
SKILLING
CENTER
TECMILENIO

SKILLING CENTER TECMILENIO

Visualización y Programación en Python

Geolocalización

Introducción

A Brenda le informaron que se requiere obtener algunos datos económicos de México para un nuevo proyecto. Esta información se puede obtener desde algunos centros de datos como el INEGI, y con éstos generar mapas temáticos e incluso interactivos.

Referente a los gráficos, no tiene problema pues es algo con lo que ya está familiarizada, sin embargo, respecto a los mapas, ella debe aprender a generarlos a partir de datos georeferenciados.

En esta experiencia educativa aprenderás a realizar mapas en Python utilizando diferentes orígenes de datos y diferentes librerías para lograrlo.

Explicación

La geolocalización, según Marketinhouse (s.f.), consiste en poder obtener la ubicación geográfica real de manera muy precisa de un dispositivo digital con una conexión a internet.

- Se pueden geolocalizar cualquier dispositivo que esté conectado a internet, ya sea por seguridad, por logística o por negocio.
- Ejemplos: identificar los contactos cercanos a mi ubicación, lugares de interés cercanos, lugar en donde se tomó la fotografía, puntos de acceso de usuarios de servicios de *streaming*, por mencionar algunos.

Explicación

El módulo GeoPy

- Librería que permite ubicar cualquier cosa por medio de coordenadas.
- La opción más sencilla de geolocalizar en Python un punto de interés es importando el geolocalizador Nominatim de la siguiente forma:
  - **from geopy.geocoders import Nominatim**
  - **geolocator= Nominatim(user_agent='my_app')**
- En seguida, se introduce el nombre del lugar buscado y se obtiene la dirección y las coordenadas:
  - **location = geolocator.geocode("Universidad Tecmilenio, Ciudad Juárez")**
  - **print(location.address)**
  - **print((location.latitude, location.longitude))**

Y el resultado es el siguiente:
Universidad TecMilenio, Calle Teófilo Borunda, Ciudad Juárez, Juárez, Chihuahua, 32668, México
(31.674806, -106.3858387962321)

Explicación

Ya con las coordenadas, es posible introducirlas en una aplicación como Google Maps, y se obtiene la ubicación exacta en el mapa.

- También es posible obtener el domicilio con las coordenadas:
  - **from geopy.geocoders import Nominatim**
  - **geolocator= Nominatim(user_agent='my_app')**
  - **location = geolocator.reverse("31.674806, -106.3858387962321 ")**
  - **print(location.address)**
- Y el resultado es la dirección:
  Universidad TecMilenio, Calle Teófilo Borunda, Ciudad Juárez, Juárez, Chihuahua,
  32668, México.

Explicación

- Cuando se requiera ubicar diversos puntos de interés, es posible obtener sus coordenadas de la siguiente forma:
  - **from geopy.geocoders import Nominatim**
  - **import time**
  - **import math**
  - **geo = Nominatim(user_agent="MyApp")**
  - **lugar_list= ["Madrid", "Monterrey", "Nueva York"]**
  - **for i in lugar_list:**
  - **ubicacion = geo.geocode(i, timeout=3)**
  - **print(i +" se ubica en: "+ str(ubicacion.latitude) + " " + str(ubicacion.longitude))**
  - **time.sleep(1)**
- Y el resultado son las coordenadas de las tres ciudades:
  Madrid se ubica en: 40.4167047 -3.7035825
  Monterrey se ubica en: 25.6802019 -100.315258
  Nueva York se ubica en: 40.7127281 -74.0060152

Explicación

- Se puede optimizar el código por medio de funciones para localizar varias ciudades:

```
def get_coords(sciudad):
  try:
    geo = Nominatim(user_agent="MyApp")
    ubicacion = geo.geocode(sciudad)
    print("C# %s %s se ubica en: %s" % (ubicacion.latitude,
     ubicacion.longitude, ubicacion.address))
    return ubicacion.latitude, ubicacion.longitude
  except:
    debug.print_exception()
    return None

def get_address(lat, long):
  try:
    geo = Nominatim(user_agent="MyApp")
    scoordenadas = ("%s %s" % (lat, long))
    direccion = geo.reverse(scoordenadas)
    print("A# las coordenadas %s %s coresponden a %s" %(direccion.latitude, direccion.longitude,
direccion.address))
    return direccion.address
  except:
    debug.print_exception()
    return None
```

