Proyecto

Roberto García Salazar
417032986
Facultad de Ciencias, UNAM

15 de Junio, 2020

Abstract.-En Diciembre de 2019 en la ciudad de Wuhan, provincia de Hubei en China, surgió un nuevo coronavirus que, eventualmente se convirtió en una amenaza a la salud pública del mundo. Al pasar los días, se identificaron cada vez más casos dentro de China y posteriormente en todo el mundo, siendo necesario distribuir de manera eficiente información sobre este virus al público. Un efecto secundario de esto, es el surgimiento de noticias falsas, así como rumores y verdades a medias, producto de la ignorancia o malas intenciones; así las cosas, resulta evidente la necesidad de encontrar maneras de hallar tendencias que pueden ser de interés para el estudio de la información que se genera en línea.

1 Objetivos

Se utilizará para este fin el sitio de microblogging *Twitter* para obtener, mediante la manipulación de la metadata, una nube de palabras que obtendremos de los objetos '*tweet*' que resulten de la búsqueda de la etiqueta 'covid19', se dará también una introducción breve a la interacción con la API de Twitter mediante la biblioteca *tweepy* y se hará una visualización con una SOM(self-organizing map) del resultado de la organización por semejanza de las nubes de palabras de los perfiles de los usuarios que las cuentas *@lajornadaonline, @el_universal_mx, @reforma, @proceso, @milenio, @eleconomista* siguen. Es importante recalcar que no se tomarán duplicados.

2 Introducción

Las nubes de palabras son "representaciones gráficas del conteo de las palabras que se vierten en un escrito determinado"[1], en este caso, se tomará como corpus la colección de textos asociados a los *tweets* de los perfiles de los seguidores de las principales cuentas de periodismo que siguen al subsecretario de salud, Hugo López-Gatell Ramírez en la plataforma de miicroblogging. Para este fin se trabajará con la versión mas reciente de python(3.8.2) se instalará la biblioteca tweepy.
Se decidió utilizar twitter para este fin pues es una red social de interacción directa entre usuarios, donde

[1]Recuperado de https://dictionary.cambridge.org/dictionary/english/word-cloud

se encuentran personas de la administración pública tanto actual como pasadas, líderes de opinión, medios tradicionales, electrónicos y personas usuarias de la plataforma, también los usuarios comparte sus ideas a través de textos, imágenes y videos, así como promociones pagadas con público dirigido. Para el entrenamiento de la SOM, se utilizará una capa de texto embebida para el entrenamiento, las razones de esto es porque a la hora de trasladar nuestro espacio de palabras a vectores, este espacio resultado tendrá propiedades geométricas que codificarán propiedades semánticas del espacio de palabras donde partimos, esto es importante porque así podremos capturar una parte de la riqueza semántica de los mensajes haciendo el conjunto de estudio mas cercano a lo que es la realidad del uso de las palabras por cada usuario.

3 Interacción con la API de *Twitter* mediante *Tweepy*

La API[2] de *Twitter* está contenida en la plataforma para desarrollo de la red social, esta plataforma ofrece 4 API para el desarrollo de aplicaciones[3]:

- Standard API.
  - Que nos permite interactuar con los *Tweets* como objeto y las *timelines*.
  - Escribir y leer mensajes directos.
  - Utilizar información pública de las cuentas.
  - Hacer búsquedas.
- Premium API.
  Da opciones avanzadas de filtrado y acceso escalable a busquedas en *tweets*.
- Enterpise API.
  Permite recabar informacion en tiempo real sobre información pública de las cuentas e historial de *tweets*.
- Ads API

Siendo el último punto de la standard API de principal importancia, pues esta herramienta nos permitirá discriminar los *tweets* que nos serán de utilidad mas adelante.
En este momento se hace el supuesto que la persona que de lectura a este texto cuenta con la versión 3.8.2 de Python y cuenta con PIP funcionando.

