palabras1))
```

```
['Firehouse!!!!!', 'APPETIZERS!!!', 'amazing!!!', 'buffet!!!', 'good!!', 'it!!!!', 'DELICIOUS!!', 'amazing!!', 'shawarrrrrrma!!!!!!', 'yucky!!!', 'ste
26
```

- **Pregunta 3.**

Busca e imprime todas las palabras que están escritas totalmente en mayúsculas. Cada coincidencia debe ser una sola palabra.

Indica cuántas palabras encontraste.

```
pattern2= r'\b[A-Z]+\b'
palabras2 = []
for row in docs:
  found2=re.findall(pattern2,row)
  palabras2.extend(found2)

print(palabras2)
print (len(palabras2))
```

```
['I', 'I', 'THAT', 'A', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'APPETIZERS', 'A', 'I', 'I', 'I', 'I', 'I', 'I', 'I', 'WILL', 'NEV
455
```

- **Pregunta 4.**

Busca e imprime los comentarios en donde todos los caracteres alfabéticos (letras) están en mayúsculas.

Cada coincidencia encontrada debe ser todo el comentario/enunciado.

Indica cuántos resultados obtuviste.

```
def todas_mayusculas(texto):
    palabras = texto.split()
    return all(palabra.isupper() for palabra in palabras)
comentarios_en_mayusculas = []

for comentario in docs:
    if todas_mayusculas(comentario):
      comentarios_en_mayusculas.append(comentario)

print(comentarios_en_mayusculas)
print(len(comentarios_en_mayusculas))
```

```
['DELICIOUS!!\n', 'WILL NEVER EVER GO BACK AND HAVE TOLD MANY PEOPLE WHAT HAD HAPPENED.\n', 'TOTAL WASTE OF TIME.\n', 'AVOID THIS ESTABLISHMENT!\n']
4
```

- **Pregunta 5.**

Busca e imprime todas las palabras que tengan una vocal acentuada, del tipo á, é, í, ó, ú.

Indica cuántos resultados obtuviste.

```
pattern3= r'\b\w*[áéíóú]\w*\b'
palabras3 = []
for row in docs:
  found3=re.findall(pattern3,row)
  palabras3.extend(found3)

print(palabras3)
print (len(palabras3))
```

```
['fiancé', 'Café', 'puréed']
3
```

- **Pregunta 6.**

Busca e imprime todas las cantidades numéricas monetarias, enteras o con decimales, que inician con el símbolo $.

Indica cuántos resultados obtuviste.

```
pattern4= r'\$\d+\.\d+'
palabras4 = []
for row in docs:
  found4=re.findall(pattern4,row)
  palabras4.extend(found4)

print(palabras4)
print (len(palabras4))
```

```
['$4.00', '$7.85', '$11.99']
3
```

- **Pregunta 7.**

Busca e imprime todas las palabras que sean variantes de la palabra "love", sin importar si incluyen mayúsculas o minúsculas, o la manera en que esté conjugada o alguna otra variación que se haga con dicha palabra.

Indica cuántos resultados obtuviste.

```
pattern5= r'\b[Ll][Oo][vV]\w*\b'
palabras5 = []
for row in docs:
  found5=re.findall(pattern5,row)
  palabras5.extend(found5)

print(palabras5)
print (len(palabras5))
```

```
['Loved', 'loved', 'Loved', 'love', 'loves', 'LOVED', 'lovers', 'loving', 'love', 'lovers', 'Love', 'loved', 'loved', 'love', 'love', 'love', 'loved',
36
```

- **Pregunta 8.**

Busca e imprime todas las palabras, variantes de "so" y "good", que tengan dos o más "o" en "so" y 3 o más "o" en good.

Indica cuántas encontraste.

```
pattern6= r'\bso*o{2,}'
pattern6_1=r'\bgo*o{3,}d'
palabras6 = []
for row in docs:
  found6=re.findall(pattern6,row)
  palabras6.extend(found6)
  found6_1=re.findall(pattern6_1,row)
  palabras6.extend(found6_1)

print(palabras6)
print (len(palabras6))
```

```
['soo', 'soooo', 'soo', 'soo', 'goood', 'soo', 'soo', 'soooooo', 'soo', 'soo', 'soooo', 'soo', 'soo']
13
```

- **Pregunta 9.**

Busca e imprime todas las palabras que tengan una longitud mayor estrictamente a 10 caracteres alfabéticos.

No se consideran los signos de puntuación o caracteres especiales en la longitud de estas cadenas, solo caracteres alfabéticos en mayúsculas o minúsculas.

Indica la cantidad de palabras encontradas.

```
pattern7= r'\b[A-Za-z]{11,}\b'

palabras7 = []
for row in docs:
  found7=re.findall(pattern7,row)
  palabras7.extend(found7)


print(palabras7)
print (len(palabras7))
```

```
['recommendation', 'recommended', 'overwhelmed', 'inexpensive', 'establishment', 'imaginative', 'opportunity', 'experiencing', 'underwhelming', 'relat
141
```

- **Pregunta 10.**

Busca e imprime todas las palabras que inician con una letra mayúscula y terminan con una minúscula, pero que además no sea la primera palabra del comentario/string.

Indica la cantidad de resultados obtenidos.