Explicación

Y la llamada a las funciones se realiza de la siguiente forma:

- **get_address("37.3937585","-5.9976877")**
- **get_address("25.6802019","-100.315258")**
- **get_coords("La minerva, Guadalajara, Mexico")**
- **get_coords("angel de la independencia, ciudad de Mexico, Mexico")**

Y se obtiene la siguiente información:
A# las coordenadas 37.3937585 -5.9976877 coresponden a Catedral, Calle Virgen de los Buenos Libros, Encarnación-Regina, Casco Antiguo, Sevilla, Andalucía, 41001, España
A# las coordenadas 25.6798791 -100.3154889 coresponden a 239, Calle Santiago Tapia, Centro, Monterrey, Nuevo León, 64010, México
C# 20.6744485 -103.3873984 se ubica en: La Minerva, Glorieta Minerva, Arcos Vallarta, Guadalajara, Región Centro, Jalisco, 44110, México
C# 19.42699615 -99.16766910537038 se ubica en: Ángel de la Independencia, Avenida Paseo de la Reforma, Juárez, Cuauhtémoc, Ciudad de México, 06600, México

Explicación

- Otra funcionalidad de GeoPy es obtener la distancia entre dos puntos, dado sus coordenadas:
  - **import geopy.distance**
  - **coords_1 = (52.2296756, 21.0122287)**
  - **coords_2 = (52.406374, 16.9251681)**
  - **print (geopy.distance.geodesic(coords_1, coords_2).km)**
- El resultado es de 279.3529 kilómetros.

- Adicionalmente con base en coordenadas, ciudades o país en donde vives, con la librería Folium es posible obtener el mapa completo con ciertas utilerías como colores, zoom, etc. Como referencia, consulta la documentación oficial de Folium en:

El siguiente enlace es externo a la Universidad Tecmilenio, al acceder a él considera que debes apegarte a sus términos y condiciones.

Folium. (s.f.). *Python data, leaflet.js maps*. Recuperado de https://python-visualization.github.io/folium/latest/

Explicación

- La librería Plotly puede visualizar datos en forma de mapas interactivos.
- Como ejemplo de su uso, se desea visualizar la población de diversas ciudades del mundo sobre un mapa. Los pasos para seguir son los siguientes:
  a) Importar las librerías.
  b) Leer el archivo de datos pob_urbana.csv.
  c) Seleccionar la columna de población proyectada para 2025.
  d) Generar la figura y da estilo a los puntos a mostrar en el mapa.
  e) Mostrar el mapa.
- El código se puede obtener en el enlace del tema.

Actividad

Para reforzar tu aprendizaje del tema, realiza lo siguiente:

Considera que te están contratando por la mesa de seguridad de la ciudad en donde radicas, participando en el proyecto para prevenir y reducir la ocurrencia de conductas delictivas por medio del análisis de datos históricos que identifiquen patrones y tendencias de eventos y ubicaciones.

Consigue o genera una tabla con datos ficticios que contenga registros de la ocurrencia de conductas delictivas de los últimos 90 días, que incluya las variables de fecha de ocurrencia, edad del delincuente y código postal del evento.

1. Elabora el histograma con la frecuencia de eventos en relación con la edad del delincuente.
2. Elabora la gráfica con la cantidad de eventos registrados en períodos semanales, ordenados cronológicamente del más antiguo al más reciente.
3. Elabora la gráfica de caja para mostrar la distribución de la edad del delincuente en tres códigos postales.
4. Elabora tus comentarios y recomendaciones sobre tus hallazgos.

Presenta tu análisis mostrando el código Python que utilizaste así, como la gráfica correspondiente.

