UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER EN CIBERSEGURIDAD Y SEGURIDAD DE LA INFORMACIÓN

TRABAJO FIN DE MÁSTER

Sistema de detección de fake news a través de Twitter

Javier Vilches Sarasate

septiembre, 2020

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Departamento de Sistemas Informáticos

MÁSTER EN CIBERSEGURIDAD Y SEGURIDAD DE LA INFORMACIÓN

Sistema de detección de fake news a través de Twitter

Autor: Javier Vilches Sarasate

Director: Félix Brezo Fernández

Septiembre, 2020

Resumen

Los problemas de la desinformación online y las “fake news” han ido ganando cada vez más relevancia en una era en la que los contenidos son creados por los propios usuarios y las redes sociales son claves en la formación y difusión de nuevas historias. A través de plataformas como Twitter y Facebook a menudo se publica y se difunde ampliamente información poco fiable. Como resultado, periodistas y editores se ven en la necesidad de herramientas que les ayuden a acelerar el proceso de verificación de contenido que proviene de redes sociales.

Publicar noticias con contenido gráfico en un contexto falso, como usar fotografías pasadas como noticias actuales, es cada vez más frecuente. Motivado por esta problemática, se presenta un sistema semiautomático de detección de fake news centrado en la búsqueda inversa de fotografías pasadas en publicaciones en Twitter (tweets). El sistema extrae características de los tweets y las cuentas que lo publican para obtener una medida de la credibilidad de la fuente, y subjetividad y polaridad de la publicación.

El sistema puede analizar tanto cuentas como tweets individuales y genera una tabla resumen con los casos encontrados, con los datos más relevantes, como las fechas del tweet y de las imágenes encontradas, el id del tweet y el enlace de la web que contiene la imagen. El sistema genera otras tablas complementarias, con todas las características que se han recogido y tablas que se pueden usar, y se han usado para su uso en machine learninig.

Debido a las limitaciones del método de búsqueda inversa y a la dificultad de determinar el contexto de una noticia con una imagen de forma automática, el sistema no puede dar resultados categóricos, pero sí ser un apoyo al analista, ya que puede recoger gran cantidad de información expuesta de una forma resumida.

Agradecimientos

Me gustaría agradecer a mis tutores, Félix Brezo y Yaiza Rubio por sus buenos consejos a la hora de enfocar el proyecto y a su buena predisposición en todo momento.

—¿Qué hay de las fake news? Fuiste objeto de ellas

—Oh, ¿sí? ¿Qué dijeron? ¿Que me presento a presidente? ¿No? ¿Qué dijeron?

—Que apoyabas la candidatura de Trump. ¿Cómo lidias con ello, con las fake news?

—Si no lees los periódicos, no estás informado. Y si lees los periódicos, estás mal informado

—¿Y qué podemos hacer?

—Es una buena pregunta.

Entrevista a Denzel Washington[1] en referencia a una fake news de la que fue víctima[2]

[1] https://youtu.be/UG0aMavVGV4

[2] https://magnet.xataka.com/idolos-de-hoy-y-siempre/la-respuesta-de-denzel-washington-que-resume-de-forma-brillante-el-principal-problema-del-periodismo

La verdad es difícil.

Difícil de encontrar.

Difícil de saber.

La verdad es ahora más importante que nunca.

Campaña de el *New York Times* en 2017[3] en respuesta a Donald Trump.

[3] https://elpais.com/internacional/2017/02/27/estados_unidos/1488157527_469114.html

ÍNDICE

INTRODUCCIÓN

El fenómeno de las "fake news" ha tomado una relevancia destacada en los últimos años y su impacto sobre la opinión pública, el periodismo y la democracia es cada vez mayor. Este impacto se sustenta en el hecho de que, por ejemplo en España, sólo uno de cada tres ciudadanos confía en las noticias publicadas en los medios (Negredo, 2020) y según un estudio de la Pew Research Center de 2016, para el 62% ciudadanos las redes sociales son la fuente principal de noticias.

El cambio de paradigma que supuso la transformación digital de los medios tradicionales, el hecho de que los ingresos dependan de los clics que reciben las noticias y a la inmediatez demandada por el consumidor, ha llevado a la profesión a una falta de rigor cada vez mayor y a la desconfianza de la sociedad. Las "fake news", o, mejor dicho, los bulos que se propagan por redes sociales e internet, dinamitan la confianza de la sociedad en los medios de comunicación, aplanando el terreno para que tanto las noticias verdaderas como las falsas se perciban de igual manera, dañando la reputación de las primeras. El peligro está en que, si no hay confianza en las noticias, sólo creeremos las que apoyen nuestras ideas independientemente de si son ciertas o no.

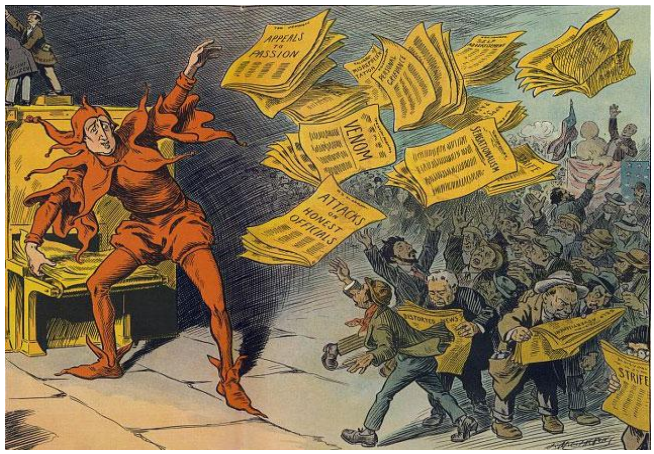
Pueden verse dos casos claros en los que se aprovecha esto como arma política. Donald Trump asocia de forma constante las "fake news" con los medios tradicionales dejando un vacío informativo que sólo él puede proveer con su (post-)verdad. Otro caso en España es el del partido político Vox, que plantea eliminar la verificación [4] de noticias, por ejemplo, del verificador Maldita[5], un arma muy importante contra los bulos. Si se elimina toda resistencia a las fake news, la información pasará a ser propaganda partidista y los hechos quedarán en segundo plano. Es por esta razón por la que son tan peligrosas.

ORÍGENES Y CARACTERÍSTICAS DE LAS FAKE NEWS

A pesar de esto, el concepto no es nuevo, ya que las mismas técnicas ya se usaban durante la propaganda nazi en la primera y segunda Guerra Mundial para manipular a las masas. Incluso podemos encontrar un caso anterior, cuando en 1898, la publicación de noticias falsas por el periódico New York Journal logró que EE.UU. declarara la guerra a España por la independencia de Cuba (Garcia, 2018). Pero a diferencia de éstas, las "fake news" de hoy no están limitadas a un entorno geográfico particular; hoy en día, las redes sociales e internet son el medio ideal para su propagación. La influencia partidista y geopolítica de las fake news ha sido objetivo de numerosos estudios y en época de elecciones, como las de EE. UU o el Brexit en 2016, son claros

[4] https://magnet.xataka.com/en-diez-minutos/batalla-verificacion-noticias-que-vox-quiere-prohibir-fact-checking
[5] https://www.maldita.es

ejemplos del peligro que suponen para la salud del sistema democrático en países de todo el mundo. Un informe de 2017 de la ONG Freedom House (Kelly, Truong, Shahbaz, Earp, & White, 2017) revela que 30 países generan noticias para distorsionar la información a su favor.



¿Quién está involucrado en las “fake news”?

A grandes rasgos se pueden difernciar tres grupos: los creadores, los propagadores de alto nivel y los propagadores de bajo nivel[6]

Los creadores son, como su nombre indica, individuos que crean el contenido falso para difundirlo. Suelen tener intencionalidad partidista o ser simplemente trolls de la post-verdad. También los hay los que viven de ello, sólo por el dinero, pero sin ningún interés político detrás. Los propagadores de alto nivel son grandes operativos que difunden la desinformación a una gran audiencia y con poco esfuerzo por medio de blogs, grupos de Facebook y de Whatsapp, sitios webs de fake news, Twitter, etc). Los propagadores de bajo nivel son la mayoría, los “consumidores” del contenido falso y que además ayudan a difundirlo compartiendo o retwitteandolo.

Con todo esto podemos sacar una conclusión importante; las fake news se crean con un objetivo claro, que puede ser económico, político o ideológico. Uno de los problemas del concepto de las fake news es que no tiene definición clara y consensuada para todo el mundo. Por ejemplo, si el Washington Post dice que una historia es fake news no tiene el mismo significado que si Donald Trump llama a fake news a la CNN.

Esto da lugar a malentendidos y usos partidistas de este término, ya que da a entender que son “noticias falsas”, cuando en realidad el fenómeno es mucho más complejo y no se limita únicamente a, lo que solemos tener en mente de, un formato determinado de noticia periodística, sino que incluye videos, fotos, capturas, memes, audios… y se propaga no sólo en redes sociales como Twitter o Facebook, sino también en Discord, Instagram o foros como 4Chan o Reddit, redes mucho más difíciles de analizar y verificar. Muchas campañas de desinformación y “fakes news” se originan en grupos cerrados de Facebook, Whatsapp o

[6] https://www.reddit.com/r/fakenews/comments/5ggakp/the_rfakenews_faq/

Telegram. El problema principal actualmente radica en entender como las fake news se comparten entre las redes originales antes de saltar a Facebook, Twitter o Youtube.

DEFINICIÓN Y TIPOS DE DESINFORMACIÓN

En este sentido, podemos utilizar la definición del Diccionario Collins:

> Las fake news son informaciones falsas diseñadas para hacerse pasar por noticias con el objetivo de difundir un engaño o una desinformación deliberada para obtener un fin político o financiero.

Los matices son importantes y la clave está en la palabra desinformación, que es la que realmente define el fenómeno. Puede verse la desinformación como la intersección entre la mala información (*Misinformation*; contenido falso no intencionado) y la intención de hacer el mal (*Malinformation*), y se refiere a algo que no solamente es falso, sino fuera de contexto. Esta forma de visualizar la información/desinformación creada por investigadores del proyecto First Draft[7] de Harvard Kennedy School of Government es muy util y explica bien el ecosistema de las "fake news".

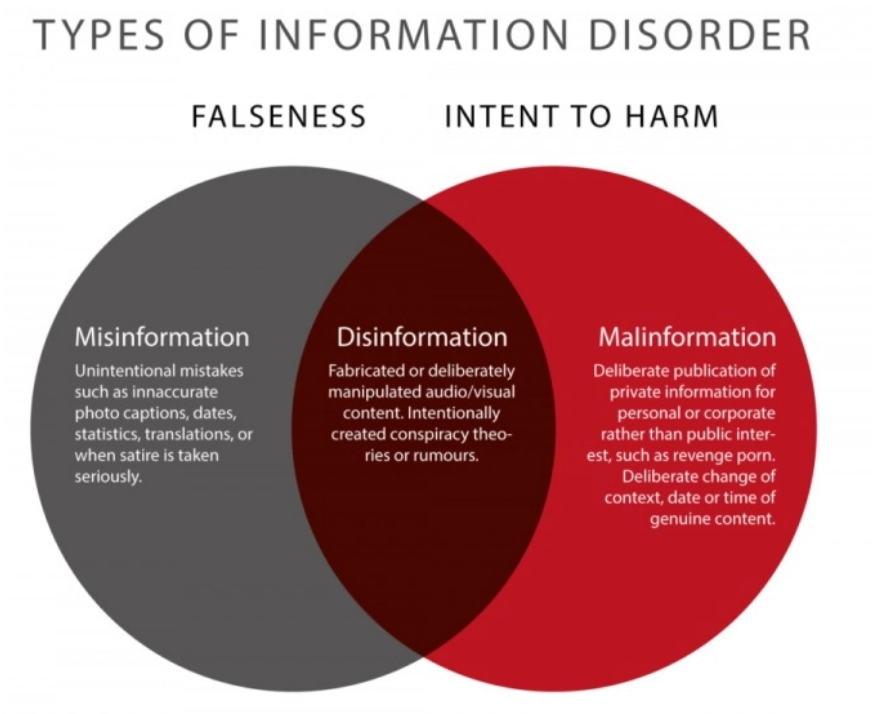


Figura 1. Tipos de desordenes de la información. Crédito: Claire Wardle & Hossein Derakshan, 2017

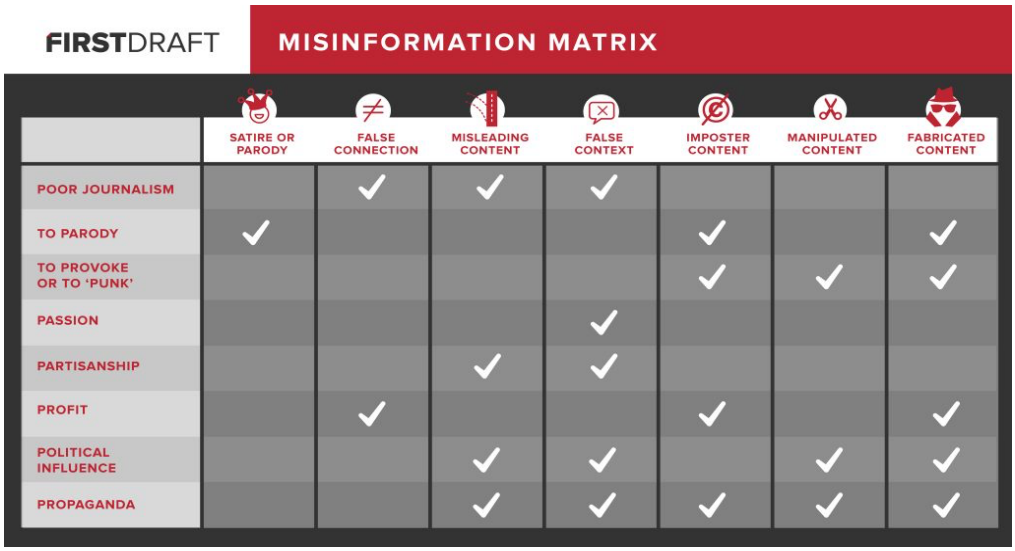
Partiendo de esa intencionalidad vista anteriormente hemos llegado a la desinformación como piedra angular sobre la que gira todo lo demás, y es en este sentido más amplio, sobre el que deben entenderse las fake news; las fake news no son noticias, aunque lo parezcan; y de hecho, según la recomendación del grupo de expertos de la UE se debería abandonar el uso del término

[7] https://firstdraftnews.org/

"fake news" y hablar de desinformación, o bulos en su lugar[8]. También es preferible referirse a contenido en lugar de noticias en estos casos (Directorate-General for Communications Networks, Content and Technology, 2018). Por eso, intentaré referirme al fenómeno de la manera adecuada.

