Actividad 2 - Aplicación de análisis de sentimientos

October 27, 2023

1 Actividad 2: Aplicación de análisis de sentimientos

Alfonso Pineda Cedillo | A01660394

Fecha de entrega: 27 de Octubre de 2023

1.1 Instrucciones

Parte 1:

- Seleccione un corpus a trabajar.
  - Puede ser el corpus utilizado en los ejercicios anteriores.
  - Un dataset de tweets podría ser un buen ejemplo.
  - Dataset minado de alguna red social como todos los comentarios de un post de una figura pública.
- Aplicar un modelo pre entrenado del "sentiment analisis" y analizar reacciones de un post.

Parte 2:

Del corpus utilizado en la parte 1, realiza lo siguiente:

- Separe las oraciones después de cada punto.
- Contabilice el número total de oraciones.

1.2 Parte 1

Como primer paso, se importan las librerías necesarias para el desarrollo de la actividad.

[1]:
```
import pandas as pd
from transformers import TFAutoModelForSequenceClassification, AutoTokenizer,␣
 ↪AutoConfig
import numpy as np
from scipy.special import softmax
import re
import urllib.request
import csv
```

Posteriormente, realizamos la carga de nuestro dataset, el cual contiene 1,600,000 tweets extraídos usando el API de Twitter. El dataset contiene 6 columnas, de las cuales solo nos interesa la columna

de `text` y la columna `target`, la cual indica si el tweet es positivo, negativo o neutral. Para este proceso, ocupamos la librería `pandas` y la función `read_csv` para leer el archivo.

[2]:
```
# Cargar el conjunto de datos desde el archivo CSV
DATASET_COLUMNS=['target','ids','date','flag','user','text']
dataset = pd.read_csv('datasets/tweets.csv', encoding='ISO-8859-1',␣
 ↪names=DATASET_COLUMNS)
```

Asimismo, realizamos un pequeño análisis exploratorio de los datos, para ver el formato general de la base de datos y ver si hay datos faltantes. Para esto, ocupamos la función `head()` para ver los primeros 5 registros de la base de datos, y la función `info()` para ver el tipo de dato de cada columna y si hay datos faltantes.

[3]:
```
dataset.head()
```

[3]:

	target	ids	date	flag	\
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	

	user	text
0	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t…
1	scotthamilton	is upset that he can't update his Facebook by …
2	mattycus	@Kenichan I dived many times for the ball. Man…
3	ElleCTF	my whole body feels itchy and like its on fire
4	Karoli	@nationwideclass no, it's not behaving at all…

[4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   target  1600000 non-null  int64
 1   ids     1600000 non-null  int64
 2   date    1600000 non-null  object
 3   flag    1600000 non-null  object
 4   user    1600000 non-null  object
 5   text    1600000 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

Para probar el modelo pre-entrenado, ocupamos únicamente la columna `text`, la cual contiene el contenido del tweet. Para esto, almacenamos los datos de la columna en una variable llamada `data` y posteriormente, observamos su contenido.

[17]:
```
data = dataset[['text', 'target']]
data
```

[17]:

	text	target
0	@switchfoot http://twitpic.com/2y1zl - Awww, t…	0
1	is upset that he can't update his Facebook by …	0
2	@Kenichan I dived many times for the ball. Man…	0
3	my whole body feels itchy and like its on fire	0
4	@nationwideclass no, it's not behaving at all…	0
…	…	…
1599995	Just woke up. Having no school is the best fee…	4
1599996	TheWDB.com - Very cool to hear old Walt interv…	4
1599997	Are you ready for your MoJo Makeover? Ask me f…	4
1599998	Happy 38th Birthday to my boo of alll time!!! …	4
1599999	happy #charitytuesday @theNSPCC @SparksCharity…	4

[1600000 rows x 2 columns]

A continuación, realizamos la carga del modelo pre-entrenado y del tokenizador. Para esto, se define la ruta al modelo pre-entrenado que se desea utilizar. El modelo en cuestión se llama "cardiffnlp/twitter-roberta-base-sentiment". Este modelo ha sido pre-entrenado en datos relacionados con Twitter y está diseñado para realizar tareas de análisis de sentimiento en texto de Twitter.