Cierre

En este tema has aprendido la utilidad de crear y visualizar mapas geográficos, además de obtener ubicaciones físicas, distancias entre puntos específicos, población de ciudades, etc. Con esta experiencia, al igual que Brenda, expandes nuevamente los beneficios que ofrecen las librerías de Python para generar mapas, agregando datos económicos que publica el INEGI como el PIB, censos económicos, entre otros, creando valor para la gestión de logística y mercadotecnia.

Bibliografía

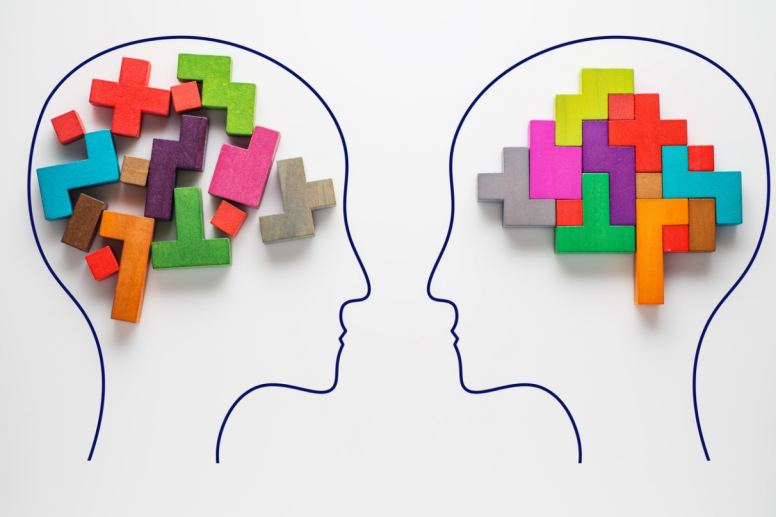
- Folium. (s.f.). *Python data, leaflet.js maps.* Recuperado de https://python-visualization.github.io/folium/latest/
- Marketinhouse. (s.f.). *Qué es la geolocalización y cómo funciona.* Recuperado de https://www.marketinhouse.es/que-es-la-geolocalizacion/

SKILLING CENTER TECMILENIO

Visualización y Programación en Python

Análisis de sentimientos

Introducción



La empresa Marketing Digital obtuvo recientemente un contrato con el gobierno para mejorar su presencia en línea y generar un acercamiento con sus gobernados. El gobierno tiene una fuerte presencia en las redes sociales, pero quería aprovechar mejor su presencia en la plataforma X para obtener una mejor comprensión de cómo se sentía la población respecto a su desempeño.

En esta experiencia educativa conocerás cuáles son las herramientas necesarias y la metodología que se sugiere para poder hace un análisis de sentimiento en la población.

Explicación

- En las redes sociales se genera información continuamente.
- Esta información contiene comentarios, opiniones, intereses, entre otros elementos.
- También las noticias viajan en las redes y se convierten en tendencias e impactos en temas específicos.
- La riqueza de esta información es que es posible modelar el flujo de información, su interpretación y estados de ánimo, tendencias y pronósticos.
- De acuerdo con Woodward (2024), los 436 millones de usuarios activos de la Plataforma X son una fuente relevante de información alternativa para percibir sentimientos, que puede ser aprovechada para aumentar la ventaja competitiva en los mercados.
- El análisis de sentimientos utiliza el procesamiento de lenguaje natural para extraer información cuantitativa.

