Caso práctico: creencias sobre el calentamiento climático

Ahora vamos a proceder a la determinación de opiniones a partir de un conjunto de observaciones procedente de la red social Twitter. No nos vamos a detener en las distintas fases de cálculo, sino que utilizaremos el lenguaje Python para realizar la misión de clasificación.

1. ¿Cómo obtener datos?

Lo primero que tenemos que hacer es obtener un conjunto de observaciones Para ello podemos utilizar las distintas redes sociales, como Facebook o Twitter, a fin recoger un conjunto de conversaciones. En efecto, las redes sociales son propicias para la clasificación de opiniones porque los usuarios las utilizan para publicar sus pareceres sobre distintos temas.

Además, gracias a los *hashtag* es fácil seleccionar las conversaciones relacionadas con un tema que nos interesa. Al seleccionar las conversaciones que tienen el hashtag #inteligenciaartificial, hay muchas posibilidades de que se refieran a la inteligencia artificial.

Como ejemplo, vamos a estudiar las distintas opiniones que los estadounidenses han dejado en Twitter sobre el calentamiento climático. Intentaremos clasificarlas en dos grupos: los que son conscientes del calentamiento y los que no creen en él. Esta clasificación podría utilizarse con fines políticos o de sensibilización, usted decide.

2. Creación de un proyecto Python

Los módulos de Python que vamos a utilizar en este proyecto están relacionados con la manipulación de datos de texto y de tablas.

Por eso vamos a utilizar las siguientes bibliotecas:

* NLTK (*Natural Language ToolKit*)
* Numpy
* Pandas
* Scikit-Learn

3. Adquisición y preparación de los datos

Los datos que vamos a utilizar se pueden descargar en el sitio del editor. Para evitarle todas las fases de preparación de los datos (normalización de las etiquetas de clasificación y eliminación de las observaciones no categorizadas), ya hemos realizado está tarea por usted. Sin embargo, también tiene la posibilidad de descargar el archivo original, con el que podrá ejercitarse en las diversas tareas de preparación.

**a. Carga del archivo**

La primera etapa consistirá en cargar el archivo que contiene las observaciones y que previamente hemos colocado en un directorio del proyecto llamado datas. El almacenamiento de los datos se hará en un Dataframe al que llamaremos mensajesTwitter.

import pandas as pnd

mensajesTwitter =

pnd.read\_csv("datas/calentamientoClimatico.csv",delimiter=";")

Hay que destacar el uso del parámetro delimiter=";" para indicar que las características de cada observación están separadas por un ;.

Después de finalizar esta etapa, podemos verificar la cantidad de observaciones contenidas en el Dataframe, así como los nombres de las distintas características:

print(mensajesTwitter.shape)

print(mensajesTwitter.head(2))

Dando como resultado:

**(4225, 3)**

                                              TWEET CREENCIA

 CONFIANZA

0  Global warming report urges governments to act...      Yes

   1.0

1  Fighting poverty and global warming in Africa ...      Yes

   1.0

Por lo tanto, disponemos de 4226 observaciones y 3 características con los nombres TWEET, CREENCIA, CONFIANZA.

La característica TWEET corresponde al mensaje dejado en Tweeter.

La característica CREENCIA corresponde a la opinión expresada por el mensaje. Si la característica es igual a Yes, significa que el mensaje expresa una opinión sobre el calentamiento climático.

La característica CONFIANZA indica el grado de confianza del tipo de opinión expresada: 1 significa que estamos seguros al 100 % de que la opinión expresada corresponde a la creencia o no en el calentamiento climático.

Vamos a modificar el contenido de la característica CREENCIA para que tome el valor 1 cuando el valor original es Yes y 0 cuando el valor original es No.

