



UNIVERSIDAD  
**COMPLUTENSE**  
MADRID

MÁSTER EN ESTADÍSTICAS OFICIALES E INDICADORES SOCIALES Y  
ECONÓMICOS

2018-2019

---

**Ánalysis de sentimiento**

---

*Autor:*

Raúl Fernández González

*Asignatura:*

Big data, Open data y Cloud data

21 de abril de 2019

# Índice general

<b>1. Resumen</b>	<b>2</b>
<b>2. Introducción teórica</b>	<b>3</b>
2.1. Metodología . . . . .	5
<b>3. Análisis de Twitter</b>	<b>7</b>
3.1. Extracción de datos . . . . .	7
3.1.1. Autentificación . . . . .	7
3.2. Limpieza de datos . . . . .	11
3.3. Análisis de la base de datos . . . . .	14
3.4. Análisis del contenido . . . . .	16
3.5. Análisis de sentimiento . . . . .	23
3.6. Análisis de emociones . . . . .	27
3.7. Clasificador mediante aprendizaje supervisado . . . . .	30
3.8. Análisis de los seguidores . . . . .	38
<b>4. Licencia del trabajo</b>	<b>40</b>
<b>5. Conclusiones</b>	<b>41</b>

# Capítulo 1

## Resumen

La expansión del uso de internet ha provocado en los últimos años un crecimiento exponencial de la información disponible. Cada día se generan millones de GigaBytes de nuevos datos susceptibles a ser analizados provenientes de distintas fuentes: redes sociales, bases de datos tradicionales, ciertos dispositivos conectados, etc. Este trabajo pretende elaborar un mecanismo de análisis de estos datos, usando como fuente la red social Twitter y centrando la atención en el sentimiento generado en el aspecto político. Para ello, mediante el uso de distintas librerías disponibles tanto en R como en python, se elabora una base de datos con los mensajes de los cinco líderes de los partidos políticos más influyentes de la actualidad. Usando algunos principios de análisis de redes sociales y modelos predictivos, se modifica la base de forma que seamos capaces, por un lado, de analizar el sentimiento que generan los mensajes de cada personaje y por el otro, poder hacer predicciones sobre nueva información.

El contenido del trabajo se distribuye como sigue. En primer lugar se hace una pequeña introducción teórica donde se incorporan el estado del arte, la metodología y algunos conceptos claves para el entendimiento posterior del proyecto. A continuación se muestra el proceso de extracción de los mensajes de la red social *twitter* así como un análisis de la base de datos que nos proporciona el servicio web. El grueso del trabajo se encuentra en el análisis de sentimientos y de emociones que nos lleva a los modelos predictivos finales donde se propone un algoritmo capaz de determinar la autoría de diferentes mensajes. Finalmente, se añaden unas breves conclusiones.

## Capítulo 2

### Introducción teórica

La ingesta cantidad de datos que ofrecen las plataformas digitales en la actualidad ponen en relevancia la necesidad de encontrar herramientas capaces de encontrar relaciones y conclusiones interesantes en los mismos. En esto, desde el surgimiento de la web 2.0, cobra relevancia la presencia de las redes sociales que dotan a los usuarios de la capacidad de generar contenido propio y compartirlo de manera pública con total facilidad. El análisis de las redes sociales consiste en la extracción, almacenamiento y posterior estudio de la enorme cantidad de información producida en las redes sociales.

Desde las empresas, aunque también desde el ámbito público, se otorga gran valor a las conclusiones extraídas de estos servicios. El hecho de que los usuarios puedan interactuar entre sí, manifestando sus propias opiniones, gustos, comentarios y creando debates permite a estos colectivos establecer estrategias que se adapten a las necesidades de la ciudadanía. Es innegable la ventaja competitiva que supone en el proceso de toma de decisión conocer la opinión de los consumidores. El marketing comercial puede aprovechar este análisis de datos para anticipar y planificar el comportamiento futuro del mercado, mejorar el servicio al cliente, entender su comportamiento e innovar. Hasta hace pocos años no existían herramientas capaces de obtener estos resultados. No obstante, en la actualidad los avances en informática ponen al alcance de cualquiera llegar a estos objetivos.

De todas las redes sociales existentes, Facebook se mantiene como la más exitosa con más de 2 mil millones de usuarios activos a nivel mundial. En términos nacionales, Facebook lidera también la clasificación junto con Whatsapp [1], afirmando su uso diario el 87 % de los usuarios de redes sociales. Bastante alejadas se encuentran tanto Instagram como Twitter, con un 48 % de la cuota. A pesar de estas cifras, en este trabajo se elige a Twitter como la fuente de nuestra información a analizar. Las razones son variadas y colocan a esta red social como la más adecuada

para la finalidad del proyecto. En primer lugar, se trata de una red social de microblogging, esto es, destinada a publicar información a través de breves comentarios en formato texto con un máximo de caracteres limitado. Esto claramente facilita el análisis posterior de su contenido. Además, se calcula que se envían 500 millones de *tweets* cada día, todos ellos de acceso público, con contenido relacionado con todos los sectores, desde el deporte, el entretenimiento hasta temas políticos. Frente a otros servicios, la principal razón de la publicación de estos mensajes es expresar el punto de vista y la opinión de cada usuario, lo que agranda su interés. No obstante, si en algo destaca Twitter es por su accesibilidad. Resulta extremadamente sencillo buscar y encontrar información sobre cualquier temática, convirtiéndose en un foco de rastreo noticias y eventos de actualidad. Estas razones colocan a Twitter como el mejor lugar para recolectar datos en tiempo real.

Para poder extraer conclusiones de los datos que circulan públicamente a través de la red se requiere de diversas herramientas que constituyen lo que se conoce como *Web Scraping*. En nuestro caso, la extracción de información útil a partir de los comentarios presentados en redes sociales es una técnica conocida como minería de opinión o análisis de sentimiento. Básicamente, se trata del área de estudio encargada de evaluar las actitudes, sentimientos y emociones que las personas muestran ante otros individuos, ciertos servicios o cualquier tema en concreto. Aunque llevado a la práctica resulta más sencillo de entender, es necesario saber que en el análisis de sentimiento de cualquier red social existen dos grandes metodologías.

En primer lugar, se tiene aquella basada en la orientación semántica. Fundamentalmente, se evalúa el sentimiento de un comentario en base a una clasificación previamente definida de las palabras. Así, estas son orientadas, por ejemplo, en positivas o negativas si se busca un análisis de polaridad, o en función de si se relacionan con un sentimiento de enfado, tristeza, alegría, etc. si la intención es un análisis de emociones. En función de lo que prime en cada mensaje se clasificará de una manera y otra.

Por otro lado, se tiene la clasificación basada en un aprendizaje automático. En este caso, se utiliza un corpus, o una colección de datos representativos, que permite enseñar a los clasificadores y clasificar los datos. Se necesitan por lo tanto dos conjuntos diferentes. Un conjunto de entrenamiento que ejercita el clasificador y uno de prueba que evalúe su eficacia. En el trabajo se presentan ambos modelos, con mayor o menor éxito, con la finalidad de lograr distintas herramientas que permitan el análisis de cualquier temática.

En cuanto a esto último, centramos el objetivo del trabajo en un análisis político de Twitter. En concreto, se busca conocer la repercusión y el sentimiento generado por los grandes líderes

políticos de nuestro país. La principal razón es la necesidad de compatibilizar este trabajo con otros proyectos académicos. A pesar de esto, y aunque la utilidad del estudio pueda ser menor a las necesidades de las empresas anteriormente mencionadas, desde un punto de vista investigador tiene un claro provecho. Conocer lo que la ciudadanía opina sobre determinadas medidas puede ayudar a determinar a qué candidato político votará en las próximas elecciones. Además, estudiar el comportamiento de los líderes políticos nos puede crear una idea de la estrategia seguida por cada uno y de los errores que pueda cometer. En cualquier caso, la metodología empleada es válida para el estudio de otras temáticas, con lo que se crea un buen mecanismo de análisis en Twitter.

## 2.1. Metodología

Terminar la introducción comentando la estructura posterior del proyecto que compone nuestra metodología. Como todo mecanismo de análisis de redes sociales, el proceso se compone de cuatro grandes fases. En primer lugar, se tiene la descarga o extracción de los datos, en nuestro caso, la obtención de los *tweets*. Este primer proceso se realiza mediante *R*. La razón es la posibilidad de utilizar paquetes de sencillo uso, especialmente diseñadas para este fin. Además, debido al tiempo que requiere el proceso de extracción de datos desde la red social, usar este programa estadístico nos permite ir analizando los resultados con otros *softwares* de manera simultánea a la obtención de nuestra base de datos. En segundo lugar, se procede a procesar el texto de los mensajes. En este caso, el programa a utilizar será *python*, por comodidad, aunque podría haber sido otro perfectamente. Existen paquetes y librerías propias tanto de *R* como de *python* preparadas no solo para el análisis estadístico sino también para el estudio semántico concreto que se lleva a cabo en este escrito.

Por resumir, aunque se irá comentando posteriormente las claves del código, el análisis se basa en la construcción de un *dataframe* que contenga tanto el texto del mensaje como otras variables que consideremos importantes. Es necesario antes de proceder al estudio semántico eliminar aquellos símbolos y palabras que carecen de significación. En consecuencia, a la vez que se obtiene la base de datos o el *dataframe* se requiere de filtrar el texto. Posteriormente, se presentan las diferentes técnicas de clasificación de palabras y de análisis de la base de datos. En cuanto a las primeras, la elegida será el uso de un diccionario donde un determinado número de palabras son clasificadas en función del sentimiento o emoción que generan. Finalmente, se lleva a cabo el análisis de sentimientos utilizando estas clasificaciones. Este análisis es complementado

con métodos predictivos para decidir la autoría de diferentes mensajes.

# Capítulo 3

## Análisis de Twitter

### 3.1. Extracción de datos

Como se comentó previamente, el primer paso en el proceso de análisis de Twitter es obtener la información desde la red social. La facilidad con la que Twitter permite a los usuarios acceder a dicha información ayuda a la rapidez de este mecanismo. La extracción de los *tweets* se lleva a cabo mediante el uso de *R*. Este programa fue especialmente diseñado para el análisis estadístico de grandes volúmenes de datos y, aunque en la actualidad *python* es el lenguaje que domina el sector, existen librerías en *R* que facilitan las capacidades del análisis de texto. En nuestro caso estas serán dos: *twitterR*, que nos permite extraer los datos más relevantes bien de los *tweets* o bien de los diferentes usuarios, y *ROAuth*, relacionado con la autentificación en el proceso de descarga de mensajes que se explica a continuación.

#### 3.1.1. Autentificación

Como informa Twitter en su página web, para poder compartir información en la plataforma, se proporciona a las empresas, los desarrolladores y a cualquier usuario acceso a sus datos a través de sus API, abreviatura de *Application Programming Interfaces* o interfaces de programación de aplicaciones. Se trata de una especificación formal de cómo un software interacciona con otro. Más simplemente, podemos definir este concepto como la manera en que los programas informáticos se comunican entre sí para solicitar y compartir datos. Así, se permite que el software llame al punto de conexión, haciendo uso de parte de las funciones existentes en Twitter. La plataforma API de esta red social da acceso a la información que los usuarios han decidido compartir públicamente, no así a mensajes directos u otros contenidos privados.

Twitter ha variado las APIs que proporciona en sus nuevas versiones. Aunque en la actualidad,

se intenta simplificar la documentación en nuevos servicios en desarrollo. No obstante, históricamente Twitter ofrece tres tipos de APIs en función de nuestras necesidades. El *Streaming API* nos proporciona los *tweets* en tiempo real mediante una conexión permanente por usuario con los servidores de Twitter. Se tiene acceso a un alto volumen de mensajes con baja latencia. Los resultados se pueden filtrar por palabras clave o usando usuarios. Mayor interés tienen las dos siguientes opciones. El Rest API ofrece a desarrolladores acceso al core de los datos de Twitter. Así, se puede leer el perfil de los usuarios, distintos datos de los mismos como sus seguidores, los seguidos, mensajes enviados y demás información que podemos encontrar vía web. Por último, existe la Search API, que suministra los mensajes ajustados a la solicitud enviada.

Aunque en un principio no parezca complicado acceder a los datos, Twitter pone ciertas complicaciones. En primer lugar, la Search API únicamente ofrece información de los *tweets* enviados con un retraso en el tiempo de unos 9 – 10 días. Esto impide el acceso histórico a mensajes antiguos. Por otro lado, se mantiene una limitación en el tiempo que depende del método solicitado. Esta restricción impone esperas de cerca de 15 minutos entre consultas y afecta de forma diferente al tipo de búsqueda. Así, la búsqueda de mensajes en la Search API se restringe a 18.000 cada 15 minutos, mientras que la búsqueda de perfiles se limita a unos 75.000.

En cualquier caso, para acceder a cualquier API de Twitter se necesita de autentificarse usando *OAuth*, un protocolo que permite la autorización segura. Esto requiere no sólo disponer de una cuenta en esta red social sino también registrar una aplicación en Twitter. Twitter proporciona lo que denota por *Twitter App* para acceder a sus contenidos. Para conocer el procedimiento, se hace un resumen en lo que sigue. En primer lugar, se accede a la página <https://dev.twitter.com/apps> donde, una vez registrado con nuestra cuenta en la red social, se selecciona *Create new app*. Cumplimentados los campos requeridos, se accede a la pestaña *Keys and Access Tokens* y se pulsa en la opción *Generate My Access Token and Token Secret*. Twitter proporciona en estas opciones las claves de acceso y los tokens, término relacionado con la seguridad de autenticación. Básicamente, estos códigos nos dan la posibilidad de acceder a la aplicación y conseguir la información.

En este punto, utilizamos los paquetes de *R* previamente comentados. Instalando el paquete *twitterR* tenemos acceso a la función *setup\_twitter\_oauth()*, que nos permite autenticarnos en la aplicación de Twitter desde *R*.

Listing 3.1:

```
library("ROAuth");
library("twitteR");

consumer_key <- 'YOUR CONSUMER KEY'
```

1  
2  
3  
4

```

consumer_secret <- 'YOUR CONSUMER SECRET KEY'
access_token <- 'YOUR ACCESS TOKEN'
access_secret <- 'YOUR ACCESS SECRET TOKEN'

setup_twitter_oauth(consumer_key, consumer_secret,
access_token, access_secret)

```

5  
6  
7  
8  
9  
10

En este punto, podemos trabajar de diferentes maneras. Si se desea obtener los *tweets* filtrando por una palabra o *hashtag* determinado, se emplea la función *searchTwitter*. En este trabajo, simplemente se extraen el listado de Tweets de los líderes de los grandes partidos en el panorama político actual, esto es, el *timeline* de Albert Rivera, Pedro Sánchez, Pablo Iglesias, Pablo Casado y Santiago Abascal. Esto nos permite remontarnos en el tiempo hasta un máximo de aproximadamente 3000, que son los que permite la aplicación de Twitter recuperar. La función del paquete *twitterR* empleada es *get\_timeline*, donde se indica el número de mensajes que se desean recuperar. Se incluye el comando *retryonratelimit*, con el objetivo de que la aplicación continúe buscando *tweets* incluso cuando se alcance el límite permitido. Así, teniendo en cuenta que cada mensaje se encuentra cifrado con un número, *id*, que determina su fecha de publicación, usando el último valor devuelto es posible llamar de nuevo a la función anterior indicando el máximo *id* que deseamos recuperar.