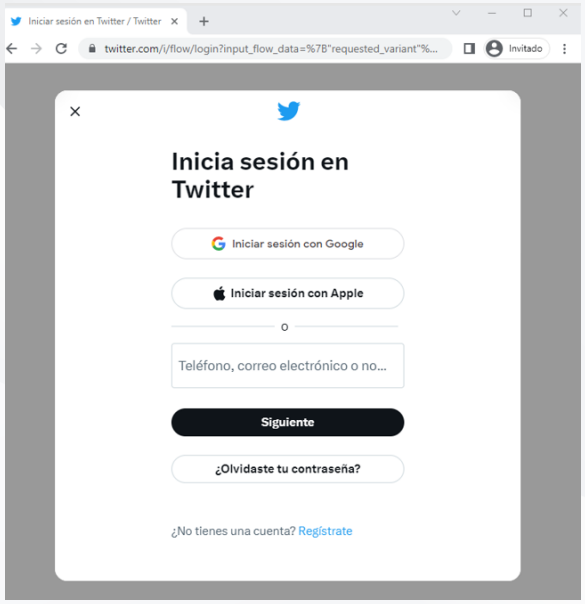
Explicación

Generación de API (*Application Programming Interface*) en plataforma X

- Permite conectarse a la API de X para extraer datos históricos y en tiempo real.
- En general, las redes sociales son datos no estructurados y utilizan jerga, regionalismos, emoticonos, etc., lo cual dificulta el análisis.
- Se requiere instalar las siguientes librerías:
  - **pip install tweepy**
  - **pip install textblob**
- También importar las librerías necesarias:
  - **from textblob import TextBlob**
  - **import sys**
  - **import tweepy**
  - **import matplotlib.pyplot as plt**
  - **import pandas as pd**
  - **import numpy as np**
  - **import os**
  - **import nltk**
  - **import pycountry**
  - **import re**
  - **import string**
  - **from wordcloud import WordCloud, STOPWORDS**
  - **from PIL import Image**
  - **from nltk.sentiment.vader import SentimentIntensityAnalyzer**
  - **from langdetect import detect**
  - **from nltk.stem import SnowballStemmer**
  - **from nltk.sentiment.vader import SentimentIntensityAnalyzer**
  - **from sklearn.feature_extraction.text import CountVectorizer**

Explicación

- Según Amazon Web Services (s.f.), el procesamiento de lenguaje natural o PNL permite a las máquinas comprender, interpretar e incluso responder al lenguaje humano analizando patrones y estructuras complejas del habla.
- En este punto, es necesario crear una cuenta de desarrollador en X. Este procedimiento tardará 1 o 2 días para cuentas básicas, pero se sugiere que obtengas una cuenta académica para disponer de mayor acceso a los datos de manera gratuita.



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

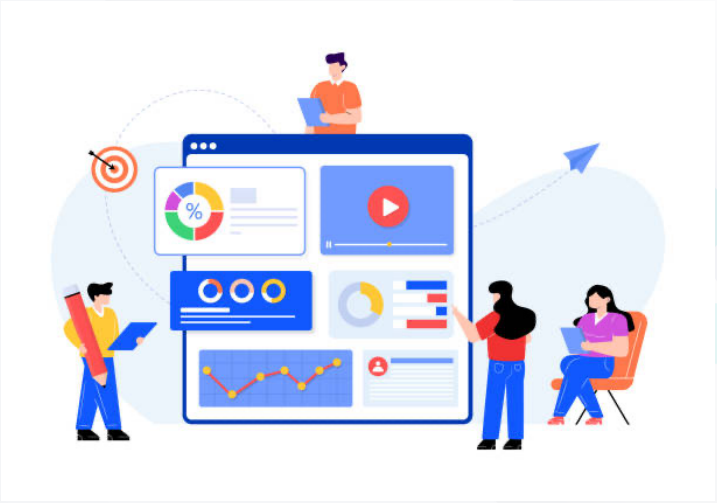
- Una vez creada la cuenta, es necesario hacer la autenticación en la API a través de tu aplicación, para lo cual es necesario definir primero las variables con las llaves necesarias para ello:
  - **# Authentication**
  - **consumer_key = "Type your consumer key here"**
  - **consumer_secret = "Type your consumer secret here"**
  - **access_token = "Type your accedd token here"**
  - **access_token_secret = "Type your access token secret here"**

- Y finalmente se utilizan los métodos de la librería tweepy para acceder:
  - **auth = tweepy.OAuthHandler(consumer_key, consumer_secret)**
  - **auth.set_access_token(access_token, access_token_secret)**
  - **api = tweepy.API(auth)**

Explicación

Extracción de tweets

- Para comenzar el análisis, se desarrolla el siguiente proceso:
  - Función para obtener el porcentaje.
  - Dos *inputs* para solicitar al usuario la palabra o *hashtag* que se va a buscar.
  - Cursor para la consulta con los parámetros obtenidos.
  - Creación de las variables positivo, negativo, neutral y polaridad, además la lista de tweets, lista de negativos, de positivos y de neutrales.
  - Barrido del cursor con un bucle *for* para catalogar el sentimiento de cada tweet.



