

# DIBUJAR TWEETS EN UN MAPA CON PYTHON

## Contenido

Objetivo de la práctica.....	2
Requisitos .....	2
Instalación de requisitos:.....	2
Comprobar que Python está instalado: .....	2
Instalar Package Installer for Python (PIP):.....	3
Instalar paquetes:.....	4
Token de acceso a la API de Twitter .....	5
Ejecución de la práctica .....	6
Credentials.py .....	6
I_DescargarTweets .....	6
II_ConvertirTweets.py.....	10
III_GraficarMapa.py .....	12
IV_AnalisisSentimientos.py .....	17

## Objetivo de la práctica

Graficar los tweets de una ciudad o zona específica, así como aplicar un análisis de sentimientos para determinar su polaridad (negativa o positiva).

## Requisitos

### Lenguaje:

- Python 3.8.x

### Herramientas y paquetes:

- Package Installer for Python (PIP)
- Token de acceso a la API de Twitter
- Spyder IDE (OPCIONAL)
- tweepy 3.9.0
- pandas 1.0.5
- numpy 1.19.0
- ipython 7.16.1
- osmviz 3.2.0
- Pillow 7.2.0
- colour 0.1.5
- imageio 2.9.0
- lxml 4.5.2
- nltk 3.5
- scikit-learn 0.23.2
- langdetect 1.0.8
- textblob 0.15.3
- joblib 0.16.0
- heatmap
- langid

## Instalación de requisitos:

### Comprobar que Python está instalado:

1. Abrir el símbolo de sistema o terminal y ejecutar el comando:

```
Python --version
```

Debe mostrar la versión de Python instalada

```
Python 3.8.3
```

En caso contrario, descargar e instalar Python del siguiente enlace:

<https://www.python.org/downloads/>

## Instalar Package Installer for Python (PIP):

Es una herramienta de línea de comandos que permite instalar, reinstalar y desinstalar paquetes de Python.

Las últimas versiones de MacOS ya tienen integrado Python y PIP. En el caso de Windows, las últimas versiones de Python cuentan con PIP integrado.

Comprobar que PIP ya está instalado:

Abrir el símbolo del sistema o terminal y ejecutar el comando:

```
pip --help
```

Debe mostrar los comandos disponibles

```
Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements format.
  list              List installed packages.
  show              Show information about installed packages.
  check             Verify installed packages have compatible dependencies.
  config            Manage local and global configuration.
  search            Search PyPI for packages.
  cache             Inspect and manage pip's wheel cache.
  wheel             Build wheels from your requirements.
  hash              Compute hashes of package archives.
  completion        A helper command used for command completion.
  debug             Show information useful for debugging.
  help              Show help for commands.

General Options:
  -h, --help        Show help.
  --isolated         Run pip in an isolated mode, ignoring environment variables and user configuration.
  -v, --verbose     Give more output. Option is additive, and can be used up to 3 times.
  -V, --version     Show version and exit.
```

Esto quiere decir que ya tenemos PIP instalado, en caso contrario seguir con los siguientes pasos para Windows (si se tiene otro sistema operativo consultar: [https://www.neoguias.com/como-instalar-pip-python/#Como\\_instalar\\_PIP\\_en\\_Mac](https://www.neoguias.com/como-instalar-pip-python/#Como_instalar_PIP_en_Mac)):

1. Ir al siguiente enlace, dar clic derecho y guardar el script en la computadora <https://bootstrap.pypa.io/get-pip.py>
2. Abrir el símbolo del sistema y navegar hasta el directorio en donde esta guardado el script anterior.
3. Ejecutar el siguiente comando:

```
python get-pip.py
```

Nos debe mostrar lo siguiente:

```
D:\>python get-pip.py
Collecting pip
  Downloading pip-20.2.2-py2.py3-none-any.whl (1.5 MB)
    | 1.5 MB 409 kB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2
    Uninstalling pip-20.2:
      Successfully uninstalled pip-20.2
Successfully installed pip-20.2.2

D:\>_
```

## Instalar paquetes:

### *Tweepy*

Paquete que permite consumir la API de Twitter.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install tweepy`

### *Pandas*

Paquete que permite el análisis y la manipulación de datos.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install pandas`

### *Numpy*

Paquete de funciones matemáticas, vectorización, entre otros.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install numpy`

### *Ipython*

Shell interactivo que añade funcionalidades extra al modo interactivo incluido con Python.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install ipython`

### *osmviz*

Paquete que permite la visualización de OpenStreetMaps.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install osmviz`

### *Pillow*

Paquete para la manipulación de imágenes.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install Pillow`

### *Colour*

Paquete que permite convertir y manipular representaciones de color (RGB, HSL, ...)

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install colour`

### *Imageio*

Paquete que permite leer y escribir un amplio rango de datos de imágenes.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install imageio`

### *lxml*

Paquete que permite la lectura de archivos XML, entre otros.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install lxml`

### *nltk*

Paquete para trabajar con datos de lenguaje natural, permite la tokenización. stemming, entre otros.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install nltk`

### *Scikit-learn*

Paquete para machine learning.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install scikit-learn`

### *Langdetect*

Paquete que permite detectar el idioma de un texto.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install langdetect`

### *Textblob*

Paquete para procesar datos textuales.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install textblob`

### *Joblib*

Paquete para programación paralela.

- Abrir el símbolo del sistema y ejecutar el comando:  
`pip install joblib`

## **Token de acceso a la API de Twitter**

Se debe solicitar una cuenta de Twitter de desarrollador en el siguiente enlace:

<https://developer.twitter.com/en/apply-for-access>

Una vez aceptados, se nos proporcionará los siguientes datos de acceso a la API de Twitter: CONSUMER\_KEY, CONSUMER\_SECRET, ACCESS\_TOKEN, ACCESS\_SECRET.

## Ejecución de la práctica

A continuación, se mostrará el contenido, entradas y salidas de los scripts que constituyen a la práctica.

### Credentials.py

En este script solo se definen los datos de acceso a la API de Twitter.

```
CONSUMER_KEY = "XXXXXXXXXX"
CONSUMER_SECRET = "XXXXXXXX"

ACCESS_TOKEN = "XXXXXXXXXX"
ACCESS_SECRET = "XXXXXXXXXX"
```

### I\_DescargarTweets

Este script permite descargar tweets en tiempo real de una ciudad o zona especificada.

- ***Paquetes necesarios.***

Entrada:

```
import json
import tweepy
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
```

- ***Importar los datos de acceso a la API de Twitter, definidos en el script Credentials.py.***

Entrada:

```
from credentials import *
```

- ***Establecer acceso a la API de Twitter.***

Entrada:

```
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)
api = tweepy.API(auth)
```

- ***Definir las coordenadas del área del cual se quiere recuperar los tweets. En esta práctica las ciudades elegidas son Tampico y Ciudad Madero.***

Para consultar las coordenadas de una ciudad puede utilizar el siguiente sitio web: <https://www.geodatos.net/>

geodatos Antípodas **Coordenadas** Distancias Más Q en

Inicio > Coordenadas > México

## Coordenadas geográficas de Tampico

Tampico se encuentra en la latitud 22.28519 y longitud -97.87777. Hace parte del continente de América y está ubicado en el hemisferio norte.

<b>Coordenadas decimales</b> Formato simple <b>22.28519, -97.87777</b>	<b>Coordenadas GD</b> Grados decimales <b>22.2852° N 97.8778° O</b>	<b>Coordenadas GMS</b> Grados, minutos y segundos <b>22°17'6.7" N 97°52.666' O</b>
--	---	--

geodatos Antípodas **Coordenadas** Distancias Más Q en

Inicio > Coordenadas > México

## Coordenadas geográficas de Ciudad Madero

Ciudad Madero se encuentra en la latitud 22.27228 y longitud -97.83623. Hace parte del continente de América y está ubicado en el hemisferio norte.

<b>Coordenadas decimales</b> Formato simple <b>22.27228, -97.83623</b>	<b>Coordenadas GD</b> Grados decimales <b>22.2723° N 97.8362° O</b>	<b>Coordenadas GMS</b> Grados, minutos y segundos <b>22°16'20.2" N 97°50.174' O</b>
--	---	---

### Entrada:

```
tampico_madero = [-97.87777, 22.28519, -96.87777, 23.28519,
-97.83623, 22.27228, -96.83623, 23.27228]
```

los primeros dos pares corresponden a las coordenadas de Tampico y los últimos 2 pares a Ciudad Madero.