Listing 3.2:

```

#Obtenemos los tweets para cada persona
rivera=get_timelines('Albert_Rivera',n = 3200,
retryonratelimit =TRUE)
#calculamos los primeros tweets que nos deja la aplicacion
id=c(0,tail(rivera, 1)[ "status_id"] %>% pull())
#obtenemos el ultimo valor del id del mensaje
while( id[1]!= id[2]){ #comparamos el ultimo valor del id guardado
  con el ultimo devuelto por la aplicacion
  id[1]= id[2]
  rivera2=get_timelines('Albert_Rivera',n = 3200,max_id = id[2],
  retryonratelimit =TRUE) #obtenemos los siguientes mensajes desde
  el ultimo devuelto
  id[2]=tail(rivera2, 1)[ "status_id"] %>% pull() #guardamos el id
  del ultimo mensaje
  rivera <- bind_rows(rivera, rivera2)
}

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

Este proceso se repite para las diferentes personalidades y juntamos los resultados en un único data frame que se exporta en formato *csv*. Este conjunto de datos contiene para cada político distintas variables, hasta un total de 88, de las cuales únicamente nos interesaremos por 13: el autor del *tweet*, la fecha (formato año-mes-día-hora), el id del mensaje, el propio texto, el número total de favoritos, si es o no un *retweet*, el recuento de *retweets*, la fuente con la que se escribe, los *hashtags* que presenta, las menciones, el autor y la localización en caso de no ser un mensaje propio y las coordenadas desde donde se escribe. Por último, se transforma la fecha, obviando el valor de la hora en la que se escriben.

Listing 3.3:

```

tweets <- bind_rows(rivera, iglesias, casado, sanchez, abascal)
#Nos quedamos con las variables interesantes
tweetsold<-tweets
write_as_csv(tweetsold, "..../tweetsold.csv")
tweets <- tweetsold[c("screen_name","created_at","status_id",
"text", "favorite_count", "is_retweet","retweet_count","source",
"hashtags","mentions_screen_name","retweet_screen_name",
"retweet_location","geo_coords")]
tweets<-rename(tweets,replace=c("screen_name"="autor","created_at"=
"fecha","text"="texto"))
write_as_csv(tweets,
"..../tweets.csv")

#Representamos el numero de tweets por mes
tweets_mes_dia <- mutate(tweets,mes_dia =
  format(fecha, "%Y-%m-%d"))
write_as_csv(tweets_mes_dia,
"..../tweets.csv")

```

El resumen del contenido del número de mensajes para cada personaje se expone en la Tabla 3.1. El número en todo caso son superiores a los 3000 mensajes, siendo en algunos casos más significativos. Por otro lado, en la Figura 3.1 se muestra un ejemplo de la salida de la base de datos

	Rivera	Iglesias	Casado	Sánchez	Abascal
Tweets recopilados	3.199	3.193	3.248	3.239	3.201
Porcentaje del total	5,8	15,1	30,2	12,9	12,8

Tabla 3.1: Distribución del número de *tweets* recopilado para cada autor y porcentaje del total de mensajes publicado .

	autor	fecha	status_id	texto	...	retweet_screen_name	retweet_location	geo_coords	mes_dia
0	Albert_Rivera	2019-03-19 21:21:07	xx11088116211974447105	Quiero que en España cumplir con tu obligación...	...	BalEdmundo	NaN	NA NA	2019-03
1	Albert_Rivera	2019-03-19 21:05:13	xx11088112309744461827	Hoy @Albert_Rivera se ha acercado para interesar...	...	MGutierrezCs	Madrid, Comunidad de Madrid	NA NA	2019-03
2	Albert_Rivera	2019-03-19 20:34:03	xx11088104367947829249	Los liberales del s. XXI tenemos que luchar pa...	...	NaN	NaN	NA NA	2019-03
3	Albert_Rivera	2019-03-19 18:26:34	xx1108872295644520450	🔴 #EnDirecto Encuentro Naranja con @Albert_Riv...	...	CiudadanosCs	España	NA NA	2019-03
4	Albert_Rivera	2019-03-19 14:58:47	xx110881798379851940	Las personas con discapacidad son héroes cada...	...	NaN	NaN	NA NA	2019-03
5	Albert_Rivera	2019-03-19 12:48:23	xx11087985164339236864	Soy afortunado por tener un padre al quequier...	...	NaN	NaN	NA NA	2019-03
6	Albert_Rivera	2019-03-19 10:08:47	xx11087946808745512961	Los totalitarios separatistas vuelven a actuar...	...	NaN	NaN	NA NA	2019-03
7	Albert_Rivera	2019-03-18 20:11:07	xx11087736208912859136	Le he propuesto por carta al señor @abulosmeco...	...	CiudadanoVille	NaN	NA NA	2019-03
8	Albert_Rivera	2019-03-18 17:32:41	xx11087695363294694801	Es muy grave que el gobierno Sánchez le pidier...	...	NaN	NaN	NA NA	2019-03
9	Albert_Rivera	2019-03-18 14:31:46	xx11087650886986527744	Cada año me gusta más vibrar con la mascletá e...	...	NaN	NaN	NA NA	2019-03
10	Albert_Rivera	2019-03-17 17:57:37	xx11087340226110869580	Un honor y un placer que Ciudadanos haya pensa...	...	BalEdmundo	NaN	NA NA	2019-03
11	Albert_Rivera	2019-03-17 15:58:27	xx11087310232957521922	Edmundo Bal llevó la acusación contra la trama...	...	JorgeBustos1	NaN	NA NA	2019-03
12	Albert_Rivera	2019-03-17 15:55:53	xx110873095875589978017	España ha batido el récord de asistencia a un ...	...	PatriciaReyesCs	NaN	NA NA	2019-03
13	Albert_Rivera	2019-03-17 13:18:42	xx11087270033070350336	Escuchen a Edmundo Bal (@BalEdmundo). Un hombr...	...	NaN	NaN	NA NA	2019-03
14	Albert_Rivera	2019-03-17 12:23:45	xx11087256205309898560	Contar en mi equipo con Edmundo Bal es uno de ...	...	NaN	NaN	NA NA	2019-03
15	Albert_Rivera	2019-03-17 11:07:12	xx11087236939734941698	🔴 #EnDirecto Encuentro Ciudadano con @Albert_R...	...	CiudadanosCs	España	NA NA	2019-03
16	Albert_Rivera	2019-03-17 09:30:27	xx11087212591292694533	El de hoy es para mí uno de los días más espec...	...	NaN	NaN	NA NA	2019-03
17	Albert_Rivera	2019-03-16 17:12:25	xx11086964645933468160	España es hoy un país libre y unido gracias a ...	...	NaN	NaN	NA NA	2019-03
18	Albert_Rivera	2019-03-16 16:37:54	xx11086957776473985165	Marcos de Quinto, exvicepresidente de Coca-Col...	...	europapress	NaN	NA NA	2019-03
19	Albert_Rivera	2019-03-16 13:19:26	xx11086987829145923584	Estoy muy orgulloso de contar en mi equipo con...	...	NaN	NaN	NA NA	2019-03
20	Albert_Rivera	2019-03-16 11:12:40	xx11086957929442349057	🔴 #EnDirecto Encuentro Ciudadano con @Albert_R...	...	CiudadanosCs	España	NA NA	2019-03

Figura 3.1: Ejemplo de la salida de la base de datos guardada.

Listing 3.4:

```

tweets = pd.read_csv('..../Tweets.csv')

print( len(tweets[tweets['autor']=='Albert_Rivera']))

```

### 3.2. Limpieza de datos

Lo primero que se observa en la Figura 3.1 es la presencia de distintos caracteres dentro de la variable *texto* que no son útiles para nuestro estudio. En este apartado del trabajo se procede a eliminar todo elemento incluido en los mensajes que no aporta información relevante sobre su contenido. Así, los signos de puntuación, la presencia de abreviaturas, los patrones no informativos (como por ejemplo enlaces a páginas web o a contenido multimedia), los números u otros caracteres sueltos que podemos encontrar no tendrán relevancia en el estudio posterior y por lo tanto deberán ser eliminados.

Realmente no existe una forma predeterminada de hacerlo pues depende de la finalidad que tenga el análisis. No obstante, sí existen caracteres innecesarios en cualquier estudio que serán los que nosotros borraremos. Habitualmente, esto se consigue procesando el texto de entrada mediante una técnica que se conoce como *tokenización*. Se conoce por *token* a las unidades mínimas de información. En nuestros términos, se corresponden con las cadenas de caracteres entre espacios en blanco o puntuación junto con las abreviaturas. Tanto en *R* como *python* existen librerías que automatizan este procedimiento. Sin embargo, y dado que no es excesivamente complicado, resulta más transparente implementarlo con una función. Además, en todos estos procesos el hecho de que el idioma de los textos sea el español nos va a dificultar el uso de librerías predeterminadas.

Así, se dividen los mensajes en sus palabras individuales y resto de símbolos y se filtran en primer lugar los patrones que se explican a continuación:

- Links a páginas web. Se trata de caracteres que mantienen en común el componente inicial *http*. Estos enlaces no nos proporcionan información al posterior análisis de sentimiento y, de hecho, la presencia de estos caracteres pueden provocar confusiones posteriores.
- Signos de puntuación. Las preguntas, las exclamaciones, los puntos son caracteres que enfatizan el discurso y condicionan el contenido del mensaje. No obstante, los mecanismos de análisis no son capaces de sobreentender estos enfoques.
- Números. Aunque en algunos estudios pueda ser interesante mantener estos caracteres, lo cierto es que en un análisis de sentimiento poca información se puede derivar del apartado numérico. Es por ello que nos resulta más sencillo prescindir de ellos.
- Palabras de una sola letra. La ausencia de contenido en estas palabras las convierten en desecharables.

- *hashtags* y nombre de usuarios. Los *hashtags* son conjuntos de palabras que aportan una información muy relevante para nuestro estudio. Sin embargo, el símbolo # con el que se estructuran puede crear problemas futuros. Además, el hecho de que se mantengan los *hashtags* en la propia base de datos, nos lleva a eliminar estos símbolos pero mantener su contenido. Por otro lado, los usuarios a los que se menciona en los textos también revelan importantes conclusiones pero de nuevo el símbolo @ es un elemento no gramatical que da problemas a la hora de clasificar sentimentalmente el texto.
- Cualquier otro símbolo sin significación.
- Espacios entre palabras innecesarios.

Comentar además que antes de procesar el texto de acuerdo a los preceptos anteriores se estandariza el mismo transformando todas las palabras a minúsculas. En la Figura 3.2 se muestra un ejemplo de cómo actúa la función *tokenizar* sobre un texto de prueba.

Listing 3.5:

```
#Proceso de Tokenizacion
RE_EMOJI = re.compile(r'[\U00010000-\U0010ffff]', flags=re.UNICODE)
def strip_emoji(texto):
    return RE_EMOJI.sub(r'', texto)
def tokenizar(texto):
    texto=texto.lower()
    #eliminamos web
    texto=re.sub(r'ht\w+://\w+\.\w+', "", texto)
    #eliminamos signos puntuacion
    texto=re.sub('[%$]', re.escape(string.punctuation), ' ', texto)
    #eliminamos numeros
    texto=re.sub(r'\d+', ' ', texto)
    #eliminamos palabras de una letra
    texto=re.sub(r'\b\w\b', ' ', texto)
    #eliminamos otras cosas
    texto=re.sub(r'[@#$&]', ' ', texto)
    #eliminamos emojis
    texto=RE_EMOJI.sub(r'', texto)
    #eliminamos espacios
    texto=re.sub(' +', ' ', texto)
    return texto
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21

El texto de prueba es: Esto es un ejemplo 34 del PROCESO3 de Tokenizacion ? para @Albert\_Rivera & #https://ciudadanos.com #cs  
 El texto tokenizado es: esto es un ejemplo \_del proceso de tokenizacion para albert rivera cs

Figura 3.2: Ejemplo de la salida de la función *tokenizar* con un texto de prueba.

El texto original de los *tweets* es sustituido por el texto filtrado con la función anterior. Si recordamos el objetivo del proyecto, poder hacer un análisis de sentimiento de cada líder político, existen ciertas palabras, como son los artículos, las preposiciones, los pronombres y otras palabras, frecuentes en nuestro lenguaje habitual pero que no aportan nada interesante. Estas palabras se conocen como *stopwords* y de nuevo python dispone de herramientas capaces de eliminarlas.

No obstante, funcionan mejor en inglés que en español, pero ante el enorme trabajo de crear una lista propia con palabras en castellano de uso común e innecesarias, es preferible utilizar la liberaría *nltk* que ofrece el programa. Este nuevo texto filtrado se guarda en una nueva variable, *texto \_ filtrado*, para no perder la información del anterior.

Listing 3.6:

```

stop_words=stopwords.words('spanish')
stop2=['va','asi','tras','hacer','ser','hecho','acto','hora','apoyo','dia','junto','toda','vez','hoy']
for i in range( len(stop2)):
    stop_words.append(stop2[i])
wordcloud(tweets,'texto_filtrado')

```

1  
2  
3  
4  
5

El texto de prueba es: Esto es un ejemplo 34 del PROCESO3 de Tokenizacion ? para @Albert\_Rivera & #https://ciudadanos.com #cs  
 El texto tokenizado es: esto es un ejemplo del proceso de tokenizacion para albert rivera cs  
 El texto tokenizado y sin stopwords es: ejemplo proceso tokenizacion albert rivera cs

Figura 3.3: Ejemplo de la salida de la función *tokenizar* y de la función *stopword* con un texto de prueba.

Un filtrado en mayor profundidad requeriría eliminar los *tokens* que hacen referencia al mismo concepto y que se diferencian por su variante morfológica. Así, conjugaciones verbales como "leo", "leí", "leemos", etc. mantienen el mismo significado y son derivados de un mismo verbo. Este proceso de eliminación de su raíz morfológica es lo que se conoce por *Stemming*. De nuevo, se trata de un procedimiento plausible en el caso de textos en inglés pero de extrema complejidad para mensajes en español. Por ello, se descarta hacer esta opción.

### 3.3. Análisis de la base de datos

En este punto, cuando tenemos el texto de los mensajes filtrado como deseamos, podemos hacer un primer análisis de su contenido. No obstante, primeramente visualizamos nuestra base de datos. A pesar de que en la Tabla 3.1 habíamos comprobado que el número de *tweets* se reparte de manera equitativa entre los 5 políticos, la Figura 3.4 muestra que su distribución en el tiempo es bastante desigual. Se añaden acontecimientos políticos importantes que tuvieron lugar en el periodo estudiado con el fin de explicar los días donde se alcanzan máximos. Así, por ejemplo Pablo Casado alcanza su mayor actividad durante el proceso de elección del líder de su partido. De la misma manera, Santiago Abascal concentra una gran cantidad de mensajes durante la campaña electoral de las elecciones andaluzas. Estos dos autores son los que mantienen una mayor actividad reciente, dado que acumulan los más de 3.000 mensajes en poco más de 8 meses. Esto supone una media de entre 15 y 25 *tweets* cada día. En el lado opuesto, se sitúa Pedro Sánchez, que reparte el mismo número en año y medio, siendo su promedio inferior a los 10 mensajes al día. En cuanto al resto, Pablo Iglesias se mostró más activo en los días previos al 12 de Octubre o durante la famosa sentencia de la manada. Durante los meses de Verano, nuestra base de datos no tiene registros de actividad de este autor, tiempo durante el que se tuvo que hacer cargo de sus hijos primerizos. Por último, Albert Rivera es quien menos variabilidad presenta, siendo eclipsado en todo momento por sus adversarios.