Según la web First Draft News, se pueden distinguir 7 tipos de desinformación (Wardle, 2017):

- Sátira/parodia. No tienen intención de causar daño, pero pueden llevar a engaño
- Contenido engañoso.: uso engañoso de la información enmarcada en un asunto o individuo.
- Contenido impuesto: cuando se suplantan las fuentes originales. Contenido fabricado: es contenido 100% falso, diseñado para engañar y causar daño.
- Contenido falsamente conectado: cuando los titulares, imágenes o capturas no coinciden con el contenido
- Contenido manipulado: cuando se manipulan imágenes o información para engañar.
- **Contexto falso**: *cuando se comparte contenido genuino con información contextual falsa.*



[8] https://maldita.es/maldita-te-explica/2019/11/20/dejemos-de-hablar-de-fake-news-y-de-noticias-falsas-2/

DESINFORMACIÓN DE TIPO FALSO CONTEXTO

Las "fake news" es un fenómeno complejo que involucra varios actores, como medios de comunicación, webs de humor, creadores, periodistas, grupos de presión, lobbies, influencers, partidos políticos o bots, y con motivaciones económicas o políticas. También incluye los 7 tipos de desinformación vistos anteriormente, y cada uno de ellos debe abordarse de una manera diferente. Este proyecto se centra en las desinformaciones de tipo *falso contexto* en las que se comparte contenido, pero fuera de contexto para crear noticias falsas. Un caso particular es el de usar fotografías en noticias ya publicadas con anterioridad en otras noticias. Esto de por sí no tiene porque ser desinformación, ya que es muy común que los medios usen fotos de agencias (EFE/AP/Reuters) sin una intención dañina. En otros casos se puede ver que hay una intención clara de engañar, por ejemplo, en el tweet mostrado abajo, la foto no es de 2020 como indica la foto, sino de 2016 (publicada por Vanitatis[9])

PANORAMA ACTUAL Y FUTURO

El problema de las fake news es, en mi opinión, un probema complejo que debe abordarse desde varios frentes, tanto informáticos como sociales, jurídicos y políticos. La mayoría de los bulos tienen como principal interés ganar dinero y hasta que dejen de ser rentables, seguirán existiendo. Por eso, los anunciantes deberían dejar de invertir en sitios que alojen estos contenidos.

También debería haber una legislación que castigue estas prácticas. Las propias redes como Twitter o Facebook deben tomar responsabilidad igualmente y aceptar el hecho de que parte de la población las usa para infomarse, y por ello deben cumplir los criterios de rigor y objetividad que se le pide a otros medios de información.

Mientras tanto, muchos medios de comunicación tradicionales de gran prestigio como el *New York Times* o el *Washington Post* han entendido que la única manera de buscar la veracidad y permanecer imparciales es saliendo de la dictadura del clic y cambiar a un modelo de negocio por subscripción, es decir, ser independientes de poderes políticos y económicos. La información de calidad en la era de las fake news tiene un precio, y en EE.UU en 2017 el 53% pagaba por las noticias[10]. Mientras las fake news no desaparezcan, posiblemente ese sea el futuro del periodismo.

---

[9] https://www.vanitatis.elconfidencial.com/multimedia/album/noticias/2016-08-23/pedro-sanchez-disfruta-de-sus-vacaciones-familiares-en-ibiza_1249855

[10] https://www.americanpressinstitute.org/publications/reports/survey-research/paying-for-news/

Como último punto, y puede que el más importante, no podemos olvidar que las "fake news" existen porque se propagan por la red, como un virus, y lo hacen porque nosotros las compartimos. Queremos información instantánea y normalmente sólo leemos los titulares. La inmediatez nos lleva a caer en la trampa y compartir sin reflexionar. Cuando empecemos a tomarnos un minuto para verificar una noticia antes de hacer "retweet", será el principio del fin de las "fake news".

MOTIVACIÓN Y OBJETIVOS

Mi interés en este proyecto no es otro que mi interés en una sociedad con información veraz y rigurosa y en mi preocupación por el daño que hace la desinformación y los bulos en un mundo globalizado. En particular, este año con la pandemia de COVID-19 se han disparado los bulos sobre este tema, lo que se conoce como infodemia, la sobreabundancia de información mucha de la cual son bulos o rumores, con consecuencias muy peligrosas para la salud pública[11]. Es por esto que, cualquier esfuerzo encaminado a desenmascarar y eliminar las fake news, merecerá la pena.

Twitter es una plataforma de comunicación bidireccional con naturaleza de red social que limita sus mensajes (tuits o tweets) a 280 caracteres. La plataforma de Twitter nació en octubre de 2006 en San Francisco (Estados Unidos). El objetivo de este proyecto consiste en el desarrollo de una herramienta semiautomática para determinar si un tweet o una lista de cuentas de Twitter contienen imágenes usadas con anterioridad a dicha publicación. El poder de la imagen y la rapidez de difusióne en redes como Twitter crean un campo propicio para las "fake news"

Por otro lado, la herramienta recogerá los últimos tweets de una cuenta (el límite es de 3200) y extraerá un conjunto datos del tweet, como la URL de una imágen si tuviera una. Después realizará una búsqueda inversa usando la API de MRISA (Meta Reverse Image Search API[12]) y se extraerán las fechas de los resultados siempre que sea posible y se comparará con la fecha del tweet. El resultado final en este caso será una tabla con la información básica del tweet original, la imagen y el enlace donde se ha encontrado dicha imagen. Los resultados con fechas del tweet anteriores a las del enlace son descartadas.

Sin embargo, la herramienta recopila muchos más datos, tanto a nivel de usuario como de tweet, con la idea de generar tablas más generales que puedan ser usadas para otras finalidades distintas. Por ejemplo, a partir de los atributos de la cuenta de usuario el sistema determina un índice de credibilidad que se aplicará a cualquier publicación de dicha cuenta. También se aplica análisis de sentimiento al texto del tweet para extraer atributos de polaridad y subjetividad a cada tweet[13] . Las tablas intermedias obtenidas con todas las características, que no se ve en el

[11] https://maldita.es/malditobulo/2020/09/16/coronavirus-bulos-pandemia-prevenir-virus-covid-19/
[12] https://github.com/vivithemage/mrisa
[13] https://textblob.readthedocs.io/en/dev/

resultado final son usadas para reducir en los posible los falsos positivos en la búsqueda inversa de imágenes.

PROPUESTA

Se propone crear una herramienta que genere tablas a partir de datos extraídos con la API de Twitter, de tal forma que se puedan usar para, pero no de forma exclusiva, el objetivo principal de la búsqueda de imágenes ya publicadas con anterioridad en internet.

Buscamos recopilar todos los tweets que nos permita la API de Twitter, que son 3200 como máximo, con una serie de condiciones, y extraer algunas características. Se han separado por una parte las correspondientes a la parte del usuario y por otra las que son específicas del propio tweet. Las condiciones que se toman para la obtención de datos son:

1. El tweet debe tener una imágen
2. La imágen no debe ser una captura de vídeo ni miniatura de un vídeo
3. El tweet debe tener un número de retweets superior a un umbral (min_retweets)

RECOGIDA Y PROCESAMIENTO DE CARACTERÍSTICAS

CARACTERÍSTICAS DE TIPO USUARIO

Las características de usuario que se recogen son:

- screen_name : nombre de la cuenta (sin @). Por ejemplo: *el_pais*
- user_descr : descripción de la cuenta
- verified : comprueba si la cuenta está verificada (1) o no (0)
- created_at: fecha de creación de la cuenta
- location : localización de la cuenta (la pone el usuario en su perfil)
- def_prf_img : establece si el usuario ha cambiado su imagen de perfil
- def_prf : establece si el usuario ha cambiado su perfil

- n_publications : número de tweets publicados por la cuenta
- n_listed : número de listas en las que aparece la cuenta
- user_url : url que aparece en el perfil de usuario (por ejemplo https://elpais.com)
- user_fav : número de "me gusta" dado por el usuario desde la creación de la cuenta.
- followers: número de seguidores del usuario
- friends : número de amigos del usuario
- user_id : identificador numérico de 10 dígitos único del usuario en Twitter
- lang : idioma principal de los tweets del usuario

En el siguiente paso se calculan otros parámetros a partir de los que se señalan arriba. Para las características del usuario se pretende obtener un valor entre 0 y 1 para la credibilidad de la cuenta. Basándose en otros estudios (Krzysztof Lorek, 2015) y (Rico & Berná, 2019) se proponen los siguientes marcadores de credibilidad:

Parámetros P_j	**Descripción**	**valor**
AgP[14]	Parámetro de edad de la cuenta	$\frac{age}{twitter_{max}} \in (0,1)$
ppdP	Parámetro de tweets medios diarios	1 si ppdP<1 0 si ppdP ≥1
FrFoP	Parámetro ratio amigos/seguidores	$1 - \frac{friends}{followers}$si friends>0 0 si friends=0
DLenP	Parámetro longitud del texto de la descripción	0 si descr_len = 0 1 si desr_len > 0
PrfParam	Parámetro perfil por defecto	1 si def_prf = 0 0 si def_prf = 1
FolParam	Parámetro seguidores	1 si followers > 700[15] 0.5 si followers < 700 0 si followers = 0
FavsParam	Parámetro favoritos	1 si user_favs> 200 0.5 si user_fav < 200 0 si user_fav= 0
NoBotParam	Inverso al botometer_score normalizado a 1	$1 - \frac{scorebot}{5}$
ImPrfParam	Parámetro imagen perfil por defecto.	1 si def_prf_img = 0

[14] AgP [Age Parameter (0-1)] =age/(twitter_max). Es la ratio entre la edad de la cuenta y la edad máxima posible, twitter_max, que es el tiempo desde que salío Twitter el 21 de marzo de 2006
[15] The Average Twitter User Now has 707 Followers. 2016. https://kickfactory.com/blog/average-twitter-followers-updated-2016/

		0 si def_prf_img = 1
url_count	Indica si la cuenta tiene un enlace web	1 si user_url≠ 0 0 si user_url = 0
verified	Indica si la cuenta está verificada	1 si verified = True 0 si verified = False

Obtenidos los parámetros de credibilidad, todos ellos normalizados entre 0 y 1, se calcula la credibilidad mediante la media aritmética de todos ellos:

$$credibility = \langle P_j \rangle$$

Esto nos da una medida de la credibilidad de la cuenta que publica el tweet, que se calcula como la media de todos los marcadores de credeibilidad.

Características de tipo tweet

Las características que se recogen de cada tweet son:

- tweet_id : identificador único del tweet
- screen_name : usuario que ha publicado el tweet
- created_at : fecha de publicación del tweet
- text : texto del tweet (240 caracteres)
- lang : idioma del tweet
- retweets: número de retweets del tweet (nº veces compartido)
- favs : número de "me gusta" que tiene el tweet
- image_url: URL de la imágen del tweet (http://pbs.twimg.com/media/[...])
- image_name : nombre de la imágen
- urls : extracción de urls del texto del tweet
- img_loc : localización en disco de la imagen descargada (~/images/screen_name)
- hashtags: hastags del tweets (parseo de palabras que empiezan por #)
- hash : función hash MD5 que se aplica a cda imagen descargada.
- total_length : longitud total del texto del tweet
- capitals : nº de mayúsculas del texto del tweet
- caps_vs_length : proporción de mayúsculas en el texto del tweet
- num_words : nº de palabras en el texto del tweet
- num_unique_words : nº de palabras únicas en el texto del tweet
- words_vs_unique : proporción entre nº de palabras y palabras únicas en el tweet
- num_exclamation_marks : nº de signos de exclamación en el tweet.
- num_question_marks : nº de signos de interrogación (?) en el tweet.
- num_question_marks_es' : nº de signos de interrogación (¿) en el tweet.
- question_marks_ratio : diferencia entre num(?) y num(¿). Relevante en español.
- num_punctuation : nº de signos de puntuación en el texto del tweet.
- num_symbols : nº de caracteres no alfanuméricos en el texto del tweet.
- num_smilies : nº de emoticonos sonrientes

- num_sad : nº de emoticonos tristes
- mean_word_len : longitud media de las palabras en el tweet
- RT : contiene “RT @” en el texto del tweet. Indica que es un retweet de otro tweet.
- orig_user : si RT=1, muestra el usuario del tweet original
- polarity [-1,1] : polaridad del texto del tweet. 1 significa sentimiento positivo y -1 sentimiento negativo.
- subjectivity [0,1] : subjetividad del texto del tweet. Un valor de 0 significa que el texto es muy objetivo y un valor cercano a 1 implica un texto muy subjetivo.