\* Nota: las coordenadas deben proporcionarse de la siguiente manera:  
[sw\_longitude, sw\_latitude, ne\_longitude, ne\_latitude]

- **Definir la clase que permitirá el uso del Twitter Streaming API, el cual permite obtener tweets publicados en tiempo real.**

### Entrada:

```
class listener(tweetpy.StreamListener):

    def __init__(self, numero_tweets):
        self.received_tweets_counter = 0
        self.max_number_tweets = numero_tweets
        #Directorio y nombre del archivo donde guardaremos los
        #tweets que se vayan publicando
        self.file = open('tweets/tweets_tampico_madero.txt', 'a', encoding=
"UTF-8")
        super(listener, self).__init__()

    def on_data(self, data):
        if (self.received_tweets_counter < self.max_number_tweets):

            self.received_tweets_counter += 1
```

```

        #La API de Twitter devuelve datos en formato JSON,
        #asi que hay que decodificarlos.
        try:
            decoded = json.loads(data)
        except Exception as e:
            print(e)
            return True

        #No todos los usuarios tienen habilitada la opcion de
        #geolocalizacion
        #Por ello hay que dar formato a cuando no este disponible
        if decoded.get('geo') is not None:
            location = str(decoded.get('geo').get('coordinates'))
        else:
            location = '[,]'

        #Extraer los datos que nos interese de los tweets
        text = decoded['text'].replace('\n', ' ')
        user = '@' + decoded.get('user').get('screen_name')
        created = decoded.get('created_at')

        #Escribir los tweets en el archivo de texto
        self.file.write(user + "|" + location + "|" + created + "|" + t
ext + "\n")

        return True
    else:
        self.file.close()
        print('Done!')
        return False

#def on_status(self, status):
#    print(status.text)

def on_error(self, status):
    print(status)
    #Debido a que el uso de la API de twitter tiene un limite diario
    #debemos
    #desconectarnos del stream cuando excedamos dicho limite
    if status == 420:
        print('status code 420: ' + status)
        self.file.close()
        #Retornando un False en on_error Conseguimos desconectarnos de
        #Stream
        return False
    self.file.close()

```



- **Comenzar a capturar los tweets en el archivo de texto**  
(tweets\_tampico\_madero.txt) **definido en la clase anterior.**

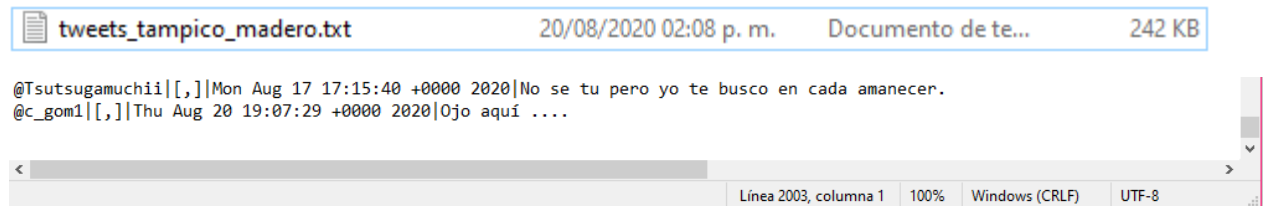
Entrada:

```
if __name__ == '__main__':  
  
    print('Starting...')  
    #Crear un Stream, estableciendo el número de  
    #tweets que se desean guardar  
    twitterStream = tweepy.Stream(auth, listener(numero_tweets = 1))  
    #Filtrar los tweets que coincidan con las coordenadas especificadas  
    twitterStream.filter(locations = tampico_madero)
```

Salida:

```
Starting...  
Done!
```

Así como el archivo de texto con los tweets.



\* Nota 1: El proceso de filtrar y guardar tweets en el archivo de texto (tweets\_tampico\_madero.txt) puede tomar mucho tiempo en terminar, dependiendo del número de tweets establecidos para ser guardados.

\* Nota 2: Debido a que no todos los usuarios tienen habilitada la opción de “Acceso a la ubicación” en sus dispositivos, la información de las coordenadas viene vacía, por ello es recomendable guardar la mayor cantidad de tweets posibles para intentar conseguir un buen número de tweets con coordenadas.

Más información en: <https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters>

## II\_ConvertirTweets.py

Este script convierte los tweets del archivo de texto (tweets\_tampico\_madero.txt) a un data frame y a un archivo CSV (tweets\_tampico\_madero.csv), para su posterior análisis y manipulación. También genera un archivo (tweets\_heatmap\_tam\_mad) que contiene solamente las latitudes y longitudes de cada tweet.

- **Paquetes necesarios.**

Entrada:

```
import pandas as pd
import numpy as np
```

- **Leer el archivo de texto (tweets\_tampico\_madero.txt) y definir un data frame para almacenar los tweets de ese archivo.**

Entrada:

```
tweets_raw = pd.read_table('tweets/tweets_tampico_madero.txt', header=None,
iterator=True)
tweets_2 = pd.DataFrame()
```

- **Convertir los datos del archivo de texto (tweets\_tampico\_madero.txt) a un data frame y a un archivo CSV (tweets\_tampico\_madero.csv).**

Entrada:

```
while 1:
    tweets = tweets_raw.get_chunk(1000) #1000 filas por chunk
    tweets.columns = ['tweets']
    tweets['len'] = tweets.tweets.apply(Lambda x: len(x.split('|')))
    tweets[tweets.len < 4] = np.nan
    del tweets['len']
    tweets = tweets[tweets.tweets.notnull()]
    #Establecer las columnas y valores que tomarán
    tweets['user'] = tweets.tweets.apply(Lambda x: x.split('|')[0])
    tweets['geo'] = tweets.tweets.apply(Lambda x: x.split('|')[1])
    tweets['timestamp'] = tweets.tweets.apply(Lambda x: x.split('|')[2])
    tweets['tweet'] = tweets.tweets.apply(Lambda x: x.split('|')[3])
    tweets['lat'] = tweets.geo.apply(Lambda x: x.split(',')[0].replace('[',
''))
    tweets['lon'] = tweets.geo.apply(Lambda x: x.split(',')[1].replace(']',
''))
    del tweets['tweets']
    del tweets['geo']
    #Convertir las latitudes y longitudes de string a float
    tweets['lon'] = pd.to_numeric(tweets['lon'], downcast="float")
    tweets['lat'] = pd.to_numeric(tweets['lat'], downcast="float")
    #Cambiar la zona horaria de UTC a GMT-5
    tweets['timestamp'] = pd.to_datetime(tweets['timestamp'], utc = False)
```

```

tweets = tweets.set_index('timestamp').tz_convert('America/Mexico_City')
).reset_index()
#Almacenar los tweets en el dataframe
tweets_2 = tweets_2.append(tweets, ignore_index = True)
#Guardar los tweets en un archivo CSV
tweets.to_csv('tweets/tweets_tampico_madero.csv', mode='a', header=False, index=False)

```

Salida:

StopIteration

Así como un archivo CSV con los tweets

tweets_tampico_madero.csv		20/08/2020 05:48 p. m.	Archivo de valores...	234 KB
2006	2020-08-20 14:33:56-05:00	@OSCARLSAUCEDOH restauranteelflaco #cdmadero #video #youtube #Y% OSCARLSH #ostiones #mariscos #camarones #ostionesensuconcha #madero&#x2113; https://t.co/ZYaaLDHyjL	22.25297	-97.84775
2007	2020-08-20 14:34:38-05:00	@DayaniYatra A DÁ*NDE VAN SebastiÁn Yatra		

\* Nota: Ya que la cantidad de datos en el archivo de texto (tweets\_tampico\_madero.txt) es muy grande, se recomienda procesar cierta cantidad de filas, en lugar de cargar todo el archivo en memoria. En el ciclo while anterior se procesan 1000 filas del archivo de texto hasta que ya no haya más filas. Dependiendo de la cantidad de tweets el proceso puede tardar en finalizar.