Asimismo, ocupamos la función `AutoTokenizer.from_pretrained` para cargar el tokenizador. El tokenizador es una parte esencial del procesamiento de texto, ya que se encarga de dividir el texto en unidades más pequeñas, como palabras o subpalabras, que son comprensibles para el modelo. En este caso, el tokenizador se adapta al modelo pre-entrenado y está diseñado específicamente para procesar texto de Twitter.

Luego, se carga la configuración del modelo utilizando la función AutoConfig.from_pretrained(MODEL). La configuración del modelo incluye detalles sobre la arquitectura y las configuraciones específicas del modelo, como el número de capas, el tamaño del embedding, entre otros. Esta información es esencial para asegurarse de que el modelo se utilice de manera adecuada.

Finalmente, se carga el modelo en sí utilizando la función TFAutoModelForSequenceClassification.from_pretrained(MODEL). El modelo pre-entrenado es una red neuronal profunda que ha sido entrenada en una gran cantidad de datos para realizar tareas de clasificación de secuencias, en este caso, análisis de sentimiento. El modelo ya ha aprendido patrones y características relevantes en el lenguaje natural a partir de los datos de Twitter.

[6]:
```
# Cargar el modelo preentrenado y el tokenizador
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
config = AutoConfig.from_pretrained(MODEL)
model = TFAutoModelForSequenceClassification.from_pretrained(MODEL)
```

```
All model checkpoint layers were used when initializing
TFRobertaForSequenceClassification.
```

```
All the layers of TFRobertaForSequenceClassification were initialized from the
model checkpoint at cardiffnlp/twitter-roberta-base-sentiment.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use TFRobertaForSequenceClassification for predictions without
further training.
```

Posteriormente, se define una función denominada `preprocess` con el propósito de realizar el preprocesamiento de texto. Este procedimiento es fundamental en el procesamiento de lenguaje natural y se utiliza para limpiar y estandarizar el texto de entrada antes de realizar análisis más avanzados. La función se encarga de aplicar dos principales transformaciones al texto ingresado como argumento.

En primer lugar, la función efectúa la eliminación de menciones de usuario en el texto. Esto se logra mediante el uso de expresiones regulares (regex). Las menciones de usuario en plataformas de redes sociales como Twitter suelen comenzar con el símbolo '@', seguido de un nombre de usuario que puede consistir en letras, números y guiones bajos. La función `re.sub` se utiliza para reemplazar todas las menciones de usuario en el texto original por la etiqueta '@user'. Esta acción es relevante ya que permite eliminar información específica de usuarios en el texto, lo que puede ser útil en tareas de análisis de sentimiento y procesamiento de texto en general.

En segundo lugar, la función lleva a cabo la eliminación de enlaces (URLs) del texto de entrada. Las URLs suelen comenzar con "http" o "https" y pueden ser seguidas por caracteres diversos que representan una dirección web. Al igual que en el caso de las menciones de usuario, la función `re.sub` se utiliza para reemplazar todas las URLs en el texto original por la etiqueta 'http'. Esta acción es significativa porque permite suprimir enlaces web que no suelen contener información relevante para el análisis de sentimiento o procesamiento de texto en sí.

[7]:
```python
# Función para preprocesar el texto
def preprocess(text):
    # Eliminar menciones de usuario
    text = re.sub(r'@[A-Za-z0-9_]+', '@user', text)
    # Eliminar enlaces
    text = re.sub(r'http\S+', 'http', text)
    return text
```

A continuación, definimos una función llamada `analyze_sentiment`, que se utiliza para realizar el análisis de sentimiento en un tweet dado. Esta función realiza una serie de pasos para evaluar la polaridad del sentimiento (positivo, negativo o neutral) en el texto del tweet.

En primer lugar, el código invoca la función antes mencionada, llamada `preprocess(text)`. Después del preprocesamiento, se codifica el texto limpio utilizando un tokenizador previamente cargado. El texto procesado se pasa al modelo pre-entrenado para realizar la predicción de sentimiento. El modelo genera una puntuación (score) para cada clase de sentimiento (positivo, negativo y neutral).