REVIMG. BÚSQUEDA INVERSA DE IMÁGENES CON MRISA

PASOS PREVIOS Y PREPARACIÓN DEL SERVIDOR MRISA

La API de MRISA (Meta Reverse Image Search API) es una API RESTful que toma una URL de una imágen y realiza una búsqueda inversa de la misma en Google Imágenes, devolviendo los resultados en una colección de tipo JSON. A diferencia de otras APIs como rapidapi[16], usada por botometer antes o SerpAPI de GoogleSearch, MRISA se instala en una máquina local. Para ello, siguiendo los pasos del repositorio de github[17], se instala en una máquina virtual con Kali Linux 2020.4. Debajo se ven los pasos de instalación, puesta en marcha y registro de una petición realizada con respuesta.

```
bradamon@kali:~/TFM$ #git clone https://github.com/vivithemage/mrisa.git
bradamon@kali:~/TFM$ #cd mrisa/
bradamon@kali:~/TFM$ cd mrisa/
bradamon@kali:~/TFM/mrisa$ #pip install -r requirements.txt
bradamon@kali:~/TFM/mrisa$ # Iniciamos el server con...
bradamon@kali:~/TFM/mrisa/src$ ccd mrisa/
bradamon@kali:~/TFM/mrisa/src$ python3 server.py
```

CONEXIÓN AL SERVIDOR Y EXTRACCIÓN DE LAS FECHAS

Desde la parte del cliente se conecta usando la librería urllib.request de Python

[16] https://rapidapi.com/
[17] https://github.com/vivithemage/mrisa

```
data = {
"image_url":image_url,
"resized_images":False,
"cloud_api":False,
"google_domain": str("google."+dom),
"gl": str(df['lang'][n]),
"hl": str(df['lang'][n])
}

try:
    r=requests.post(url_mrisa, headers=headers, data=json.dumps(data),timeout=(2,10))
```

- image_url : indica la URL de la imágen que se quiere buscar
- resized_images : busca imágenes iguales de distinto tamaño. No se usa porque da muchos errores.
- cloud_api : activa o desactiva la API de Google vision. No es compatible con Meta Reverse Image Searching
- google_domain : dominio de google en el que buscar. Para mejorar los resultados se cambia a google.es en el caso de que el idioma del tweet sea español.
- gl y hl indican la localización, que se mapea con el idioma del tweet para mejorar resultados.

Para la petición POST se amplian los tiempos de espera de respuesta y tiempo de espera de lectura (tiempo que espera a una respuesta una vez se ha establecido la conexión (https://realpython.com/python-requests/#timeouts)

```
bradamon@kali:~/TFM/mrisa/src$ python3 server.py
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
server.py:62: DeprecationWarning: PY_SSIZE_T_CLEAN will be required for '#' formats
  conn.perform()
Successful search
192.168.135.1 - - [16/Sep/2020 20:25:09] "POST /search HTTP/1.1" 200 -
SS
```

En el servidor MRISA en Kali se ve el registro de esta conexión:
El siguiente paso es, tomar cada imágen de cada tweet del paso anterior y usar MRISA para extraer los siguientes atributos de cada imagen buscada. Por las pruebas realizadas, se suelen obtener entre 1 y 7 resultados para cada imágen, que se enumeran con el índice “i”.

- i : índice del resulado i-esimo
- descriptions : Texto que describe la imágen ,obtenido del propio link de resultado
- links: el enlace a la web donde está la imágen (no siempre)
- best_guess : intento de la API de describir la imágen en 1-3 palabras.
- alert: parámetro que indica si la fecha de la imagen es anterior al tweet (1) o posterior (0)

Nos interesa sobre todo la fecha de publicación de la imágen. El problema es que no devuelve un campo específico con la fecha, sino que aparece (algunas veces) dentro del texto del campo

descriptions. Por ejemplo, como se ve en la captura de abajo al buscar la imagen https://pbs.twimg.com/media/EgppeEHXcAENzXs.jpg correspondiente al tweetid 1299990238727516160:


```
import requests, json
url='https://pbs.twimg.com/media/EgppeEHXcAENzXs.jpg'
url_mrisa = "http://192.168.135.130:5000/search"
image_url = url
lang = 'es'
data = {
    "image_url":image_url,
    "resized_images":False, # Or True
    "cloud_api":False,
    "google_domain": 'google.com',
    "gl": lang,
    "hl": lang
}

headers = {'Content-type': 'application/json'}
r = requests.post(url_mrisa, headers=headers, data=json.dumps(data))
descriptions=r.json()[u'descriptions']
descriptions
```

```
['A clinic is a health facility that is primarily focused on the care of outpatients. Clinics can be
privately operated or publicly managed and funded. They typically\xa0...',
 'With your collaboration, you help other patients have a better experience and aid Hospital Clínic d
e Barcelona to further develop healthcare, advance research,\xa0...',
 '1024 × 576 - Aug 27, 2020 - Ángel Rodríguez de Guzmán ha abandonado la UCI del Gregorio Marañón a s
us 70 años tras 158 días ingresado por complicaciones\xa0...',
 '640 × 360 - Aug 30, 2020 - Ángel ha pasado más de cinco meses en la UCI por covid, más que ninguna
otra persona en España. Pero por fin, este jueves ha pasado a\xa0...',
 '640 × 360 - The latest Tweets from Cuca Sales (@cucasalesmoreno). Periodista. Reportera en Espejo P
úblico. Antena 3. Madrid-Zaragoza.',
 '1024 × 576 - Aug 27, 2020 - El Hospital Gregorio Marañón de Madrid ha trasladado a planta al pacien
te con covid-19 que más tiempo ha estado en una UCI en España,\xa0...',
 '640 × 360 - De seneste tweets fra Cuca Sales (@cucasalesmoreno). Periodista. Reportera en Espejo Pú
blico. Antena 3. Madrid-Zaragoza.']
```

En algunos resultados aparece la fecha, y en otros no. Además, no siempre siguen el mismo patrón. Sin embargo, en las pruebas realizadas, la mayor parte de las fechas se parsean correctamente con uno de las tres expresiones regulares siguientes:

1. regex1: `\d*\s\W\s\d*\s-\s(?P<date1>.+?\d{4})\s[-]\s(?P<desc1>\w.*)`
2. regex2: `(?P<date2>\d{1,2} days? ago).*- (?P<desc2>.*)\\*`
3. regex3: `(?P<date3>\w{3} \d{1,2}, \d{4}).*- (?P<desc3>.*)\\*`
4. regex4: `\d*\s\W\s\d*\s-\s.*(?P<date4>\d{2}-\d{2}-\d{4}`

La web de pythex.org[18] es de gran ayuda para verificar las regex. En el ejemplo de arriba, tenemos 3 resultados con fecha. Con regex 3 se extrae la fecha y descripción de los resultados i=3,4, y 6:

POST-PROCESAMIENTO DE DATOS Y GENERACIÓN DE TABLAS

Este proceso nos genera un nuevo subgrupo para cada tweet_id, que puede tener entre 1 y 7 registros, cada uno correspondiente a un link distinto con una fecha determinada. La herramienta recorre todos los elementos de este subgrupo, y aplica finditer al campo descriptions para extraer la fecha y la descripción de cada uno. En el caso de que no se encuentre una fecha, el elemento se descarta. El resultado se almacena en una tabla "tabla_tweets_RIS" como la que se muestra a continuación (un fragmento):

```
df_RIS.head()
```

	screen_name	tweet_id	tweet_date	img_date	i	link	best_guess	img link description	img_url	alert
0	_anapastor_	1276659003238596608	2020-06-26 23:30:00	2020-06-28	1.0	https://www.reddit.com/r/europe/comments/hgv5x...	logo guardia civil lgbt	Jun 28, 2020 - 2.4m members in the europe comm...	http://pbs.twimg.com/media/EbeGLr4XsAgXSA1.jpg	0
1	_anapastor_	1276659003238596608	2020-06-26 23:30:00	2020-06-27	3.0	https://www.heraldo.es/noticias/nacional/2020/...	logo guardia civil lgbt	Jun 27, 2020 - La Guardia Civil luce en su per...	http://pbs.twimg.com/media/EbeGLr4XsAgXSA1.jpg	0
2	_anapastor_	1276659003238596608	2020-06-26 23:30:00	2020-06-27	5.0	https://www.noticiasdenavarra.com/vivir-on/cie...	logo guardia civil lgbt	Jun 27, 2020 - Partidos, empresas e institucio...	http://pbs.twimg.com/media/EbeGLr4XsAgXSA1.jpg	0
3	_anapastor_	1280383043086307328	2020-07-07 06:08:00	2020-07-06	2.0	https://twitter.com/_anapastor_/status/1280383...	screenshot	Jul 6, 2020 - Precisamente eso es lo que se cr...	http://pbs.twimg.com/media/EcTVMaRXYAI_XVh.jpg	1
4	_anapastor_	1280432574180077569	2020-07-07 09:24:49	2020-07-07	5.0	https://twitter.com/ikercasillas/status/128041...	Iker Casillas	Jul 7, 2020 - Tocaste la gloria... Pocos puede...	http://pbs.twimg.com/media/EcTzKvbWsAAg0qH.jpg	1

Como puede verse, hay tweets repetidos (1276659003238596608) porque se ha encontrado más de un link con la imagen de dicho tweet.

Posteriormente, a partir de la tabla obtenida (tabla_tweets_RIS) se realiza un postprocesamiento de esta tabla mediante la función postprocess_RIS. Esta función por una parte extrae datos de la URL de los links encontrados, como la longitud, número de puntos, número de dígitos, etc. Y por otro lado realiza las siguientes acciones:

¿Aparece el nombre de la cuenta de Twitter en el dominio del link encontrado en la búsqueda inversa? Los elementos de la tabla que cumplan esta condición se filtran, ya que sería normal que un medio de comunicación publicara la misma imagen en su cuenta de Twitter y en su

[18] https://pythex.org

página web. El parámetro “ALERT” muestra esta condición, siendo 1 si en el link no aparece el nombre de la cuenta de Twitter (caso interesante)

También hace uso de un filtro adicional (listado ‘filtro_medios_2.txt') que reduce los resultados a únicamente aquellos con links pertenecientes al listado. Este listado incluye 228 medios de comunicación nacionales e internacionales.

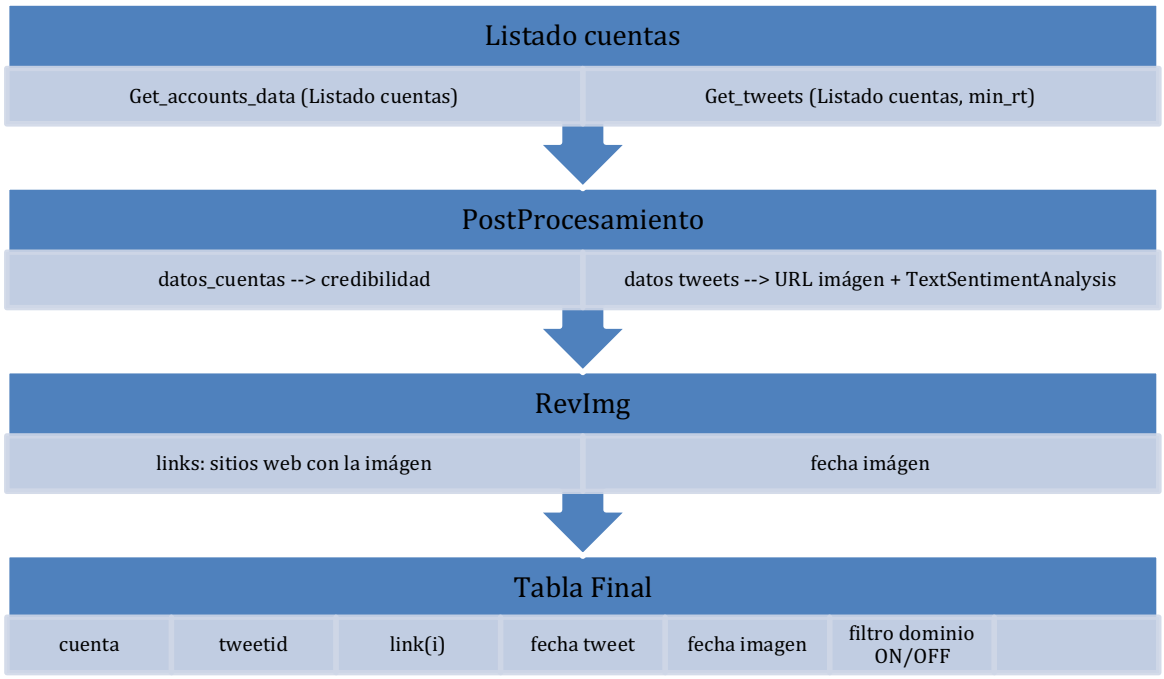
Los tweets con fechas de publicación anteriores a sus imágenes en principio no son interesantes para este proyecto. Sin embargo, no se pierden, ya que su inclusión/exclusión dependen del atributo “alert”.

Se calcula el grado de similitud entre el texto del tweet y el texto descriptivo del link encontrado con la función RevImg. Esto se realiza mediante el método SequenceMatcher de la libreria de python difflib. Si este parámetro es alto (coincidencia de palabras en ambos) significa que es probable que la imagen del link sea la que se está buscando.

La nueva tabla también calcula el tiempo entre la publicación de la imágen y la publicación del tweet, en días (delta_t). Si este tiempo es de 1 día o menos, el resultado se toma como “no interesante” (ALERT=0). Es necesario hacerlo así y descartar la información de la hora del Tweet porque, como hemos visto antes, debe compararse con fechas de los enlaces encontradas por MRISA y éstas se extraen de un campo de texto que sólo da información del día, mes y año. Por tanto, todas las fechas se presuponen a hora 00:00:00, aunque estrictamente no sea así. Sin embargo, esto no es un problema porque en coincidencias con delta_t de más de un día, el error en el redondeo de las horas se vuelve muy pequeño.