- **Modificar el data frame con los tweets, para almacenar solo aquellos que tengan coordenadas y que a su vez se encuentren en el rango del área que deseamos.**

Entrada:

```

min_lon = -97.99
min_lat = 22.20
max_lon = -96.0
max_lat = 23.30

#Descartar las filas con latitud y longitud nulas
tweets_2 = tweets_2[(tweets_2.lat.notnull()) & (tweets_2.lon.notnull())]

#Filtrar los tweets que estén dentro de la zona que nos interese
tweets_2 = tweets_2[(tweets_2.lon >= min_lon) & (tweets_2.lon <= max_lon) &
(tweets_2.lat >= min_lat) & (tweets_2.lat <= max_lat)]

```

- **Guardar las coordenadas de los tweets en un archivo.**

Entrada:

```

with open('heatmap-files/tweets_heatmap_tam_mad','w') as file:
    file.write(tweets_2[['lat','lon']].to_string(header=False, index=False)
)

```

## Salida:

Un archivo con las coordenadas de los tweets

tweets_heatmap_tam_mad	20/08/2020 06:43 p. m.	Archivo	1 KB
tweets_heatmap_tam_mad: Blo			
Archivo	Edición	Formato	Ver
22.264406	-97.785187		
22.255301	-97.868599		
22.279510	-97.886620		
22.264650	-97.786034		
22.255301	-97.868599		

### III\_GraficarMapa.py

Este script permite graficar las coordenadas del archivo (tweets\_heatmap\_tam\_mad) en un mapa, con ayuda del script heatmap.py

- **Paquetes necesarios.**

Entrada:

```
import PIL
import osmviz
from PIL import Image
from PIL import ImageDraw, ImageFont
from IPython.display import Image
from IPython import get_ipython
import numpy as np
import imageio
from colour import Color
import pandas as pd
from copy import deepcopy
```

- **Graficar las coordenadas en un mapa.**

#### OPCIÓN 1: Usando el símbolo del sistema

Abrir el símbolo del sistema y navegar hasta el directorio de nuestro proyecto y ejecutar el siguiente comando.

Entrada:

```
python heatmap.py -o maps/map_tweets_tam_mad_01.png --width 1920 -p heatmap-files/tweets_heatmap_tam_mad -b black -P equirectangular --osm --osm_base http://tile.memomaps.de/tilegen --decay 0.8 -r 10 --zoom 0 --margin 15
```

-o Directorio y nombre del archivo.

--width Ancho de la imagen en pixeles.

-p Directorio y nombre del archivo que contiene las coordenadas.

--osm Especificar que se usará un mapa editable, OpenStreetMaps.

--osb\_base Estilo del mapa (tiles url). Puede consultar mas estilos en:

[https://wiki.openstreetmap.org/wiki/Tile\\_servers](https://wiki.openstreetmap.org/wiki/Tile_servers).

**OPCION 2:** Usando el IDE Spyder, Jupiter Qt console o Jupiter notebook  
Simplemente ejecutar la siguiente línea

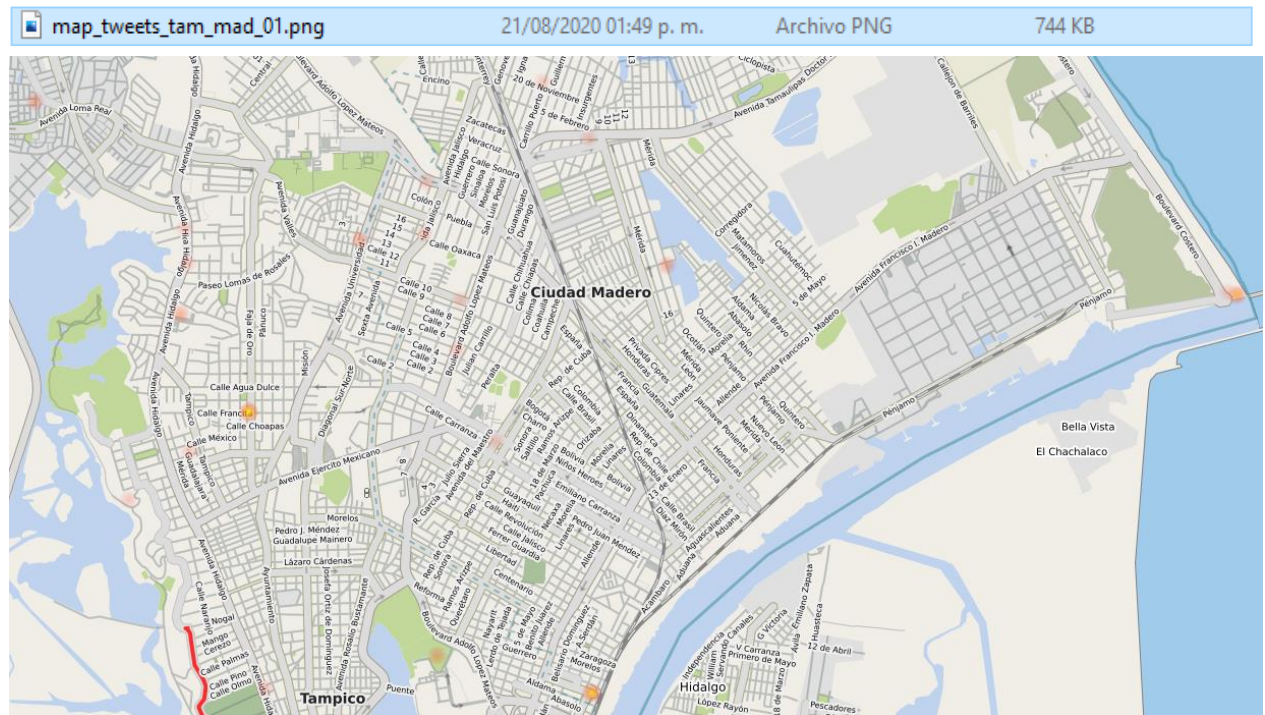
Entrada:

```
get_ipython().system('python heatmap.py -o maps/map_tweets_tam_mad_01.png -  
-width 1920 -p heatmap-files/tweets_heatmap_tam_mad -b black -  
P equirectangular --osm --osm_base http://tile.memomaps.de/tilegen --  
decay 0.8 -r 10 --zoom 0 --margin 15')
```

Salida:

```
Fetching tiles:  
100%|  
18/18 [00:00<00:00, 142.70tile/s]
```

Así como una imagen con las coordenadas graficadas en un mapa.



- **Agregar una leyenda al mapa de los tweets; como el nombre de la ciudad, la cantidad de tweets graficados y la fecha.**