[2] API es un acrónimo en inglés para Aplication Programming Interface.
[3] Obtenido de https://developer.twitter.com/en/docs/basics/getting-started

3.1 Instalando Tweepy

Para la instalación de la biblioteca basta con abrir la *Terminal, bash o consola de comandos* y escribir la instrucción:

```
$ pip3 install tweepy
```

Una vez terminada la instalación de la biblioteca se crea la siguiente función:

```
1 import tweepy
2
3 def openConnection():
4   auth=tweepy.OAuthHandler(API_key, API_secret_key)
5   auth.set_access_token(access_token, access_token_secret)
6   api=tweepy.API(auth)
7   return api
```

donde *API_key, API_secret_key, acces_token y access_token_secret* son llaves que nos da *Twitter* después de registrar una cuenta para desarrollador y generar una app en el dashboard; para generar el objeto API, se tiene que:

api=openConnection()

Con esta ultima línea ya se tiene lo necesario para empezar la interacción con la API.

3.2 Obteniendo los *usuarios*

Se escribe el siguiente segmento de código como se muestra en la siguiente imagen:

```
1 import multiprocessing as mp
2
3 nodes=['lajornadaonline', 'el_universal_mx', 'reforma', 'proceso', 'milenio', 'eleconomista']
4 following=[]
5 for node in nodes:
6   aux=api.friends_ids(node, count=5000)
7   for i in range(len(aux)):
8     try:
9       if (aux[i] not in following and len(api.get_user(aux[i]).followers_count)>100000 and api.get_user(aux[i]).protected==False):
10        following.append(api.get_user(aux[i]).screen_name.lower())
11    except:
12     pass
```

Es importante hacer la precisión de que la instancia de api se escribe dentro del ciclo que corre sobre la lista *nodes* porque crea una nueva lista de seguidores de cada cuenta cada vez que se obtengan datos de los *tweets* saturando la cantidad de conexiones posibles en un periodo de tiempo(si se quieren obtener de múltiples usuarios que es nuestro caso); de lo anterior se tiene que esta acción es imposible de agilizar debido a las restricciones de la API. Dicha restricción también incluye, por supuesto a obtener los tweets, que dicho sea de paso se formula una función con computo en paralelo, vía multiprocessing para maximizar las solicitudes de tweets en el máximo de usuarios posibles.
Puesto el párrafo anterior, se tiene la necesidad de evitar repetir este proceso, por lo tanto, se define la funcion *createTargetFile(iterable)* que guarda todos los usuarios en un archivo llamado *following*, entonces también se crea una función llamada *get_followers()*.

3.3 Obteniendo los tuits

Se tiene que hacer uso de una funcion de la api, llamada user_timeline que tiene como argumentos al usuario y un conteo de resultados. A continuación se ilustra la funcion *get_tweet_words y trim_tweet*:

```python
import regex as re
def trim_tweet(text):
  words=[]
  skipped_words=['de', 'la', 'que ', 'el', 'en', 'y', 'a', 'los', '
                 'que', 'pues', 'creen','dia','noche','esto','esta'
  tweet=text.lower()
  for word in tweet.split():
    if word not in skipped_words:
      word=re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL', word)
      word=word.strip('\¿?,.-_")(:;¿¡!')
      word=re.sub(r'@([^\s]+)',r'\1',word)
      word=re.sub(r'#([^\s]+)',r'\1',word)
      word=re.sub(r'á',r'a',word)
      word=re.sub(r'é',r'e',word)
      word=re.sub(r'í',r'i',word)
      word=re.sub(r'ó',r'o',word)
      word=re.sub(r'ú',r'u',word)
      words.append(word)
  return words
```

```python
def get_tweets_words(follower):
  dic={}
  tweets=api.user_timeline(follower, count=5000)
  text=[]
  for tweet in tweets:
    aux=trim_tweet(tweet.text)
    text.extend(aux)
  dic[follower]=text
  return dic
```

3.4 SOM

El código de la SOM se encuentra en el github:
https://github.com/ReallyStonedApe/proyecto_final_seminarioCC/blob/master/som.ipynb