**mensajesTwitter['CREENCIA'] =**

**(mensajesTwitter['CREENCIA']=='Yes').astype(int)**

print(mensajesTwitter.head(100))

**b. Normalización**

Ahora que los datos están cargados en un Dataframe, vamos a proceder a la normalización de los datos con una función a la que llamaremos normalizacion tomando como parámetro un mensaje. Aquí podemos ver la función:

import re

def normalizacion(mensaje):

   mensaje = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',

mensaje)

   mensaje = re.sub('@[^\s]+','USER', mensaje)

   mensaje = mensaje.lower().replace("ё", "е")

   mensaje = re.sub('[^a-zA-Zа-яА-Я1-9]+', ' ', mensaje)

   mensaje = re.sub(' +',' ', mensaje)

   return mensaje.strip()

Esta función utiliza lo que se llama expresiones regulares (*regular expressions*) que permiten buscar un conjunto de caracteres específicos en una cadena de caracteres. Una vez encontrados los caracteres, entonces podemos aplicarles un tratamiento.

La función sub del módulo Re (*regular expression*) se utiliza de la siguiente manera:

text = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL', text)

Lo que significa que sustituimos todos los caracteres www con la palabra «URL» dentro de la cadena de caracteres text. En otras palabras, se elimina todo rastro de enlace de Internet, que se sustituye con el término «URL». También podemos ver que el conjunto de los mensajes se ha transformado en minúsculas con ayuda de la función lower. En cuanto al uso de la función strip(), su trabajo es eliminar las tabulaciones, los espacios y los retornos de línea contenidos en el mensaje.

Para efectuar la normalización, tenemos que llamar al método que acabamos de crear como se indica a continuación:

**mensajesTwitter["TWEET"] =**

**mensajesTwitter["TWEET"].apply(normalizacion)**

print(mensajesTwitter.head(10))

La función normalizacion se aplica a cada uno de los mensajes contenidos en la característica TWEET. Los resultados obtenidos se pueden mostrar usando la función head() del Dataframe.

**c. Eliminación de las stop words**

Para eliminar las *stop words* de nuestros mensajes en inglés, vamos a utilizar las que conoce el módulo nltk.corpus:

from nltk.corpus import stopwords

stopWords = stopwords.words('english')

A continuación, vamos a utilizar una función lambda (anónima) que comprueba todas las palabras contenidas en los mensajes y verifica si la variable stopWords las conoce.

Esta función lambda crea una cadena vacía. Si la variable stopWords no conoce la primera palabra del mensaje, la guarda y la concatena (join) con la cadena vacía. En caso contrario, la ignora. Entonces puede empezar el análisis de la palabra siguiente. Si la variable stopWords no conoce la segunda palabra del mensaje, la guarda y la concatena (join) con la cadena que contiene la primera palabra. Entonces puede empezar el análisis de la tercera palabra y así sucesivamente.

mensajesTwitter['TWEET'] = mensajesTwitter['TWEET'].apply(lambda

mensaje: ' '.join([palabra for palabra in mensaje.split() if palabra

not in

(stopWords)]))

print(mensajesTwitter.head(10))

**d. El stemming**

La aplicación de *stemming* a los mensajes se hace con ayuda del módulo ntlk y de la clase SnowballStemmer. Nosotros también utilizamos una función lambda, como hemos hecho en la eliminación de las *stop words*.

**from nltk.stem.snowball import SnowballStemmer**

**stemmer = SnowballStemmer('english')**

mensajesTwitter['TWEET'] = mensajesTwitter['TWEET'].apply(lambda

mensaje: ' '.join([stemmer.stem(mot) for palabra in

mensaje.split(' ')]))

print(mensajesTwitter.head(10))

**e. La lematización**

La lematización de los mensajes también se hace con el módulo ntlk, pero además esta vez también con la clase WordNetLemmatizer:

**from nltk.stem import WordNetLemmatizer**

**lemmatizer = WordNetLemmatizer()**

mensajesTwitter['TWEET'] = mensajesTwitter['TWEET'].apply(lambda

mensaje: ' '.join([lemmatizer.lemmatize(palabra) for palabra in

mensaje.split(' ')]))