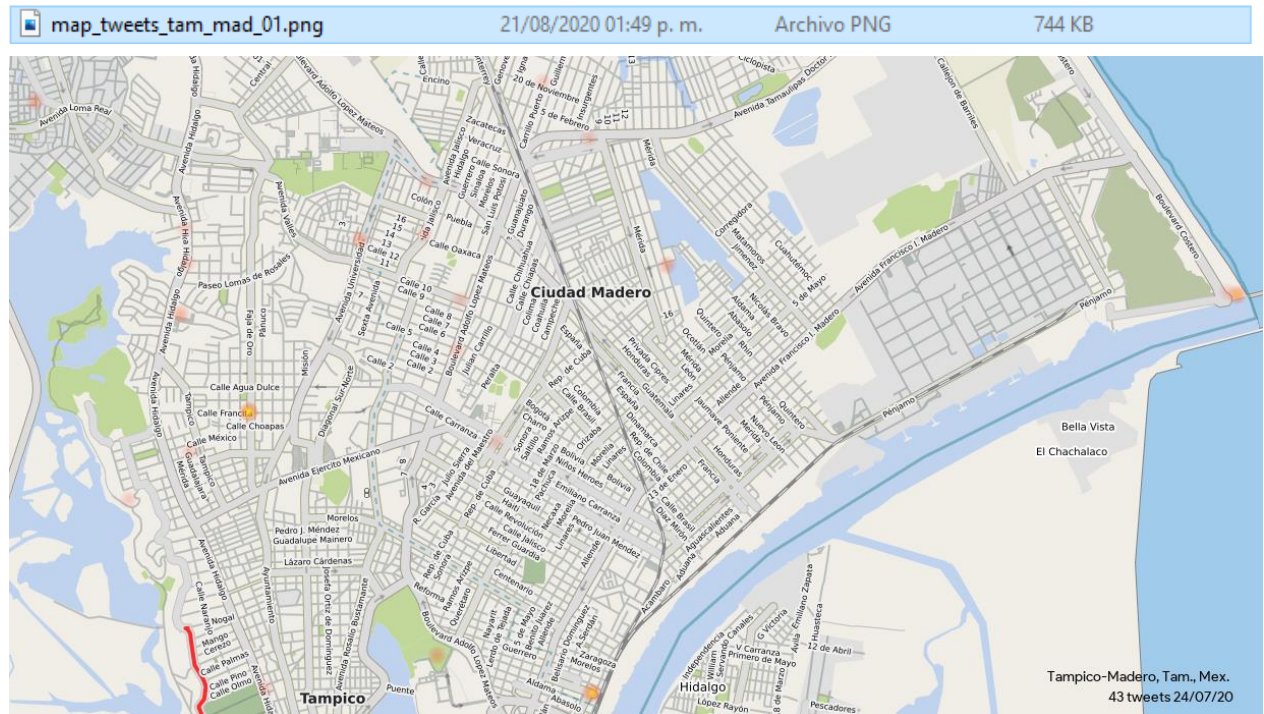
Entrada:

```
im = PIL.Image.open('maps/map_tweets_tam_mad_01.png')  
draw = ImageDraw.Draw(im)  
font = ImageFont.truetype("fonts/ProductSans.ttf", 14)  
draw.text((1020, 600), "Tampico-  
Madero, Tam., Mex.", fill = "black", font=font)  
draw.text((1080, 620), "43 tweets 24/07/20", fill = "black", font=font)  
  
im.save('maps/map_tweets_tam_mad_01.png')
```



## Salida:

La imagen de los puntos graficados en un mapa, pero ahora con el texto que agregamos.



- **Definir un nuevo gradiente de color que sustituya los puntos amarillos que marca heatmap.py por defecto.**

## Entrada:

```
hsva_min = Color()
hsva_min.hex_l = '#008dcd'

hsva_max = Color()
hsva_max.hex_l = '#4dccff'

color_gradient = list(hsva_max.range_to(hsva_min,256))
alpha = np.arange(0,256)[::-1]

gradient = []
for i, color_point in enumerate(color_gradient):
    rgb = list(color_point.get_rgb())
    rgb = [int(e * 255) for e in rgb]
    rgb.append(alpha[i])
    gradient.append([rgb])
color_gradient = np.array(gradient)

width = 43
from copy import deepcopy
```

```

color_gradient_row = deepcopy(color_gradient)

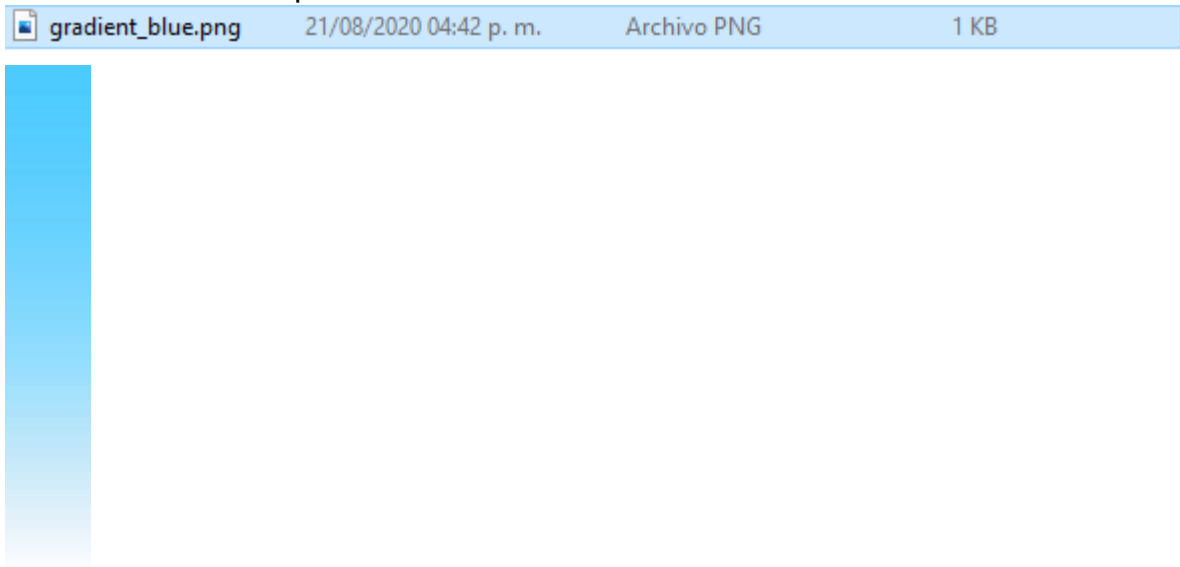
for col in range(width-1):
    color_gradient = np.hstack((color_gradient, color_gradient_row))

imageio.imwrite('gradients/gradient_blue.png', color_gradient)

```

**Salida:**

Gradiente del color que definimos anteriormente



- ***Graficar las coordenadas utilizando el gradiente de color definido anteriormente.***

#### **OPCIÓN 1: Usando el símbolo del sistema**

Abrir el símbolo del sistema y navegar hasta el directorio de nuestro proyecto y ejecutar el siguiente comando.

**Entrada:**

```

python heatmap.py -G gradients/gradient_blue.png -
o maps/map_tweets_tam_mad_02.png --width 1920 -p heatmap-
files/tweets_heatmap_tam_mad -b black -P equirectangular --osm --
osm_base https://basemaps.cartocdn.com/rastertiles/dark_all --decay 0.8 -
r 10 --zoom 0 --margin 15

```

**-G** Directorio y nombre de la imagen del gradiente.

**OPCION 2:** Usando el IDE Spyder, Jupiter Qt console o Jupiter notebook  
Simplemente ejecutar la siguiente línea

Entrada:

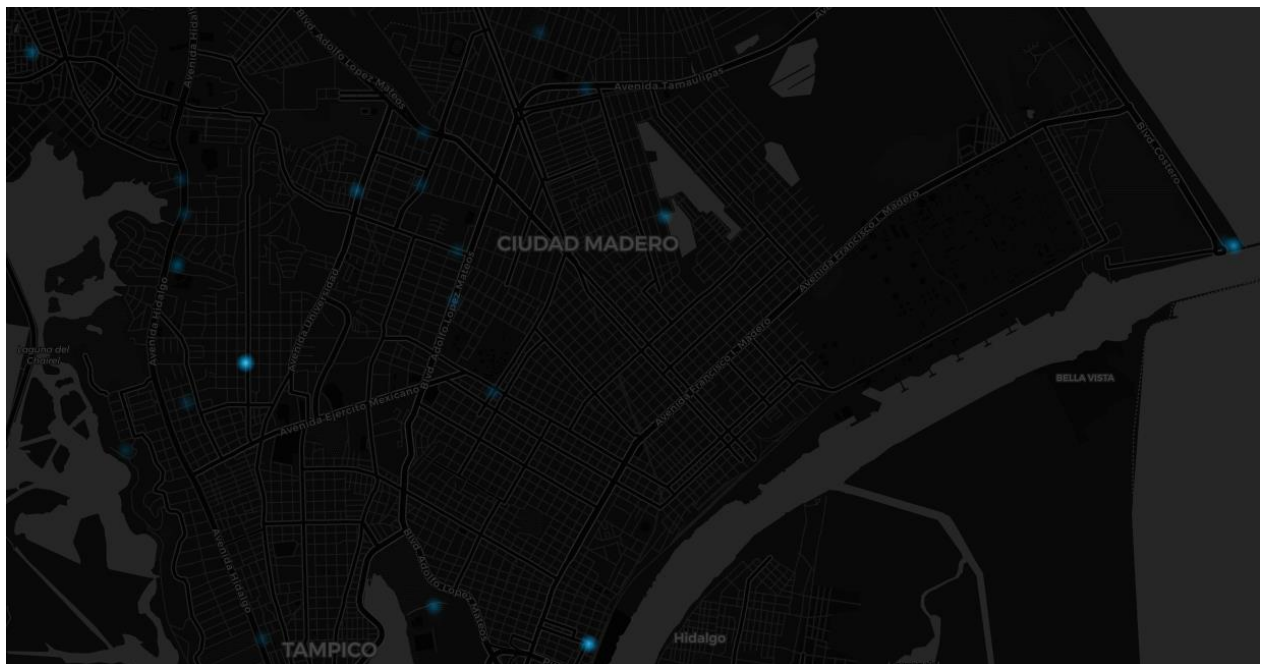
```
get_ipython().system('python heatmap.py -G gradients/gradient_blue.png -  
o maps/map_tweets_tam_mad_02.png --width 1920 -p heatmap-  
files/tweets_heatmap_tam_mad -b black -P equirectangular --osm --  
osm_base https://basemaps.cartocdn.com/rastertiles/dark_all --decay 0.8 -  
r 10 --zoom 0 --margin 15')
```