Para determinar la etiqueta de sentimiento correspondiente, el código realiza una serie de pasos adicionales. Primero, se normaliza las puntuaciones mediante una función softmax, lo que produce una distribución de probabilidad sobre las clases de sentimiento. Luego, las etiquetas se recuperan a partir de un archivo de mapeo (mapping) ubicado en una URL específica. Las etiquetas representan

las clases de sentimiento a las que se asocian las puntuaciones.

Finalmente, se ordenan las puntuaciones en orden descendente, lo que permite identificar la etiqueta de sentimiento más probable. Las etiquetas y las puntuaciones se imprimen en la consola, lo que brinda información sobre el sentimiento estimado del tweet, junto con la confianza asociada a esa estimación.

[31]:
```python
def analyze_sentiment(text):
    text = preprocess(text)
    encoded_input = tokenizer(text, return_tensors='tf')
    output = model(encoded_input)
    scores = output[0][0].numpy()
    scores = softmax(scores)
    ranking = np.argsort(scores)
    ranking = ranking[::-1]

    # Definir el mapeo de etiquetas
    mapping = {
        0: 'negative',
        1: 'neutral',
        2: 'positive'
    }

    sentiment_result = {}
    for i in range(scores.shape[0]):
        label_id = ranking[i]  # Obtener la etiqueta como entero
        label = mapping[label_id]  # Obtener el nombre de la etiqueta
        score = scores[ranking[i]]
        sentiment_result[label] = np.round(float(score), 4)
        # print(f"{i+1}) {label} {np.round(float(score), 4)}")

    return sentiment_result
```

Por último, obtenemos 10 tweets aleatorios del dataset previamente importado y aplicamos la función `analyze_sentiment` a cada uno de ellos. Esto nos permite observar el funcionamiento del modelo pre-entrenado en la práctica y ver cómo se comporta en diferentes casos de uso.

El dataset escogido para probar el modelo es diferente que el utilizado para su entrenamiento. Esto due hecho a propósito para ver cómo se comporta el modelo en un dataset que no ha visto antes. Los resultados obtenidos se imprimen primero mostrando el tweet analizado y posteriormente, la etiqueta de sentimiento y la confianza asociada a esa estimación.

[30]:
```python
# Elegir 10 tweets aleatorios
random_tweets = data.sample(10)

# Aplicar el análisis de sentimiento a los 10 tweets aleatorios
for index, row in random_tweets.iterrows():
    text = row['text']
    print(f"Tweet: {text}")
```

```
    analyze_sentiment(text)
    print("\n-----------------------------\n")
```

```
Tweet: @ChrisBaragar Me too
1) neutral 0.7331
2) positive 0.1897
3) negative 0.0772

-----------------------------

Tweet: the &quot;history of Art &quot; is so,well BORING! and long, so long
1) neutral 0.4315
2) positive 0.3709
3) negative 0.1976

-----------------------------

Tweet: @Tymethief Glad to hear it  bummer on the electric though.
1) positive 0.7591
2) neutral 0.2077
3) negative 0.0331

-----------------------------

Tweet: @wynlim I looked at my network settings and I don't see anythig there…
1) neutral 0.4985
2) negative 0.4697
3) positive 0.0319

-----------------------------

Tweet: Ahhhhhh! My backyard.
1) positive 0.6056
2) neutral 0.3381
3) negative 0.0563

-----------------------------

Tweet: @giblahoj so don't wake up
1) neutral 0.6138
2) negative 0.3334
3) positive 0.0528

-----------------------------

Tweet: My sweety has appeared  Almost…
1) positive 0.5533
```

```
2) neutral 0.437
3) negative 0.0097

------------------------------

Tweet: @barbedwyer ha ha!  So hav u bn to Northern Ireland? Oh thank you re:
photo of my mum! My father is v lucky! xxx
1) positive 0.9616
2) neutral 0.0356
3) negative 0.0028

------------------------------

Tweet: I really don't want to give up the fun that was this weekend
1) positive 0.5409
2) neutral 0.2878
3) negative 0.1713

------------------------------

Tweet: @Tamara_RJ nah i had the regular ones   but they were still great, LOL
1) positive 0.9453
2) neutral 0.0498
3) negative 0.005

------------------------------
```