El resultado final es una tabla con los links encontrados para cada tweet, los enlaces a las imágenes y las fechas correspondientes. La herramienta también puede aceptar como argumento de entrada un sólo tweet mediante la función get_tweet_features(). Además, se crean las siguientes tablas adicionales.

- tabla_caracteristicas_tweets: Contiene las características de todos los tweets tras aplicar un postprocesamiento del texto
- datos_cuentas: Tabla en formato con las características de las cuentas de Twitter analizadas.
- tabla_resultados: tabla con los resultados de la búsqueda de imágenes inversa



IMPLEMENTACIÓN

Los procesos descritos en el apartado anterior se han implementado en su totalidad sobre pandas en Jypiter Notebook de Anaconda[19] . El sistema operativo en el que se ha instalado ha sido Windows 10 64 bits y la versión de python base ha sido la 3.7. Han sido necesarias instalar las siguientes librerías desde la consola de Anaconda.

pip3 install tweepy
pip3 install pandas
pip3 install tldextract
pip3 install xlrd
pip3 install openpyxl
pip3 install google-search-results
pip3 install googletrans
pip3 install botometer
pip3 install wget
pip3 install textblob
pip3 install tldextract

[19] https://jupyter.org/

Durante el transcurso de la preparación de este proyecto, actualizaron la versión de botometer a 1.6, por tanto, esa es la versión necesaria para que funcione sin problemas.

AUTENTICACIÓN Y APIS

Para acceder a la API de Twitter y llevar a cabo la extracción de tweets, es necesario autenticarse previamente con el método OAuth2, un tipo de autenticación básica usada por la API REST y de streaming standar de Twitter, API que se ha usado en este proyecto. El tipo de autenticación OAuth2 significa que se autentica la application creada en Twitter a tal efecto, en nombre del analista. Para poder usar la herramienta es necesario tener cuenta de desarrollador de Twitter [20] y poseer claves API de streaming (*consumer_key, consumer_secret, access_token_key y access_token_secret)* de Twitter para poder autenticarse con la API y realizar consultar.

Para llevar a cabo la autenticación OAuth2 se puede usar la librería **Tweepy.** Se pueden usar los métodos de la clase tweepy.API[21] para realizar todas las operaciones y consultas a Twitter de una manera más sencilla que con TwitterAPI. Por ejemplo, el usuario de una cuenta se obtiene con tweepy.API.get_user(<cuenta>)

```
consumer_key = 'XXXXXXXXXXXXXXXXXXXXXXXXX'
consumer_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
access_key = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
access_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
rapidapi_key = "XXXXXXXXXXXXXXXXXXXXXXXXX"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True, compression=True)

twitter_app_auth = {
    'consumer_key': consumer_key,
    'consumer_secret': consumer_secret,
    'access_token': access_key,
    'access_token_secret': access_secret
  }
bom = botometer.Botometer(wait_on_ratelimit=True, rapidapi_key=rapidapi_key, **twitter_app_auth)
```

Para poder usar la API de rapidapi[22] ha sido necesario realizar una subscripción a la misma, pero sin coste, ya que el número de consultas realizadas no hizo necesaria la modalidad de pago.

FUNCIONES

[20] https://developer.twitter.com/)
[21] https://tweepy.readthedocs.io/en/latest/api.html
[22] https://rapidapi.com

Se definen las siguientes funciones:

Get_accounts_data(*file*): Recopila información de un listado de cuentas de Twitter y calcula la credibilidad de las mismas. Los resultados se almacenan en dos tablas: tabla_cuentas.csv y tabla_cuentas.xlsx

get_userdata (*cuenta*): Recopila información de una cuenta de Twitter a partir del nombre de la cuenta. Calcula la credibilidad de la misma. Se almacena en dos tablas: datos_[cuenta].csv y datos_cuenta.xlsx

Get_tweets_tables (*file*, *min*_rt): Script principal para recopilar características y generar tablas a partir de un listado de cuentas de Twitter. Tiene como parámetros un fichero de texto con la lista de cuentas y un límite inferior al número de retweets de cada tweet encontrado. La función hace uso de TwitterAPI para buscar todos los tweets de las cuentas de la lista. El resultado es una tabla con los links encontrados con las imágenes. Además, se generan las siguientes tablas (en csv y xlsx):

- tabla_caracteristicas_tweets.: resultado del paso *postprocess* (tweets)
- tabla_tweets_pRIS: resultado del paso *postprocessRIS* (búsqueda inversa de imágenes)
- tabla_tweets_completa: Fusión de las dos tablas anteriores
- tabla ML: tabla para uso en Machine Learninig
- tabla_resultados: Salida de la función

check_tweet_pic (*tweetid*): Igual que el script anterior pero aplicado a un solo tweet. Genera las siguientes tablas:

- tabla_resultados_<*tweetid*>
- datos_<*cuenta*>
- tabla_ML_<*tweetid*>
- tabla_<*tweetid*>

Funciones secundarias usadas por las funciones anteriores:

postprocess(df): Procesamiento de los tweets recopilados y resultados almacenado en "tabla_carcaterísticas_tweets". La usa Get_tweets y get_tweet_features. L salida es un dataframe df_post

RevImg(df_post): Esta función usa la salida de la función anterior y realiza la búsqueda inversa de imágenes por medio de MRISA. Luego se procesarán los resultados (df_RIS) con la siguiente función.

postprocessRIS(df_RIS): Se eliminan posibles duplicados, nos quedamos sólo con las fechas ignorando las horas (ya que la búsqueda inversa de imágenes no suele dar este dato) y creamos una nueva columna ALERT mejorada, donde se tiene en cuenta, aparte de la diferencia de fechas entre la publicación del tweet y la imagen, la existencia en el link del propio nombre de la cuenta de Twitter, lo que indica que, independientemente de las fechas, el origen de la imagen es la misma cuenta de Twitter. Esto lo hacemos con postprocess_RIS aplicado al dataframe df_RIS

get_past_date (str_days_ago): Transforma los resultados de "X days ago" a formato fecha. Necesario para el detectar fechas con MRISA en este formato.

credibility(cuenta): Forma rápida de obtener la credibilidad de una cuenta. Llama a get_user_data realmente, por lo que se crea la tabla correspondiente.

hash_file(file): Función usada en la etapa de postprocess en la que se almacenan las imágenes en local. Es necesario calcular los hashes MD5 de todas para elmiminar duplicados de cada cuenta.

Get_accounts_data

```
file = str(input('Introduce el nombre del fichero con la lista de cuentas de Twitter: '))

def Get_accounts_data(file):

    bom = botometer.Botometer(wait_on_ratelimit=True, rapidapi_key=rapidapi_key, **twitter_a
pp_auth)
    with open(file, encoding="utf-8-sig") as accountdata:
        cuentas = csv.reader(accountdata)
        df_usuarios=[]
        for cuenta in cuentas:

            #User Features [F1]
            F1_data=[]
            user = api.get_user(cuenta[0])
            screen_name=user.screen_name
            display_name=user.name
            user_descr=user.description
            descr_len=len(user_descr)
            verified=user.verified
            created_at=user.created_at
            age = (datetime.datetime.today() - created_at).days
            location = user.location
            def_prf_img=user.default_profile_image
            def_prf=user.default_profile
            n_publications=user.statuses_count
            n_listed= user.listed_count
            user_url=user.url
            user_favs=user.favourites_count
            followers=user.followers_count
            friends=user.friends_count
            user_id=user.id
            date_twitter=datetime.datetime.strptime('21-03-2006', '%d-%m-%Y')
            twitter_tmax=(datetime.datetime.today()-date_twitter).days
            age = (datetime.datetime.today() - created_at).days
            location = user.location
            def_prf_img=user.default_profile_image
            def_prf=user.default_profile
            n_publications=user.statuses_count
            n_listed= user.listed_count
            user_url=user.url
            user_favs=user.favourites_count
            followers=user.followers_count
            friends=user.friends_count
            user_id=user.id
            AgP=age/twitter_tmax
            Pubs_per_day=n_publications/age
            Fr_Fo_ratio=friends/followers
            np.multiply(def_prf_img, 1)
            loc_set=1 if location!="" else 0
            ppdP = 0 if Pubs_per_day<1 else 1
            np.multiply(verified,1)
            url_count=1 if user_url!=0 else 0
            DlenP=0 if descr_len==0 else 1
            ImPrfParam=not def_prf_img
            PrfParam=not def_prf

            try:
                lang=bom.check_account(user.id)['user']['majority_lang']

            except:
                print("Botometer Error en la cuenta",screen_name)
                pass
```

```
            else:

                if lang=='en':
                     score_bot_univ = bom.check_account(user_id)['display_scores']['english'
]['overall']
                else:
                     score_bot_univ = bom.check_account(user_id)['display_scores']['universa
l']['overall']

                F1_data.append([str(user.id),screen_name,user_descr,descr_len,verified,creat
ed_at,age, location,loc_set,def_prf_img,def_prf,n_publications,Pubs_per_day, n_listed,user_u
rl,followers,friends,Fr_Fo_ratio,user_favs,score_bot_univ])

            col_names = ["user_id","screen_name","user_descr","descr_len","verified","create
d at", "account_age (d)","location","loc_set","def_img_prf","def_prf","n_pubs","Pubs_per_day
","n_listed","user_url","followers","friends",'Fr_Fo_ratio',"user_favs","score_bot"]

            df_account=pd.DataFrame(data=F1_data,columns=col_names)
            df_usuarios=df_account.append(df_usuarios)

         # Parámetro antiguedad de la cuenta: t(d)/tmax(d) age_parameter=AgP
        df_usuarios['AgP']=df_usuarios['account_age (d)']/twitter_tmax

        #Num de tweets diarios: pubs per day : ppdP
        df_usuarios['ppdP']=df_usuarios.apply(lambda row: 0 if row['Pubs_per_day']<1 else 1,
axis=1)

        #Friends/Followers ratio
        df_usuarios['FrFoP']=(
            np.select(
            condlist=[df_usuarios['friends']>0,df_usuarios['friends']==0],
            choicelist=[1-df_usuarios['Fr_Fo_ratio'],0]            )
        )

         #Verified account
        df_usuarios["verified"] = df_usuarios["verified"].astype(int)

        #website in profile
        df_usuarios['url_count']=df_usuarios['user_url'].apply(lambda x: 1 if x!=0 else 0)

        #Longitud de la descripción del perfil
        df_usuarios['DlenP']=df_usuarios.apply(lambda row: 0 if row['descr_len']==0 else 1,
                                                            axis=1)
        #Localidad
        df_usuarios['loc_set']

        #Imágen de perfil personalizada:
        df_usuarios["ImPrfParam"]=(~df_usuarios['def_img_prf']).astype(int)

        #Perfil por defecto

        df_usuarios["PrfParam"] = (~df_usuarios["def_prf"]).astype(int)

        #Followers a la cuenta
        df_usuarios['FolParam']=(
            np.select(
            condlist=[df_usuarios['followers']>700,df_usuarios['followers']<700,
                                            df_usuarios['followers']==0],
            choicelist=[1,0.5,0]
            )
        )
        #Likes del usuario (user_n_favs)
        df_usuarios['FavsParam']=(
            np.select(
            condlist=[df_usuarios['user_favs']>200,df_usuarios['user_favs']<200,
                                                    df_usuarios['user_favs']==0],
            choicelist=[1,0.5,0])
        )

        #Parámetro botometer (0:no bot - 1:bot)
        df_usuarios['NoBotParam']=df_usuarios.apply(lambda row: 1-row['score_bot']/5,axis=1)
        df_usuarios['credibility']=df_usuarios[['AgP', 'ppdP', 'FrFoP', 'DlenP', 'ImPrfParam
','PrfParam', 'FolParam','FavsParam', 'NoBotParam']].mean(axis=1)

        df_usuarios.reset_index(drop=True,inplace=True)
        df_usuarios.to_csv(f'csv/tabla_cuentas_{file}.csv', index = False)
```

```
        df_usuarios.to_excel(f'csv/tabla_cuentas_{file}.xlsx', index = False)

    display(df_usuarios)
def main():
    Get_accounts_data(file)
if __name__ == '__main__':
    main()
```

Tabla que se obtiene al ejecutar Get_account_data para un fichero con dos cuentas: el_pais y elmundoes:

```
Introduce el nombre del fichero con la lista de cuentas de Twitter: cuenta.txt
```

	user_id	screen_name	user_descr	descr_len	verified	created at	account_age (d)	location	loc_set	def_img_prf	..
0	7906082	el_pais	La mejor información en español. Con nuestra m...	155	1	2007-08-06 16:20:09	4793	Madrid	1	False	..
1	14436030	elmundoes	Cuenta oficial de EL MUNDO. https://t.co/vUaVD...	51	1	2008-04-18 18:29:25	4537	España	1	False	..