Salida:

```
Fetching tiles:  
100% |  
| 18/18 [00:00<00:00, 142.70tile/s]
```

Y un mapa con el nuevo color de puntos

map_tweets_tam_mad_02.png	27/07/2020 05:05 a. m.	Archivo PNG	285 KB
---------------------------	------------------------	-------------	--------





## IV\_AnalisisSentimientos.py

En este script se predice si los tweets son positivos o negativos, mediante machine learning, para posteriormente ser graficados.

- ***Paquetes necesarios.***

Entrada:

```
import PIL
import osmviz
from PIL import Image
from PIL import ImageDraw, ImageFont
from IPython.display import Image
from IPython import get_ipython
import numpy as np
import imageio
from colour import Color
from copy import deepcopy
import pandas as pd
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
from string import punctuation
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import SnowballStemmer
from nltk.tokenize import word_tokenize
from sklearn.model_selection import cross_val_score
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
import joblib
import langid
from langdetect import detect
import textblob
```









- ***Descargar los dataset que se usarán para predecir los sentimientos de los tweets.***

The Spanish Society for Natural Language Processing (SEPLN) proporciona varios dataset para el análisis de sentimientos de forma gratuita para propósitos académicos.

Para poder descargar los dataset debe registrarse en el siguiente enlace:

[http://tass.sepln.org/tass\\_data/download.php](http://tass.sepln.org/tass_data/download.php)

De cualquier forma, en la carpeta TASS-Dataset ya se encuentran descargados algunos archivos para el análisis de sentimientos.

 general-test-tagged.xml	26/07/2020 10:07 a. m.	Documento XML	24,251 KB
 general-test-tagged-mx.xml	26/07/2020 11:50 a. m.	Documento XML	167 KB
 general-train-tagged.xml	26/07/2020 10:02 a. m.	Documento XML	3,540 KB
 general-train-tagged-mx.xml	26/07/2020 11:49 a. m.	Documento XML	324 KB
 socialtv-test-tagged.xml	26/07/2020 10:08 a. m.	Documento XML	240 KB
 socialtv-train-tagged.xml	26/07/2020 10:07 a. m.	Documento XML	429 KB
 stompol-test-tagged.xml	26/07/2020 10:08 a. m.	Documento XML	135 KB
 stompol-train-tagged.xml	26/07/2020 10:08 a. m.	Documento XML	214 KB

- **Convertir cada dataset descargado a un archivo CSV y a un data frame**

Entrada:

```
pd.set_option('max_colwidth',1000)
try:
    general_tweets_corpus_train = pd.read_csv('TASS-Dataset/general-train-
tagged.csv', encoding='utf-8')
except:

    from lxml import objectify
    xml = objectify.parse(open('TASS-Dataset/general-train-
tagged.xml', encoding="UTF-8"))
    root = xml.getroot()
    general_tweets_corpus_train = pd.DataFrame(columns=('content', 'polarit
y', 'agreement'))
    tweets = root.getchildren()
    for i in range(0,len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'], [tweet.content
.text, tweet.sentiments.polarity.value.text, tweet.sentiments.polarity.type
.text]))
        row_s = pd.Series(row)
        row_s.name = i
        general_tweets_corpus_train = general_tweets_corpus_train.append(ro
w_s)
    general_tweets_corpus_train.to_csv('TASS-Dataset/general-train-
tagged.csv', index=False, encoding='utf-8')

try:
    general_tweets_corpus_test = pd.read_csv('TASS-Dataset/general-test-
tagged.csv', encoding='utf-8')
except:

    from lxml import objectify
    xml = objectify.parse(open('TASS-Dataset/general-test-
tagged.xml', encoding="UTF-8"))
    root = xml.getroot()
```

```

general_tweets_corpus_test = pd.DataFrame(columns=('content', 'polarity'))
tweets = root.getchildren()
for i in range(0, len(tweets)):
    tweet = tweets[i]
    row = dict(zip(['content', 'polarity', 'agreement'], [tweet.content
.text, tweet.sentiments.polarity.value.text]))
    row_s = pd.Series(row)
    row_s.name = i
    general_tweets_corpus_test = general_tweets_corpus_test.append(row_s)

general_tweets_corpus_test.to_csv('TASS-Dataset/general-test-
tagged.csv', index=False, encoding='utf-8')

try:
    stompol_tweets_corpus_train = pd.read_csv('TASS-Dataset/stompol-train-
tagged.csv', encoding='utf-8')
except:

    from lxml import objectify
    xml = objectify.parse(open('TASS-Dataset/stompol-train-
tagged.xml', encoding = "UTF-8"))
    root = xml.getroot()
    stompol_tweets_corpus_train = pd.DataFrame(columns=('content', 'polarit
y'))
    tweets = root.getchildren()
    for i in range(0, len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'], [' '.join(list
(tweet.itertext()))], tweet.sentiment.get('polarity'))))
        row_s = pd.Series(row)
        row_s.name = i
        stompol_tweets_corpus_train = stompol_tweets_corpus_train.append(ro
w_s)

    stompol_tweets_corpus_train.to_csv('TASS-Dataset/stompol-train-
tagged.csv', index=False, encoding='utf-8')

try:
    stompol_tweets_corpus_test = pd.read_csv('TASS-Dataset/stompol-test-
tagged.csv', encoding='utf-8')
except:

    from lxml import objectify

```

```

    xml = objectify.parse(open('TASS-Dataset/stompol-test-
tagged.xml', encoding = "UTF-8"))
    root = xml.getroot()
    stompol_tweets_corpus_test = pd.DataFrame(columns=('content', 'polarity
'))
    tweets = root.getchildren()
    for i in range(0,len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'], [' '.join(list
(tweet.itertext()))], tweet.sentiment.get('polarity'))))
        row_s = pd.Series(row)
        row_s.name = i
        stompol_tweets_corpus_test = stompol_tweets_corpus_test.append(row_
s)
    stompol_tweets_corpus_test.to_csv('TASS-Dataset/stompol-test-
tagged.csv', index=False, encoding='utf-8')

try:
    social_tweets_corpus_test = pd.read_csv('TASS-Dataset/socialtv-test-
tagged.csv', encoding='utf-8')
except:

    from lxml import objectify
    xml = objectify.parse(open('TASS-Dataset/socialtv-test-
tagged.xml', encoding = "UTF-8"))
    root = xml.getroot()
    social_tweets_corpus_test = pd.DataFrame(columns=('content', 'polarity'
))
    tweets = root.getchildren()
    for i in range(0,len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'], [' '.join(list
(tweet.itertext()))], tweet.sentiment.get('polarity'))))
        row_s = pd.Series(row)
        row_s.name = i
        social_tweets_corpus_test = social_tweets_corpus_test.append(row_s)
    social_tweets_corpus_test.to_csv('TASS-Dataset/socialtv-test-
tagged.csv', index=False, encoding='utf-8')

try:
    social_tweets_corpus_train = pd.read_csv('TASS-Dataset/socialtv-train-
tagged.csv', encoding='utf-8')
except:

```

```

    from lxml import objectify
    xml = objectify.parse(open('TASS-Dataset/socialtv-train-
tagged.xml', encoding = "UTF-8"))
    root = xml.getroot()
    social_tweets_corpus_train = pd.DataFrame(columns=('content', 'polarity
'))
    tweets = root.getchildren()
    for i in range(0,len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'], [' '.join(list
(tweet.itertext())) , tweet.sentiment.get('polarity')]))
        row_s = pd.Series(row)
        row_s.name = i
        social_tweets_corpus_train = social_tweets_corpus_train.append(row_
s)
    social_tweets_corpus_train.to_csv('TASS-Dataset/socialtv-train-
tagged.csv', index=False, encoding='utf-8')

try:
    general_tweets_corpus_train_mx = pd.read_csv('TASS-Dataset/general-
train-tagged-mx.csv', encoding='utf-8')
except:

    from lxml import objectify
    xml = objectify.parse(open('TASS-Dataset/general-train-tagged-
mx.xml', encoding="UTF-8"))
    #sample tweet object
    root = xml.getroot()
    general_tweets_corpus_train_mx = pd.DataFrame(columns=('content', 'pola
rity'))
    tweets = root.getchildren()
    for i in range(0,len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'], [tweet.content
.text, tweet.sentiment.polarity.value.text]))
        row_s = pd.Series(row)
        row_s.name = i
        general_tweets_corpus_train_mx = general_tweets_corpus_train_mx.app
end(row_s)
    general_tweets_corpus_train_mx.to_csv('TASS-Dataset/general-train-
tagged-mx.csv', index=False, encoding='utf-8')

try:

```

```

general_tweets_corpus_test_mx = pd.read_csv('TASS-Dataset/general-test-
tagged-mx.csv', encoding='utf-8')
except:

    from lxml import objectify
    xml = objectify.parse(open('TASS-Dataset/general-test-tagged-
mx.xml', encoding="UTF-8"))
    root = xml.getroot()
    general_tweets_corpus_test_mx = pd.DataFrame(columns=('content', 'polar
ity'))
    tweets = root.getchildren()
    for i in range(0,len(tweets)):
        tweet = tweets[i]
        row = dict(zip(['content', 'polarity', 'agreement'], [tweet.content
.text, tweet.sentiment.polarity.value.text]))
        row_s = pd.Series(row)
        row_s.name = i
        general_tweets_corpus_test_mx = general_tweets_corpus_test_mx.appe
nd(row_s)
    general_tweets_corpus_test_mx.to_csv('TASS-Dataset/general-test-tagged-
mx.csv', index=False, encoding='utf-8')

```

- **Concatenar los dataset anteriores en un solo data frame corpus.**

Entrada:

```

tweets_corpus = pd.concat([
    social_tweets_corpus_train,
    social_tweets_corpus_test,
    stompol_tweets_corpus_test,
    stompol_tweets_corpus_train,
    general_tweets_corpus_test,
    general_tweets_corpus_train,
    general_tweets_corpus_test_mx,
    general_tweets_corpus_train_mx

])

```

- **Modificar el dataframe corpus (tweets\_corpus) para que descarte los tweets con polaridad neutral.**

El data frame corpus (tweets\_corpus) tiene un nuevo campo, por tweet, llamado 'agreement', el cual solo puede tener dos posibles valores 'AGREEMENT' Y 'DISAGREEMENT'. El valor 'DISAGREEMENT' corresponde a tweets con polaridad neutral.

Entrada:

```
tweets_corpus = tweets_corpus.query('agreement != "DISAGREEMENT" and polarity != "NONE"')
```

- **Definir las funciones para la tokenización y stemming.**

Entrada:

```
#Eliminar enlaces
tweets_corpus = tweets_corpus[~
tweets_corpus.content.str.contains('^http.*$')]
spanish_stopwords = stopwords.words('spanish')
non_words = list(punctuation)
non_words.extend(['¿', '¡'])
non_words.extend(map(str, range(10)))
stemmer = SnowballStemmer('spanish')
def stem_tokens(tokens, stemmer):
    stemmed = []
    for item in tokens:
        stemmed.append(stemmer.stem(item))
    return stemmed

def tokenize(text):
    # Eliminar caracteres que no sean letras
    text = ''.join([c for c in text if c not in non_words])
    # tokenize
    tokens = word_tokenize(text)

    # stem
    try:
        stems = stem_tokens(tokens, stemmer)
    except Exception as e:
        print(e)
        print(text)
        stems = []
    return stems
```

- **Preparar el modelo que evaluará los tweets.**

Agregar una nueva columna al data frame corpus (tweets\_corpus) en donde las polaridades de los tweets serán representadas de forma binaria.

Entrada:

```
tweets_corpus = tweets_corpus[tweets_corpus.polarity != 'NEU']

tweets_corpus['polarity_bin'] = 0
tweets_corpus.polarity_bin[tweets_corpus.polarity.isin(['P', 'P+'])] = 1
tweets_corpus.polarity_bin.value_counts(normalize=True)
```


- **Definir el modelo de evaluación, realizar un GridSearch para encontrar los hiperparámetros óptimos y guardar el modelo.**

Entrada:

```
vectorizer = CountVectorizer(  
    analyzer = 'word',  
    tokenizer = tokenize,  
    lowercase = True,  
    stop_words = spanish_stopwords)  
  
pipeline = Pipeline([  
    ('vect', vectorizer),  
    ('cls', LinearSVC()),  
])  
  
parameters = {  
    'vect__max_df': (0.5, 1.9),  
    'vect__min_df': (10, 20, 50),  
    'vect__max_features': (500, 1000),  
    'vect__ngram_range': ((1, 1), (1, 2)), # unigrams or bigrams  
    'cls__C': (0.2, 0.5, 0.7),  
    'cls__loss': ('hinge', 'squared_hinge'),  
    'cls__max_iter': (500, 1000)  
}  
  
grid_search = sklearn.model_selection.GridSearchCV(pipeline, parameters, n_  
jobs=-1, scoring='roc_auc')  
grid_search.fit(tweets_corpus.content, tweets_corpus.polarity_bin)  
  
grid_search.best_params_  
  
#Guardar el modelo  
joblib.dump(grid_search, 'grid_search.pkl')
```

Salida:

El modelo

 grid_search.pkl	27/07/2020 02:48 a. m.	Archivo PKL	1,110 KB
---	------------------------	-------------	----------

\* Nota: Debido a la enorme cantidad de datos en el data frame corpus (tweets\_corpus), el modelo puede tomar mucho tiempo en terminar. En un equipo con 6gb de RAM y con un procesador de 4 nucleos a 2.2GHz, finalizó tras 12 horas.

Si quiere omitir el proceso anterior, el modelo ya se encuentra en la carpeta del proyecto. Solo debe cargarlo de la siguiente manera:

```
grid_search = joblib.load('grid_search.pkl')
```



- **Hacer una validación cruzada para mostrar el rendimiento del modelo.**

Entrada:

```
model = LinearSVC(C=.2, loss='squared_hinge',max_iter=1000,multi_class='ovr',
    random_state=None,
    penalty='l2',
    tol=0.0001
)

vectorizer = CountVectorizer(
    analyzer = 'word',
    tokenizer = tokenize,
    lowercase = True,
    stop_words = spanish_stopwords,
    min_df = 50,
    max_df = 1.9,
    ngram_range=(1, 1),
    max_features=1000
)

corpus_data_features = vectorizer.fit_transform(tweets_corpus.content)
corpus_data_features_nd = corpus_data_features.toarray()

scores = cross_val_score(
    model,
    corpus_data_features_nd[0:len(tweets_corpus)],
    y=tweets_corpus.polarity_bin,
    scoring='roc_auc',
    cv=5
)

scores.mean()
```

Salida:

AUC

0.9168214861317802

- **Convertir el archivo CSV con los tweets** (*tweets\_tampico\_madero.csv*) **en un dataframe y seleccionar aquellos que se encuentren dentro del área que nos interesa.**

Entrada:

```
tweets = pd.read_csv('tweets/tweets_tampico_madero.csv', names = ["timestamp", "user", "tweet", "lat", "lon"], encoding='utf-8')
tweets = tweets[tweets.tweet.str.len() < 150]
tweets.lat = pd.to_numeric(tweets.lat, errors='coerce')
tweets = tweets[tweets.lat.notnull()]
min_lon = -97.99
min_lat = 22.20
max_lon = -96.0
max_lat = 23.30

tweets = tweets[(tweets.lat.notnull()) & (tweets.lon.notnull())]

tweets = tweets[(tweets.lon > min_lon) & (tweets.lon < max_lon) & (tweets.lat > min_lat) & (tweets.lat < max_lat)]
```

- **Detectar el idioma de los tweets, para posteriormente guardar aquellos que estén en español.**

Entrada:

```
def langid_safe(tweet):
    try:
        return langid.classify(tweet)[0]
    except Exception as e:
        pass

def langdetect_safe(tweet):
    try:
        return detect(tweet)
    except Exception as e:
        pass

def textblob_safe(tweet):
    try:
        return textblob.TextBlob(tweet).detect_language()
    except Exception as e:
        pass

#Puede tomar un tiempo en terminar
tweets['lang_langid'] = tweets.tweet.apply(langid_safe)
tweets['lang_langdetect'] = tweets.tweet.apply(langdetect_safe)
tweets['lang_textblob'] = tweets.tweet.apply(textblob_safe)
tweets['lang_textblob'] = tweets.tweet.apply(textblob_safe)
```

- **Exportar el data frame de los tweets con el campo de identificación de idioma a un archivo CSV** (*tweets\_tampico\_madero\_2.csv*).