Explicación

Propuesta de código (primera parte)

```
#Sentiment Analysis
def percentage(part,whole):
  return 100 * float(part)/float(whole)
keyword = input("Please enter keyword or hashtag to search: ")
noOfTweet = int(input ("Please enter how many tweets to analyze: "))
tweets = tweepy.Cursor(api.search, q=keyword).items(noOfTweet)
positive  = 0
negative = 0
neutral = 0
polarity = 0
tweet_list = []
neutral_list = []
negative_list = []
positive_list = []
for tweet in tweets:
  #print(tweet.text)
  tweet_list.append(tweet.text)
  analysis = TextBlob(tweet.text)
  score = SentimentIntensityAnalyzer().polarity_scores(tweet.text)
  neg = score['neg']
  neu = score['neu']
  pos = score['pos']
  comp = score['compound']
  polarity += analysis.sentiment.polarity
```

Explicación

Propuesta de código (segunda parte)

```
        if neg > pos:
          negative_list.append(tweet.text)
          negative += 1
        elif pos > neg:
          positive_list.append(tweet.text)
          positive += 1
        elif pos == neg:
          neutral_list.append(tweet.text)
          neutral += 1
    positive = percentage(positive, noOfTweet)
    negative = percentage(negative, noOfTweet)
    neutral = percentage(neutral, noOfTweet)
    polarity = percentage(polarity, noOfTweet)
    positive = format(positive, '.1f')
    negative = format(negative, '.1f')
    neutral = format(neutral, '.1f')
```

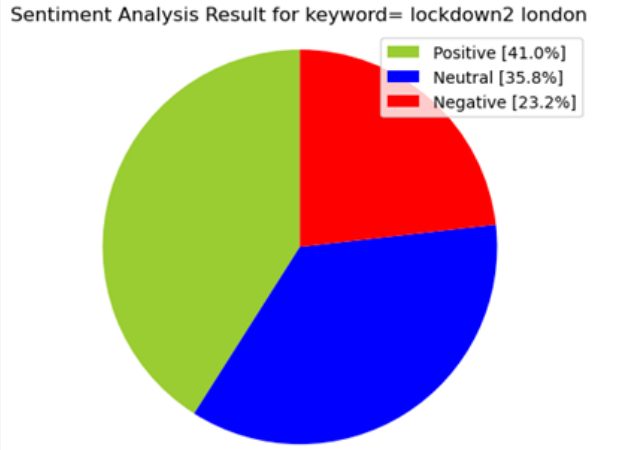
Explicación

- Para este ejercicio, se considera el sentimiento del segundo confinamiento de la ciudad de Londres y el número de tweets a analizar de 2500.
  - **Please enter keyword or hashtag to search: lockdown2 London**
  - **Please enter how many tweets to analyze: 2500**

Explicación

Después de obtener 2500 tweets, es posible ver la cantidad de tweets de cada tipo de sentimiento:

- **#Number of Tweets (Total, Positive, Negative, Neutral)**
- **tweet_list = pd.DataFrame(tweet_list)**
- **neutral_list = pd.DataFrame(neutral_list)**
- **negative_list = pd.DataFrame(negative_list)**
- **positive_list = pd.DataFrame(positive_list)**
- **print("total number: ",len(tweet_list))**
- **prnt("positive number: ",len(positive_list))**
- **print("negative number: ", len(negative_list))**
- **print("neutral number: ",len(neutral_list))**



Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

- Un análisis inicial arroja los siguientes resultados:
  - 1025 (41.0%) tweets con sentimiento positivo
  - 580 (23.2 %) tweets tienen un sentimiento negativo.
  - 895 (35.8 %) tweets tienen un sentimiento neutral.