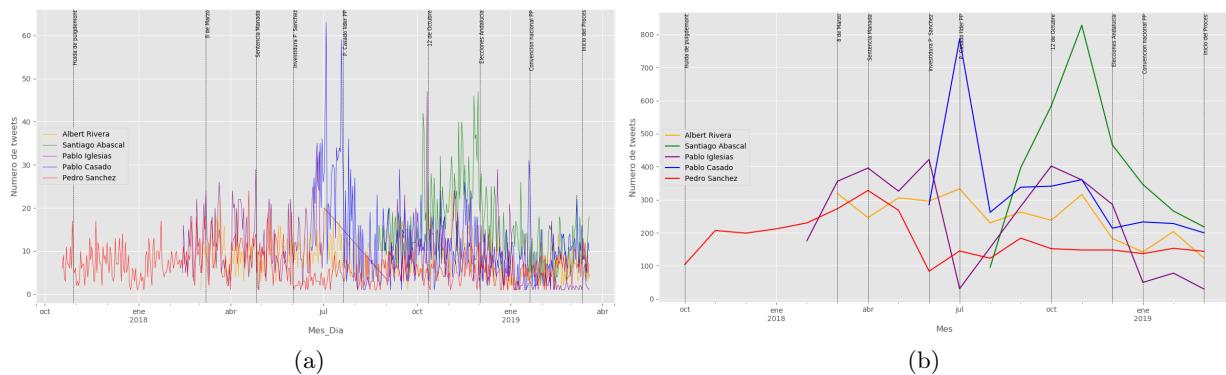


Figura 3.4: Número total de *tweets* publicados por cada autor en función del: (a): día y del mes, (b): del mes.

De la base de datos se pueden extraer otras conclusiones curiosas, como por ejemplo las horas del día donde es más frecuente la publicación de mensajes, ver Figura 3.5. Aunque solo sea por entretenimiento, se muestra que la hora predilecta por los políticos se encuentra entorno a las 12 del mediodía, siendo Albert Rivera el más madrugador y Santiago Abascal el más trasnochador.

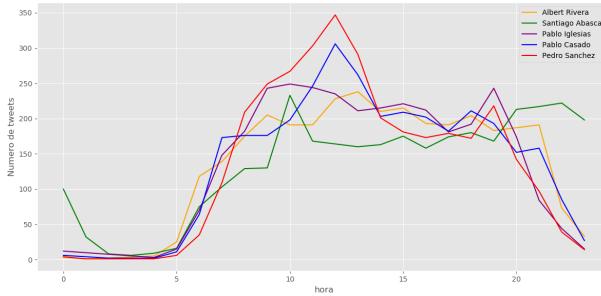


Figura 3.5: Número total de *tweets* publicados por cada autor en función de la hora.

Además, es posible hacernos una idea de la capacidad de convicción o la penetración que tiene cada político dentro de la red social. Así, en la Figura 3.6 se muestra el número total de *retweets* que logra cada político en los diferentes días del año. Se trata de una aproximación a la relevancia que tiene la información publicada. Por ejemplo, durante el conflicto de Venezuela, son Pablo Iglesias y Casado quienes lanzan mensajes con mayor atracción. De la misma forma, en el caso del barco *Aquarius*, Pedro Sánchez fue capaz de atraer a su posición a un gran número de usuarios. En el 12 de Octubre, es Rivera quién logra una mayor penetración y durante las negociaciones en Andalucía, Vox muestra un mayor apoyo entre sus seguidores. El diagrama de cajas contiguo, muestra que, efectivamente, es este último partido quien mantiene de media una mayor cantidad de *retweets*, seguido de Rivera y alejado de Casado, que se sitúa como el político que menos rentabilidad saca de esta red social.

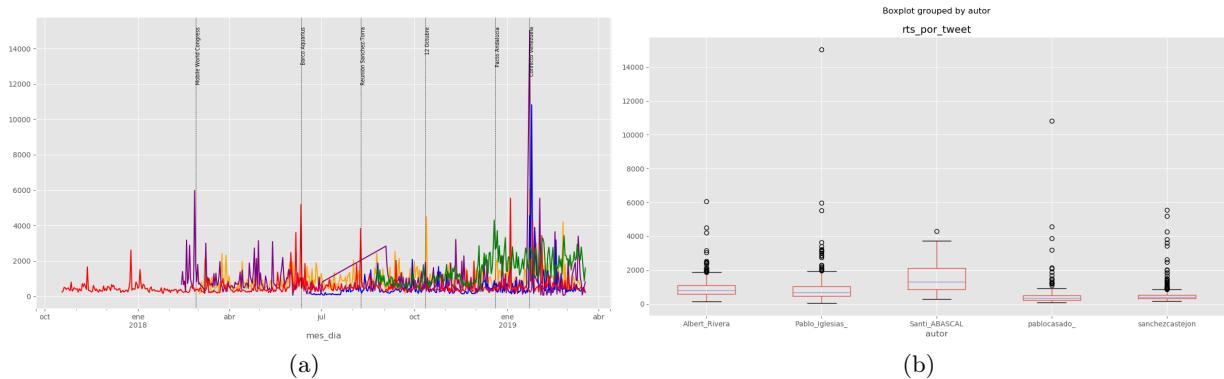


Figura 3.6: (a):Número total de *retweets* obtenidos en los últimos meses en función del total de mensajes enviados. (b): Diagrama de cajas del número total de *retweets*.

La otra manera que dispone Twitter de medir la popularidad de una cierta cuenta es a través del número de favoritos que recibe cada mensaje. Como se puede comprobar en la Figura 3.7, el total de favoritos por número de mensajes no tiene que coincidir con el total de *retweets* obtenidos. En este caso, se muestra una distribución más homogénea, donde los máximos se corresponden

con acontecimiento importantes del ámbito político. De hecho, el diagrama de cajas muestra que apenas existen diferencias por candidatos, siendo Albert Rivera quien recibe de media una mayor cantidad de favoritos y Casado de nuevo quién se aleja más de la cima de la clasificación.

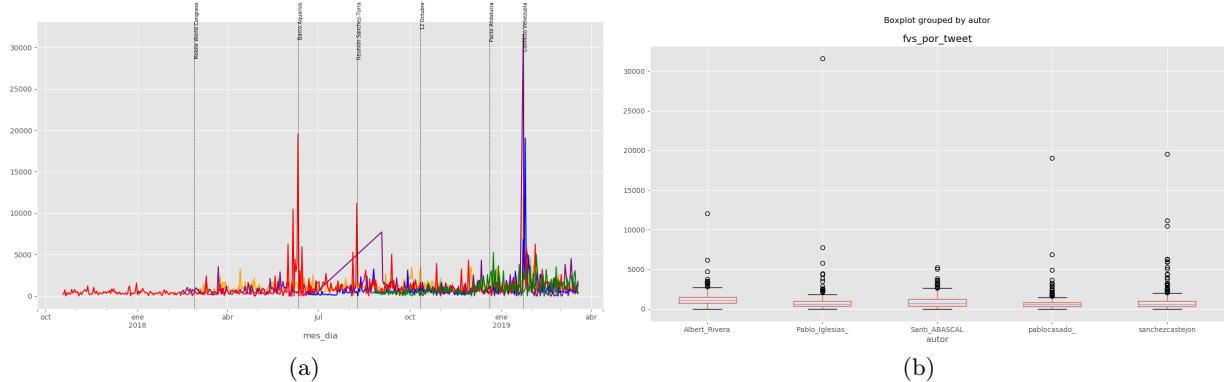


Figura 3.7: (a):Número total de favoritos obtenidos en los últimos meses en función del total de mensajes enviados. (b): Diagrama de cajas del número total de favoritos.

### 3.4. Análisis del contenido

Nos centramos en lo que sigue en el contenido de los mensajes. Para ver la importancia del la eliminación de las *stopwords*, en la Figura 3.8 se relaciona el número total de palabras empleadas en todos los mensajes por mes antes de realizar el procesado y después. Prácticamente nos quedamos con la mitad de palabras que contenían los mensajes originales. Por otro lado, en esa misma Figura se añade el caso particular para Albert Rivera, donde se muestra cómo varía nuestro análisis si no consideramos deshacernos de las palabras sin significado. Así, por ejemplo, comparando la frecuencia con la que aparece cada palabra en el texto filtrado y en el mismo sin filtrar, se comprueba que los elementos sujetos de análisis apenas aparecen en el análisis inicial.

Al igual que hicimos antes, se puede comparar a los 5 políticos en función de su uso de la red social, en este caso a través de la longitud medio de los *tweets* por usuario. Así, la Figura 3.9 muestra el diagrama de cajas del número de palabras totales escritas por mensaje. En promedio es Pablo Casado quien escribe mensajes más largos, con bastante ventaja respecto a Albert Rivera. En última posición se sitúa Santiago Abascal, quien parece no interesarle transmitir mensajes con excesivo contenido.

Una manera más adecuada de comparar el contenido de los diferentes políticos es usando las nubes de palabras o *word clouds*. En esta forma de representación visual, las palabras más importantes tienen un tamaño mayor. Simplemente contando el número de veces que aparece

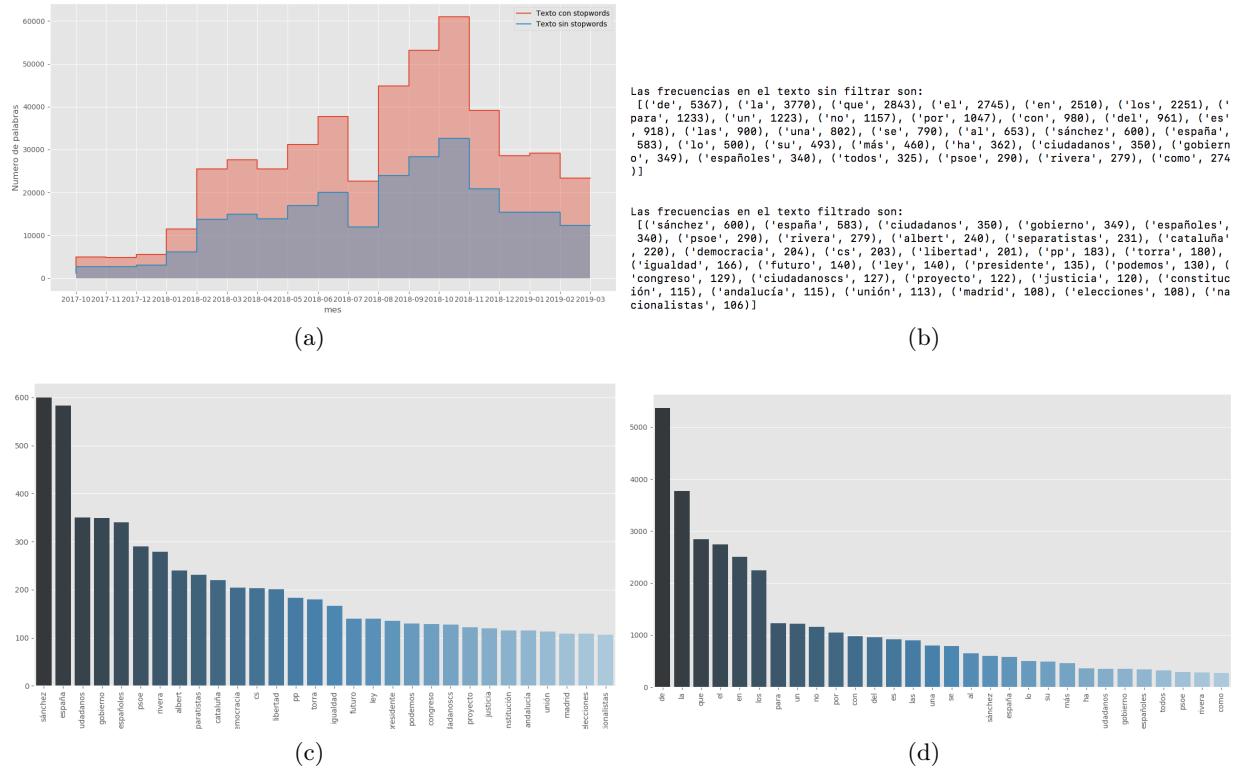


Figura 3.8: (a): Número total de palabras publicadas en los mensajes de los 5 autores en función del mes. (b), (c) y (d): Número de palabras en los mensajes publicados por Albert Rivera antes y después del proceso de tokenización.

cada palabras en los mensajes de cada autor, se crea una tabla con las frecuencias obtenidas que la librería de python *wordcloud* es capaz de interpretar y convertir en la salida que se expone en la Figura 3.10. Se observan claras diferencias en función del autor. Así, a las palabras comunes que mantienen todos los candidatos, como España, política o gobierno junto con el nombre de su propio partido, cada político añade aquellas a las que da más importancia. Albert Rivera muestra una mayor preferencia por nombrar al presidente de gobierno, centra su discurso en Cataluña y el proceso independentista (nombrando de forma común a los separatistas, a Cataluña, a Torra, a la unión o a los nacionalistas) y muestra su ideario liberal. Pablo Iglesias, en cambio, expone temas más sociales, apelando a las pensiones, a las mujeres, la ciudadanía, los presupuestos y muestra una mayor inclinación a hablar sobre el PP y Mariano Rajoy. Pablo Casado de nuevo vuelca su atención sobre Cataluña y el futuro de España, hablando sobre su propio proyecto en términos de ilusión, libertad, elecciones, unidad, etc. Pedro Sánchez se muestra más similar a Iglesias, con un discurso centrado en sus propuestas de igualdad, compromiso, pensiones, ley etc. Por último, se tiene a Santiago Abascal con un discurso mucho más centrado en su partido, mencionando constantemente bien a su persona o bien a otros integrantes del partido. Encontramos también

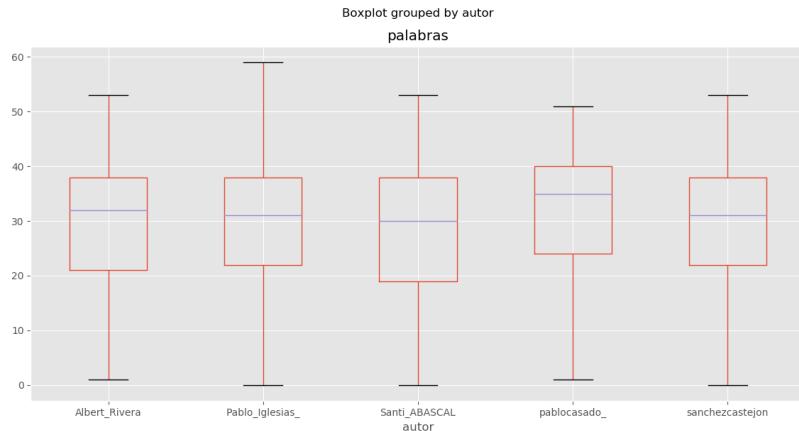


Figura 3.9: Diagrama de cajas del número total de palabras usadas por *tweet* en función del autor.

referencias continuas a las elecciones andaluzas, al proceso independentista y al gobierno.