2 rows × 31 columns

Get_userdata

```
def get_userdata():

    F1_data=[]
    date_twitter=datetime.datetime.strptime('21-03-2006', '%d-%m-%Y')
    twitter_tmax=(datetime.datetime.today()-date_twitter).days

    try:
        user=api.get_user(screen_name)

    except tweepy.error.TweepError:
        print("Usuario no encontrado")
        pass

    display_name=user.name
    user_descr=user.description
    descr_len=len(user_descr)
    verified=user.verified
    created_at=user.created_at
    age = (datetime.datetime.today() - created_at).days
    location = user.location
    def_prf_img=user.default_profile_image
    def_prf=user.default_profile
    n_publications=user.statuses_count
    n_listed= user.listed_count
    user_url=user.url
    user_favs=user.favourites_count
    followers=user.followers_count
    friends=user.friends_count
    user_id=user.id
    lang=bom.check_account(user.id)['user']['majority_lang']
    AgP=age/twitter_tmax
    Pubs_per_day=n_publications/age
```

```
    Fr_Fo_ratio=friends/followers
    np.multiply(def_prf_img, 1)
    loc_set=1 if location!="" else 0
    ppdP = 0 if Pubs_per_day<1 else 1
    np.multiply(verified,1)
    url_count=1 if user_url!=0 else 0
    DlenP=0 if descr_len==0 else 1
    ImPrfParam=not def_prf_img
    PrfParam=not def_prf


    try:
        bom.check_account(user)['display_scores']
    except:

        pass

    if lang=='en':
        score_bot = bom.check_account(user_id)['display_scores']['english']['overall']

    else:
        score_bot = bom.check_account(user_id)['display_scores']['universal']['overall']

    NoBotParam=1-score_bot/5
    FrFoP=(
        np.select(
        condlist=[friends>0,friends==0],
        choicelist=[1-Fr_Fo_ratio,0]
        )
    )

    FolParam=(
        np.select(
        condlist=[followers>700,followers<=700,followers==0],
        choicelist=[1,0.5,0]
        )
    )

    FavsParam=(
        np.select(
        condlist=[user_favs>200,user_favs<=200,user_favs==0],
        choicelist=[1,0.5,0]
        )
    )

    credibility=np.mean([AgP, ppdP, FrFoP, DlenP, ImPrfParam, PrfParam, FolParam,FavsParam,
NoBotParam])
    F1_data.append([str(user.id),screen_name,user_descr,verified,created_at,age,location,def
_prf_img,def_prf,n_publications,n_listed,user_url,followers,
                friends,user_favs,score_bot,credibility])

    col_names = ["user_id","screen_name","user_descr","verified","created at","account_age (
d)",
                 "location","def_img_prf","def_prf","n_pubs",
                 "n_listed","user_url","followers","friends","user_n_favs","score_bot","cred
ibility"]

    df_user=pd.DataFrame(F1_data,columns=col_names)
    df_user.to_csv(f'datos_{screen_name}.csv', index = False)
    df_user.to_excel(f'datos_{screen_name}.xlsx', index = False)

    display(df_user)

def main():
    get_userdata()

if __name__ == '__main__':
    main()
```

Lo probamos con la cuenta de Twitter de @elmundoes:

```
Introduce el nombre de la cuenta de Twitter: elmundoes
```

	user_id	screen_name	user_descr	verified	created at	account_age (d)	location	def_img_prf	def_prf	n_pubs	n_liste
0	14436030	elmundoes	Cuenta oficial de EL MUNDO. https://t.co/vUaVD...	True	2008-04-18 18:29:25	4537	España	False	False	309324	3040

Get_tweets_tables

```
file = str(input('Introduce el nombre del fichero con la lista de cuentas de Twitter: '))
min_retweets = int(input('Introduce el número mínimo de retweets por tweet: '))

def Get_tweets(file,min_retweets):
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True, compress
ion=True)

    with open(file, encoding="utf-8-sig") as origenDatos:
        cuentas = csv.reader(origenDatos)
        df_post=[]
        for cuenta in cuentas:
            alltweets = []
            images_dir=str('images/' + cuenta[0] + '/')
            screen_name = cuenta[0]
            try:
                os.stat(images_dir)
            except:
                os.makedirs(images_dir)

            #Petición de los tweets más recientes (200 es el máximo)
            new_tweets = api.user_timeline(screen_name = cuenta[0],count=200,tweet_mode='ext
ended',include_rts='true')

            #se almacenan los más recientes
            alltweets.extend(new_tweets)

            #save the id of the oldest tweet less one
            oldest = alltweets[-1].id - 1

            #Sigue recogiendo tweets hasta que se llegue al límite

            while len(new_tweets) > 0:

                new_tweets = api.user_timeline(screen_name = cuenta[0],count=200,max_id=olde
st,tweet_mode='extended',include_rts='true')

                alltweets.extend(new_tweets)
                #Se guarda el ID del tweet más antiguo
                oldest = alltweets[-1].id - 1

            # Características de los tweets (F2)

            F2_data = []
            for tweet in alltweets:

                if tweet.retweet_count < min_retweets:

                    continue

                try:
                    image_url=tweet.entities['media'][0]['media_url']

                except (NameError,KeyError):
                    pass

                else:
                    if image_url.find('amplify_video') > 0 or image_url.find('video_thumb')
> 0:
                        continue
```

```
                    imagesfiles = [] # this is the list I want the image urls to go into
                    image_url=tweet.entities['media'][0]['media_url']
                    file_location = str(image_url)
                    image_name = str(image_url).rsplit('/', 1)[-1]
                    imagesfiles.append(file_location)
                    file_location = images_dir + file_location.rsplit('/', 1)[-1]
                    urllib.request.urlretrieve(image_url, file_location)
                    tweet_id = str(tweet.id_str)
                    screen_name = tweet.user.screen_name
                    pub_date = tweet.created_at
                    full_text = tweet.full_text
                    lang = tweet.lang
                    rt_count = tweet.retweet_count
                    fav_count = tweet.favorite_count


                    F2_data.append([tweet_id,screen_name,pub_date,full_text,lang,rt_count,
                                    fav_count,image_url,image_name])

            F2_labels = ["tweet_id","screen_name","created_at","text","lang","retweets","fav
s","image_url","image_name"]

            df=pd.DataFrame(data=F2_data,columns=F2_labels)
            df_tweets_post=postprocess(df)
            translator = Translator()
            df_tweets_post['subjectivity']=df_tweets_post['text'].apply(lambda x:TextBlob(tr
anslator.translate(x,src=df_tweets_post['lang'][0]).text).sentiment.subjectivity)
            df_post=df_tweets_post.append(df_post)

        df_post.drop_duplicates(subset=['tweet_id'], keep = 'first', inplace=True)
        df_post['account_cred']=get_userdata(screen_name)['credibility'][0]
        df_post.reset_index(drop=True,inplace=True)
        df_post.to_excel(f'tabla_caracteristicas_tweets_{file}.xlsx',index=False)
        df_post.to_csv(f'tabla_caracteristicas_tweets_{file}.csv',index=False)
        df_RIS=RevImg(df_post)
        df_postRIS=postprocess_RIS(df_RIS)

        '''
        Eliminamos posibles duplicados, nos quedamos sólo con las fechas ignorando las horas
        (ya uqe la búsqueda inversa de imágenes no suele dar este dato) y creamos una nueva
        columna ALERT mejorada, donde se tiene en cuanta, aparte de la diferencia de fechas
        entre la publicación del tweet y la imagen, la existencia en el link del propio nomb
        re de la cuenta de Twitter, lo que indica que, independientemente de las fechas, el
        origen de la imagen es la misma cuenta de Twitter. Esto lo hacemos con postprocess_R
        IS aplicado al dataframe df_RIS:
        '''
        df_postRIS.reset_index(drop=True,inplace=True)
        df_postRIS.to_excel(f'tabla_tweets_pRIS_{file}.xlsx',index=False)
        df_postRIS.to_csv(f'tabla_tweets_pRIS_{file}.csv',index=False)
        df_tweets_completa=pd.merge(df_post, df_postRIS, on=['tweet_id','image_url','screen_
name','text'])
        df_tweets_completa.reset_index(drop=True,inplace=True)
        df_tweets_completa.drop_duplicates(subset=['link'],keep='first',inplace=True)
        df_tweets_completa.to_csv(f'tabla_tweets_completa_{file}.csv', index = False)
        df_tweets_completa.to_excel(f'tabla_tweets_completa_{file}.xlsx', index = False)

      df_tweets_completa_SEL=df_tweets_completa[['screen_name','tweet_id','retweets','RT','
favs','i','img_domain',
        'ext','lang','total_length', 'capitals', 'caps_vs_length', 'num_words', 'num_unique_
words', 'words_vs_unique',
        'num_exclamation_marks','num_question_marks', 'num_question_marks_es','question_mark
s_ratio', 'num_punctuation',
        'num_symbols', 'num_smilies','num_sad', 'mean_word_len', 'polarity', 'subjectivity',
'similarity','account_cred',
        'delta_t','link_dom_sym', 'link_digits', 'link_Dots','link_url_syms', 'link_dom_digi
ts','link_delimiter_path',
        'link_dom_dots','link_length','link_NumberRate','ALERT','filter','account_cred']]

        df_tweets_ML=df_tweets_completa_SEL[(df_tweets_completa_SEL['ALERT']==1) & (df_tweet
s_completa_SEL['filter']==1)]
        df_tweets_ML.to_csv(f'tabla_ML_{file}.csv', index = False)
        df_tweets_ML.to_excel(f'tabla_ML_{file}.xlsx',index=False)
        df_mini=df_tweets_completa[(df_tweets_completa['ALERT']==1) & (df_tweets_completa['f
ilter']==1)][['screen_name','tweet_id',
        'text','retweets','favs','link','image_url','tweet_date','img_date','delta_t','polar
ity','subjectivity','filter','account_cred']]
```

```
        df_mini_filtro_OFF=df_tweets_completa[df_tweets_completa['ALERT']==1][['screen_name'
,'tweet_id',
        'text','retweets','favs','link','image_url','tweet_date','img_date','delta_t','polar
ity','subjectivity','filter','account_cred']]
        df_mini.to_excel(f'tabla_resultados_{file}.xlsx',index=False)


    display(df_mini)

def main():
    Get_tweets(file,min_retweets)

# Execute main() function
if __name__ == '__main__':
    main()
```

FUNCIONES SECUNDARIAS

Postprocess

```
def postprocess_RIS(df):
    df2=df.copy()
    lista_medios= open('filtro_medios_2.txt').read().splitlines()
    df2['link_path']=df2['link'].apply(lambda x:urlparse(x).path)
    df2['link_query']=df2['link'].apply(lambda x:urlparse(x).query)
    df2['ext']=df2['img_url'].apply(lambda x: re.findall(r'\.([a-zA-Z]{2,3})$',str(x))[0])
    df2['link_frag']=df2['link'].apply(lambda x:urlparse(x).fragment)
    df2['link_dom_sym']=df2['link'].apply(lambda x:len(re.findall('[^A-Za-z0-9.]', str(urlparse(x)
.hostname))))
    df2['link_digits']=df2['link'].apply(lambda x:len(re.findall('\d',str(x))))
    df2['link_Dots']=df2['link'].apply(lambda x:x.count('.'))
    df2['link_url_syms']=df2['link'].apply(lambda x:len(re.findall('\W', str(x))))
    df2['link_dom_digits']=df2['link_query'].apply(lambda x:len(re.findall('\d', str(x))))
    df2['link_delimiter_path']=df2['link_path'].apply(lambda x:len(re.findall('/', x)))
    df2['link_dom_dots']=df2['link'].apply(lambda x:(str(urlparse(x).hostname)).count('.'))
    df2['link_length']=df2.apply(lambda row:int(len(row['link'])-len(row['link_frag'])),axis=1)
    df2['link_NumberRate']=df2['link'].apply(lambda x:len(re.findall('\d', x))/len(x))
    df2['img_domain']= df2['link'].apply(lambda x:re.sub(r'www\.','',urlparse(x).hostname))
    df2['tweet_id']= df2['tweet_id'].astype(str)
    df2['img_date'] = df2['img_date'].apply(lambda x: x.strftime("%d/%m/%Y"))
    #df2['img_date'] = df2['img_date'].apply(lambda x:datetime.strptime(x, '%Y-%m-%d %H:%M:%S').st
rftime("%d/%m/%Y"))
    df2['tweet_date'] = df2['tweet_date'].apply(lambda x: x.strftime("%d/%m/%Y"))
    #df2['tweet_date'] = df2['tweet_date'].apply(lambda x:datetime.strptime(x, '%Y-%m-%d %H:%M:%S'
).strftime("%d/%m/%Y"))
    df2['delta_t'] = df2.apply(lambda row: (pd.to_datetime(row['tweet_date'],format='%d/%m/%Y')-pd
.to_datetime(row['img_date'],format='%d/%m/%Y')).days,axis=1)
    #df2['delta_t'] = df2.apply(lambda row: (pd.to_datetime(row['tweet_date'])-pd.to_datetime(row[
'img_date'])).days,axis=1)
    df2['namesimple']=df2['screen_name'].apply(lambda x:re.sub(r'_\w{2}$',"", x.lower()))
    df2['name_in_link'] = df2.apply(lambda x: x.namesimple in x.link.lower(), axis=1)
    df2.rename(columns={'img_url':'image_url'},inplace=True)
    df2['tweet_url']=df2.apply(lambda row:'https://twitter.com/'+row['screen_name']+'/status/'+row
['tweet_id'], axis=1)
    df2.rename(columns={'img link description':'img_link_description'},inplace=True)
    df2['img_link_description']=df2['img_link_description'].apply(lambda x:re.sub(r'.*-\s',"", str
(x)))
    df2['filter']=df2['img_domain'].apply(lambda x: 1 if x in lista_medios else 0)
    df2['similarity']=df2.apply(lambda row: SM(None,row['text'],row['img_link_description']).ratio
(),axis=1)
    df2.reset_index(drop=True,inplace=True)
    df2['ALERT'] = ''
    df2['manual_chk']=''
    for n in range(0,len(df2)):

        if df2['name_in_link'][n] == True:
            df2['ALERT'][n] = 0

        elif df2['name_in_link'][n] == False and df2['delta_t'][n]>1:
            df2['ALERT'][n] = 1

        else:
            df2['ALERT'][n] = 0
```

```
    return df2
```