1.3 Parte 2

Debido a la longitud del dataset original (1,600,000 tweets), se decidió utilizar una muestra de 100,000 tweets para realizar el análisis de sentimiento. Para esto, se utilizó la función `groupby` para agrupar los tweets por su etiqueta de sentimiento y posteriormente, se utilizó la función `sample` para obtener una muestra de 50,000 tweets por cada etiqueta de sentimiento. Esto nos permite obtener una muestra equilibrada de tweets positivos y negativos.

[32]:
```python
# Contar la cantidad de registros para cada categoría de 'target'
target_counts = data['target'].value_counts()

# Definir la cantidad de registros que deseas muestrear por categoría
sample_size_per_category = 50000  # Puedes ajustar este valor según tus␣
 ↪necesidades

# Realizar el muestreo equilibrado
sampled_data = data.groupby('target', group_keys=False).apply(lambda x: x.
 ↪sample(min(len(x), sample_size_per_category)))

# Restablecer el índice del dataset resultante
```

```
sampled_data = sampled_data.reset_index(drop=True)

# Verificar que el muestreo sea equilibrado
sampled_target_counts = sampled_data['target'].value_counts()
print(sampled_target_counts)
```

```
target
0    50000
4    50000
Name: count, dtype: int64
```

Aplicamos el modelo de análisis de sentimiento a todo el dataset y almacenamos los resultados obtenidos en 3 nuevas columnas llamadas 'positive', 'negative' y 'neutral'. Estas columnas contienen la probabilidad de que el tweet sea positivo, negativo o neutral, respectivamente.

[ ]:
```
# Aplicar el análisis de sentimiento a todo el dataset
positive_scores = []
negative_scores = []
neutral_scores = []

for index, row in sampled_data.iterrows():
    text = row['text']
    sentiment_result = analyze_sentiment(text)
    positive_scores.append(sentiment_result.get('positive', 0))
    negative_scores.append(sentiment_result.get('negative', 0))
    neutral_scores.append(sentiment_result.get('neutral', 0))

# Agregar las columnas al dataset
sampled_data['Positive'] = positive_scores
sampled_data['Negative'] = negative_scores
sampled_data['Neutral'] = neutral_scores
```

[52]:
```
sampled_data
```

[52]:

	text	target	Positive	Negative	Neutral
0	happy fathers day to all the dads out there…	0	0.9435		
1	OMG noooo leah u put that pic of us singing '…	0	0.0850		
2	@Bliezy Why don't you like Holland? I'm sure i…	0	0.3744		
3	Miss my honey.. http://myloc.me/4zQb	0	0.0938		
4	Am sending good thoughts to my friend Carmi. H…	0	0.7920		
…	…	…	…		
99995	@jonwaldock i hope it goes well have fun!	4	0.9850		
99996	@katelyntarver my star can't wait to see u	4	0.9799		
99997	long long day. movies compressing itself in th…	4	0.1911		
99998	Back door. Good idea	4	0.8212		
99999	@chrisjsimon ah alrite. hehe well feast your e…	4	0.8873		

```
0         0.0102   0.0462
1         0.4198   0.4952
2         0.2732   0.3524
3         0.4659   0.4403
4         0.0211   0.1869
…          …       …
99995     0.0016   0.0134
99996     0.0016   0.0185
99997     0.2203   0.5886
99998     0.0125   0.1663
99999     0.0044   0.1083

[100000 rows x 5 columns]
```

Obtenemos los 5 tweets con mayor probabilidad de ser positivos, imprimiendo el texto del tweet y sus probabilidades de ser positivo. Comprobamos que los tweets con mayor probabilidad de ser positivos, efectivamente lo sean.