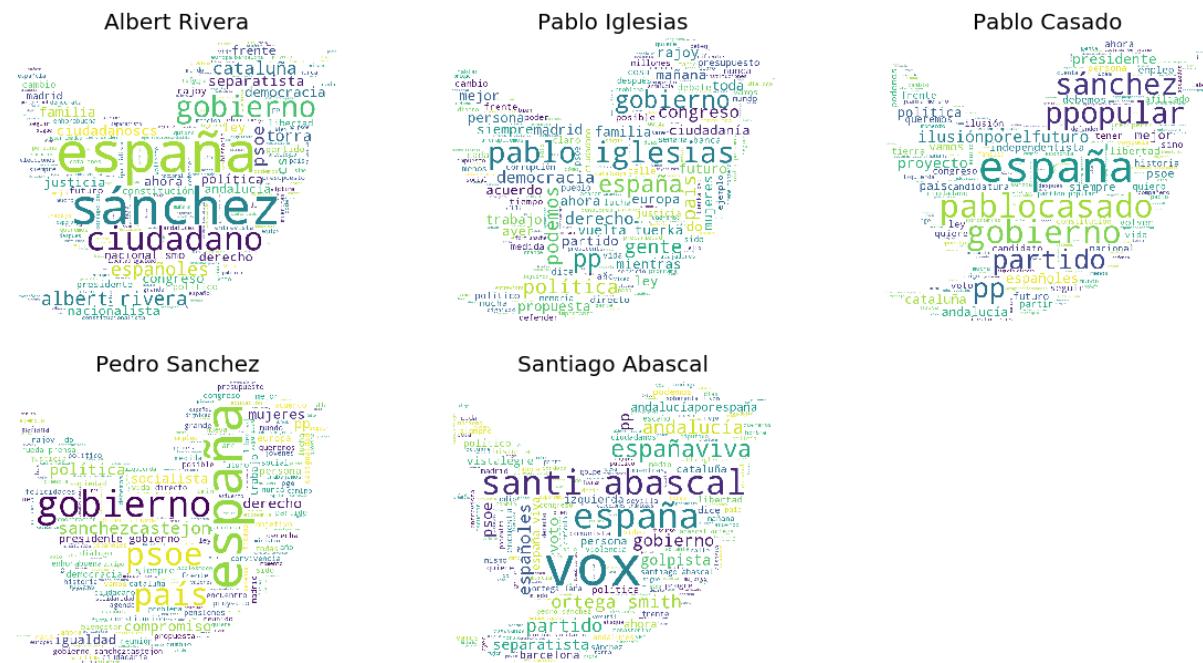


Figura 3.10: Diagrama de cajas del número total de palabras usadas por *tweet* en función del autor.

Hemos visto la existencia de palabras comunes entre las más usadas de los 5 políticos. Esta similitud entre las cuentas puede ser cuantificada mediante un procedimiento de correlación en el uso de cada palabra. El objetivo es comparar la frecuencia con la que cada autor emplea cada palabra para poder deducir la existencia de contenidos parecidos en los mensajes. Computacionalmente realizar una correlación implica primero convertir los datos en una matriz donde se

refleje la frecuencia de uso de cada palabra. En el código de texto primeramente se separa los *tweets* de cada autor por cada *token* que contiene y se crea un nuevo *dataframe* con índice dichos caracteres y columnas los autores. El resultado se expone en la Figura 3.11. Podemos fijarnos bien en la matriz de correlaciones o bien en la representación gráfica de la misma. En cualquier caso, se muestra que las mayores similitudes se alcanzan entre Casado y Rivera. Previamente en el apartado anterior ya habíamos notado el uso similar de ciertas palabras. Por otro lado, se tiene también una relación estrecha entre Iglesias y Sánchez. Por el contrario, Abascal apenas logra coincidir su vocabulario con el del resto de adversarios.

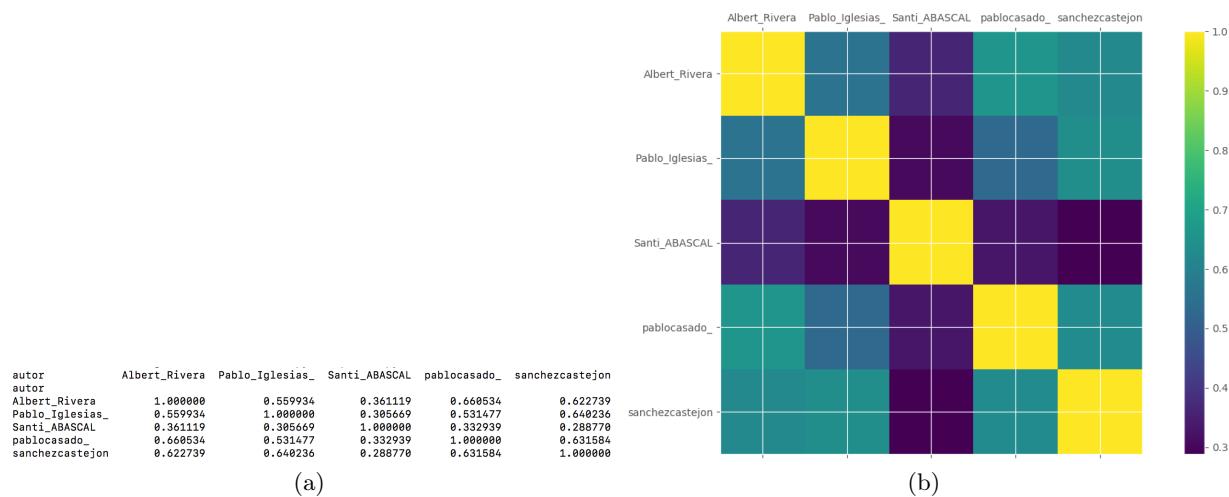


Figura 3.11: Matriz de correlaciones para la frecuencia de uso de cada palabra en función de los distintos autores.

Otra manera de llegar a la misma conclusión es empleando una gráfica de dispersión como la que se expone en la Figura 3.12. En esa misma Figura, se añade la coincidencia en el uso de ciertas palabras para el caso de Albert Rivera y de Pablo Casado. Se muestra que el tema de Cataluña, hablar sobre el gobierno y las futuras elecciones son problemáticas compartidas.

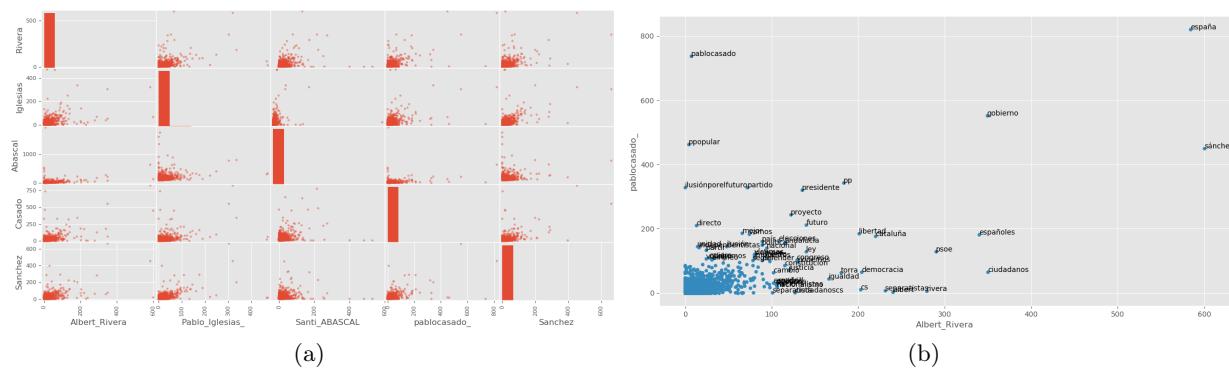


Figura 3.12: Gráfico de dispersión para la frecuencia de uso de cada palabra.

No obstante, en el análisis previo hemos comprobado que existen palabras, como España, que utilizan todos los autores de manera masiva. Por el contrario, podemos encontrarnos con vocablos que son propios de cada político. En lo siguiente nos dedicamos a analizar las palabras más diferenciadas, esto es, aquellas palabras que un autor determinado utilizada en varias ocasiones mientras que el resto apenas menciona. Esto se consigue mediante la técnica del logaritmo de los odd ratios. Se conoce por *odd* a la probabilidad de que suceda un evento dividido por la probabilidad de que no suceda e indica cuánto más probable es la ocurrencia de un evento frente a que no ocurra. En este caso, el *odd* de cada palabra se corresponde con el número de veces que cada autor la cita dividido entre el número total de palabras distintas. El caso de los *odd ratio* se corresponde con una asociación entre dos variables, que indica una razón de probabilidades. La ventaja de emplear logaritmos con los *odd ratios* es que permite una lectura simétrica de la relación entre proporciones.

En resumen, buscamos obtener una comparación a pares entre los 5 autores que cumpla la siguiente relación:

$$\text{logoddsratio} = \log \frac{(n_1/(N_1 - n_1))}{(n_2/(N_2 - n_2))} \quad (3.1)$$

siendo  $n_1$  el número de veces que aparece la palabra  $n$  en todos los comentarios del autor 1 y  $N$  el número total de palabras empleadas por dicho autor. Como ejemplo se expone el caso de la frecuencia de palabras de Albert Rivera y de Pablo Casado, Figura 3.13. Para entender el gráfico, a mayor valor positivo del *odd ratio*, mayor posibilidad de que la palabra la diga Rivera frente a que la diga Casado mientras que a mayor valor negativo mayor probabilidad de que sea Casado quien la escribe.

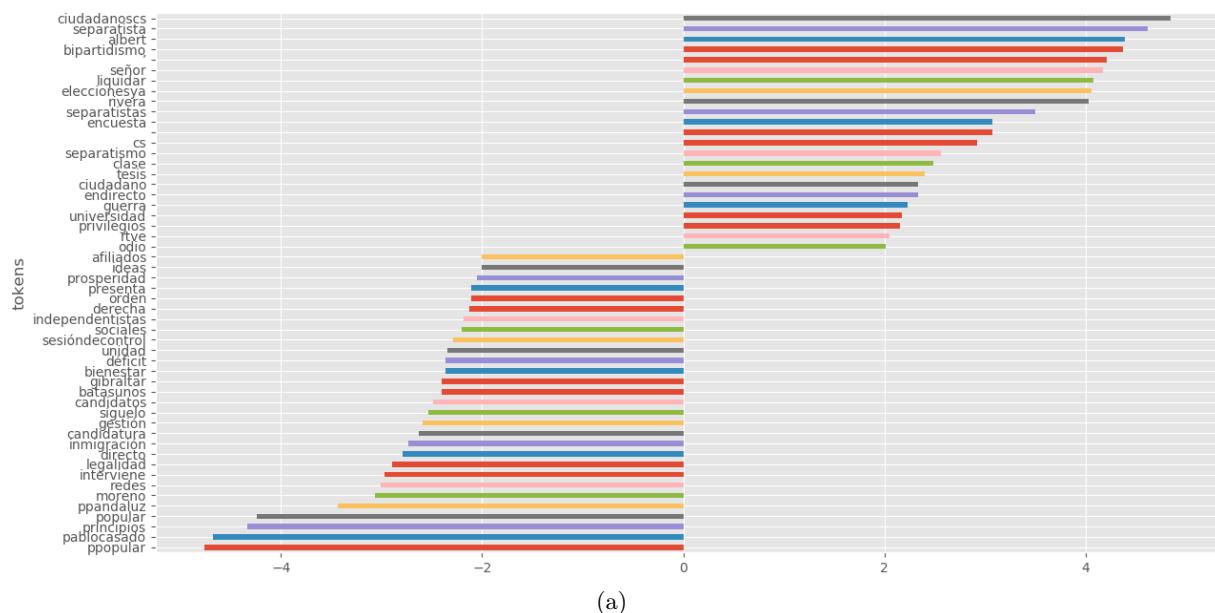


Figura 3.13: Logaritmo de los *odd ratio* para la frecuencia de palabras entre Albert Rivera y Pablo Casado.

Para terminar esta sección, se hace un pequeño análisis de los *hashtags* más usados por los políticos. Aunque pueda parecer innecesario, mucha de la información que ofrecen los mensajes los podemos encontrar en estas palabras. No obstante, y aunque un estudio en profundidad podría sacar a la luz importantes conclusiones, en este trabajo no nos centraremos en ellos. Únicamente se presenta la clasificación por orden de uso para cada autor de los diferentes *hashtags*, presente en la Figura 3.14. Así, para el caso de Rivera este sencillo y rápido estudio pone de manifiesto lo relevante de la lucha de este partido contra el bipartidismo y contra el gobierno, con la búsqueda de nuevas elecciones, y la predilección por temas polémicos como son Venezuela o Alsasua. Pablo Iglesias se centra más en cuestiones reivindicativas como la huelga feminista, el caso de la manada o el tema de las pensiones. Pedro Sánchez tiende a mostrar su oferta de gobierno mientras Santiago Abascal busca publicitarse.

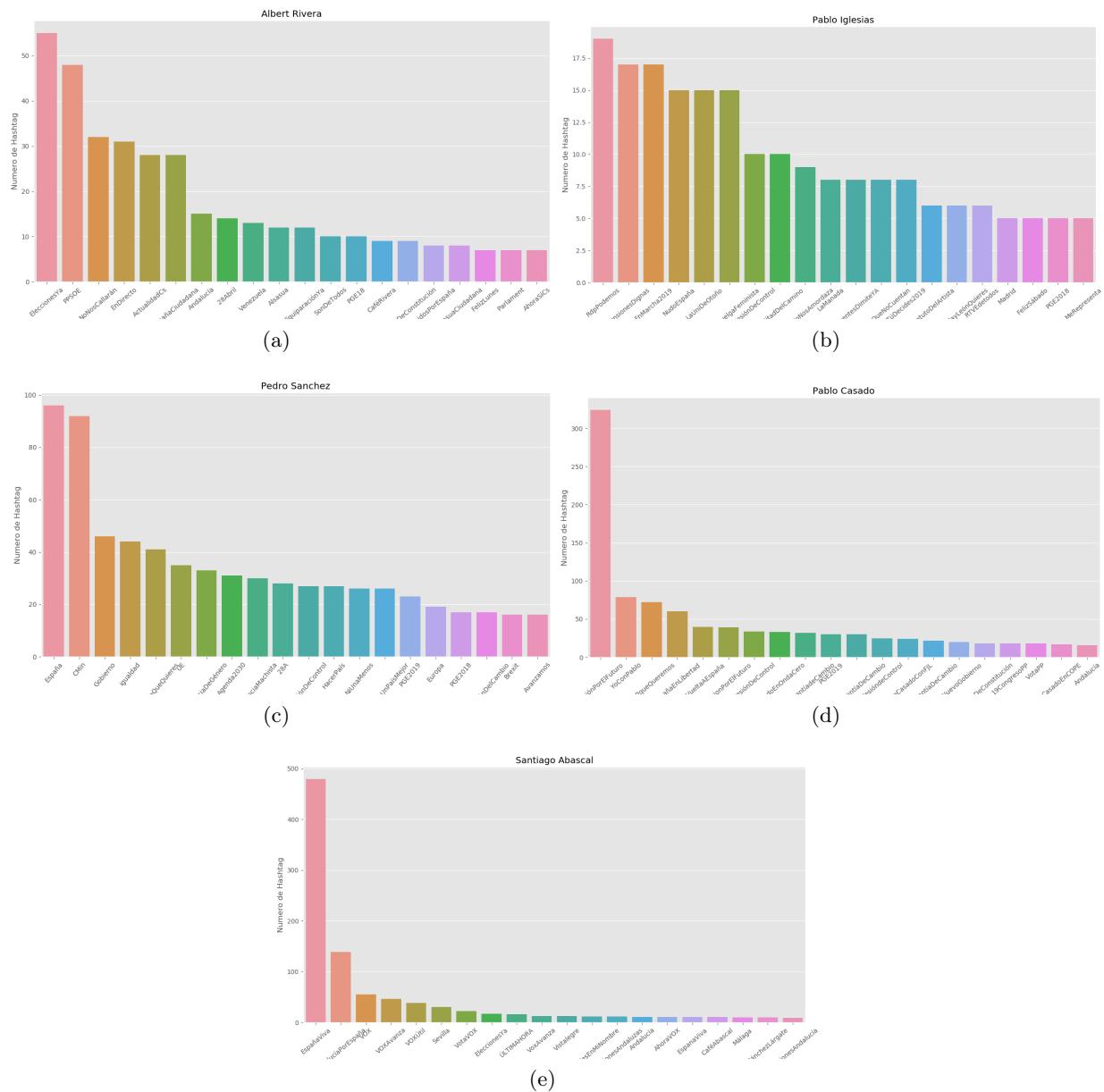


Figura 3.14: Clasificación por orden de importancia de los distintos *hashtags* usados por los políticos.