- Lo cual se puede representar en una gráfica de pastel:
  - **#Creating PieCart**
  - **labels = ['Positive ['+str(positive)+'%]' , 'Neutral ['+str(neutral)+'%]','Negative ['+str(negative)+'%]']**
  - **sizes = [positive, neutral, negative]**
  - **colors = ['yellowgreen', 'blue','red']**
  - **patches, texts = plt.pie(sizes,colors=colors, startangle=90)**
  - **plt.style.use('default')**
  - **plt.legend(labels)**
  - **plt.title("Sentiment Analysis Result for keyword= "+keyword+"")**
  - **plt.axis('equal')**
  - **plt.show()**

Explicación

- Una buena práctica es limpiar los datos que se van a analizar, para lo cual hay que eliminar los registros duplicados usando la función drop_duplicates, en este caso, de la lista de tweets:
  - **tweet_list.drop_duplicates (inplace = True)**

- Como siguiente paso es limpiar el texto, los retweets, links y signos de puntuación, y estandarizando todo en minúsculas:
  - **#Cleaning Text (RT, Punctuation etc)**
  - **#Creating new dataframe and new features**
  - **tw_list = pd.DataFrame(tweet_list)**
  - **tw_list["text"] = tw_list[0]**

  - **remove_rt = lambda x: re.sub('RT @\w+: ',"",x)**
  - **rt = lambda x: re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)","",x)**
  - **tw_list["text"] = tw_list.text.map(remove_rt).map(rt)**
  - **tw_list["text"] = tw_list.text.str.lower()**

- Con el texto limpio se calculan nuevamente los parámetros de polaridad, subjetividad, sentimiento negativo, positivo, neutral y compuesto.
  - **#Calculating Negative, Positive, Neutral and Compound values**
  - **tw_list[['polarity', 'subjectivity']] = tw_list['text'].apply(lambda Text: pd.Series(TextBlob(Text).sentiment))**
  - **for index, row in tw_list['text'].iteritems():**
  - **score = SentimentIntensityAnalyzer().polarity_scores(row)**

Continúa el código en la siguiente página.

Explicación

Continuación del código:

```
neg = score['neg']
neu = score['neu']
pos = score['pos']
comp = score['compound']
if neg > pos:
    tw_list.loc[index, 'sentiment'] = "negative"
elif pos > neg:
    tw_list.loc[index, 'sentiment'] = "positive"
else:
    tw_list.loc[index, 'sentiment'] = "neutral"
tw_list.loc[index, 'neg'] = neg
tw_list.loc[index, 'neu'] = neu
tw_list.loc[index, 'pos'] = pos
tw_list.loc[index, 'compound'] = comp
tw_list.head(10)
```

	0	text	polarity	subjectivity	sentiment	neg	neu	pos	compound
0	RT @Petethestreet1: #loweringsun on #christmas...	loweringsun on christmaslights thestrand ...	0.700	0.600000	positive	0.000	0.847	0.153	0.4404
1	RT @LondonEconomic: Protesters, very few of wh...	protesters very few of whom were wearing fac...	-0.260	0.130000	positive	0.079	0.747	0.174	0.5106
3	Photo Journal - Day 01\r\n#lockdown2 #lockdown...	photo journal day 01 lockdown2 lockdown20...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000
4	God love 'em - @SlowRichies opened the doors o...	god love em opened the doors of their res...	0.375	0.466667	positive	0.000	0.730	0.270	0.7430
5	So might wear my #addidas #prideshorts for #lo...	so might wear my addidas prideshorts for lo...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000
6	RT @basicincome_uk: BREAKING: @sianberry &...	breaking amp will be putting f click to expand output; double click to hide output					.000	0.811	0.189	0.5423
7	Praticamente è così \n#6Novembre #COVID19 #Loc...	praticamente cos 6novembre covid19 lock...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000
8	RT @ShentonStage: LOVE LETTERS, which I saw an...	love letters which i saw and loved at last...	0.400	0.488889	positive	0.000	0.649	0.351	0.8442
9	RT @erndad07: @HedgecockCentre Foodbank is supp...	foodbank is supporting and also doing foo...	-0.125	0.375000	positive	0.096	0.758	0.146	0.2500
11	Early morning walk\r\n\r\n#deserted #Lockdown2 #Lo...	early morning walk deserted lockdown2 lond...	0.100	0.300000	neutral	0.000	1.000	0.000	0.0000