RevImg

Búsqueda en la tabla de tweets de imágenes en el metabuscador MRISA:

```
def RevImg(df):

    df_data_RIS=[]
    for n in range(len(df)):
        name=df['screen_name'][n]
        image_url=df['image_url'][n]
        tweet_text=df['text'][n]
        url_mrisa = "http://192.168.135.130:5000/search"
        headers = {'Content-type': 'application/json'}

        if df.at[n,'lang']==None:
            lang='en'
        else:
            lang = df['lang'][n]

        if lang=='en':
            dom='com'
        else:
            dom=str(lang)

        data = {
        "image_url":image_url,
        "resized_images":False,
        "cloud_api":False,
        "google_domain": str("google."+dom),
        "gl": str(df['lang'][n]),
        "hl": str(df['lang'][n])
        }

        try:
            r=requests.post(url_mrisa, headers=headers, data=json.dumps(data),timeout=(2,10)
)
        except:
            print('\nError en',image_url,'\nVerifica que el servidor MRISA está accesible')
            pass
        else:
            descriptions=r.json()[u'descriptions']
            best_guess=r.json()['best_guess']
            titles=r.json()['titles']
            links=r.json()['links']
            similar_images=r.json()['similar_images']

        #pat = '\d*\s\W\s\d*\s-\s(?P<date>.+?\d{4})\s[-]\s(?P<desc>\w.*)'
        regex = re.compile(r'\d*\s\W\s\d*\s-\s(?P<date1>.+?\d{4})\s[-]\s(?P<desc1>\w.*)|(?P<
date2>\d{1,2} days? ago)\s[-]\s(?P<desc2>.*)|(?P<date3>\w{3} \d{1,2}, \d{4}).*- (?P<desc3>.*
)\\*', re.MULTILINE)
        tweet_date=df['created_at'].apply(lambda x:pd.to_datetime(x,format='%Y-%m-%d'))


        RIS_data = []
        for i in range(len(descriptions)):

            for match in regex.finditer(descriptions[i]):

                if match.group('date1')==None:

                    pass

                    if match.group('date2')==None:
                        #print("No se encuentra fecha con patrón 2")
                        pass

                        if match.group('date3')==None:
                            #print("No se encuentra fecha con patrón 3")
                            pass

                        else:
                            img_date=pd.Timestamp(match.group('date3'))
                            descr=match.group('desc3')
```

```
                    else:
                        date2=match.group('date2')
                        img_date=pd.Timestamp(get_past_date(date2))
                        #img_date=img_date.strftime("%d-%m-%Y")
                        descr=match.group('desc2')


                else:
                    img_date=pd.Timestamp(match.group('date1'))
                    descr=match.group('desc1')

                RIS_data.append({'screen_name':df['screen_name'][n],
                            'tweet_id':str(df['tweet_id'][n]),
                             'text':df['text'][n],
                            'tweet_date':tweet_date[n],
                            'img_date':img_date,
                             'i':i,
                            'link': links[i],
                            'best_guess':best_guess,
                             'img link description':descr,
                            'img_url':df['image_url'][n],
                             'tweet_text':tweet_text,
                             'alert': 1 if img_date < tweet_date[n] else 0
                            })

            df_data=pd.DataFrame(RIS_data)

        df_data_RIS=df_data.append(df_data_RIS)
        df_data_RIS.reset_index()
        print(n," de ",len(df['image_url']))

    return df_data_RIS
```

Postprocess_RIS

```
def postprocess_RIS(df):
    df2=df.copy()
    lista_medios= open('filtro_medios_2.txt').read().splitlines()
    df2['link_path']=df2['link'].apply(lambda x:urlparse(x).path)
    df2['link_query']=df2['link'].apply(lambda x:urlparse(x).query)
    df2['ext']=df2['img_url'].apply(lambda x: re.findall(r'\.([a-zA-Z]{2,3})$',str(x))[0])
    df2['link_frag']=df2['link'].apply(lambda x:urlparse(x).fragment)
    df2['link_dom_sym']=df2['link'].apply(lambda x:len(re.findall('[^A-Za-z0-9.]', str(urlpa
rse(x).hostname))))
    df2['link_digits']=df2['link'].apply(lambda x:len(re.findall('\d',str(x))))
    df2['link_Dots']=df2['link'].apply(lambda x:x.count('.'))
    df2['link_url_syms']=df2['link'].apply(lambda x:len(re.findall('\W', str(x))))
    df2['link_dom_digits']=df2['link_query'].apply(lambda x:len(re.findall('\d', str(x))))
    df2['link_delimiter_path']=df2['link_path'].apply(lambda x:len(re.findall('/', x)))
    df2['link_dom_dots']=df2['link'].apply(lambda x:(str(urlparse(x).hostname)).count('.'))
    df2['link_length']=df2.apply(lambda row:int(len(row['link'])-len(row['link_frag'])),axis
=1)
    df2['link_NumberRate']=df2['link'].apply(lambda x:len(re.findall('\d', x))/len(x))
    df2['img_domain']= df2['link'].apply(lambda x:re.sub(r'www\.','',urlparse(x).hostname))
    df2['tweet_id']= df2['tweet_id'].astype(str)
    df2['img_date'] = df2['img_date'].apply(lambda x: x.strftime("%d/%m/%Y"))
    #df2['img_date'] = df2['img_date'].apply(lambda x:datetime.strptime(x, '%Y-%m-%d %H:%M:%
S').strftime("%d/%m/%Y"))
    df2['tweet_date'] = df2['tweet_date'].apply(lambda x: x.strftime("%d/%m/%Y"))
    #df2['tweet_date'] = df2['tweet_date'].apply(lambda x:datetime.strptime(x, '%Y-%m-%d %H:
%M:%S').strftime("%d/%m/%Y"))
    df2['delta_t'] = df2.apply(lambda row: (pd.to_datetime(row['tweet_date'],format='%d/%m/%
Y')-pd.to_datetime(row['img_date'],format='%d/%m/%Y')).days,axis=1)
    #df2['delta_t'] = df2.apply(lambda row: (pd.to_datetime(row['tweet_date'])-pd.to_datetim
e(row['img_date'])).days,axis=1)
    df2['namesimple']=df2['screen_name'].apply(lambda x:re.sub(r'_\w{2}$',"", x.lower()))
    df2['name_in_link'] = df2.apply(lambda x: x.namesimple in x.link.lower(), axis=1)
    df2.rename(columns={'img_url':'image_url'},inplace=True)
    df2['tweet_url']=df2.apply(lambda row:'https://twitter.com/'+row['screen_name']+'/status
/'+row['tweet_id'], axis=1)
    df2.rename(columns={'img link description':'img_link_description'},inplace=True)
    df2['img_link_description']=df2['img_link_description'].apply(lambda x:re.sub(r'.*-\s',"
", str(x)))
    df2['filter']=df2['img_domain'].apply(lambda x: 1 if x in lista_medios else 0)
```

```
    df2['similarity']=df2.apply(lambda row: SM(None,row['text'],row['img_link_description'])
.ratio(),axis=1)
    df2.reset_index(drop=True,inplace=True)
    df2['ALERT'] = ''
    df2['manual_chk']=''
    for n in range(0,len(df2)):

        if df2['name_in_link'][n] == True:
            df2['ALERT'][n] = 0

        elif df2['name_in_link'][n] == False and df2['delta_t'][n]>1:
            df2['ALERT'][n] = 1

        else:
            df2['ALERT'][n] = 0
    return df2
```

CASOS DE ESTUDIO

CASO 1. LISTADO DE CUENTAS DE MEDIOS DE COMUNICACIÓN

Hemos definido las funciones necesarias para realizar la extracción de datos de las cuentas y de los tweets, pero falta aplicarla a un caso real. Vamos a aplicar las herramientas creadas a un listado de cuentas de Twitter de medios de comunicación nacionales e internacionales (fichero media_all.txt). La lista la componen las siguientes 55 cuentas:

elmundoes	rtve	PatchTweet
sextaNoticias	20m	Storyzy
elperiodico	rne	tassagency_en
okdiario	elperiodistadi1	InfoWarsMedia
La_SER	elespanolcom	nytimes
elespanolcom	CNN	Reuters
eldiarioes	PRNewswire	lavozdeasturias
JotDownSpain	bing	lanuevaespana
el_pais	wikinoticias	21_horas
EFEnoticias_ES	PressReader	huzlers
elconfidencial	YahooNoticias	newsbuzzdaily
elindepcom	BBC	TNROnline
ElHuffPost	euronews	newsdailyreport
publico_es	googlenews	gatewaypundit
ctxt_es	TheAtlantic	newswarz
msn_es	dpa	BurrardStreetJ
cnnespana	1stHeadlines	NacionDigital2
abc_es	foxheadlines	MediterraneoDGT

Viendo la lista se puede constatar dos ausencias muy relevantes: EFEnoticias y AP. La razón de excluir estas dos asociaciones de prensa tan importantes es que generan muchísimos tweets con fotos y son la fuente de noticias y recursos gráficos de otros medios. Por tanto, analizar las fotografías de los tweets de la agencia EFE o Associated Press resultaría en un gasto excesivo de recursos con prácticamente ninguna posibilidad de obtener resultados interesantes para nuestro propósito de encontrar fake news por fotografías ya usadas con anterioridad.

Al analizar la lista anterior con get_tweets_tables se ha obtenido una tabla con 9885 elementos y 54 características, que son:

'screen_name', **'tweet_id'**, **'tweet_date'**, **'img_date'**, 'created_at',
'text', 'lang', **'retweets'**, **'favs'**, **'image_url'**, 'image_name', 'ext',
'urls', 'hashtags', 'total_length', 'capitals', 'caps_vs_length',
'num_words', 'num_unique_words', 'words_vs_unique',
'num_exclamation_marks', 'num_question_marks', 'num_question_marks_es',
'question_marks_ratio', 'num_punctuation', 'num_symbols', 'num_smilies',
'num_sad', 'mean_word_len', 'polarity', 'subjectivity', 'account_cred',
'RT', 'orig_user', 'i', 'link', 'img_link_description', 'similarity',
'link_dom_sym', 'link_digits', 'link_Dots', 'link_url_syms',
'link_dom_digits', 'link_delimiter_path', 'link_dom_dots',
'link_length', 'link_NumberRate', 'img_domain', 'delta_t',
'name_in_link', 'tweet_url', 'filter', 'ALERT', 'manual_chk'

```
1 df_final_g1=pd.read_excel('Base/csv/final/tabla_final_g1_full.xlsx')
```

```
1 df_final_g1.shape
```

```
(9885, 54)
```

Tras el filtrado por los campos ALERT=1 y filter=1 se reducen los resultados a 1026 casos. Estos filtros sirven para que se muestren unicamente los casos de tweets con fecha (tweet_date) posterior a la fecha de la imagen (tweet_date). El filtro "filter" se aplica sobre los dominios de los enlaces de las búsquedas inversas para incluir sólo los que están en la lista de "filtro_medios_2.txt"

yahoo.com	rtve.es	vanitatis.elconfidencial.com
news.yahoo.com	abc.es	lavanguardia.com
nytimes.com	tass.com	elperiodico.com
bbc.co.uk	bangkokpost.com	barrons.com
bbc.com	elespanol.com	washingtonpost.com
efe.com	france24.com	archyde.com
msn.com	publico.es	blogspot.com
cnn.com	rtl.lu	huffingtonpost.es
dailymail.co.uk	thesun.co.uk	alamy.com
theguardian.com	eldiario.es	reuters.com
elpais.com	m.eldiario.es	ctvnews.ca
lamoncloa.gob.es	maldita.es	indiatimes.com
english.elpais.com	newtral.es	express.co.uk
ctxt.es	elconfidencial.com	scmp.com

voanews.com
beforeitsnews.com
elmundo.es
radiotimes.com
okdiario.com
gulfnews.com
metro.co.uk
go.com
channelnewsasia.com
rfi.fr
telegraph.co.uk
nbcnews.com
dw.com
20minutos.es
cbc.ca
lavozdegalicia.es
threader.app
tekdeeps.com
wsj.com
nst.com.my
newsweek.com
forbes.com
sky.com
mirror.co.uk
periodistadigital.com
inews.co.uk
globalnews.ca
world-today-news.com
thenational.ae
libertaddigital.com
usatoday.com
wokv.com
inform.kz
europapress.es
deccanherald.com
businessinsider.com
apa.az
cnbc.com
occupydemocrats.com
ft.com
thejakartapost.com
theatlantic.com
bloomberg.com
marca.com
euronews.com
ondacero.es
as.com
okdiario.com
independent.co.uk
timesofisrael.com
foxnews.com
nypost.com
rappler.com
esdiario.com
eldiario.es
npr.org
standard.co.uk
phys.org
aljazeera.com
vozpopuli.com
arabnews.com
axios.com
theaustralian.com.au
abc.net.au
teletrader.com
wcax.com
elindependiente.com
thetimes.co.uk
tv6.news
thesun.ie
medium.com
newsbeezer.com
elnacional.cat
lasexta.com
dawn.com
thetop10news.com
levante-emv.com
knowledia.com
thehill.com
rnz.co.nz
eleconomista.es
dailystar.co.uk
larazon.es
laprensalatina.com
madridmetropolitan.com
trendsmap.com
burgeronreport.com
cgtn.com
ibtimes.com
hindustantimes.com
the-sun.com
atalayar.com
apnews.com
tumblr.com
inforos.ru
thelocal.it
politico.eu
thestar.com
digitalspy.com
usnews.com
thelocal.es
sfchronicle.com
elcorreo.com
flipboard.com
insider.com
lainformacion.com
antena3.com
sueddeutsche.de
euroweeklynews.com
majorcadailybulletin.com
factba.se
mb.com.ph
theglobeandmail.com
latimes.com
theconversation.com
foreignpolicy.com
corealpha.org
seattletimes.com
threadreaderapp.com
tweet-per-sec.com
dailysabah.com
lasprovincias.es
chicagotribune.com
augc.org
thehindu.com
irishtimes.com
themoscowtimes.com
rte.ie
diariopatriota.com
capitalthinkingblog.com
telesurenglish.net
thestar.com.my
bluewaterhealthyliving.com
mediterraneodigital.com
inquirer.net
elnortedecastilla.es
rferl.org
farodevigo.es
news.com.au
diariovasco.com
corporatedispatch.com
dailyhunt.in
cope.es
nuevaalcarria.com
cadenaser.com
canalnueve.tv
ceutatv.com