### 3.5. Análisis de sentimiento

El análisis de sentimiento de un texto es un estudio que consiste en extraer las opiniones y emociones que genera dicho escrito usando técnicas de lenguaje automatizadas. Dentro de este campo, podemos encontrar varias metodologías que nos lleven a este fin. Una de las más empleadas es aquella que construye el sentimiento que genera un cierto mensaje como suma de los sentimientos que producen las palabras que lo contienen por separado. De esta forma, al igual que se ha hecho antes, se dividen los mensajes en sus componentes y se evalúa el sentimiento de cada uno. En la actualidad, los programas estadísticos ofrecen diferentes opciones para llevar a cabo este proceso, pero para entender mejor su funcionamiento podemos llevar a cabo el procedimiento a través de un diccionario que asocie cada palabra a un sentimiento.

En una primera aproximación, se procede a clasificar la opinión de cada *tweet* como positiva o negativa. No se consideran más detalles en cuanto al sentimiento específico que esconde la opinión. Esto puede traer consigo cierto sesgo consecuencia de la presencia de ironía además del no pequeño error de medición. No obstante, Twitter es una buena opción para este tipo de análisis pues el hecho de que el número de caracteres en cada mensaje sea limitado hace que los usuarios suelen lanzar su opinión sin incluir elementos innecesarios. En resumen, nuestra tarea consiste en elaborar un diccionario donde se coloquen aquellas palabras del castellano que puedan evocar un sentimiento positivo, marcado por +1 o uno negativo, indicado con -1. Dado un mensaje, simplemente debemos sumar el valor del sentimiento total generado para poder determinar el sentimiento final de la oración. En el caso del castellano, existen diversos diccionarios (o lexicón) disponibles para valorar el sentimiento de una palabra. Para nuestro modelo, se ha empleado uno ya existente: el ISOL, elaborado por [3] y que se compone de un total de 2.509 palabras positivas y 5.626 palabras negativas.

Los archivos con las palabras negativas y positivas son leídos con python de forma que para cada mensaje sólo es necesario contar el número de veces que coincide una palabra contenida en el mismo con una palabra del diccionario. Como ejemplo, en la Figura 3.15 se muestra la polaridad que se extra de algún *tweet*. En el primer caso, se tiene un mensaje neutro, donde la suma de las componentes negativas y positivas se anulan. Los dos siguientes casos, son *tweets* positivos, con un nivel de positividad de 1 en el primer caso y de 4 en el segundo. Por último, se incluye un mensaje negativo. En general, se pueden comprobar las bondades y deficiencias del modelo con este simple ejemplo.

Procediendo de esta manera, se obtiene en la Figura 3.16 el sentimiento medio que generan

(a) Edmundo Bal (@BalEdmundo)

Quiero que en España cumplir con tu obligación y ser fiel a la verdad no sea motivo de castigo. Quiero que en mi país los servidores públicos puedan trabajar sin miedo.

Por eso me sumo al proyecto de @CiudadanosCs convencido de que vamos a conseguir una España mejor. 🇪🇸

(b) Albert Rivera (@Albert\_Rivera)

Las personas con discapacidad son héroes cada día, superan obstáculos permanentemente. No habrá igualdad hasta que logremos su plena inclusión. Gracias a @Fundacion\_ONCE y @Cermi\_Estatal por esta visita a la Fundación del Perro-Guía. Me ha encantado dejarme guiar por Leva. 🐶😍

(c) Albert Rivera (@Albert\_Rivera)

Soy afortunado por tener un padre al que quiero y admiro, por haberme enseñado tanto y por haberme cuidado siempre.

Y tengo la suerte de ser el padre de Daniela, a la que amo y amaré siempre incondicionalmente. A todos los padres, ¡#FelizDíaDelPadre!

[instagram.com/p/BvMIJGkB8gr/...](https://instagram.com/p/BvMIJGkB8gr/)

13:40 - 19 mar. 2019

183 Retweets 1.149 Me gusta

(d) Albert Rivera (@Albert\_Rivera)

Los totalitarios separatistas vuelven a acosar e insultar a @manuelvalls en Barcelona. Tienes todo mi apoyo, Manuel, seguiremos defendiendo la libertad, la unión y la democracia por toda Cataluña y por toda España. #NoNosCallarán

Figura 3.15: Ejemplos de *tweets* y su clasificación en función de nuestra técnica de análisis de sentimiento. En rojo las palabras consideradas negativas y en verde aquellas positivas.

todos los mensajes de cada candidato político así como el porcentaje de mensajes positivos y negativos en relación con el total. Pedro Sánchez es quién tiene una media más alta, notando un mayor sentimiento positivo en sus *tweets*, aunque seguido de cerca por Casado. En el lado opuesto se sitúa Santiago Abascal que es el único que mantiene una media de sentimiento negativo en sus mensajes.

Para mayor claridad, en la Figura 3.17 se muestra un histograma con la frecuencia de *tweets* de cada autor que devuelven cada nivel de sentimiento. En todos casos, la mayoría de los mensajes son neutros aunque tienden a decantarse hacia el lado positivo.

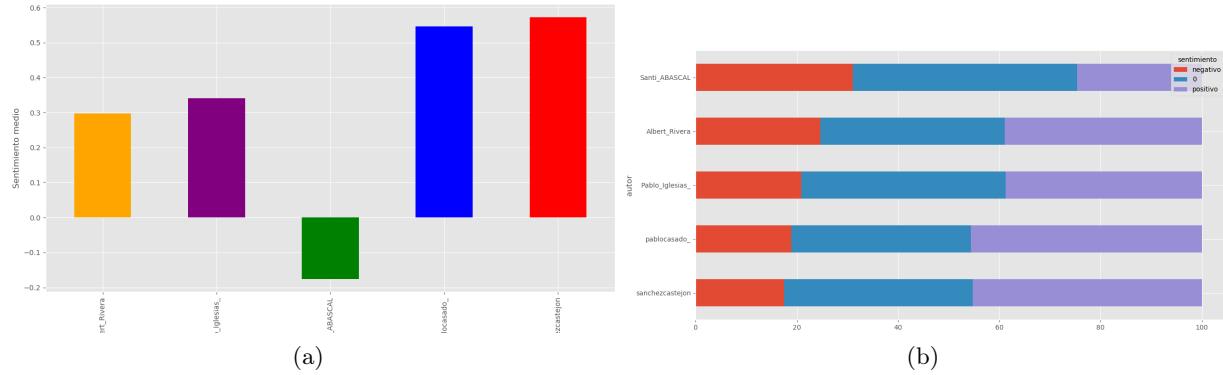


Figura 3.16: Sentimiento medio que generan todos los mensajes de cada político.

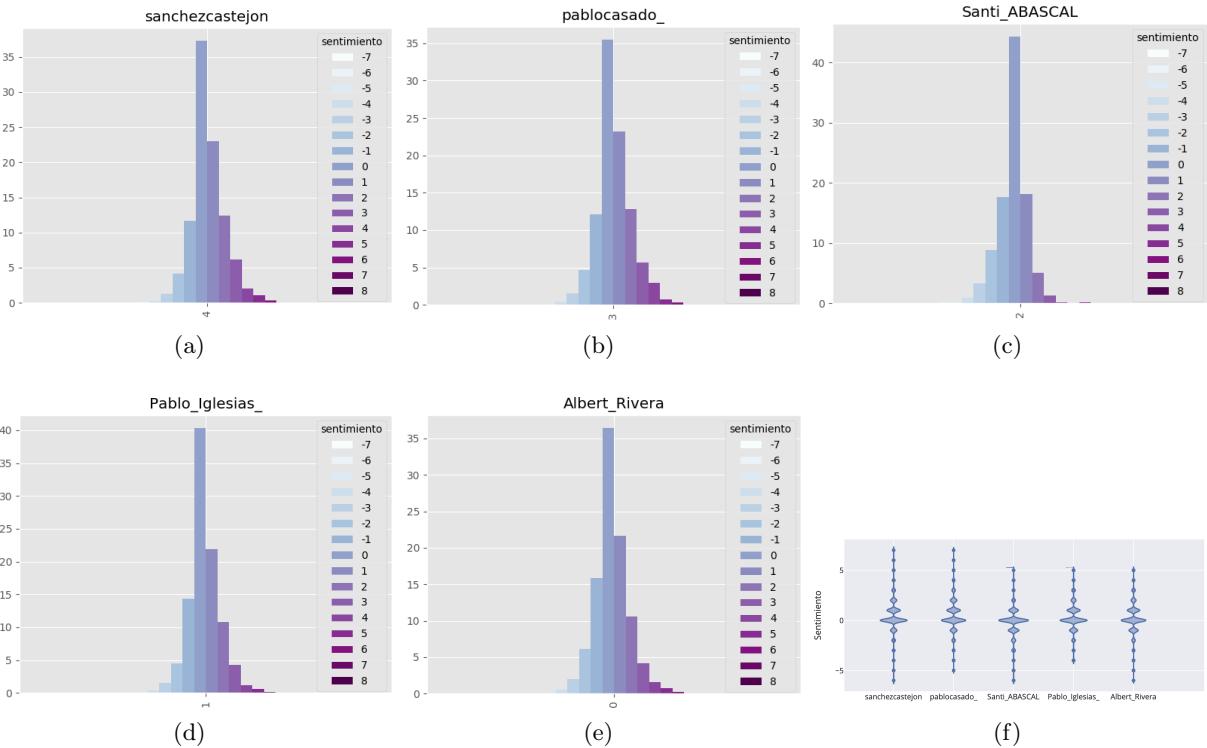


Figura 3.17: Número de tweets en proporción de cada autor en función del nivel de sentimiento que generan.

Es interesante comprobar la evolución de los sentimientos generados en su conjunto durante los últimos meses. Así, en la Figura 3.18 nos encontramos con que la mayoría de los tweets negativos se concentran durante períodos de convulsión política, como las elecciones andaluzas y en los últimos días con el juicio del proceso independentista. Se muestra también en la Figura 3.19 la posible relación entre el sentimiento que genera un tweet y el número de retweets o influencia de dicho mensaje. Realmente no podemos concluir que exista una relación evidente.

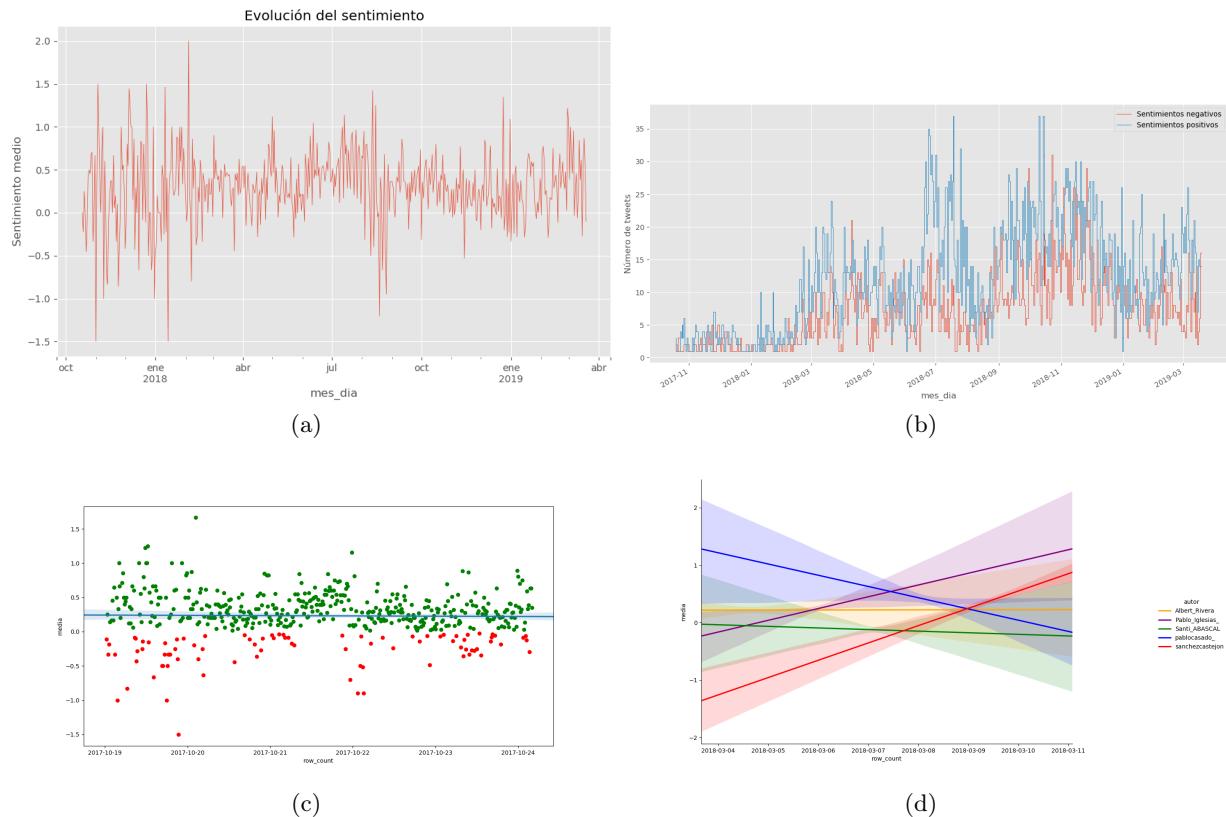


Figura 3.18: (a): Evolución del sentimiento medio generado por todos los *tweets*. (b): Número total de *tweets* negativos y positivos. (c): Línea de tendencia de una regresión con la evolución del sentimiento medio.