[ ]:
```python
top_positive = sampled_data.nlargest(5, 'Positive')

# Mostrar los 5 tweets más positivos
print("================================")
print("== Top 5 tweets más positivos ==")
print("================================\n")
for index, row in top_positive.iterrows():
    text = row['text']
    positive_score = row['Positive']
    print(f"Tweet: {text}")
    print(f"Positive Score: {positive_score}")
    print("\n-----------------------------\n")
```

```
================================
== Top 5 tweets más positivos ==
================================

Tweet: laying in bed, smiling the biggest smile. It's beautiful out and last
night was so fun &amp;lt;3  Happy Mothers Day!
Positive Score: 0.9942

-----------------------------

Tweet: @itschristablack i love reading your tweets. they make me happy and can't
wait to see you guys aug 25 in nashville  be blessed today.
Positive Score: 0.9941

-----------------------------

Tweet: @Jonasbrothers congrats once again. haha. i'm lovin' it.  you guys are
```

```
the best! &lt;3 come to singapore pleeeease.
Positive Score: 0.994

------------------------------

Tweet: Nice date night with my hubby, @charlienesdahl.  Celebrated good test
result.  Love him so much!
Positive Score: 0.994

------------------------------

Tweet: @ kirstenamber Me n kirst are being the greatest big sisters ever
today… yay so excited its gonna be a fabulous day!!
Positive Score: 0.994

------------------------------
```

Obtenemos los 5 tweets con mayor probabilidad de ser negativos, imprimiendo el texto del tweet y sus probabilidades de ser negativo. Comprobamos que los tweets con mayor probabilidad de ser negativos, efectivamente lo sean.

[ ]:
```python
top_negative = sampled_data.nlargest(5, 'Negative')

# Mostrar los 5 tweets más negativos
print("================================")
print("== Top 5 tweets más negativos ==")
print("================================\n")
for index, row in top_negative.iterrows():
    text = row['text']
    negative_score = row['Negative']
    print(f"Tweet: {text}")
    print(f"Negative Score: {negative_score}")
    print("\n------------------------------\n")
```

```
================================
== Top 5 tweets más negativos ==
================================

Tweet: My eyes have been hurting ridiculously for the past few days and I hate
it. What have I done wrong to deserve this?!
Negative Score: 0.985

------------------------------

Tweet: i hate it that i've had this stupid cough for THREE weeks!! leave me
alone!!
Negative Score: 0.9849
```

```
------------------------------

Tweet: I HATE LIVING IN TAMPA! I'm really lonely and extremely bored!!
Negative Score: 0.9846

------------------------------

Tweet: I HATE JETLAG!!! its 2 pm but why do i feel like its night time already?
i miss you canada
Negative Score: 0.984

------------------------------

Tweet: FUCK MY LIFE  uggggh! SO pissed up :'( STUPID CAREERS PROJECTS DON&quot;T
HELP EITHER!
Negative Score: 0.9838

------------------------------
```

Obtenemos los 5 tweets con mayor probabilidad de ser neutros, imprimiendo el texto del tweet y sus probabilidades de ser positivo, negativo o neutral. Comprobamos que los tweets con mayor probabilidad de ser neutros, efectivamente lo sean.

[ ]:
```python
top_neutral = sampled_data.nlargest(5, 'Neutral')

# Mostrar los 5 tweets más neutros
print("================================")
print("=== Top 5 tweets más neutros ===")
print("================================\n")
for index, row in top_neutral.iterrows():
    text = row['text']
    neutral_score = row['Neutral']
    print(f"Tweet: {text}")
    print(f"Neutral Score: {neutral_score}")
    print("\n------------------------------\n")
```

```
================================
=== Top 5 tweets más neutros ===
================================

Tweet: pssst: Rebecca Taylor footwear sale coming: http://www.thesavvy.com/ May
27th, 2009, 12:00 PM EST (Noon) - May 30th 12:00 AM EST
Neutral Score: 0.9507

------------------------------
```

```
Tweet: This is a status update to twitter from ICE  Timestamp: Mon Jun 15
11:02:44 CEST 2009
Neutral Score: 0.9465

------------------------------

Tweet: This is a status update to twitter from ICE  Timestamp: Sun May 10
10:57:44 CEST 2009
Neutral Score: 0.9464

------------------------------

Tweet: This is a status update to twitter from ICE  Timestamp: Mon Jun 15
14:02:34 CEST 2009
Neutral Score: 0.9463

------------------------------

Tweet: @amyomy : How long have ya been doing yoga?
Neutral Score: 0.9416

------------------------------
```