diariodealicante.net	nbcnews.com	diariodeavila.es
diariodeteruel.es	metro.news	diariodecadiz.es
ecodiario.eleconomista.es	diariodenavarra.es	diariodeferrol.com
edition.cnn.com	canariasdiario.com	diariodeleon.es
edtimes.in	diariodealicante.net	diariogol.com
egyptianstreets.com	nhdiario.es	diarioinformacion.com
elcierredigital.com	canariasdiario.com	diariopalmero.es
elcorreodeespana.com	elsaltodiario.com	diariosur.es
eldigital.com.do	eldiariodecanarias.com	listindiario.com
elforodeceuta.es	blogs.diariovasco.com	nuevodiario.es
europost.eu	albacetediario.es	mallorcadiario.com
intereconomia.com	eldiarioalerta.com	
buzzfeednews.com	diariocordoba.com	
catalannews.com	diariocritico.com	

```
df_final_SEL=df_final_g1[(df_final_g1['ALERT']==1) & (df_final_g1['filter']==1)]
display(df_final_SEL.head())
df_final_SEL.shape
```

	screen_name	tweet_id	tweet_date	img_date	created_at	text	lang	retweets	favs	
4	20m	1296473210553238016	20/08/2020	18/08/2020	2020-08-20 15:44:35	#Directo | Fernando Simón: "Que nadie se con...	es	37	29	http://pbs.twimg.com/media
5	20m	1296473210553238016	20/08/2020	17/08/2020	2020-08-20 15:44:35	#Directo | Fernando Simón: "Que nadie se con...	es	37	29	http://pbs.twimg.com/media
6	20m	1296473210553238016	20/08/2020	17/08/2020	2020-08-20 15:44:35	#Directo | Fernando Simón: "Que nadie se con...	es	37	29	http://pbs.twimg.com/media
7	20m	1296473210553238016	20/08/2020	18/08/2020	2020-08-20 15:44:35	#Directo | Fernando Simón: "Que nadie se con...	es	37	29	http://pbs.twimg.com/media
11	20m	1296093646949961984	19/08/2020	04/08/2020	2020-08-19 14:36:20	#ÚltimaHora | 'La isla de las tentaciones' p...	es	11	12	http://pbs.twimg.com/media/

5 rows × 54 columns

```
(1026, 54)
```

El resultado final es una tabla reducida, “tabla_resultados” como se muestra a continuación en una captura de Excel.

screen_name	tweet_id	text	retweets	favs	link	image_url	weet_dat	img_date	delta_t	polarity	ubjectivit	ccount_cred
eldiarioes	1,29E+18	DIRECTO | Pedro Sánchez	402	1600	https://ca	http://pbs	29/07/202	22/07/201	373	0	0	0,8958167
eldiarioes	1,29E+18	DIRECTO | Pedro Sánchez	402	1600	https://w	http://pbs	29/07/202	04/01/202	207	0	0	0,8958167
eldiarioes	1,29E+18	DIRECTO | Pedro Sánchez	402	1600	https://ca	http://pbs	29/07/202	25/07/201	370	0	0	0,8958167
eldiarioes	1,29E+18	DIRECTO | Abascal anunc	57	149	https://w	http://pbs	29/07/202	04/01/202	207	0	0,2	0,8958167
eldiarioes	1,29E+18	DIRECTO | Abascal anunc	57	149	https://w	http://pbs	29/07/202	08/01/202	203	0	0,2	0,8958167
eldiarioes	1,29E+18	DIRECTO | Abascal anunc	57	149	https://m	http://pbs	29/07/202	03/01/202	208	0	0,2	0,8958167
eldiarioes	1,29E+18	DIRECTO | Abascal anunc	57	149	https://w	http://pbs	29/07/202	07/01/202	204	0	0,2	0,8958167
eldiarioes	1,29E+18	DIRECTO | Pedro Sánchez	41	133	https://w	http://pbs	29/07/202	20/05/202	70	0,25	0,5	0,8958167
eldiarioes	1,29E+18	DIRECTO | Fernando Sim	16	21	https://w	http://pbs	27/07/202	11/03/202	138	-0,01667	0,5	0,8958167
eldiarioes	1,29E+18	ÚLTIMA HORA | El juez reab	58	95	https://w	http://pbs	27/07/202	15/03/202	134	-0,05	0,083333	0,8958167
eldiarioes	1,29E+18	ÚLTIMA HORA | El juez reab	58	95	https://w	http://pbs	27/07/202	03/07/202	24	-0,05	0,083333	0,8958167
eldiarioes	1,29E+18	1 PSOE: 32,4% 2 PP: 19,4% 3 U	55	164	https://w	http://pbs	27/07/202	22/10/201	279	0	0	0,8958167
eldiarioes	1,29E+18	150 años de postales: ¿m	11	33	https://w	http://pbs	26/07/202	09/07/202	17	0	0	0,8958167
eldiarioes	1,29E+18	150 años de postales: ¿m	11	33	https://w	http://pbs	26/07/202	19/06/202	37	0	0	0,8958167
eldiarioes	1,29E+18	150 años de postales: ¿m	11	33	https://w	http://pbs	26/07/202	08/07/202	18	0	0	0,8958167
elespanolcom	1,3E+18	Fallece el primer médico co	13	6	https://w	http://pbs	18/08/202	10/06/202	69	0,25	0	0,8171443
elespanolcom	1,3E+18	Fallece el primer médico co	13	6	https://w	http://pbs	18/08/202	15/06/202	64	0,25	0	0,8171443
elespanolcom	1,3E+18	La herencia #ElZarpazo de @	11	18	https://w	http://pbs	17/08/202	12/08/202	5	0	0	0,8171443
elespanolcom	1,3E+18	La herencia #ElZarpazo de @	11	18	https://w	http://pbs	17/08/202	08/07/202	40	0	0	0,8171443
elespanolcom	1,3E+18	La herencia #ElZarpazo de @	11	18	https://w	http://pbs	17/08/202	07/06/202	71	0	0	0,8171443
elespanolcom	1,29E+18	El PP vuelve a exigir al Gobi	13	17	https://w	http://pbs	15/08/202	22/05/202	85	0	0	0,8171443
elespanolcom	1,29E+18	El PP vuelve a exigir al Gobi	13	17	https://w	http://pbs	15/08/202	22/07/202	24	0	0	0,8171443
elespanolcom	1,29E+18	El nuevo final de Casablanca	10	16	https://w	http://pbs	15/08/202	02/08/201	379	0,136364	0,454545	0,8171443
elespanolcom	1,29E+18	Simón niega el riesgo de col	13	9	https://w	http://pbs	14/08/202	31/07/202	14	0	0	0,8171443
elespanolcom	1,29E+18	Mil toneladas de petróleo s	12	12	https://w	http://pbs	11/08/202	21/11/201	994	0	0	0,8171443

Esta tabla contiene todos los resultados de imágenes encontradas con fecha extraída, la fecha del tweet, diferencia en días (delta_t), el link de la imagen, el texto del tweet, polaridad y subjetividad del texto y credibilidad de la cuenta.

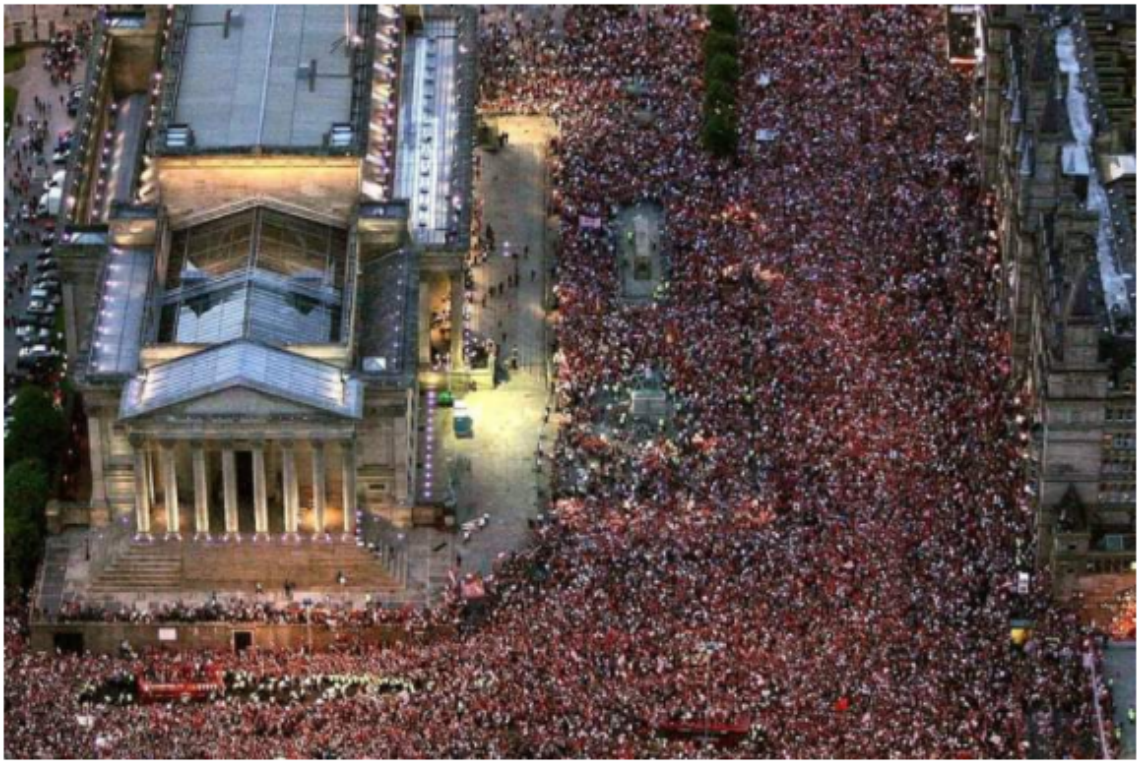
CASO 2. UN TWEET INDIVIDUAL.

CASO 2.A

La herramienta también puede analizar un tweet individual a partir de su ID con el script check_tweet_pic. Un ejemplo de uso sobre un tweet con id 1299775142827040768 se muestra debajo, un caso conocido de bulo[23]

[23] https://maldita.es/malditobulo/2020/08/31/fotos-manifestacion-mascarillas-coronavirus-berlin/)

```
Introduce el id del tweet a analizar: 1299775142827040768
100% [.....................................................................................] 90740 /
90740
```



La imágen más antigua encontrada se publicó el 11/06/2018 en https://www.standard.co.uk/news/uk/tommy-robinson-activists-claim-photo-of-liverpool-champions-league-victory-parade-actually-shows-a3859901.html , 810 días antes de la publicación del tweet.

Fake score: 2.4557216886357542

El parámetro "Fake score" es un intento de cuantificar el grado de falsedad del tweet y se calcula como:

$$\frac{subjetividad}{credibilidad^2} \; si \; polaridad > 0$$

$$\frac{subjetividad+|polaridad|}{credibilidad^2} \; si \; polaridad \; < 0$$

CASO 2.B

Analizamos el tweet visto en la página 10 como ejemplo de fotografía fuera de contexto. La herrmienta nos da el resultado esperado como se puede ver en la ejecución de *check_tweet_pic* en Jupyter Notebook abajo:

```
Introduce el id del tweet a analizar: 1295417329673818112
100% [.....................................................................] 161054 / 161054
```


```
La imágen más antigua encontrada se publicó el 24/08/2016 en  https://www.vanitatis.elconfidencial.com/multimedia/album/noticias/2016-08-23/pedro-sanchez-disfruta-de-sus-vacaciones-familiares-en-ibiza_1249855/ , 1454 días antes de la publicación del tweet.

Fake score: 0.21384006370561637
```

etweets	favs	link	image_url	tweet_date	img_date	delta_t	polarity	subjectivity	account_cred	filte
0	1	https://www.vanitatis.elconfidencial.com/notic...	http://pbs.twimg.com/media/Efo-q3ZXoAAIUn4.jpg	17/08/2020	25/08/2016	1453	0.066667	0.133333	0.789632	
0	1	https://www.vanitatis.elconfidencial.com/multi...	http://pbs.twimg.com/media/Efo-q3ZXoAAIUn4.jpg	17/08/2020	24/08/2016	1454	0.066667	0.133333	0.789632	