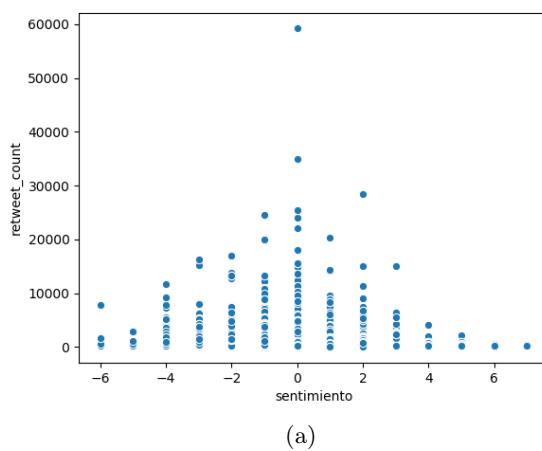
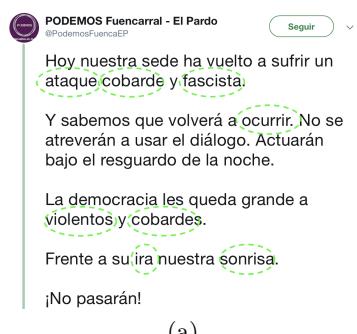


Figura 3.19: Gráfico de dispersión con el sentimiento que genera cada mensaje y el número de *retweets*

### 3.6. Análisis de emociones

Dentro de las múltiples opciones que ofrece el análisis de sentimiento, en la sección anterior nos hemos decantado por una técnica que clasifica palabras según su polaridad. No obstante, perdemos información relativa al sentimiento en concreto que cada mensaje trae consigo. Es por ello, que en lo que sigue se aplica un procedimiento muy similar al anterior añadiendo una clasificación por distintas emociones. La metodología es muy similar al análisis de polaridad. En primer lugar, se requiere un diccionario donde se puntúe cada palabra. En este trabajo, se acude al léxico *NRC Emolex*, disponible en [2] y consistente en un total de 14.182 palabras clasificados en 8 emociones básicas: ira o enfado, miedo, anticipación, confianza, sorpresa, tristeza, alegría y asco o disgusto. Si una palabra pertenece a una de estas categorías se indica con un 1, en caso contrario se asigna un 0.

Como es de suponer, el uso de este diccionario presenta varios problemas. En primer lugar, se trata originalmente de una lista de palabras en inglés que ellos mismos traducen a otros idiomas entre los que se encuentra el castellano. En consecuencia, puede haber errores de traducción, fallos en los sentimientos que evocan cada palabras e incluso ausencia de palabras usadas en nuestro idioma para designar las ocho emociones básicas. Además, si ya era complicado clasificar un mensaje en función de su polaridad, el reto es mucho mayor cuando se trata de encontrar la emoción concreta que refleja. En esto, es importante notar que ahora no vamos a clasificar los *tweets* con una sola emoción, sino que se calcula el porcentaje presente de todas ellas. Se expone un ejemplo de como funciona el clasificador en la Figura 3.20.



(a)

Figura 3.20: Ejemplo de un mensaje y las emociones que nuestro diccionario registra. Así, la palabra *ataque* se relaciona con miedo y enfado, la palabra *cobarde* con asco, miedo y tristeza, la palabra *ira* con enfado y miedo, *violentos* con enfado, asco, miedo y sorpresa y así sucesivamente. En resumen, este artículo destacaría por un 23 % de enfado, un 17 % de asco, un 30 % de miedo y un 20 % de tristeza.

Procedemos a calcular las emociones medias que generan todos los mensajes de cada autor.

Así, en la Figura 3.21 se muestra el porcentaje con el que aparece cada sentimiento en los *timeline* de cada político. Aunque estos resultados tienen un elevado error de medición, confianza es la emoción que destaca en todos los autores. En cuanto al resto de emociones, apenas existen diferencias. Únicamente destaca el caso de Santiago Abascal donde el porcentaje de enfado y de miedo supera en media a los valores del resto de candidatos.

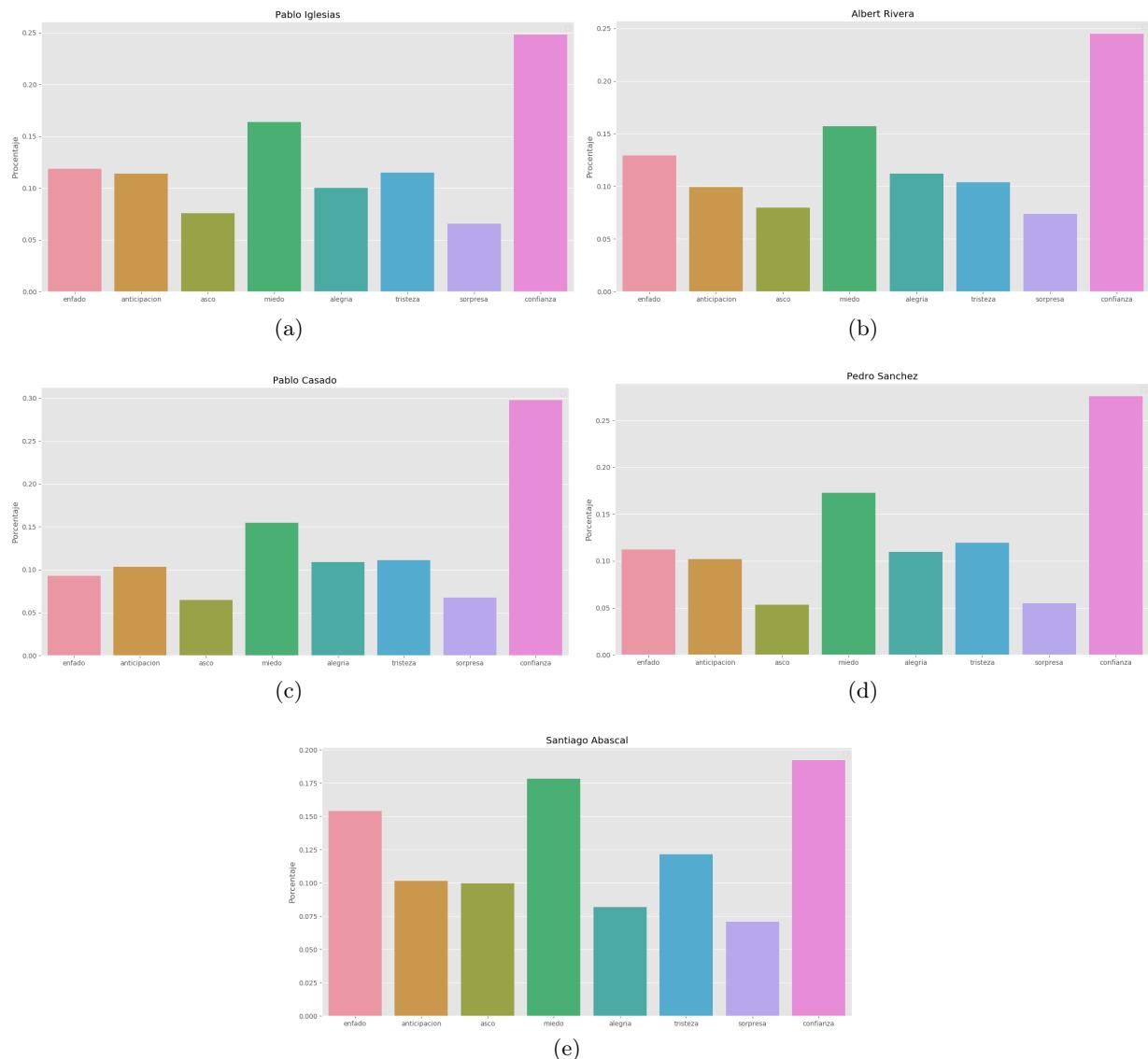


Figura 3.21: Porcentaje de emociones que podemos encontrar en los mensajes de los diferentes autores.

La metodología presenta numerosas equivocaciones. De hecho, si comprobamos cuáles son las palabras que más contribuyen a cada sentimiento, nos encontramos con algunos errores claros. Haciendo un *wordcloud* similar al llevado a cabo para el número de palabras empleadas por autor, R dispone de una librería que nos devuelve qué palabras se asignan a cada sentimiento

en mayor manera. Esto se expone en la Figura 3.22. Vemos que para el sentimiento de miedo se incluye sorprendentemente al gobierno. En el caso de asco, se añade también separatista o congreso. Esto son solo algunos ejemplos de los cambios que se deberían hacer manualmente en una base de datos obtenida de una traducción del inglés.



(a)

Figura 3.22: *wordcloud* para las palabras que contribuyen en mayor medida a cada sentimiento.

### 3.7. Clasificador mediante aprendizaje supervisado

Esta otra técnica de análisis de contenido se basa en el uso de algoritmos de aprendizaje automático, incorporados dentro de la clasificación de *machine learning*. El objetivo es encontrar patrones trabajando sobre una base de datos con gran cantidad de información mediante procesos automáticos. Su funcionamiento se basa en generalizar reglas a partir de ejemplos dados donde se apliquen dichos patrones. De hecho, la calificación de supervisado se debe al hecho de que se requiere de un grupo de ejemplo previamente clasificado que de lugar a un modelo utilizado el cual dará lugar a posteriori al algoritmo de selección. Todo proceso de aprendizaje automático supervisado precisa de dos conjuntos: un conjunto de entrenamiento, encargado de enseñar al clasificador, y un conjunto de prueba que determina la eficacia del mismo. Notar que los datos deben ser previamente etiquetados de forma manual. En nuestro caso, el análisis de sentimiento realizado en las secciones previas nos deja una base de datos donde las palabras están clasificadas por su polaridad. El modelo de aprendizaje que propone este trabajo se basa en la posibilidad de crear un algoritmo que determine, por un lado la polaridad del mensaje, y por el otro su autoría.

Durante el proceso de entrenamiento, el algoritmo de clasificación que elijamos establece una relación entre los elementos del conjunto, agrupándolos en tantos conjuntos como etiquetas existan. No cabe duda de que cuanto mayor sea el conjunto de datos entrenados, mayor será la eficacia de nuestro clasificador. Existen numerosos clasificadores, indicados para distintas situaciones. En nuestro caso, vamos a hacer referencia a 3 diferentes. En primer lugar, se presenta el algoritmo de los k vecinos más cercanos (*k Nearest Neighbour*), basado en criterios de vecindad. Este método de clasificación estima la probabilidad de que un cierto elemento pertenezca a una cierta clase según tenga k vecinos más cerca de un grupo o de otro. Se obtiene la distancia de cada nuevo elemento con los ya almacenados, y se determina el grupo al que pertenece de acuerdo. Otro método que emplearemos será el *Random forest*, un método de tipo árbol de decisión que se basa en un conjunto de condiciones y acciones que relacionan unos factores con una decisión final. El último algoritmo será el SVM o máquinas de soporte vectorial, basado en el concepto de hiperplanos y la búsqueda de separar las observaciones.

En cualquier caso, el procedimiento que se sigue es bastante análogo. Así, en primer lugar se separa cada mensaje en los componentes o *tokens* que lo componen. Cada componente recibe una tupla formada por dos valores, el primero de ellos se corresponde con el número del *tweet* al que pertenece y el segundo es relativo a cada *token* que aparece en el conjunto de mensajes. Estos son los valores con los que se entrena el método. No obstante, la tarea de clasificación de cualquier

método presenta una problemática a la hora de llevar a cabo su entrenamiento. Nos encontramos con ciertas palabras, como *gobierno* o *españa* que aparecen constantemente referenciadas en los mensajes. En consecuencia, estamos atribuyendo una importancia excesiva a términos que realmente no aportan mucha información. Para evitar estos desajustes se utiliza el método *tf-idf*. Este método otorga una mayor relevancia a aquellos términos que aparecen con más frecuencia en todo nuestro documento pero en pocos mensajes del mismo. Su funcionamiento resulta muy sencillo de entender. En primer lugar, se calcula la frecuencia con la que nos encontramos cada *token* en nuestro conjunto de datos (*tf* o *term-frequency*). En términos matemáticos:

$$tf(token) = \frac{n_{token}}{longitudbasedatos}. \quad (3.2)$$

Este valor es ponderado por la inversa de la frecuencia con la que dicho *token* aparece en los distintos mensajes de la base de datos. De nuevo, en términos matemáticos:

$$idf(token) = \log\left(\frac{n_{tweets}}{n_{tweetsconeltoken}}\right). \quad (3.3)$$

Finalmente, el estadístico *tf-idf* mide la importancia de cada término combinando ambas expresiones:

$$tf - idf(token) = tf(token)idf(token). \quad (3.4)$$

Este es el procedimiento habitual a seguir por los distintos métodos de aprendizaje automático con supervisado que podemos emplear. No obstante, teniendo en cuenta que nuestra base de datos se compone de mensajes enviados por políticos diferentes, que aluden constantemente a ciertos vocablos, como su propio partido o su nombre, hace que esta ponderación no sea muy exitosa. Palabras como *Vox* son muy utilizadas por Santiago Abascal y, en consecuencia, el estadístico *tfidf* debería ser pequeño. No obstante, como el resto de candidatos apenas alude a este partido, nos encontramos con un valor mucho más elevado. Por otro lado, términos como *España* o *españoles* no debería tener tanta relevancia pero el hecho de no haber eliminado los morfemas nos lleva a error. Por otro lado, el uso de logaritmos da mayor importancia a la presencia relativa de cada término. En nuestra base de datos, existen diferencias enormes entre *tokens* que aparecen de forma habitual y quien lo hace esporádicamente. No obstante, si ponderamos la frecuencia relativa y el porcentaje de aparición en mensajes por igual, el resultado carece de significación. Este método es implementado directamente por python a la hora de vectorizar los mensajes, aunque para entender su funcionamiento en el código se dispone de su implementación manual. Las diferencias, como se ha comentado, no son muchas y se muestran en la Figura 3.23.