Posteriormente obtenemos la cantidad de oraciones que componen cada tweet. Para esto, definimos una función llamada `count_sentences` que se encarga de contar el número de oraciones en un texto dado. Esta función utiliza la librería `nltk` y la función `sent_tokenize` para dividir el texto en oraciones. La función `len` se utiliza para contar el número de oraciones en el texto. Es importante mencionar que consideramos como oración a cualquier secuencia de caracteres que termina con un punto.

Una vez obtenido el número de oraciones en cada tweet, almacenamos los resultados en una nueva columna llamada 'sentences'. Posteriormente, imprimimos los 5 tweets con mayor número de oraciones, junto con el número de oraciones en cada uno de ellos.

[66]:
```python
import nltk
from nltk.tokenize import sent_tokenize
import spacy

# Función para contar las oraciones en un texto y agregarlo como columna␣
 ↪'sentences'
def count_sentences(text):
    sentences = text.split('.')  # Dividir el texto por puntos
    # Eliminar elementos vacíos resultantes de la división
    sentences = [sentence.strip() for sentence in sentences if sentence.strip()]
    return len(sentences)

# Aplicar la función a cada fila del DataFrame y agregar la columna 'sentences'
```

```
sampled_data['sentences'] = sampled_data['text'].apply(count_sentences)
```

[67]:
```
top_sentences = sampled_data.nlargest(5, 'sentences')[['text', 'sentences']]

# Mostrar los 5 tweets con más oraciones
print("=======================================")
print("=== Top 5 tweets con más oraciones ===")
print("=======================================\n")
for index, row in top_sentences.iterrows():
    text = row['text']
    sentences = row['sentences']
    print(f"Tweet: {text}")
    print(f"Sentences: {sentences}")
    print("\n-----------------------------\n")
```

```
=======================================
=== Top 5 tweets con más oraciones ===
=======================================

Tweet: Heyy.thnxx.or.inviting.me.to.ur.party.rob.ill.be.down.there.on.june.24th.
ill.miss.u.when.u.go.to.italy…   &lt;333333333 txt.me.hon!!!
Sentences: 27

-----------------------------

Tweet: bob. bob. bob. bob. bob. hair. bob. bob. bob. aaaaaahhh 80. translation:
i cut my hair REALLY short. kinda like it though…
Sentences: 12

-----------------------------

Tweet: inspired… ahhh.. i love that feeling.. i.n.s.p.i.r.e.d
Sentences: 11

-----------------------------

Tweet: coffee… guitar lesson… food… interview… food… open mic…
food… open mic… food… sleep  Alright Monday, let's do this!
Sentences: 10

-----------------------------

Tweet: Laundry. Dishes. Deck cleaning. Virtuality. Sonic Boom. Propane. Beers.
BBQ. Boozin. Join me
Sentences: 10

-----------------------------
```

Finalmente, guardamos el dataset con los resultados obtenidos en un nuevo archivo CSV llamado 'tweets_sentiment.csv' y utilizamos la función `head()` para ver el formato del nuevo dataset, observando los primeros 5 registros.

[69]:
```
sampled_data.head()
```

[69]:

	text	target	Positive	Negative	Neutral	sentences
0	happy fathers day to all the dads out there…	0	0.9435	0.0102	0.0462	2
1	OMG noooo leah u put that pic of us singing '…	0	0.0850	0.4198	0.4952	1
2	@Bliezy Why don't you like Holland? I'm sure i…	0	0.3744	0.2732	0.3524	1
3	Miss my honey.. http://myloc.me/4zQb	0	0.0938	0.4659	0.4403	3
4	Am sending good thoughts to my friend Carmi. H…	0	0.7920	0.0211	0.1869	2

[70]:
```
# Guardar el DataFrame actualizado con la nueva columna
sampled_data.to_csv('tweets_sentiment.csv', index=False)
```