```
pattern8= r'(?<!^)\b[A-Z][a-z]*[a-z]\b'

palabras8 = []
for row in docs:
  found8=re.findall(pattern8,row)
  palabras8.extend(found8)


print(palabras8)
print (len(palabras8))
```

```
['Loved', 'May', 'Rick', 'Steve', 'Cape', 'Cod', 'This', 'Vegas', 'Burrittos', 'Blah', 'The', 'The', 'They', 'This', 'Mexican', 'Took', 'Luke', 'Our',
517
```

- **Pregunta 11.**

Busca e imprime la secuencia de dos o más palabras que están separadas por un guion, "-", sin que tengan espacios en blanco entre ellas.

Por ejemplo "Go-Kart" sería válido, pero "Go -Kart" o "Go - Kart" no lo serían.

Indica la cantidad de resultados obtenidos.

```
pattern9= r'\b[A-Za-z]+(?:-[A-Za-z]+)+\b'

palabras9 = []
for row in docs:
  found9=re.findall(pattern9,row)
  palabras9.extend(found9)


print(palabras9)
print (len(palabras9))
```

```
['flat-lined', 'hands-down', 'must-stop', 'sub-par', 'Service-check', 'in-house', 'been-stepped-in-and-tracked-everywhere', 'multi-grain', 'to-go', 'n
19
```

- **Pregunta 12.**

Busca e imprime todas las palabras que terminan en "ing" o "ed".

Indica la cantidad de palabras que encontraste de cada una.

```
pattern10= r'\b\w+ing\b'
pattern10_1= r'\b\w+ed\b'

palabras10 = []
for row in docs:
  found10=re.findall(pattern10,row)
  palabras10.extend(found10)
  found10_1=re.findall(pattern10_1,row)
  palabras10.extend(found10_1)


print(palabras10)
print (len(palabras10))
```

```
['Loved', 'during', 'Stopped', 'loved', 'getting', 'being', 'being', 'ended', 'overpriced', 'tried', 'disgusted', 'shocked', 'recommended', 'amazing',
614
```

Parte 3. Proceso de limpieza.

- **Pregunta 13.**

Ahora realiza un proceso de limpieza del corpus que incluya los siguientes procesos:

- Solo se deben considerar caracteres alfabéticos. Es decir, se eliminan todos los signos de puntuación y caracteres especiales.
- Todos los caracteres alfabéticos se transforman a minúsculas.
- Se deben eliminar todos los espacios en blanco adicionales que se puedan encontrar en cada comentario.

Al finalizar dicho proceso de limpieza, imprime el resultado de los primeros 10 comentarios resultantes.

```
no_az_pattern = r'[^a-zA-Z\s]'

new_docs = []



for row in docs:
    new_row = re.sub(no_az_pattern, '', row)
    new_row = new_row.lower()
    new_row = ' '.join(new_row.split())
    new_docs.append(new_row)

new_docs[1:10]
```

```
['crust is not good',
 'not tasty and the texture was just nasty',
 'stopped by during the late may bank holiday off rick steve recommendation and loved it',
 'the selection on the menu was great and so were the prices',
 'now i am getting angry and i want my damn pho',
 'honeslty it didnt taste that fresh',
 'the potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer',
 'the fries were great too',
 'a great touch']
```

- **Pregunta 14.**

Con el resultado de la limpieza obtenido en la pregunta anterior, realiza ahora un proceso de tokenización por palabras del corpus.

Es decir, al final de este proceso de tokenización, debes tener como resultado una lista de listas, donde cada comentario estará tokenizado por palabras.

Al terminar calcula el total de tokens obtenido en todo el corpus.

```
new_docs_tok= []

for r in new_docs:
    tokens = r.split()
    new_docs_tok.append(tokens)


sum(len(tokens) for tokens in new_docs_tok)
```

```
 10777
```

- **Pregunta 15.**

Finalmente, en este ejercicio definiremos nuestro conjunto de palabras "stopwords", las cuales deberás eliminar de todo el corpus.

Recuerda que ejemplos de stopwords son artículos, adverbios, conectivos, etcétera, que tienen frecuencias de aparición muy altas en cualquier documento, pero que no brindan mucho significado en cuanto al significado de un enunciado.

Con base a la lista de stopwords que se te proporciona, realiza un proceso de limpieza eliminando todas estas palabras del corpus obtenido en el ejercicio anterior.

Obtener cuántos tokens/palabras quedan finalmente en todo el corpus.

Obtener cuántos de estos tokens/palabras son diferentes, es decir, cuántos tokens únicos tendrá lo que llamaremos más adelante nuestro vocabulario.

```
# Considera la siguiente lista como tu conjunto de stopwords:
mis_stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',



new_docs_tok_sin_stopwords = []

for comentario_tokens in new_docs_tok:
    tokens_sin_stopwords = [w for w in comentario_tokens if w.lower() not in mis_stopwords]
    new_docs_tok_sin_stopwords.extend(tokens_sin_stopwords)



print("palabras en todo el corpus: ", len(new_docs_tok_sin_stopwords))
```

```
 palabras en todo el corpus:  5776
```

```
print("palabras en todo el corpus unicas : ", len(set(new_docs_tok_sin_stopwords)))
```

```
 palabras en todo el corpus unicas :  1941
```

- **Comentarios**

Incluye finalmente tus comentarios de la actividad.

<< Esta actividad nos proporcionó experiencia práctica en el preprocesamiento de texto utilizando Python, expresiones regulares y operaciones de listas. Estos son conceptos fundamentales en el procesamiento del lenguaje natural y son ampliamente utilizados en una variedad de aplicaciones, como la clasificación de texto, la agrupación de documentos y la generación de texto, entre otros. >>

Fin de la Actividad de la semana 2.

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.