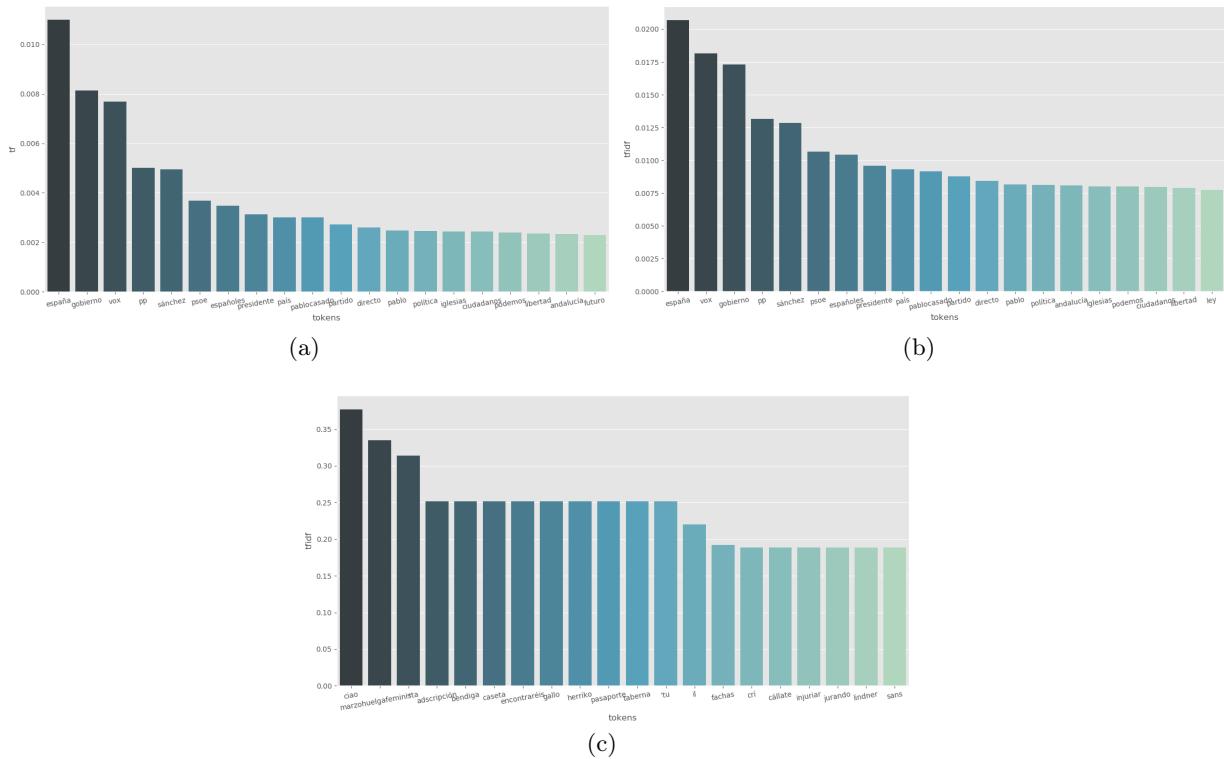


Figura 3.23: Método de ponderación de la importancia de distintas palabras basado en el estadístico  $tfidf$ . En (a): se muestra el peso relativo de cada palabra dentro del conjunto total de nuestra base de datos. En (b) se tiene la ponderación realizada. En (c) se usa la ponderación con igual peso de la frecuencia relativa de cada palabra y el número de mensajes en los que aparece.

En estas condiciones, podemos proceder a entrenar nuestro algoritmo de clasificación. El proceso de vectorización, esto es, la asignación del número de *tweet* al que cada *token* pertenece y un valor que lo caracteriza, así como el ponderado previo se lleva a cabo mediante la secuencia *vectorizer.fit\_transform*. Precisamente, el comando *fit* permite fijar las correspondencias creando un diccionario en el cual cada palabra lleva asignada un número determinado. Esto nos permite, posteriormente, transformar un texto cualquiera de acuerdo en lo establecido en la vectorización. En la Figura 3.24 se muestra un ejemplo de la salida que nos devuelve.

A continuación, los datos se dividen en los conjuntos de entrenamiento y de prueba. Para este fin, se emplea la secuencia *train\_test\_split*, la cual reparte el 50 % de los datos para el entrenamiento de forma aleatoria, fijando un *seed* a través de *random\_state*. El resultado se compone del conjunto de prueba, guardado en *X\_test*, un array con los sentimientos correspondientes, *y\_test* y un equivalente para el resto de datos.

(0, 23943)	0.31809890294189946
(0, 23843)	0.14069453836985177
(0, 10590)	0.08834613104841876
(0, 11478)	0.16379756926138983
(0, 7464)	0.18761594783820446
(0, 5989)	0.06445648092522618
(0, 28931)	0.14782072361057044
(0, 20367)	0.19895743339833494
(0, 26521)	0.1177292647353969
(0, 12713)	0.24323790768166445
(0, 16917)	0.040342626605916254
(0, 29630)	0.169719455998717
(0, 20104)	0.06808294574068097
(0, 26248)	0.1530245298928844
(0, 19464)	0.20606997483886857
(0, 7719)	0.10617465693461485
(0, 4730)	0.2535872870894985
(0, 18989)	0.11270243657890285
(0, 21523)	0.11823256169892953
(0, 17831)	0.05506021306577656
(0, 26569)	0.2077284992775273
(0, 23831)	0.16018749059583665
(0, 23699)	0.18719234501250576
(0, 28548)	0.17365991130100197
(0, 26804)	0.12059092373447997

(a)

Figura 3.24: Ejemplo de la salida vectorizada de nuestra base de datos.

### KNeighborsClassifier

En primer lugar, utilizamos el clasificador de vecinos más cercanos a k. Este algoritmo se basa en clasificar elementos asignándoles la clase que poseen los elementos similares. Así, busca los puntos de datos más similares (por cercanía) aprendidos durante la fase de entrenamiento. El algoritmo comprueba las clases a la que pertenecen estos elementos más cercanos y en función de ello se decidirá su clasificación final. Implementar este método en python resulta muy sencillo con el comando *KNeighborsClassifier(n\_neighbors=3)* (el número de vecinos elegidos es 3), pues solamente requiere entregar los conjuntos de prueba y de entrenamiento, los cuales son usados para enseñar al algoritmo mediante *clf.fit*.

Observando los resultados de nuestro modelo predictivo, se tiene que la calidad en la muestra de prueba es del 39 %. La calidad del clasificador por lo tanto es baja, lo cual no es una sorpresa. En primer lugar, se ha construido un clasificador sobre una metodología que ya de por si tenía cierto sesgo a la hora de definir sentimiento. En la Figura 3.25 se muestra cómo nuestro modelo ha etiquetado los diferentes mensajes en función a cómo estaban etiquetados originalmente. Es claro que se tiende a considerar neutros la mayoría de los mensajes. Quizás el método de los vecinos no sea el más adecuado en nuestro caso. Cuando nosotros introducimos un texto nuevo, el método busca los vecinos más cercanos y analiza su puntuación. Dado el enorme desnivel que existe en cuanto a mensajes neutros, son mucho más numerosos que el resto, esto puede estar influyendo en el resultado. En cuanto al resumen de nuestro clasificador, para entender la tabla se explica los componentes más importantes: La precisión hace referencia a la habilidad

del clasificador para, dado una estimación en cada clase, no equivocarse; el *recall* nos indica en cada clase cuántos se han clasificado correctamente; *support* es el número de veces que ocurre cada clase en nuestra base de datos.

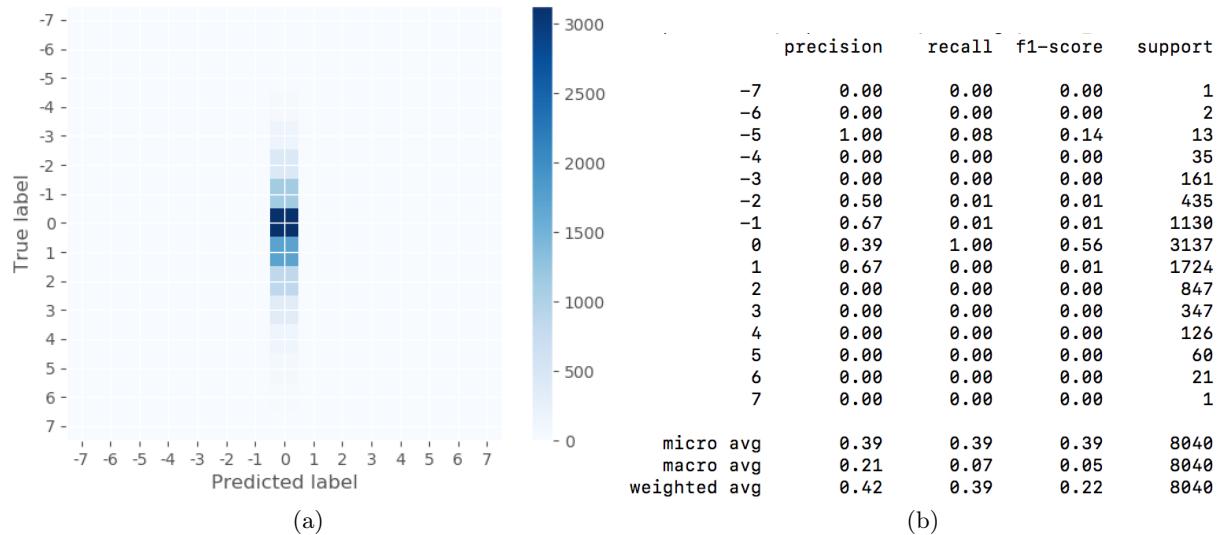


Figura 3.25: Clasificación del sentimiento de los *tweets* usando el algoritmo de los k vecinos.

Una opción para intentar mejorar el modelo podría ser eliminar la nivelación de los sentimientos y considerar únicamente aquellos negativos, positivos o neutros. No obstante, este mismo clasificador nos devuelve una calidad ligeramente superior pero innecesaria, como se comprueba en la Figura 3.26, donde se muestra la probabilidad de encontrar cada tipo de sentimiento una vez pasado el clasificador a nuestros datos de prueba.

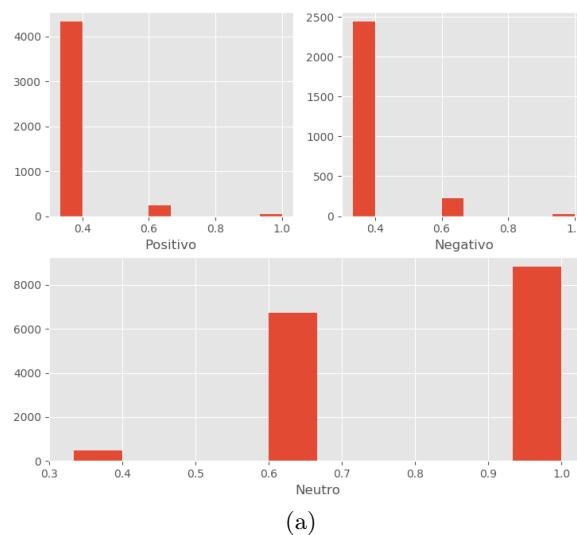


Figura 3.26: Clasificación del sentimiento de los *tweets* usando el algoritmo de los k vecinos con solo tres clases.

En consecuencia, vamos a volver a intentarlo cambiando esta vez de algoritmo de clasificación. Escogemos en este caso el método de *Random Forest*. Su funcionamiento se basa en realizar múltiples árboles de decisión en función de diferentes combinaciones. De lo que se determine en cada árbol se clasificará un mensaje con un sentimiento y otro. La calidad de nuestro predictor aumenta hasta el 62 %, un porcentaje insuficiente, pero mucho mayor que en el caso anterior. Los resultados, expuestos en la Figura 3.27, muestran que ya no existe tanta tendencia a considerar todos los mensajes neutros, sino que entre los positivos (denotados por 1), un alto porcentaje se encuentra bien etiquetado.

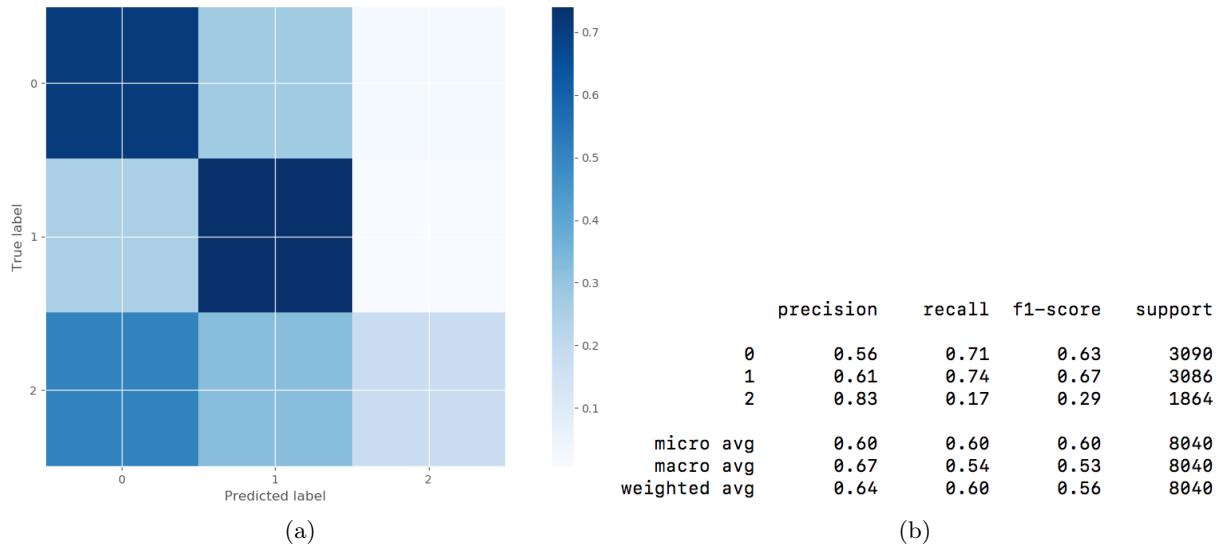


Figura 3.27: Clasificación del sentimiento de los *tweets* usando el algoritmo de Random Forest, donde 0 es un sentimiento neutro, 1 es un sentimiento positivo y 2 es un sentimiento negativo.

Finalmente, se construye un clasificador que nos prediga si un determinado *tweet* pertenece a un líder político. El modelo en esta ocasión se enmarca dentro de la estrategia de clasificación uno contra todos, puesto que ciertos artículos muestran un mal funcionamiento de ciertos clasificadores cuando las clases no son binarias. La sentencia utilizada es *onevsrestclassifier*, la cual, indicando el algoritmo deseado, nos devuelve la probabilidad de que un cierto mensaje se corresponda a las distintas clases. Básicamente, se ajusta a cada clase un clasificador, de forma que las muestras de dicha clase son tomadas como positivas mientras el resto se consideran negativas. Para etiquetar un nuevo caso, cada clasificador calcula la probabilidad de que pertenezca a su clase. La calidad del nuevo modelo supera el 80 %, lo que mejora considerablemente lo anterior, y, permite clasificar de forma adecuada la autoría de cada mensaje. Así, en la Figura 3.28, se muestra que únicamente existen pequeñas discrepancias a la hora de distinguir si el autor del mensaje es Pablo Iglesias o Pedro Sánchez, pero en el resto de ocasiones la decisión es acertada.