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Ahora se puede separar la lista de tweets (tw_list) en tres grupos de sentimientos y hacer un recuento de cada uno y los porcentajes:

```
#Creating new data frames for all sentiments (positive, negative and neutral)
tw_list_negative = tw_list[tw_list["sentiment"]=="negative"]
tw_list_positive = tw_list[tw_list["sentiment"]=="positive"]
tw_list_neutral = tw_list[tw_list["sentiment"]=="neutral"]

def count_values_in_column(data,feature):
  total=data.loc[:,feature].value_counts(dropna=False)
  percentage=round(data.loc[:,feature].value_counts(dropna=False,normalize=True)*100,2)
  return pd.concat([total,percentage],axis=1,keys=['Total','Percentage'])
#Count_values for sentiment
count_values_in_column(tw_list,"sentiment")
```

El resultado obtenido se pasa a la siguiente tabla:

	Total	Percentage
positive	497	38.80
neutral	476	37.16
negative	308	24.04

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Finalmente, puedes generar una nube de palabras que muestra la frecuencia de las palabras utilizadas en los tweets.

```
#Function to Create Wordcloud
def create_wordcloud(text):
    mask = np.array(Image.open("cloud.png"))
    stopwords = set(STOPWORDS)
    wc = WordCloud(background_color="white",
                   mask = mask,
                   max_words=3000,
                   stopwords=stopwords,
                   repeat=True)
    wc.generate(str(text))
    wc.to_file("wc.png")
    print("Word Cloud Saved Successfully")
    path="wc.png"
    display(Image.open(path))
#Creating wordcloud for all tweets
create_wordcloud(tw_list["text"].values)
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Actividad

Para reforzar el aprendizaje del tema, realiza lo siguiente:

1. Menciona algunas aplicaciones que puedes desarrollar en Python aprovechando las oportunidades que ofrece el lenguaje de programación para el análisis de sentimiento.
2. Comenta los beneficios que tiene el análisis de sentimiento, específicamente de los noticieros digitales en el área de finanzas, como apoyo a la evaluación y selección de inversiones en la bolsa de valores.
3. Imagina que estás desarrollando un modelo analítico para identificar la percepción de los padres de familia sobre la escuela en donde estudian sus hijos. La base del modelo es un diccionario con palabras clasificadas por tipo de sentimiento, como positivo, neutral y negativo. Planeas recolectar su percepción por medio de una encuesta, con la cual sus respuestas se van a clasificar de acuerdo con el parecido o similitud con las palabras en el diccionario.
    a) Menciona 20 palabras de cada tipo de sentimiento que pueden coincidir con la respuesta del encuestado.
    b) Describe la forma en que vas a presentar los resultados del proyecto.
    c) ¿De qué otra manera puedes utilizar el análisis de sentimiento como un servicio adicional a los padres de familia de esa escuela?

Cierre

No cabe duda de la extensa aplicación de las utilerías del lenguaje Python para explorar en la información resolver algún problema o mejorar algún proceso. Con el aprendizaje de análisis de sentimiento en redes sociales, la empresa Marketing Digital tiene un gran panorama de posibilidades para aprovechar su disponibilidad y ofrecer estrategias para llegar a segmentos de mercado específicos y, por lo tanto, incrementar sus clientes y sus ingresos.

Bibliografía

- Amazon Web Services. (s.f.). *¿Qué es NLP?* Recuperado de https://aws.amazon.com/es/what-is/nlp/
- Woodward, M. (2024). *Twitter user statistics 2024: what happened after "x" rebranding?* Recuperado de https://www.searchlogistics.com/grow/statistics/twitter-user-statistics/

Tecmilenio no guarda relación alguna con las marcas mencionadas como ejemplo. Las marcas son propiedad de sus titulares conforme a la legislación aplicable, estas se utilizan con fines académicos y didácticos, por lo que no existen fines de lucro, relación publicitaria o de patrocinio.