Entrada:

```
tweets.to_csv('heatmap-files/tweets_tampico_madero_2.csv', encoding='utf-8')
```

Salida:

Un archivo CSV con los tweets y su identificador de idioma

tweets_tampico_madero_2.csv									
21/08/2020 09:45 p. m. Archivo de valores... 9 KB									
	A	B	C	D	E	F	G	H	I
1		timestamp	user	tweet	lat	lon	lang_langid	lang_langde	lang_textblob
2	7	2020-07-22 1	@OSCARLSA	Sigan ðŸ'‰g	22.264406	-97.78519	eo	es	es
3	74	2020-07-22 1	@ElliePerez	ðŸ'š en Tamp	22.2553	-97.8686	si	es	es

- **Modificar el data frame de los tweets para solo guardar aquellos que estén en idioma español.**

Entrada:

```
tweets = tweets.query(''' lang_langdetect == 'es' or lang_langid == 'es' or lang_textblob == 'es' ''')
```

- **Utilizar el modelo entrenado para predecir los sentimientos de lo tweets y exportar el data frame resultante a un archivo CSV** (*tweets\_polarity\_bin.csv*) **y a un archivo de texto** (*tweets\_heatmap\_polarity\_binary*) **para ser graficado posteriormente.**

Entrada:



```
pipeline = Pipeline([
    ('vect', CountVectorizer(
        analyzer = 'word',
        tokenizer = tokenize,
        lowercase = True,
        stop_words = spanish_stopwords,
        min_df = 50,
        max_df = 1.9,
        ngram_range=(1, 1),
        max_features=1000
    )),
    ('cls', LinearSVC(C=.2, loss='squared_hinge', max_iter=1000, multi_class='ovr',
        random_state=None,
        penalty='l2',
        tol=0.0001
    )),
])
pipeline.fit(tweets_corpus.content, tweets_corpus.polarity_bin)
```

```

tweets['polarity'] = pipeline.predict(tweets.tweet)
tweets[['tweet', 'lat', 'lon', 'polarity']].to_csv('heatmap-
files/tweets_polarity_bin.csv', encoding='utf-8')
with open('heatmap-files/tweets_heatmap_polarity_binary','w') as file:
    file.write(tweets[['lat','lon', 'polarity']].to_string(header=False, in
dex=False))

```

Salida:

 tweets_heatmap_polarity_binary	21/08/2020 09:49 p. m.	Archivo	2 KB
 tweets_polarity_bin.csv	21/08/2020 09:49 p. m.	Archivo de valores...	7 KB

```

22.264406 -97.785190 1
22.255300 -97.868600 1
22.279510 -97.886620 1
22.255300 -97.868600 1
22.268698 -97.859190 1
22.260176 -97.850970 1

```

	A	B	C	D	E
1		tweet	lat	lon	polarity
2	7	Sigan ðŸ™‰%	22.264406	-97.78519	1
3	74	ðŸ™‰ en Tamp	22.2553	-97.8686	1

- **Definir dos nuevos gradientes de color para identificar a los tweets positivos (verde) y los negativos (rojo).**

Entrada:

```

# Verde = tweets positivos

hsva_min = Color()
hsva_min.hex_l = '#24b736'

hsva_max = Color()
hsva_max.hex_l = '#24b736'

color_gradient = list(hsva_max.range_to(hsva_min,256))
alpha = np.arange(0,256)[::-1]

gradient = []
for i, color_point in enumerate(color_gradient):
    rgb = list(color_point.get_rgb())
    rgb = [int(e * 255) for e in rgb]
    rgb.append(alpha[i])
    gradient.append([rgb])
color_gradient = np.array(gradient)

width = 43

```

```

color_gradient_row = deepcopy(color_gradient)

for col in range(width-1):
    color_gradient = np.hstack((color_gradient, color_gradient_row))

imageio.imwrite('gradients/gradient_green.png', color_gradient)

# Rojo = tweets negativos

hsva_min = Color()
hsva_min.hex_l = '#ff3639'

hsva_max = Color()
hsva_max.hex_l = '#ff3639'

color_gradient = list(hsva_max.range_to(hsva_min,256))
alpha = np.arange(0,256)[::-1]

gradient = []
for i, color_point in enumerate(color_gradient):
    rgb = list(color_point.get_rgb())
    rgb = [int(e * 255) for e in rgb]
    rgb.append(alpha[i])
    gradient.append(rgb)
color_gradient = np.array(gradient)

width = 43



color_gradient_row = deepcopy(color_gradient)

for col in range(width-1):
    color_gradient = np.hstack((color_gradient, color_gradient_row))

imageio.imwrite('gradients/gradient_red.png', color_gradient)

```

### Salida:

 gradient_blue.png	21/08/2020 06:04 p. m.	Archivo PNG	1 KB
 gradient_green.png	27/07/2020 03:41 a. m.	Archivo PNG	1 KB

- **Convertir el archivo de texto de las coordenadas y polaridades** (*tweets\_heatmap\_polarity\_binary*) en un data frame.

Entrada:


```
tweets = pd.read_table('heatmap-
files/tweets_heatmap_polarity_binary', encoding='utf-
8', sep=' ', header=None)
del tweets[tweets.columns[0]]
del tweets[tweets.columns[2]]
tweets.columns = ['lat', 'lon', 'polarity']
```

- **Exportar las coordenadas con polaridad negativa a un archivo de texto.**

Entrada:

```
with open('heatmap-files/tweets_with_polarity_negative','w') as file:
    file.write(tweets[['lat','lon']][tweets.polarity==0].to_string(header=F
alse, index=False))
```

Salida:


 tweets_with_polarity_negative	27/07/2020 03:46 a. m.	Archivo	1 KB
---	------------------------	---------	------

- **Exportar las coordenadas con polaridad positiva a un archivo de texto.**

Entrada:

```
with open('tweets_with_polarity_positive','w') as file:
    file.write(tweets[['lat','lon']][tweets.polarity==1].to_string(header=F
alse, index=False))
```

Salida:

 tweets_with_polarity_positive	27/07/2020 03:46 a. m.	Archivo	1 KB
---	------------------------	---------	------

- **Graficar la capa de tweets negativos.**

**OPCIÓN 1:** Usando el símbolo del sistema

Abrir el símbolo del sistema y navegar hasta el directorio de nuestro proyecto y ejecutar el siguiente comando.