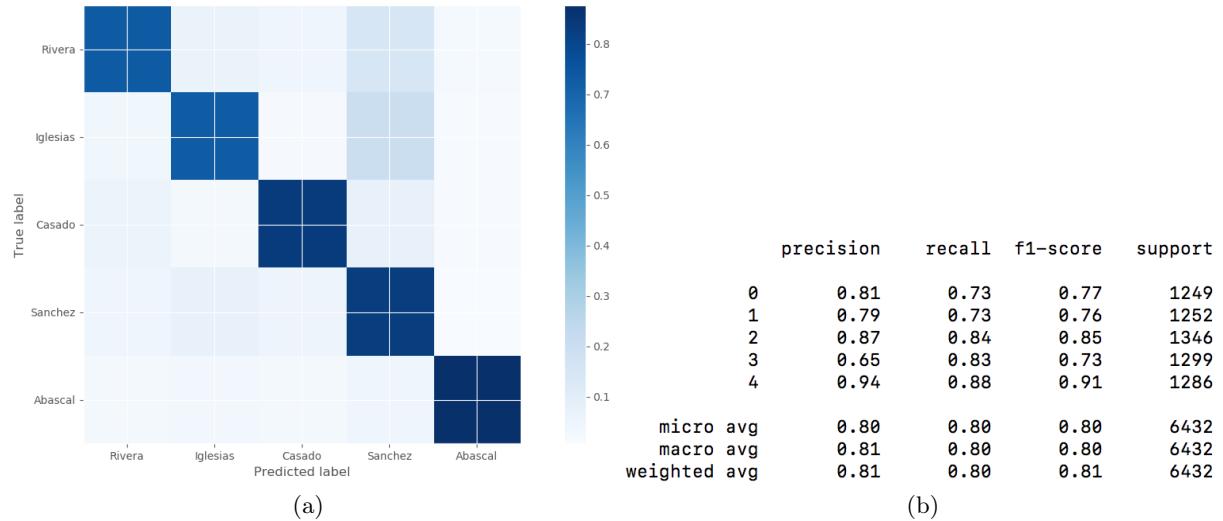


Figura 3.28: Clasificación de la autoría de los mensajes.

Para terminar la sección, se evalúan algunos de los últimos *tweets* de los políticos, para validar nuestro modelo predictivo. En la Figura 3.29 se representa cada mensaje utilizado y la probabilidad de acuerdo al algoritmo de que dicho mensaje pertenezca a cada líder político. Exceptuando ciertos casos en los que existe duda, la eficacia del modelo es absoluta.

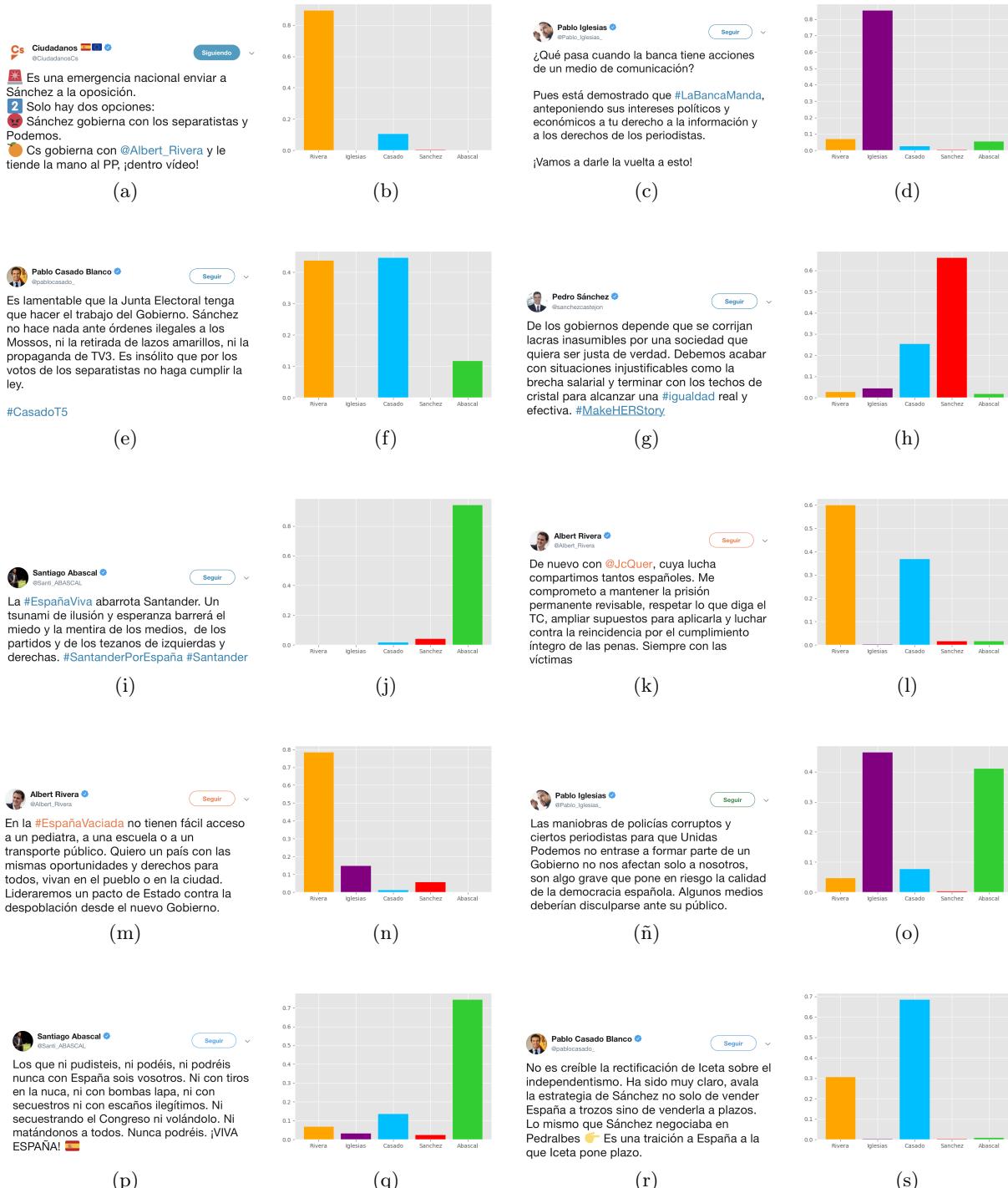


Figura 3.29: Clasificación de la autoría de alguno de los últimos mensajes de cada político. A la izquierda el mensaje a analizar y a la derecha las probabilidades que asigna nuestro modelo.

### 3.8. Análisis de los seguidores

Para finalizar el estudio, se incluye como introducción un análisis más extendido de *twitter*, incluyendo en el mismo los seguidores de cada líder político. Usando como base de nuevo R, el comando que nos permite recuperar los seguidores es *get\_followers*. Se tiene de nuevo el problema con la limitación de búsquedas por minuto, para lo cual se añade la opción *retryonratelimit*, que ejecuta el código cada vez que se para. La información sobre los seguidores se guarda en un archivo csv que posteriormente es leído en python. El dataframe resultante muestra que Pablo Iglesias es el político con más número de seguidores en la red social, ver Figura 3.30. Podemos aplicar un procedimiento análogo al que vimos con la relación existente entre mensajes y observar las similitudes que presentan los seguidores de cada personaje. Así, se comprueba que la mayor cercanía, entendiendo como tal el mayor número de seguidores compartidos, se encuentra entre Rivera y Sánchez o Casado. El hecho de que Abascal tenga pocos seguidores puede conllevar que buena parte de sus seguidores, quizás medios de comunicación o personajes que siguen a más políticos, coincidan no solo con otros personajes de la derecha sino también con Sánchez o Iglesias.

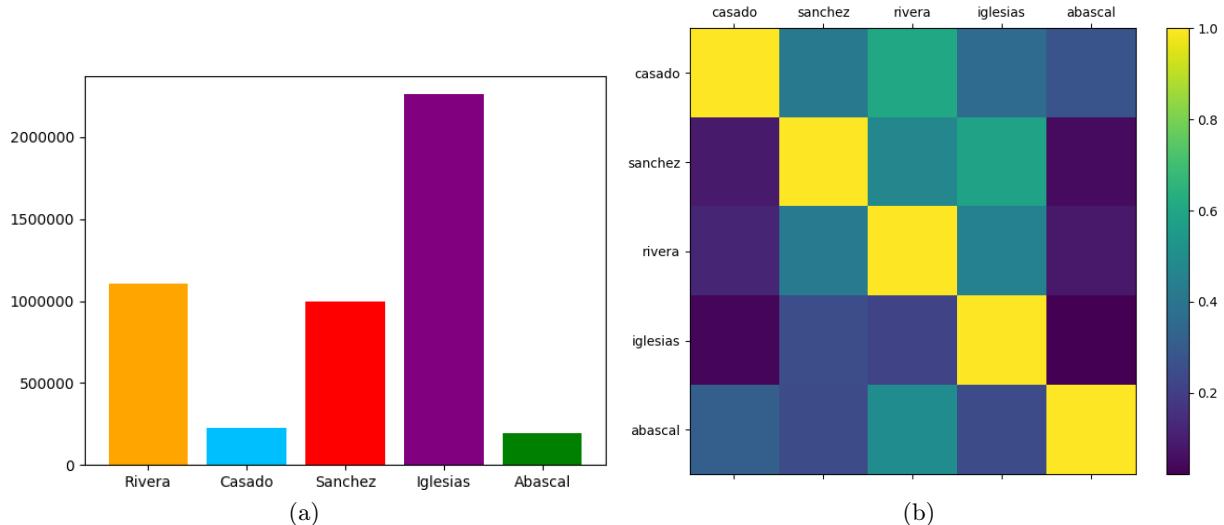


Figura 3.30: (a): Número total de seguidores por político. (b): Porcentaje de coincidencia de seguidores. La tabla se entiende como sigue: Porcentaje de seguidores del político de la izquierda que son a su vez seguidores del político de arriba.

Por último, se comprueba la existencia de seguidores falsos. En esto, vamos a definir un seguidor falso como aquel que tiene cero seguidores. Evidentemente, pueden existir cuentas sin seguidores que no sean falsas, y usen Twitter como medio de información, pero en su mayoría se trata de cuentas creadas para aumentar seguidores. Así, en la Figura 3.31 se muestra que es

Pablo Iglesias quien más seguidores falsos tiene, seguido de cerca por Pedro Sánchez.

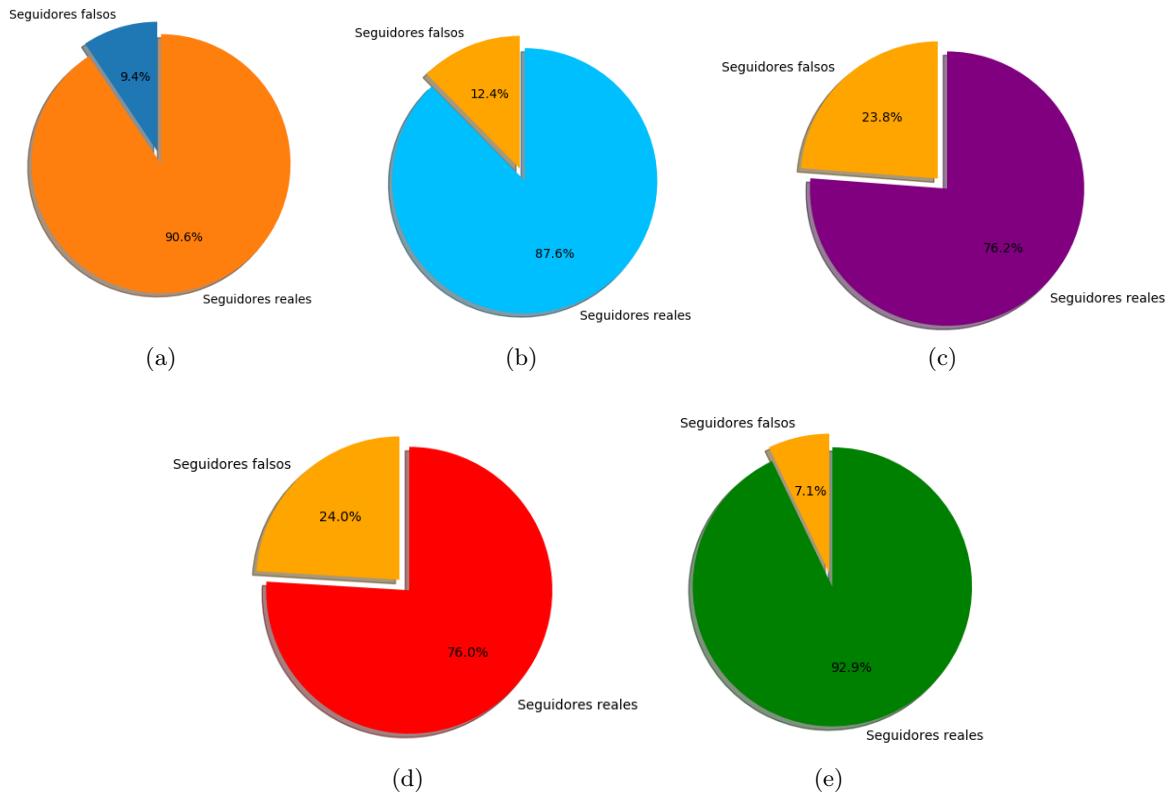


Figura 3.31: Porcentaje de seguidores de cada político que se pueden considerar falsos.

## Capítulo 4

### Licencia del trabajo

La licencia de uso que se emplea en este trabajo es la *Creative Commons Reconocimiento (CC BY 3.0)*, que permite utilizar los datos del trabajo de manera libre únicamente haciendo referencia a la fuente principal y al autor. La razón es que los datos empleados en el proyecto son abiertos, fáciles de obtener por cualquier usuario y sin excesivo interés por ocultar. El objetivo del trabajo no era llegar a ninguna conclusión novedosa, sino simplemente elaborar una herramienta que permita analizar el sentimiento de los mensajes publicados en una red social. En consecuencia, no tiene sentido poner restricciones a su uso. Además, la manipulación de los datos no ha requerido mayores complicaciones que las computacionales, con lo que el código del proyecto puede ayudar a otras personas a seguir enriqueciendo el análisis.

En resumen, se permite compartir, copiar o distribuir el material presente en este trabajo en cualquier medio o formato. Además, se puede modificar o transformar y hacer cualquier tipo de nuevas creaciones con independencia de su finalidad, incluida la comercial. Las condiciones bajo las cuales se admiten estos actos es el reconocimiento de la autoría.



(a)

# Capítulo 5

## Conclusiones

Este trabajo no pretendía ser un estudio exhaustivo de la situación política de España, sino el objetivo era únicamente elaborar técnicas y procedimientos que nos permitan analizar los mensajes publicados en redes sociales, en este caso *twitter*. De esta forma, se ha explicado la metodología a seguir en caso de querer extraer información sobre esta red social, así como algunos de los principales procedimientos para analizar la base de datos ofertada. Es importante llevar a cabo una limpieza o filtrado de los mensajes antes de analizarlos, pues se vio que más de la mitad del contenido en realidad no nos sirve para nada. En cuanto al análisis propio del sentimiento, aunque existen diversas formas de realizarlo, la más sencilla, usando un simple diccionario clasificador de palabras, nos sirve para determinar qué políticos tienen unos mensajes más positivos y cuáles más negativos. Realmente interesante fue el caso de los modelos predictivos que nos devolvían con elevada precisión la autoría de diferentes mensajes hechos después de la elaboración de la base de datos. En general, se ha puesto de relieve la fuerza que tienen estos mecanismos para el análisis de diferentes redes sociales y, usando un mero ejemplo sin importancia, hemos sido testigos de la valiosa información que de forma sencilla se puede extraer.

# Bibliografía

- [1] Interactive Advertising Bureau. Estudio anual de redes sociales. I, 2018.
- [2] S. M. Mohammad and P.D. Turney. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34, 2010.
- [3] M. D. Molina González, E. Martínez Cámara, M. T. Martín Valdivia, and J. M. Perea Ortega. Semantic orientation for polarity classification in spanish reviews. expert systems with applications. 40(18):7250–7257, 2013.