VERIFICACIÓN DE RESULTADOS

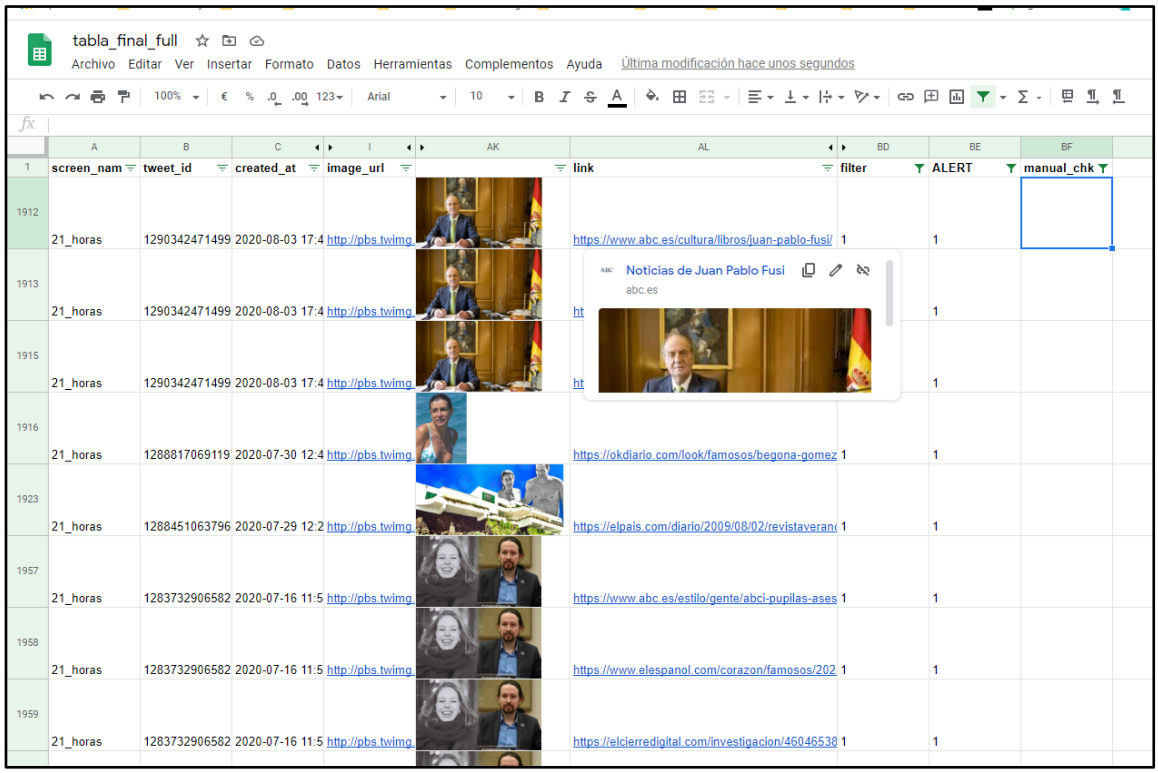
La tabla de resultados obtenida contiene toda la información necesaria para que el analista pueda determinar si el contexto de una fotografía en Twitter es correcto o no. Sin embargo, los resultados contienen muchos falsos positivos y los enlaces encontrados no

contienen la foto buscada. Esto ocurre por ejemplo con todas las fotografías que son diagramas o capturas de pantalla de Twitter. En ese aspecto tiene una debilidad. Para poder determinar los puntos débiles del sistema, se ha verificado manualmente los resultados de la tabla anterior de 750 registros de los 1026. Para hacerlo se ha añadido una nueva columna "manual_chk" (chequeo manual) y se ha usado la plataforma de Google Excel online, ya que tiene dos funciones que facilitan mucho el proceso, que son HYPERLINK() que hace de una celda con una URL que sea un hiper-enlace, y IMAGE() que muestra una miniatura de la imagen en la celda. Como se puede ver en la captura de abajo, las miniaturas corresponden a la columna de la imagen del tweet (img_url) y al lado la columna "link" , web donde se ha encontrado supuestamente la imagen. Al pasar el ratón sobre la url se puede ver la imágen y de esa forma se compara con la miniatura de la celda contigua. En otras ocasiones la imagen no aparece y fue necesario entrar en el enlace y comprobarlo. Este método manual se ha seguido para etiquetar cada registro según la fotografía efectivamente estaba en el link (manual_chk=1) o no lo estaba y era un falso positivo (manual_chk=0).

De los 750 registros analizados a mano, 449 son verdaderos positivos y 301 falsos positivos:

```
1  df_final_SEL[df_final_SEL['manual_chk']!=''].groupby('manual_chk')['manual_chk'].count()
```

```
manual_chk
0.0    301
1.0    449
Name: manual_chk, dtype: int64
```



Esto sirvió para eliminar determinar que, por ejemplo, los resultados de ciertos dominios son casi siempre falsos positivos, por ejemplo, slideshare. Youtube es otro ejemplo y por eso se descarta en el filtro de dominios.

El paso siguiente fue tomar la tabla_ML con la etiqueta "manual_chk" y aplicarle clasificadores de Machine Learning para tratar de predecir si un resultado a será correcto (1) o incorrecto (0) a partir de las siguientes características contenidas en la tabla, que son:

''screen_name', 'tweet_id', 'retweets', 'RT', 'favs', 'i', 'img_domain',

'ext', 'lang', 'total_length', 'capitals', 'caps_vs_length',
'num_words', 'num_unique_words', 'words_vs_unique',
'num_exclamation_marks', 'num_question_marks', 'num_question_marks_es',
'question_marks_ratio', 'num_punctuation', 'num_symbols', 'num_smilies',
'num_sad', 'mean_word_len', 'polarity', 'subjectivity', 'similarity',
'account_cred', 'delta_t', 'link_dom_sym', 'link_digits', 'link_Dots',
'link_url_syms', 'link_dom_digits', 'link_delimiter_path',
'link_dom_dots', 'link_length', 'link_NumberRate', 'ALERT', 'filter',

Se han usado dos modelos de clasificación, Random Forests y Gradiente Estocástico[24], quedándonos con la media de ambos. Se ha elegido ambos algoritmos tras obtener puntuaciones en F1-score de 80.0% y 80.7% respectivamente en el conjunto de datos de pruebas.

Entrenamiento con clasificadores Random Forests y Gradiente Estocástico y almacenamiento en disco (serialización de los modelos)

```
df_cred=pd.read_csv('Base/csv/tabla_credibilidad_final.csv')
df = pd.read_excel('Base/csv/tabla_ML.xlsx')
df.set_index(['screen_name','tweet_id'],inplace=True)
df['lang'].fillna("",inplace=True)
X = df.drop('manual_chk',axis=1)
Y = df['manual_chk']

for col in ['img_domain' ,'ext','lang']:
    df[col] = LabelEncoder().fit_transform(df[col])
    X[col] = LabelEncoder().fit_transform(X[col])


#Divisiónd el conjunto de datos
X_train, X_test, Y_train, Y_test, df_train, df_test = train_test_split(X, Y, df, test_size=0.33)

rf_model = RandomForestClassifier(n_estimators=100,random_state=42,n_jobs=-1)
rf_model.fit(X_train, Y_train)

xgb_model=GradientBoostingClassifier()
xgb_model.fit(X_train,Y_train)

# save the model to disk
filename_rf = 'Base/RF_model_RevImgValidator.sav'
filename_xgb = 'Base/XGB_model_RevImgValidator.sav'
pickle.dump(rf_model, open(filename_rf, 'wb'))
pickle.dump(xgb_model, open(filename_xgb, 'wb'))
```

captura 1. Modelos entrenados almacenados en RF_model_RevImgValidator.sav y XGB_model_RevImgValidator.sav

Arriba se ve como, primero se entrenan los algoritmos de RandomForestClassifier() [RF] y GradientBoostingClassifier() [XGB] sobre los datos de la tabla_ML, con etiqueta manual_chk [0,1].

Usando el módulo pickle de Python se puede almacenar los dos modelos para usarlos a posteriori:

[24] https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a

```
x.set_index(['screen_name','tweet_id'],inplace=True)
for col in ['img_domain' ,'ext','lang']:
    x[col] = LabelEncoder().fit_transform(x[col])

rf_model = pickle.load(open(filename_rf, 'rb'))
xgb_model = pickle.load(open(filename_xgb, 'rb'))
y_pred_rf=rf_model.predict(x)
y_pred_xgb=xgb_model.predict(x)
y_pred=np.mean([y_pred_rf[0],y_pred_xgb[0]])
return y_pred
```

La predicción para un registro de la tabla ML sin etiquetar será la media de los clasificadores RF y XGB. Podemos tratar de predecir el acierto de MRISA con la siguiente función, que lee la tabla que teníamos tabla_ML.xlsx y le aplica dos modelos de clasificación entrenados.

```
import pandas as pd
from openpyxl import load_workbook
import pickle
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split


filename_rf = 'RF_model_RevImgValidator.sav'
filename_xgb = 'XGB_model_RevImgValidator.sav'

file = str(input('Introduce el nombre de la tabla_ML de Excel: '))

def RevImgValidator(file):
    x=pd.read_excel(file)
    x.set_index(['screen_name','tweet_id'],inplace=True)
    for col in ['img_domain' ,'ext','lang']:
        x[col] = LabelEncoder().fit_transform(x[col])
    display(x)

    rf_model = pickle.load(open(filename_rf, 'rb'))
    xgb_model = pickle.load(open(filename_xgb, 'rb'))
    y_pred_rf=rf_model.predict(x)
    y_pred_xgb=xgb_model.predict(x)
    y_pred=(y_pred_rf+y_pred_xgb)/2
    print('Predicción:',y_pred)


def main():
    RevImgValidator(file)

# Execute main() function
if __name__ == '__main__':
    main()
```

Esta funcionalidad no se ha implantado en la herramienta de análisis pero muestra que las tablas generadas se pueden utilizar para casos más amplios que el objetivo inicial.

CONCLUSIONES Y PROPUESTAS

Como conclusión sobre el trabajo realizado se puede afirmar que la herramienta desarrollada cumple el objetivo principal de la búsqueda de tweets con imágenes con fechas anteriores a la publicación. Además, buscando otras funcionalidades futuras el sistema trata de ser lo más general posible para adaptarse a otros usos futuros. A diferencia de otras propuestas, la herramienta creada usa información de usuario, del tweet y la imagen. El método de obtención de fechas, con MRISA tampoco es el habitual, siendo más común el uso de otras APIs que puedan contener la fecha en uno de los campos.

El sistema propuesto presenta varias formas de mejora y posibles correcciones, como pueden ser:

- Añadir otro sistema auxiliar de búsqueda de imágenes, como...

Motor de búsqueda	URL	API
Berify	https://berify.com	si
CamFind	https://camfindapp.com	si
Flickr	https://secure.flickr.com	si
Google Images	https://images.google.com	si
TinyEye	https://tineye.com	si

Durante gran parte del proyecto se usó la API (suscripción de pago mensual) de serpapi de Google SearchImages, pero un error en un script a la hora de analizar un grupo muy elevado de tweets provocó que se alcanzará el límite de 5000 consultas y no se pudo volver a usar.

- Se podría usar en MRISA la API de Google Vision en lugar del metabuscador.
- Se ganaría mucho tiempo si se pudiera calcular el hash de una imagen a partir de su URL sin tener que descargarla previamente.
- Se podría incluir la ejecución de MRISA en los propios scripts, simplificando mucho los requerimientos

BIBLIOGRAFÍA

LIBROS Y ARTICULOS

Referencias

(s.f.).

Directorate-General for Communications Networks, Content and Technology. (2018). *A multi-dimensional approach to disinformation.* EU publications.

Garcia, M. A. (2018). *Fake News. La verdad de las noticias falsas.* Plataforma Editorial.

Kelly, S., Truong, M., Shahbaz, A., Earp, M., & White, J. (2017). *Freedom of the Net 2017: Manipulating Social Media to Undermine Democracy.* Washington: Free House. Obtenido de https://fredomhouse.org/report/freedom-net/freedom-net-2017

Krzysztof Lorek, J. S.-W.-L. (2015). Automated Credibility Assessment on Twitter. *Computer Science*, 156-168. doi:https://doi.org/10.7494/csci.2015.16.2.157

Negredo, S. (2020). Obtenido de Digital News Report España 2020: http://www.digitalnewsreport.es

Rico, M., & Berná, J. (2019). Análisis de la credibilidad de cuentas en Twitter. *Universidad de Alicante*.

Wardle, C. (enero de 2017). *Fake news. It's complicated*. Obtenido de https://firstdraftnews.org/latest/fake-news-complicated/

Otras fuentes consultadas

https://www.theguardian.com/media/2016/dec/02/fake-news-facebook-us-election-around-the-world

https://www.abc.net.au/news/2016-11-14/fake-news-would-have-influenced-us-election-experts-say/8024660

https://www.bellingcat.com/resources/2020/03/27/investigating-coronavirus-fakes-and-disinfo-here-are-some-tools-for-you/

https://www.ericsson.com/en/blog/2018/11/fake-news-on-social-media-whose-responsibility-is-it

https://www.forbes.com/sites/niallmccarthy/2018/06/14/where-exposure-to-fake-news-is-highest-infographic/ - 3e523b764a4d

https://www.lavanguardia.com/vida/20180210/44669387611/solo-14--de-espanoles-sabe-distinguir-un-fake-pero-60--cree-que-puede.html

https://www.eldiario.es/tecnologia/astroturfing-electoral-segmentacion-campana-peligrosamente_1_1631910.html

https://www.eldiario.es/tecnologia/aprobada-permitira-partidos-electoral-consentimiento_1_1829454.html

http://www.accionenred-andalucia.org/wp-content/uploads/2019/04/dosier-sobre-desinformaci%c3%b3n-fake-news-y-posverdad1.pdf

https://www.nytimes.com/2018/03/08/technology/twitter-fake-news-research.html

https://www.journalism.org/2016/05/26/news-use-across-social-media-platforms-2016/

https://coschedule.com/blog/how-often-to-post-on-social-media/

http://www.extradigital.es/quienes-son-los-periodistas-y-los-medios-de-comunicacion-mas-relevantes-en-twitter/

https://reverseimagesearch.net/

https://www.bellingcat.com/resources/how-tos/2019/12/26/guide-to-using-reverse-image-search-for-investigations/

CONTENIDO DEL FICHERO ADJUNTO

En el contenido del fichero que acompaña a la memoria podemos encontrar los siguientes recursos:

- Memoria del trabajo en los formatos PDF y DOCX.
- Código fuente de los scripts dentro del directorio scripts python.
- Código de los notebooks de Jupyter en el directorio notebooks
- Algoritmos de clasificación entrenados con tabla_ML
- Tablas de resultados del caso 1 y 2.B en formatos CSV y XLSX