Entrada:

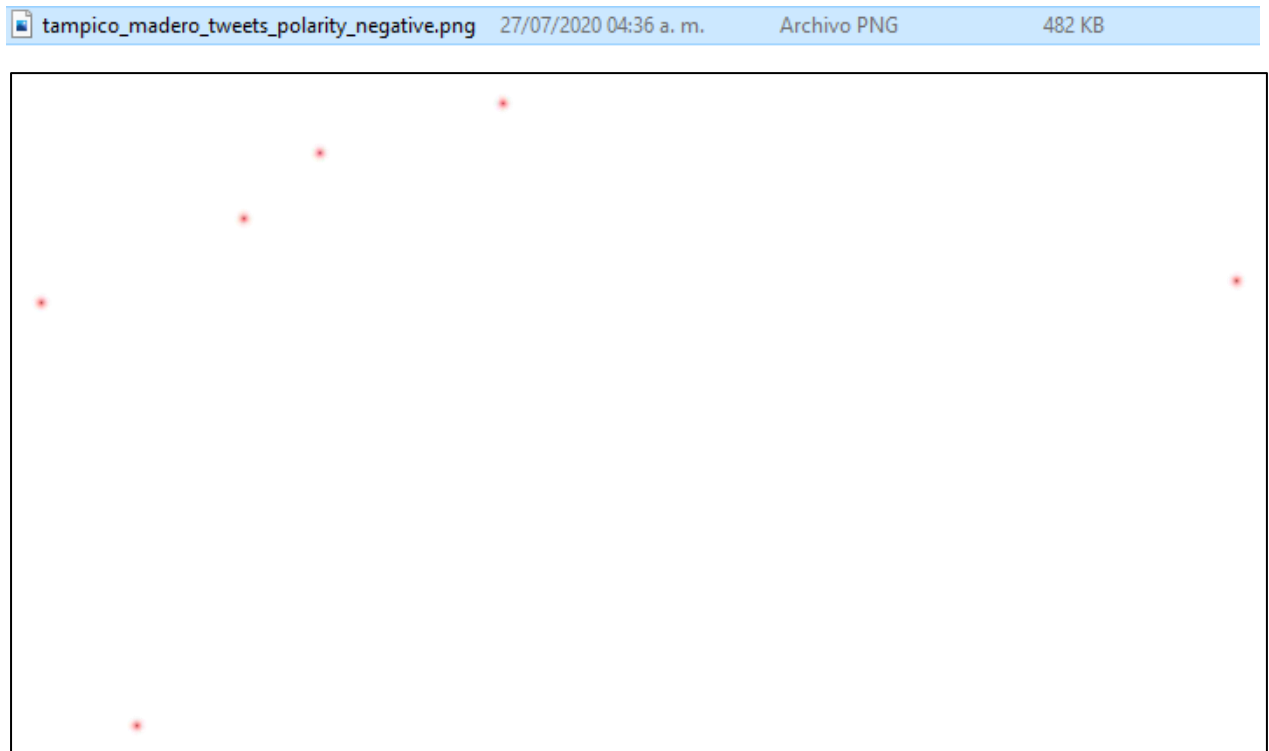
```
python heatmap.py -o maps/tampico_madero_tweets_polarity_negative.png --
width 1920 -p heatmap-files/tweets_with_polarity_negative -
P equirectangular --osm --
osm_base https://basemaps.cartocdn.com/rastertiles/light_all --decay 0.8 -
r 10 --zoom 0 --margin 15 -G gradients/gradient_red.png --layer
```

**OPCION 2:** Usando el IDE Spyder, Jupiter Qt console o Jupiter notebook  
Simplemente ejecutar la siguiente línea

Entrada:

```
get_ipython().system('python heatmap.py -  
o maps/tampico_madero_tweets_polarity_negative.png --width 1920 -p heatmap-  
files/tweets_with_polarity_negative -P equirectangular --osm --  
osm_base https://basemaps.cartocdn.com/rastertiles/light_all --decay 0.8 -  
r 10 --zoom 0 --margin 15 -G gradients/gradient_red.png --layer')
```

Salida:



- ***Graficar la capa de tweets positivos.***

**OPCIÓN 1:** Usando el símbolo del sistema

Abrir el símbolo del sistema y navegar hasta el directorio de nuestro proyecto y ejecutar el siguiente comando.

Entrada:

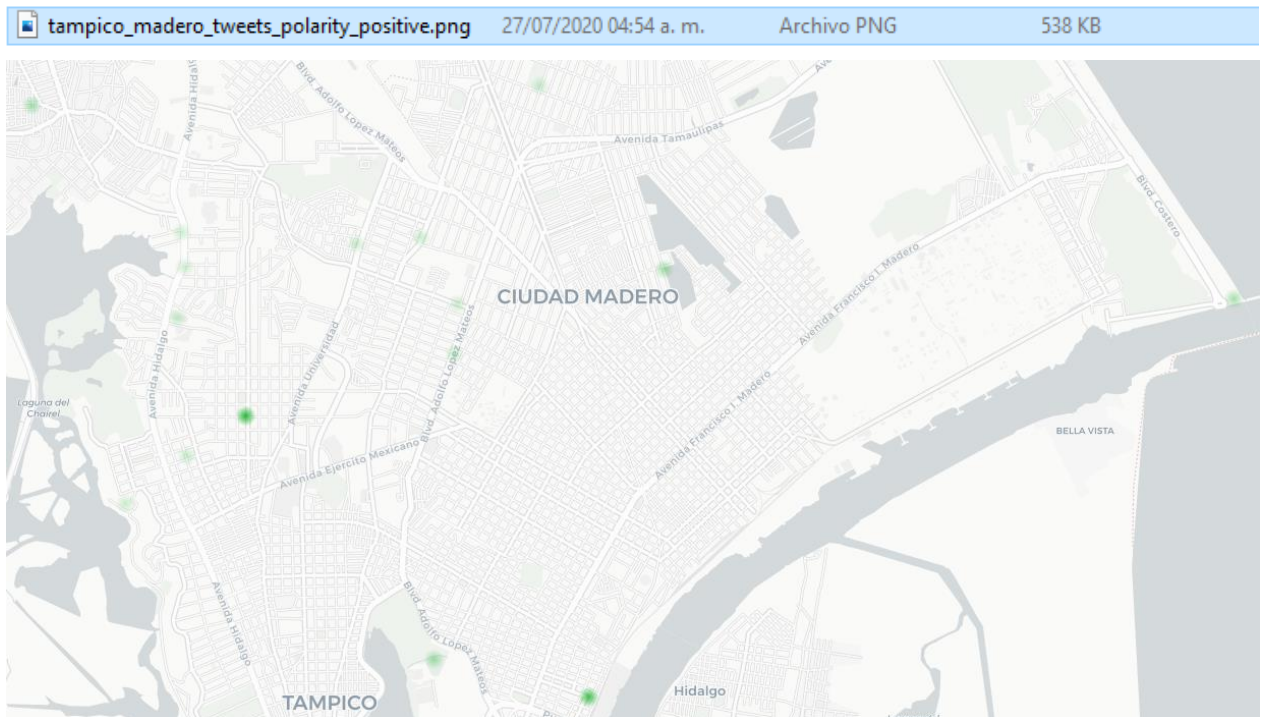
```
python heatmap.py -o maps/tampico_madero_tweets_polarity_positive.png --  
width 1920 -p heatmap-files/tweets_with_polarity_positive -  
P equirectangular --osm --  
osm_base https://basemaps.cartocdn.com/rastertiles/light_all --decay 0.8 -  
r 10 --zoom 0 --margin 15 -G gradients/gradient_green.png
```

**OPCION 2:** Usando el IDE Spyder, Jupiter Qt console o Jupiter notebook  
Simplemente ejecutar la siguiente línea

Entrada:

```
get_ipython().system('python heatmap.py -  
o maps/tampico_madero_tweets_polarity_positive.png --width 1920 -p heatmap-  
files/tweets_with_polarity_positive -P equirectangular --osm --  
osm_base https://basemaps.cartocdn.com/rastertiles/light_all --decay 0.8 -  
r 10 --zoom 0 --margin 15 -G gradients/gradient_green.png')
```

Salida:



- **Combinar las imágenes de los tweets positivos y negativos en un solo archivo.**

Entrada:

```
background = PIL.Image.open('maps/tampico_madero_tweets_polarity_positive.p  
ng')  
foreground = PIL.Image.open('maps/tampico_madero_tweets_polarity_negative.p  
ng')  
  
background.paste(foreground, (0, 0), foreground)  
  
draw = ImageDraw.Draw(background)  
font = ImageFont.truetype("fonts/ProductSans.ttf", 14)
```



```
draw.text((1020, 600),"Tampico-  
Madero, Tam., Mex.", fill = "black", font=font)  
draw.text((1020, 620),"Tweets positivos y negativos", fill = "black", font=  
font)  
  
background.save('maps/tampico_madero_tweets_polarity.png')
```

Salida:

