FACULTAD DE MATEMÁTICAS

DEPARTAMENTO DE ESTADÍSTICA E INVESTIGACIÓN OPERATIVA

TÉCNICAS ESTADÍSTICAS EN ANÁLISIS DE REDES SOCIALES

Memoria realizada por Gaspar Ríos Alcobendas

Dirigido por:
Dr. Jose Luis Pino Mejías

Resumen

El gran auge de las tecnologías de la comunicación ha supuesto un gran cambio en la forma en que seres humanos nos relacionamos. En particular nos hemos acogido notoriamente al uso de redes sociales, en las cuales nos relacionamos virtualmente emulando la vida real. Es por esto que, con el propósito de conocer un poco más el comportamiento de los usuarios resulta de gran interés la aplicación de técnicas estadísticas en redes sociales.

En la introducción del trabajo hablamos de la importancia que tiene el estudio de las redes sociales. Además de ello presentamos la red social Twitter, servicio de microblogging, y el software estadístico R.

El el primer capítulo hablaremos de cómo se extraen dichos mensajes así como la posterior organización de estos mensajes con varios ejemplos.

En el segundo capítulo explicamos con detalle el Análisis de Frecuencia, aplicándolo a un conjunto de datos basado en mensaje de Twitter. En este vemos la construcción de la matriz término-documento, fundamental a lo largo del trabajo, y una aplicación de esta en forma de wordcloud de los mensajes de la cuenta oficial de Donald Trump.

En el tercer capítulo vemos el Análisis Clustering o de Conglomerados. En él comenzamos introduciendo el concepto de distancia entre documentos. Posteriormente explicamos detalladamente los métodos jerárquicos y los divisivos. Además, se incluye una aplicación de ambos sobre las mensajes de la famosa cuenta @wikileaks, observando resultados bastante claros a la hora de agrupar los temas principales de los que se habla en dicha cuenta.

En el cuarto capítulo nos centramos en el Análisis de Sentimientos. Resul-

ta ser un tema de gran importancia en lo que aplicación a Twitter se refiere ya que dicha red social es comúnmente usada para volcar las opiniones las opiniones personales sobre toda índole de temas. Primeramente explicamos en qué consiste el análisis de polaridad, exponiendo un ejemplo en el que la opinión de usuarios angloparlantes sobre la corrupción en España es claramente negativa. Por otro lado vemos el análisis de sentimientos basado en técnicas clasificatorias como el SVM, Naives Bayes o el algoritmo de máxima entropía.

Abstract

The great rise of communication technologies has brought about a significant change in the manner in which we as human beings relate. We have notably welcomed the use of social networks, through which we interact virtually emulating real life. It is for this reason that, with the purpose of knowing a little more about the behavior of users, the application of statistical techniques in social networks is of great interest.

In the introduction of the work we talk about the importance of the social nets studies. In addition we present the social network Twitter, microblogging service, and the statistical software R.

In the first chapter we present how such messages are extracted as well as the subsequent organization of these messages with several examples.

In the second chapter we explain in detail the Frequency Analysis, applying it to a data set based on Twitter messages. In this we see the construction of the matrix document-term, fundamental throughout the work, as well as its application in wordcloud format of the messages sourced from the official Twitter account of Donald Trump.

In the third chapter we see Clustering or Conglomerate Analysis. In the beginning we introduce the concept of distance between documents. Subsequently, we explain the hierarchical and divisive methods in detail. In addition, an application of both is included-sourced from the messages of the famous @wikileaks account, observing quite clear results when grouping the main subjects of which that it is spoken of in said account.

In the fourth chapter we focus on the Analysis of Feelings. It turns out to

be an issue of great importance in which the application of Twitter is referred to since this social network is commonly used to overturn personal opinions on a variety of topics. First, we explain the polarity analysis, setting forth an example in which the English-speaking users' opinion on corruption in Spain is clearly negative. On the other hand we see the analysis of feelings based on classification techniques such as the SVM, Naives Bayes or the maximum entropy algorithm.

Índice general

Introducción

El gran auge de las tecnologías de la comunicación ha supuesto un gran cambio en la forma en que seres humanos nos relacionamos. En particular nos hemos acogido notoriamente al uso de redes sociales, en las cuales nos relacionamos virtualmente emulando la vida real.

Los estudios y el análisis de redes sociales sociales han crecido a su vez de forma significativa, suponiendo una importante abertura en el camino de este tipo de trabajos. No es esto de extrañar ya que conocer el comportamiento y la opinión o sentimiento de los usuarios nos permite predecir y gestionar diferentes procesos de índole social, económica y empresarial. Por poner algunos ejemplos podemos optimizar la comercialización de un producto en base a las opiniones de los usuarios relacionadas con el producto, encontrar un problema político en la difamación de falsos mensajes o por ejemplo encontrar zonas de nuestro territorio altamente machistas, encontrar usuarios que ejerzan gran influencia sobre el resto...

Con este tipo de objetivos en mente nos planteamos entonces la realización de este trabajo sobre técnicas estadísticas en análisis de redes sociales. Más allá de los fines u objetivos que persigamos nos centraremos sobre todo en la base matemática que tienen estas técnicas, es decir, en la teoría que subyace a los algoritmos y aplicaciones que son usualmente utilizados.

Nos centraremos en Twitter, una de las redes sociales que más se prestan a nuestro estudio, y en el software estadístico R. Es por esto que pasamos a introducirlos con más detalle.

Twitter Social Network

Twitter se ha convertido en uno de los mayores servicios de redes sociales. Se trata de un servicio de microblogging que permite enviar mensajes de texto plano con un máximo de 140 caracteres. Estos mensajes se llaman tweets y aparecen en la página principal del usuario. Su contenido va desde conversaciones entre dos usuarios a comentarios generales sobre temas de interés público, resultando entonces muy práctico para el análisis de las tendencias sociales. Cada usuario puede seguir a otros usuarios y ver sus tweets de forma automática en su ventana de inicio.
Una de las características más importantes es la existencia del llamado hashtag o almohadilla (#) seguido de la etiqueta que se quiera usar, por el cual los usuarios pueden agrupar sus publicaciones. Además, puede mencionarse expresamente a otro usuario con (@) seguido del nombre de usuario que quiera mencionarse.

Su fundación, en California, data del año 2006, fecha relativamente moderna. Se estima que tiene más de 500 millones de usuarios, generando 400 [9] millones de tweets al día y manejando más de 800 000 peticiones de búsqueda diarias.Es por esto que parece conveniente basarnos en dichos mensajes a fin de conocer y análizar los comentarios y opiniones de ciertas poblaciones, sobre todo las más informatizadas.

Sin embargo, no toda la información reside en el propio mensaje. Podemos tambien encontrar interesante conocer la distribución de los seguidores o followers, la distribución geográfica en datos agrupados por el hashtag o por minería de datos con R, la velocidad de propagación de ciertos mensajes y tendencias, etc...

Se puede acceder a Twitter desde la web (www.twitter.com), con aplicaciones para smartphones, o a través de SMS en ciertos países.

R

R es un lenguaje y entorno de código libre orientado a la realización de cálculos y gráficos estadísticos. Ofrece una gran variedad de técnicas estadísticas, disponibles como funciones que son llamadas desde código fuente

R. Aunque el entorno gráfico de usuario no es demasiado amigable, se han desarrollado diversos sistemas que facilitan las tareas, por ejemplo RStudio y recientemente Eclipse.

Usaremos R para la realización de un caso práctico al final del trabajo, dado que nos ofrece muchas ventajas.Algunas de estas son: la capacidad de combinar análisis de tipo general con análisis específicos, ofrecer gráficos de alta calidad, existencia de una gran comunidad de usuarios y de extensiones y librerías expecíficas...

Uno de las características más importantes es la existencia de librerias auxiliares, algunas de las cuales serán de gran utilidad en este trabajo. Estos paquetes son creados por los propios usuarios y subidos a un repositorio oficial (CRAN), los cuales nos permiten extender las funcionalidades de R.

Capítulo 1

Tratamiento de los datos

Como hemos dicho en la introducción el flujo de información que existe en las redes sociales es enorme. En este trabajo nos centraremos en la información que los usuarios de Twitter escriben como ”tweets”.

Veremos en este capítulo cómo extraer dichos datos en forma de mensaje, o tweet, y cómo podemos plantearnos su manipulación en función de la organización de estos.

1.1. Extracción de datos

Se nos presenta entonces la pregunta: ¿cómo podemos extraer esta información? En este trabajo profundizaremos sobre la información histórica de los usuarios y no en un estudio en tiempo real de los mensajes.Es por esto que llevaremos esta tarea a cabo mediante un proceso API de arquitectura REST, mientras que para estudios en tiempo real sería más conveniente usar un proceso Streaming API.
Las APIs REST proporcionan acceso programado para leer y escribir datos de Twitter pudiendo crear nuevos tweets y obtener información de usuarios, tendencias y localizaciones. En concreto la API REST identifica las aplicaciones de Twitter con los usuarios que utilizan OAuth, generando las respuestas en formato JSOn.
OAuth (Open Authorization) es un protocolo que permite flujos simples de autorización para sitios web o aplicaciones informáticas. Se trata de un protocolo que permite autorización segura de una API de modo estándar y simple

para aplicaciones de escritorio, móviles y web. En nuestro caso utilizaremos el software R, un entorno de programación estadística.

Procedemos entonces a autenticarnos, cargando en R nuestras credenciales. Por motivos de seguridad no incluiremos las credenciales reales, sin embargo podemos ver que tienen esta forma:

```
requestURL <- "https://api.twitter.com/oauth/request_token"
accessURL <- "https://api.twitter.com/oauth/access_token"
authURL <- "https://api.twitter.com/oauth/authorize"

consumerKey<-"xxxxxxxxxxxxxxxxxxxxxxxxx"
consumerSecret<-"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
as<-"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
at<-"xxxxxxxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

```
> setup_twitter_oauth(consumer_key=consumerKey, consumer_secret
+                     =consumerSecret, access_token=at, access_secret=as)

[1] "Using direct authentication"
```

Una vez autenticados podemos cargar, a modo ilustrativo, tweets de la cuenta de la universidad de Sevilla @unisevilla. Vemos algunos de ellos:

```
> #tweetsUS= userTimeline("unisevilla",3200)
> load("tweetsUS.RData")
> tweetsUS.df <- twListToDF(tweetsUS)
> for (i in c(5,6,7,13)) {
+ cat(paste0("[", i, "] "))
+ writeLines(strwrap(tweetsUS.df$text[i], 40))
+ }

[5] "Bob Dylan-Leonard Cohen: dos maestros
de la palabra" en @FFilologiaUS
https://t.co/GaSjyNAWuY #culturasev
https://t.co/NSI33KtzZi
[6] ¿Estuvisteis en el #SalónUS? Más de
16.000 se divirtieron y orientaron con
la oferta académica y deportiva del
Saló. https://t.co/wTteol4Bn0
[7] .@usvirtualempleo os ofrece mañana un
Taller de #CV y carta de presentación
https://t.co/QAVYlpPf4o #empleo
```

```
https://t.co/2WUc4Ebafz
[13] .@MSF_Espana organiza #SeguirConVida:
cómo sobreviven los civiles a la guerra
https://t.co/ChyZ55Ymbv
https://t.co/d302E3owFk

>
```

O cargar tweets que contengan las palabras universidad de Sevilla. Vemos los 5 primeros de ellos:

```
> #tweetsUS2<-searchTwitter("universidad+sevilla")
> load("tweetsUS2.RData")
> tweetsUS2.df <- twListToDF(tweetsUS2)
> for (i in c(1:5)) {
+ cat(paste0("[", i, "] "))
+ writeLines(strwrap(tweetsUS2.df$text[i], 40))
+ }
[1] RT @EsEndometriosis: Investigaciones
sobre #endometriosis en Sevilla: Inebir
colabora con la Universidad de Sevilla
en este proyecto https:.
[2] RT @Davidpati: @flechagob Esta
mola...142 años de funcionamiento para
la Universidad de Sevilla
[3] RT @juantorreslopez: Me satisface a mí
y me alegro por mis alumnos
https://t.co/8JO70uSirg
[4] @flechagob Esta mola...142 años de
funcionamiento para la Universidad de
Sevilla
[5] RT @dzaguilar: Los empresarios
consideran la @unisevilla la mejor de
España para estudiar Economía y
Derecho. https://t.co/JTRcEnA3Rr vía @.
```

Inevitablemente estamos sufriendo un sesgo importante en la muestra. Tanto el escaso tiempo que llevamos usando Twitter o cualquier otra red social como la necesidad de un status económico y social que permita el uso de esta tecnología hacen que debamos tener cuidado al sacar conclusiones.

1.2. Organización de los datos

Los conjuntos de datos nos llegan de muy diversas formas, por lo que cabe plantearse cómo organizar y modelar los datos para un mayor aprovechamiento de los mismo.

1.2.1. Modelo Booleano

No hay duda de que el modelo booleano es uno de los más utilizados en conjuntos de datos aleatorios en morfología matemática, geometría estocástica y estadísticas espaciales. Esta basado en la teoría conjuntista y el álgebra de Boole.

En este, cada documento se reprenta por un conjunto de términos donde cada uno se trata como una variable Booleana que recibe el valor 1 si pertenece al documento y 0 si no. Dicho términos se recogen en un fichero adicional en el que guardaremos el conjunto de términos de la colección de documentos que estemos explorando así como la correlación entre los mismos.

Las consultas se arman con términos vinculados por operadores lógicos. Estos operadores son un símbolo o palabra que se utiliza para conectar dos fórmulas bien formadas o sentencias. Rn este caso se usarán los operadores AND (verdadero sólo si los dos elementos son verdaderos), OR (verdadero si cualquiera de los elementos es verdadero), XOR (verdadero si cualquiera de las expresiones, pero no ambas, es verdadera) y NOT (cambia el valor de Falso a Verdadero y viceversa).

Los resultados son referencias a documentos cuya representacín satisface las restricciones lógicas de la expresión de búsqueda. Sin embargo existe el incoveniente de que no establece un ranking de relevancia sobre el conjunto de respuestasa una consulta, todos los documentos tienen la misma importancia. Ademas, podemos tener respuestas irreleventas y demasiado elevadas con lo que a documentos respecta. A su favor tiene el su fácil uso, pudiendo resultar útil para un usuario medio así como la facilidad para hacer consultas muy concretas funcionando bien con volúmenes grandes de datos.

1.2.2. Modelo espacio vectorial

Se trata de un modelo algebraico para representar documentos de texto(o cualquier objeto en general) identificandolos como vectores. Es un modelo muy utilizado para el filtro y recuperación de información, indexación y creación de rankings de relevancia. En este modelo tanto los documentos como las consultas son representadas como vectores.
Permite introducir un orden en los documentos recuperados, por lo que surge la necesidad de crear una función de similitud entre el documento y la consulta. Si un término aparece en un documento su valor en el vector será no nulo. Ahora bien, este valor o peso será el que nos marque la relevancia sobre el resto. Se ha desarrollado un gran número de técnicas con las que medir dicho valor. Una de las más conocidas es la llamada "term frecuency-inverse document frequency model". Este, a diferencia del más trivial método de conteo de ocurrencia, incorpora información global y local.
En este caso, el vector con pesos para un documento d será $v_d = (w_{1,d}, w_{2,d}, ..., w_{N,d})'$, donde el término de peso se define como:

$$w_{i,d} = tf_{i,d} * log(\frac{D}{df_{i,d}})$$

donde $tf_{i,d}$ es el número de veces que el término i aparece en el documento; $df_{i,d}$ el número de documentos en los que aparece el término i y D el número de documentos de la base o colección.

Dicho modelo sin embargo tiene algunos defectos. El tiempo de computación puede llegar a ser demasiado alto, no dejandonos trabajar. Ademas, y algo muy importante en lo que a práctica respecta, es que al añadir un nuevo vector (en este caso un nuevo documento) se tendrían que recalcular todos los vectores.

1.2.3. Modelo de grafo

Un grafo es una combinación de nodos y aristas. Los nodos representan diferentes objetos mientras que las aristas representan cierta conexión entre ellos. Más concretamente diremos que un grafo es un par G=(V,E) donde V es el conjunto de nodos y E el de aristas tal que $E \subseteq [V]^2$. Se trata de un modelo que nos representa de forma idónea procesos de la vida real como conexiones de usuarios.

Podemos clasificar los grafos en dos grandes grupos, los direccionados y no direccionados, basándonos en el tipo de aristas que lo compongan:

- Grafos no direccionados:
  Las aristas que componen estos grafos no tienen orientación. A la hora de modelizar situaciones las relaciones entre nodos siempre serán bilaterales.

- Grafos direccionados:
  Dichos grafos son los que contienen aristas orientadas, es decir, que las relaciones o conexiones entre nodos no tienen por qué ser recíprocas. Es un modelo muy interesante para representar los follows o seguimientos entre cuentas en twitter ya que un usuario puede seguir a otro sin que este siga al primero.

Un concepto importante en los modelos de grafos es la matriz de adyacencia. Esta es una matriz $v \times v$, donde $v = |V|$, en la que se pone de manifiesto la existencia o no de aristas entre vértices o nodos representando entonces un grafo de forma unívoca. Para grafos no dirigidos la componente ij es igual a 1 si los nodos i y j están relacionados y 0 en caso contrario, siendo entonces una matriz simétrica. Por otro lado para grafos dirigidos la componente ij será igual a 1 si existe una arista que vaya del nodo i al j y 0 en caso contrario.

Grafos de Twitter

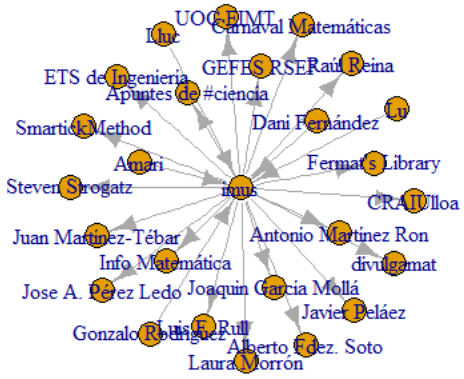
Como hemos comentado podemos definir un grafo dirigido en el cual los nodos sean los propios usuarios y las aristas la acción de seguir. Es decir, si seguimos a un usuario existirá una arista desde mi nodo al suyo. A modo ilustrativo vemos cómo contruir un subgrafo que represente las relaciones que mantiene la cuenta @*blog_imus*, cuenta oficial del Instituto de Matemáticas de la Universidad de Sevilla, con el resto de usuarios.

```
> # Cargamos librerias requeridas
> library("twitteR")
> library("igraph")
> # Obtiene el usuario, puede ser el tuyo o cualquier otro de
> # twitter
> user <-getUser("blog_imus")
```

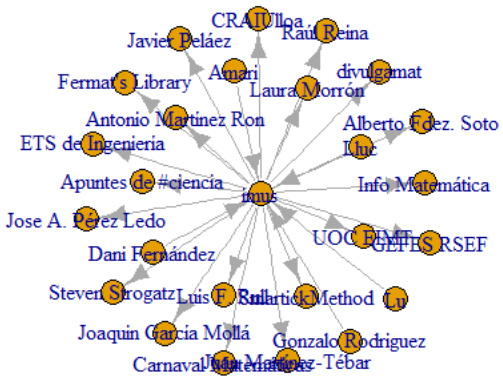
```
> # Obtiene IDs de amigos y seguidores
> friends.object <-lookupUsers(user$getFriendIDs())
> followers.object1 <-lookupUsers(user$getFollowerIDs())
> followers.object<-followers.object1[1:50]
> # Obtiene los nombres de amigos y seguidores
> # Se puede limitar el nombre de amigos y seguidores a visualizar,
> # ajustando las variables n y m
> # Si dichas variables no tienen nada, el máximo de amigos y
> # seguidores se desplegará
>
> #n <-length(friends.object)
> n=20
> #m <-length(followers.object)
> m=30
> friends <- sapply(friends.object[1:n],name)
> followers <- sapply(followers.object[1:m],name)
> # Crea un dataframe con relaciona los amigos y seguidores
> # al usuario
> relations <- merge(data.frame(User='imus', Follower=friends),
+                 data.frame(User=followers, Follower='imus'), all=T)
> # Crea un grafo de las relaciones
> g <- graph.data.frame(relations, directed = T)
> # Asigna las etiquetas (nombres) al grafo
> V(g)$label <- V(g)$name
> # Dibuja el grafo usando tkplot(), el cual es muy útil para la
> # manipulación descriptiva del grafo
> #Arreglar estos plots luego meterlos en el latex
> tkplot(g)
```

```
[1] 1
```

```
> plot(g)
```




```
> set.seed(3)
> layout1<-layout.fruchterman.reingold(g)
> #Vemos una representación distinta:
> plot(g,layout=layout1)
```



Capítulo 2

Análisis de frecuencia

El análisis de frecuencia, en este caso de términos o palabras, es una de las técnicas más básicas y usuales en lo que minería de datos se refiere. Introduciremos el concepto de matriz término-documento veremos un ejemplo de su uso en la composición de un wordcloud.

2.1. Matriz término-documento

Una matriz de término-documento representa la relación entre términos y documentos, en nuestro caso palabras y tweets. Cada fila corresponde a un términos y cada columna a un documento. El conjunto de términos que conforman las filas de esta matriz se guardan en un corpus. Pueden usarse como corpus desde un conjunto de palabras que deseamos buscar específicamente, el propio diccionario de una o varias lenguas hasta uno compuesto por el conjunto de palabras de todos los documentos de la colección que estemos estudiando. Por lo tanto el valor de $t_{i,j}$ corresponde al número de ocurrencias del término i en el documento j, donde T=$(t_{i,j})$ es la matriz término-documento.

Ilustramos a continuación la construcción de un corpus y la matriz término-documento(tdm) de un subconjunto de los tweets de Donald Trump en su cuenta @realDonaldTrump obtenidos en Febrero de 2017.

```
> #tweets = userTimeline("realDonaldTrump",n=3200)
> load("Trump.RData")
> length(tweets)
```

```
[1] 756
```

Cabe comentar que aunque hayamos especificado que queremos 3200 mensajes ha cargado bastantes menos, seguramente por temas de la conexión API.

```
> #Volcamos los datos en un data frame
> df <- twListToDF(tweets)
> ##Creación de un corpus:
> library(tm)
> # Hacemos uso de la función Corpus del paquete tm
> myCorpus <- Corpus(VectorSource(df$text))
> # Borramos URLs
> removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
> myCorpus <- tm_map(myCorpus, content_transformer(removeURL))
> # Borramos cualquier otra cosa que palabras en inglés
> # y espacios en blanco
> removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
> myCorpus <- tm_map(myCorpus, content_transformer(removeNumPunct))
> # Borramos palabras vacias de significado (en inglés stopwords)
> myStopwords <- c(setdiff(stopwords('english'), c("a", "my","i")),
+   "use","if","the","a","i","you","it","", "see", "used", "via", "amp")
> myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
> # Borramos los espacios en blanco extras
> myCorpus <- tm_map(myCorpus, stripWhitespace)
> # Hacemos una copia del corpus por si luego se quiere
> # modificar aparte
> myCorpusCopy <- myCorpus
```

Ahora que hemos construido el corpus pasamos a la matriz tdm

```
> tdm <- TermDocumentMatrix(myCorpus,
+               control = list(wordLengths = c(1, Inf)))
> m <- as.matrix(tdm)
> #Calculamos la frecuencia de las palabras y las
> # ordenamos en función de esto
> wordFreq <- function(corpus, word) {
+   results <- lapply(corpus,
+  function(x) { grep(as.character(x), pattern=paste0("\\<",word)) }
```

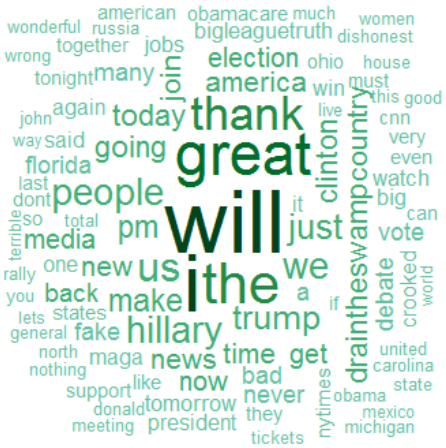
```
+   )
+   sum(unlist(results))
+ }
> term.freq <- rowSums(as.matrix(tdm))
> term.freq <- subset(term.freq, term.freq >= 2)
> freq.df <- data.frame(term = names(term.freq), freq = term.freq)
```

2.2. Wordcloud

Despues de haber construido esta matriz de frecuencia de términos podemos visualizar la importancia de cada uno de ellos mediante una nube de palabras(o en inglés wordcloud). En esta los términos más frecuentes se agrupan en una especie de nube bidimensional en la que el tamaño de cada término indica su importancia o frecuencia. Podemos ver un ejemplo con la cuenta del nuevo presidente de EEUU Donald Trump para ver de qué ha hablado más en su cuenta @realDonaldTrump.

Con este fin haremos uso del paquete wordcloud [6]. Después de calcular una ordenación de las palabras por frecuencias hacemos uso de la función wordcloud, que comparte nombre con el paquete al que pertenece.

```
> library(wordcloud)
> library(tmap)
> library(RColorBrewer)
> #La frecuencia total de las palabras en el conjunto
> # total de mensajes se obtiene sumando por filas
> # en la matriz de ocurrencias.
> word.freq <- sort(rowSums(m), decreasing = T)
> # pal es para definir qué colores usar
> pal <- brewer.pal(9, "BuGn")[-(1:4)]
> wordcloud(words = names(word.freq),max.words =100,
+           freq = word.freq, min.freq = 3,
+           random.order = F, colors = pal)
```



En dicha representación podemos observar términos involucrados en la campaña electoral que Trump desarrolló para las últimas elecciones, en las que salió electo, como por ejemplo:

- Obamacare: seguro médico que Obama quiso implantar, ya que en EEUU no hay nada análogo a la Seguridad Social de España.
- Hillary y Clinton: son el nombre y apellido de su rival directa en la votación final. No es de extrañar que haya hablado mucho de ella.
- México: está siendo un tema muy polémico el del requerimiento de una muralla con México.

Así como palabras de uno de sus mayores esloganes, ”Make America Great Again”.

Capítulo 3

Clustering

Un algoritmo de agrupamiento, o clustering en inglés, es como su nombre indica, una técnica para la organización y clasificación de los datos. Una de las mayores ventajas de su uso es que no necesita información previa para sacar conclusiones. Por esta razón, el clustering ha sido usado tradicionalmente considerándose una técnica de aprendizaje no supervisado. Sin embargo, tambien existen métodos y algoritmos que admiten guías o ajustes.

El análisis cluster es un conjunto de técnicas multivariantes utilizadas para clasificar a un conjunto de individuos en grupos homogéneos. Pertenece, al igual que otras tipologías y que el análisis discriminante al conjunto de técnicas que tiene por objetivo la clasificación de los individuos. La diferencia fundamental entre el análisis cluster y el discriminante reside en que en el análisis cluster los grupos son desconocidos a priori y son precisamente lo que queremos determinar; mientras que en el análisis discriminante los grupos son conocidos y lo que pretendemos es saber en qué medida las variables disponibles nos discriminan esos grupos y nos pueden ayudar a clasificar o asignar los individuos en/a los grupos dados.

Se trata, fundamentalmente, de resolver el siguiente problema: Dado un conjunto de N individuos caracterizados por la información de n variables $\{Xj\}_{j=1,2,...,n}$, nos planteamos el reto de ser capaces de clasificarlos de manera que los individuos pertenecientes a un grupo (cluster) (y siempre con respecto a la información disponible) sean tan similares entre sí como sea posible, siendo los distintos grupos entre ellos tan disimilares como sea posible.

Como puede comprenderse fácilmente el análisis cluster tiene una extraordinaria importancia en la investigación científica, en cualquier rama del saber. Téngase presente que la clasificación es uno de los objetivos fundamentales de la ciencia .Y en la medida en que el análisis cluster nos proporciona los medios técnicos para realizarla, se nos hará imprescindible en cualquier investigación.

Generalmente se dividen las técnicas y métodos de clustering en dos grandes grupos, basándonos en las propiedades de los grupos o clusters. Estos son el clustering particional y el clustering jerárquico. El primero trata de dividir directamente los puntos del conjunto de datos en número preestablecido de grupos sin ninguna estructura más mientras que los jerárquicos organizan los grupos en una secuencia encajada de particiones que van desde un grupo atómico de un único individuo hasta uno que incluya a todos y viceversa.

3.1. Distancias usuales

Como se ha comentado, la parte fundamental de estas técnicas reside en el agrupamiento homogéneo. Por lo tanto, será fundamental establecer un criterio de semejanza u homogeneidad entre individuos o términos.

Empezamos recordando la definición de distancia. Se da, en general, el nombre de distancia o disimilaridad entre dos individuos i y j a una medida, indicada por d(i,j), que mide el grado de semejanza, o a mejor decir de desemejanza, entre ambos objetos o individuos, en relación a un cierto número de características cuantitativa y/o cualitativas. El valor de d(i,j) es siempre un valor no negativo, y cuanto mayor sea este valor mayor será la diferencia entre los individuos i y j. Toda distancia debe verificar, al menos, las siguientes propiedades:

- $d(i,j) \geq 0 \forall i,j$
- $d(i,i) = 0 \forall i$
- $d(i,j) = d(j,i) \forall i,j$ (simetría)

Dada entonces la importancia de la distancia que usemos pasamos a describir algunas de las más utilizadas.

Diremos que una distancia es euclidiana cuando pueda encontrarse un espacio vectorial de dimensión igual o inferior a la dimensión del espacio de las variables en el que podamos representar a los individuos por puntos cuya

distancia euclídea ordinaria coincida con la distancia utilizada.
Es decir si existe un espacio vectorial Rm , con m <n (siendo n el número de variables consideradas para representar a los individuos) y dos puntos de ese espacio, Pi y Pj de coordenadas : $Pi = (Pi_1, Pi_2, \cdots, Pi_m)$ y $Pj = (Pj_1, Pj2, \cdots, Pj_m)$ verificándose que la distancia que estamos considerando entre los individuos i y j es igual a la distancia euclídea entre los puntos Pi y Pj en Rm ; esto es: Si $d(i,j) = |Pi - Pj|$, diremos que la distancia $d(i,j)$es euclidiana. Cuando la distancia es euclidiana se verifica además que:

- $d(i,j) < d(i,t) + d(t,j) \forall i,j,t$
- $d(i,j) > 0$ si $i \neq j$

3.1.1. Distancia Euclidea

$$d(x,y) = d([x_1, x_2, ..., x_n], [y_1, y_2, ..., y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3.1.2. Distancia de Minkowski

Puede considerarse una generalización de la distancia euclidea. La distancia entre dos puntos está definida como sigue:

$$d(x,y) = d([x_1, x_2, ..., x_n], [y_1, y_2, ..., y_n]) = (\sum_{i=1}^{n} (x_i - y_i)^p)^{1/p}$$

3.1.3. Distancia de Chebyshev

Se define dicha distancia como el máximo de la diferencia en valor absoluto de la componentes de los vectores cuya distancia estemos midiendo.

$$d(x,y) = d([x_1, x_2, ..., x_n], [y_1, y_2, ..., y_n]) = \max_{i \in 1..n} |x_i - y_i|$$

3.1.4. Distancia de Mahalanobis

Esta medida de distancia está basada en la correlación que existe entre variables. Formalmente se define la distancia entre un vector $x = (x_1, x_2, ..., x_n)$

y un conjunto de valores con media $\nu = (\nu_1, \nu_2, ..., \nu_n)$ con matriz de covarianzas S como:

$$D_M(x) = \sqrt{(x-\nu)' S^{-1} (x-\nu)}$$

Además, se puede definir la distancia entre dos vectores como:

$$d(x,y) = d([x_1, x_2, ..., x_n], [y_1, y_2, ..., y_n]) = \sqrt{(x-y)' S^{-1} (x-y)}$$

Si la matriz de covarianzas es diagonal, lo cual ocurre si x e y son dos vectores aleatorios independientes, esta distancia tambien es llamada distancia euclidea estandarizada y se expresa como:

$$d(x,y) = d([x_1, x_2, ..., x_n], [y_1, y_2, ..., y_n]) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{s_i^2}}$$

donde s_i son los elementos de la diagonal de S.

3.1.5. Distancia entre vectores binarios

Se puede definir la distancia entre dos vectores binarios $x, y \in \{0,1\}^n$ como la proporción de variables donde un sólo caso es 1 respecto del total de casos donde al menos una variable es 1. Es decir:

$$d_{bin}(x,y) = \frac{\sum_{i=1}^{n}(x_i + y_i)}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i - \sum_{i=1}^{n}(x_i y_i)}$$

3.2. Métodos particionales

A diferencia del clustering jerárquico, éste solo se encarga de dividir el conjunto de datos de puntos en un número k de clusters o grupos en base a un criterio de optimalidad de cierta función. más específicamente, dado un conjunto de vectores $\{x_i\}_{i=1..N} \in \mathcal{R}^d$ buscamos dividirlos en K subconjuntos $\{C_1, ..., C_k\}$ maximizando o minimalizando cierta función J.
En un principio podriamos pensar en enumerar todas las posibles particiones y ver cuál es la óptima, sin embargo, esto es imposible en la práctica dado el alto tiempo de computación, dado por (Liu, 1968):

$$P(N,K) = \frac{1}{K!} \sum_{m=1}^{K} (-1)^{K-m} \binom{K}{m} m^N$$

Incluso para conjuntos pequeños de datos, como 30 objetos, y un número reducido de clusters, digamos 3, el número de particiones es de $2{,}10^{14}$. Por esto, se usan métodos heurísticos para obtener aproximaciones plausibles.

3.2.1. Método k-means

El algoritmo de k-means o k-medias (Forgy, 1965; MacQueen, 1967) es uno de los más usados y conocidos para hacer clustering. La idea básica reside en la búsqueda de una partición óptima que minimice el criterio de la suma del cuadrado de los errores. Supangamos que tenemos un conjunto de vectores $x_j \in \mathcal{R}^d \ \forall j \in \{1, ..., N\}$ buscamos dividirlos en K subconjuntos $\{C_1, ..., C_k\}$, entonces dicho criterio se expresa mediante la función:

$$J_s(\Gamma, M) = \sum_{i=1}^{K} \sum_{j=1}^{N} \gamma_{ij} |x_j - m_i|^2 = \sum_{i=1}^{K} \sum_{j=1}^{N} \gamma_{ij} (x_j - m_i)^T (x_j - m_i)$$

donde
$\Gamma = \{\gamma_{ij}\}$ es la matrix de partición, en la que $M = [m_1, ..., m_K]$ es la matriz de medias o prototipo cluster; y $m_i = \frac{1}{N_i} \sum_{j=1}^{N} \gamma_{ij} x_j$ es la media muestral para el grupo o cluster i-ésimo con N_i objetos.
Con esto, el método k-means puede resumirse como:

1. Inicializamos con una K-partición aleatoria. Obviamente conseguiremos convergencias más rápidas si usamos conocimiento previo para formar esta partición. A continuación calculamos la matriz de grupos prototípica $M = [m_1, \cdots, m_K]$;

2. Asignamos cada objeto x_j al grupo más cercano C_l. Es decir:

$$x_j \in C_l, if |x_j - m_l| < |x_j - m_i| \forall j = 1, ..., N i \neq l, y i = 1, \cdots, K;$$

3. Recalculamos la matriz prototipo basada en la actual partición,

$$m_i = \frac{1}{N_i} \sum_{x_j \in C_i} x_j;$$

4. Repetimos los pasos 2 y 3 hasta que no haya cambios en la partición, es decir, que converja el algoritmo.

Notemos que la partición que se hace en el paso 2 se trata de una partición del tipo diagrama de Voronoi en los que los centroides son los puntos que definen las células. Dicho diagrama se trata de una partición del espacio en la que cada punto de una determinada región Voronoi está más cerca de vector de Voronoi, en este caso los centroides, que del resto.

Ilustramos con un ejemplo dicho proceso. Usaremos los datos de la famosa cuenta de wikileaks, pretendiendo agrupar los temas más relevantes de los que se ha hablado. Lo primero como siempre será extraer los datos, formar un corpus y con este una matriz término-documento:

```
> library(twitteR)
> #wikileaks=userTimeline("wikileaks",3200)
> load("wikileaks.RData")
> df=twListToDF(wikileaks)
> date<-Sys.Date()
> date<-as.character(date)
> name<-paste(date,".RData")
> #Guardamos en .RData
> save(df, file =name)
> ##Creación de un corpus:
>
> library(tm)
> # build a corpus, and specify the source to
> # be character vectors
> myCorpus <- Corpus(VectorSource(df$text))
> # convert to lower case
> myCorpus <- tm_map(myCorpus,
+                    content_transformer(tolower))
> # remove URLs
> removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
> myCorpus <- tm_map(myCorpus,
+                    content_transformer(removeURL))
> # remove anything other than
> # English letters or space
> removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*",
+                                    "", x)
> myCorpus <- tm_map(myCorpus,
```

```
+                    content_transformer(removeNumPunct))
> # remove stopwords
> myStopwords <- c(setdiff(stopwords('english'),
+                          c("a", "my","i")),"use","if",
+                  "the","a","i","you","it","", "see",
+                  "used", "via", "amp")
> myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
> # remove extra whitespace
> myCorpus <- tm_map(myCorpus, stripWhitespace)
> # keep a copy for stem completion later
> myCorpusCopy <- myCorpus
> #Construimos matriz:
> tdm <- TermDocumentMatrix(myCorpus,
+             control = list(wordLengths = c(1, Inf)))
> m <- as.matrix(tdm)
> #Eliminación de términos dispersos
> tdm2 <- removeSparseTerms(tdm, sparse=0.95)
> #Crea una sparse=máxima dispersión permitida en el rango.
> #Se usa para quitar elementos sin interés.
> #head(inspect(tdm2))
> m2 <- as.matrix(tdm2)
> # Transponemos la matriz
> m3 <- t(m2)
> ##Algorítmo de las k-medias
>
> #Fijamos semilla
>
> set.seed(122)
> # Grupos por el algorítmo de las k-medias
> # Tomamos k=7
>
> k <- 7
> kmeansResult <- kmeans(m3, k)
> # Centros
>
> round(kmeansResult$centers, digits=3)

  assange   cia  cias comey director election   fbi french  full james macron
```

```
1   0.150 0.050 0.100 0.000        0    0.000 0.000  0.000 0.000 0.000  0.100
2   0.143 0.143 0.000 1.000        0    0.000 0.714  0.000 0.143 0.714  0.000
3   0.000 0.000 0.000 0.000        0    0.000 0.000  0.000 0.333 0.667  0.000
4   0.000 0.000 0.000 0.667        1    0.167 1.000  0.000 0.333 0.833  0.000
5   0.000 0.250 0.083 0.000        0    0.167 0.000  0.000 0.000 0.083  0.083
6   0.105 1.053 0.105 0.000        0    0.105 0.000  0.158 0.105 0.000  0.158
7   0.077 0.000 0.038 0.000        0    0.058 0.000  0.077 0.019 0.000  0.096
  macronleaks media   now obama today trump us vault  vote wikileaks
1       0.150 0.100 0.100 0.000 0.000 0.050  0 0.000 0.000     1.050
2       0.000 0.000 0.000 0.000 0.000 0.286  0 0.000 0.000     0.429
3       0.000 0.000 0.000 1.000 1.000 0.333  0 0.000 0.000     0.333
4       0.000 0.000 0.500 0.000 0.000 0.000  0 0.000 0.000     0.000
5       0.083 0.000 0.083 0.000 0.167 0.000  1 0.000 0.000     0.417
6       0.000 0.053 0.000 0.158 0.053 0.053  0 0.263 0.053     0.000
7       0.231 0.077 0.077 0.000 0.000 0.019  0 0.038 0.096     0.000
```

```
> #Revisamos tres primeras palabras de cada cluster
> for (i in 1:k) {
+   cat(paste("cluster ", i, ": ", sep=""))
+   s <- sort(kmeansResult$centers[i,], decreasing=T)
+   cat(names(s)[1:3], "\n")
+   # Imprimir tweets de cada cluster
+   # print(rdmTweets[which(kmeansResult$cluster==i)])
+ }
```

```
cluster 1: wikileaks assange macronleaks
cluster 2: comey fbi james
cluster 3: obama today james
cluster 4: director fbi james
cluster 5: us wikileaks cia
cluster 6: cia vault french
cluster 7: macronleaks macron vote
```

Con esto podemos determinar que existen un par de temas con mayor importancia: mensajes sobre Emmanuel Macron, el nuevo presidente de Francia y sobre James Comey, el séptimo director del FBI de US.

3.2.2. Método k-medoids

En este método de k-medoids o k-mediodes se buscan k puntos representativos de k conglomerados, llamados mediodes, de que modo que la suma de las distancias de los puntos a su mediode más cercano sea mínima. Es decir, la función objetivo será:

$$\min_{m_1,\ldots,m_k} \sum_{i=1}^{n} \min_{t=1,\ldots,k} d(i, m_t)$$

Cada caso será entonces asignado al mediode más cercano, formando así k clusters.

Se trata de un método más robusto que el k-means ya que es menos sensible a casos atípicos.

```
> ##Algorítmo de k-medoids
> library(fpc)
> # Estimar el número de grupos
> pamResult <- pamk(m3, metric="manhattan")
> #pamk devuelve: un objeto pam, el nº óptimo
> # de grupos y p-values del test Duda-Hart.
>
> pamResult$pamobjec
```

```
Medoids:
     ID assange cia cias comey director election fbi french full james macron
5     5       0   0    0     1        1        0   1      0    0     1      0
117 117       0   0    0     0        0        0   0      0    0     0      0
118 118       0   0    0     0        0        0   0      0    0     0      0
115 115       0   1    0     0        0        0   0      0    0     0      0
59   59       0   0    0     0        0        0   0      0    0     0      0
36   36       0   1    0     0        0        1   0      1    0     0      1
86   86       0   0    1     0        0        0   0      0    1     0      0
    macronleaks media now obama today trump us vault vote wikileaks
5             0     0   0     0     0     0  0     0    0         0
117           0     0   0     0     0     0  0     0    0         0
118           0     0   0     0     0     0  0     0    0         1
115           0     0   0     0     0     0  0     0    0         0
59            1     0   0     0     0     0  0     0    0         0
```

```
36           0     0   0     1     0     0  0     0    0         0
86           0     0   0     0     0     0  0     1    0         0
Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  2
  1   2   3   1   1   1   1   1   1   1   1   1   4   3   2   4   4   4   5
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  4
  5   2   2   2   2   2   3   3   2   2   2   3   3   6   5   6   5   2   2
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  6
  2   5   3   5   3   5   2   2   2   5   3   5   5   5   6   5   5   5   5
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  8
  2   2   4   2   7   4   3   4   2   4   4   4   6   3   3   2   2   3   4
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 10
  3   4   2   2   2   7   7   4   4   3   3   2   4   2   2   2   2   3   3
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
  4   4   3   2   2   2   2   3   2   3   2   2   6   2   4   2   2   3   4
Objective function:
    build      swap
1.0000000 0.9915966

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
 [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"

> #PAM:Partitioning Around Medoids.
> #El objeto pam contiene: centroides, id de los
> #centroides, el vector que indica a que grupo
> #pertence cada término, el valor de la función
> #objetivo después del primer y segundo paso del algorítmo.
> #Available components te indica que más información
> #puedes saber del objeto pam:
> #https://stat.ethz.ch/R-manual/R-devel/library/
> #cluster/html/pam.object.html.
>
> #El nº óptimo de grupos
> k <- pamResult$nc
> k

[1] 7
```

```
> #P-values test Duda-Hart: indica el nº posible de grupos.
> #En este caso, el mayor p-value es el segundo.
> #La hipótesis nula se refiere a considerar homogeneidad
> #con 1 grupo, con 2,...
>
> pamResult$crit

 [1] 0.0000000 0.2442855 0.2826291 0.3271309 0.3205107 0.3558933 0.3702369
 [8] 0.3687245 0.3650473 0.3690415

> pamResult <- pamResult$pamobject
> pamResult

Medoids:
     ID assange cia cias comey director election fbi french full james macron
5     5       0   0    0     1        1        0   1      0    0     1      0
117 117       0   0    0     0        0        0   0      0    0     0      0
118 118       0   0    0     0        0        0   0      0    0     0      0
115 115       0   1    0     0        0        0   0      0    0     0      0
59   59       0   0    0     0        0        0   0      0    0     0      0
36   36       0   1    0     0        0        1   0      1    0     0      1
86   86       0   0    1     0        0        0   0      0    1     0      0
    macronleaks media now obama today trump us vault vote wikileaks
5             0     0   0     0     0     0  0     0    0         0
117           0     0   0     0     0     0  0     0    0         0
118           0     0   0     0     0     0  0     0    0         1
115           0     0   0     0     0     0  0     0    0         0
59            1     0   0     0     0     0  0     0    0         0
36            0     0   0     1     0     0  0     0    0         0
86            0     0   0     0     0     0  0     1    0         0
Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
  1   2   3   1   1   1   1   1   1   1   1   1   4   3   2   4   4   4   5   2
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  5   2   2   2   2   2   3   3   2   2   2   3   3   6   5   6   5   2   2   2
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  2   5   3   5   3   5   2   2   2   5   3   5   5   5   6   5   5   5   5   2
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
```

```
  2   2   4   2   7   4   3   4   2   4   4   4   6   3   3   2   2   3   4
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 10
  3   4   2   2   2   7   7   4   4   3   3   2   4   2   2   2   2   3   3
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
  4   4   3   2   2   2   2   3   2   3   2   2   6   2   4   2   2   3   4
Objective function:
    build      swap
1.0000000 0.9915966

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
 [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"

> # Imprimir los centroides
> pamResult$medoids

     Terms
Docs  assange cia cias comey director election fbi french full james macron
  5         0   0    0     1        1        0   1      0    0     1      0
  117       0   0    0     0        0        0   0      0    0     0      0
  118       0   0    0     0        0        0   0      0    0     0      0
  115       0   1    0     0        0        0   0      0    0     0      0
  59        0   0    0     0        0        0   0      0    0     0      0
  36        0   1    0     0        0        1   0      1    0     0      1
  86        0   0    1     0        0        0   0      0    1     0      0
     Terms
Docs  macronleaks media now obama today trump us vault vote wikileaks
  5             0     0   0     0     0     0  0     0    0         0
  117           0     0   0     0     0     0  0     0    0         0
  118           0     0   0     0     0     0  0     0    0         1
  115           0     0   0     0     0     0  0     0    0         0
  59            1     0   0     0     0     0  0     0    0         0
  36            0     0   0     1     0     0  0     0    0         0
  86            0     0   0     0     0     0  0     1    0         0

> for (i in 1:k) {
+   cat(paste("cluster", i, ": "))
+   cat(colnames(pamResult$medoids)
```

```
+       [which(pamResult$medoids[i,]==1)], "\n")
+ }
```

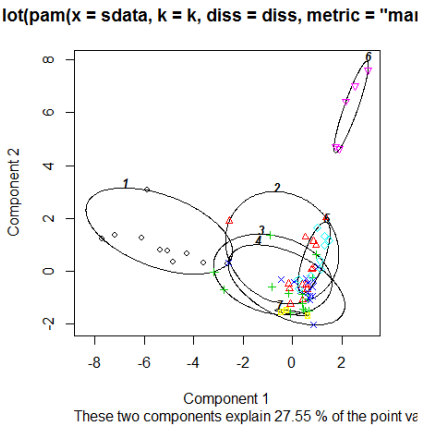
```
cluster 1 : comey director fbi james
cluster 2 :
cluster 3 : wikileaks
cluster 4 : cia
cluster 5 : macronleaks
cluster 6 : cia election french macron obama
cluster 7 : cias full vault
```

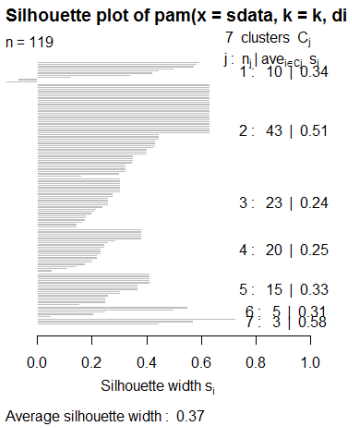
```
> # Gráfico
>
> par(mfrow=c(1,2))
> plot(pamResult, color=F, labels=4,
+      lines=0, cex=.8, col.clus=1,
+      col.p=pamResult$clustering)
> #Para cada observación i, el ancho de silueta s(i)
> #se define de la siguiente manera:
> #a(i) = disimilitud media entre i y todos los
> #  demás puntos del grupo al que i pertenece.
> #Para todos los otros grupos C, poner d(i, C)
> #= disimilaridad media de i a todas las observaciones de C.
> #El menor de estos d(i, C) es b(i) y puede
> #verse como la disimilitud entre i y su agrupación
> #"vecina", es decir, la más cercana a la que no pertenece.
> #Por tanto,
> #s(i) := ( b(i) - a(i) ) / max( a(i), b(i) ).
> #Las observaciones con una gran s(i) (casi 1)
> #están muy bien agrupadas,
> #una pequeña s(i) (alrededor de 0) significa
> q#ue la observación se encuentra entre dos grupos,
```

```
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<bytecode: 0x000000003b79ea58>
<environment: namespace:base>
```

```
> #y las observaciones con un negativo s(i)
> #están en un grupo incorrecto.
```

```
>
> library(cluster)
> summary(silhouette(pamResult)) #RStudio no dibuja la silueta

Silhouette of 119 units in 7 clusters from pam(x = sdata, k = k, diss = diss,
 Cluster sizes and average silhouette widths:
       10        43        23        20        15         5         3
0.3393578 0.5097617 0.2418872 0.2478013 0.3284608 0.3114018 0.5805029
Individual silhouette widths:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.1163  0.2467  0.3515  0.3702  0.5001  0.7261
```





Podemos ver primero un gráfico de 2 dimensiones (2 componentes principales) las distribución de los puntos, que representan cada uno de los mensajes, y su agrupamientos en clusters con recintos en forma de elipses. Por otra parte vemos tambien el gráfico de barras que representa la distancia entre mensaje y grupos.

3.3. Métodos jerárquicos

Los métodos jerárquicos de clustering persiguen contruir una estructura del conjunto de puntos estableciendo una jerarquía entre grupos. Podemos decir entonces que dichos métodos se dividen en dos grandes grupos: aglomerativos y divisivos. Mientras que los primeros van de una mayor a menor homogeneidad los segundos se establecen al contrario, de menor a mayor homogeneidad. Pero no solo nos referimos a la cuestión estética de la orientación

del árbol obtenido sino que el algoritmo utilizado es basicamente distinto.

básicamente en los del tipo aglomerativo cada observación comienza en su propio cluster y posteriormente los pares de grupos van siendo mezclados conforme se sube en la jerarquía. A diferencia de los del tipo anterior los divisivos comienzan con todos los puntos en un único cluster, el cual se va diviendo conforme bajamos niveles de jerarquía

Los dendrogramas se usan para la representación en forma de árbol de la jerarquía de los clusters anteriormente obtenida. En esta, las sucesiones fusiones de ramas a distintos niveles nos indican el grado de homogeneidad que comparten teniendo en cuenta que a mayor nivel menor homogeneidad.

Permite apreciar muy bien las relación de agrupación entre individuos aunque en contra no se apreciará bien las relaciones de cercanía o similaridad entre categorías o grupos.

3.3.1. Algoritmo aglomerativo

Para describir este algoritmo usaremos el concepto de matriz de proximidad entre clusters. Sea $\{\mathcal{C}_1, ..., \mathcal{C}_k\}$ un conjunto de k clusters se define la matriz de proximidad de dicho conjunto como $M_P = [m_{i,j}]$ con $m_{i,j} = D(\mathcal{C}_i, \mathcal{C}_j)$ y $D(\mathcal{C}_i, \mathcal{C}_j)$ la distancia entre cluster anteriormente descrita. Cabe destacar que dicha matriz es simétrica y de diagonal 0.
Pasamos a describir entonces el algoritmo:

1. Comenzamos con N clusters formados por las N palabras o puntos. Calculamos la matriz de proximidad para estos; NA

2. Actualizamos la matriz de proximidad entre clusters;

3. Repetimos los pasos 2 y 3 hasta que nos quede un único cluster.

En el caso general la complejidad del algoritmo será de $\mathcal{O}(N^3)$, lo cual ya es bastante intenso computacionalmente.

Unión de clusters

Como hemos observado en el paso 2 del algoritmo anterior la unión de los clusters es lo que determina la estructura final. Es por esto que depende entonces de la definición de distancia entre clusters. Procederemos a explicar e ilustrar las distancias que dan lugar a los métodos más usuales.

1. Ward: Con este se pretende determinar conglomerados esféricos. Ward argumentó que los conglomerados debían contruirse de tal forma que la pérdida de información al fundirse dos clusters fuera mínima. Dicha cantidad de información se cuantifica como la suma de las distancias al cuadrado de cada elemento respecto del centroide de su cluster. Por lo tanto en casa paso se unirán los clusters que den lugar a un menor incremento de la SCE, la cual se define como:

$$SCE = \sum_{g=1}^{k} \sum_{i \in g} |x^i - m_i|^2 = tr(W)$$

   donde k es el número de clusters, $sum_{i \in g}(x^i)$ se refiere al sumatorio de todos los vectores que pertenecen al cluster, m_i el centroide del cluster o conglomerado i y W la matriz de suma de cuadrados y productos dentro de los grupos.

2. Single: está muy relacionado con el problema del mínimo árbol de unión (minimal spanning tree) y es sensible a posibles encadenamientos. La medida de distancias entre los clusters $\mathcal{C}_l$ y $\mathcal{C}_{ij}$ (donde el cluster $\mathcal{C}_{ij}$ el que resulta de la unión de $\mathcal{C}_i$ y $\mathcal{C}_j$)que define el método viene dada por:

$$D(\mathcal{C}_l, \mathcal{C}_{ij}) = min(D(\mathcal{C}_l, \mathcal{C}_i), D(\mathcal{C}_l, \mathcal{C}_j))$$

3. Complete: busca formar cluster cuyos elementos sean muy parecidos entre si. La distancia se forma de manera homóloga al método single, pero con el max:

$$D(\mathcal{C}_l, \mathcal{C}_{ij}) = max(D(\mathcal{C}_l, \mathcal{C}_i), D(\mathcal{C}_l, \mathcal{C}_j))$$

4. Average (WPGMA): la distancia se define como el promedio de la distancia entre los grupos de puntos que van formando cada cluster, es decir:

$$D(\mathcal{C}_l, \mathcal{C}_{ij}) = \frac{1}{2}(D(\mathcal{C}_l, \mathcal{C}_i) + D(\mathcal{C}_l, \mathcal{C}_j))$$

5. Weighted average (WPGMA): la idea es la similara la del método anterior, solo que le otorgamos pesos en función de la cantidad de puntos de cada cluster a la hora de hacer el promedio, es decir:

$$D(\mathcal{C}_l, \mathcal{C}_{ij}) = \frac{n_i}{n_i + n_j} D(\mathcal{C}_l, \mathcal{C}_i) + \frac{n_j}{n_i + n_j} D(\mathcal{C}_l, \mathcal{C}_j)$$

donde n_i es el número de puntos que contiene el cluster i.

6. Centroid (UPGMC): este método basa la unión de los clusters en función de sus centroides. Puede producir inversiones(distancias de unión que son menores en una etapa que en la anterior) lo cual dificulta la interpretación. Es por ellos que definiremos la distancia entre clusters como:

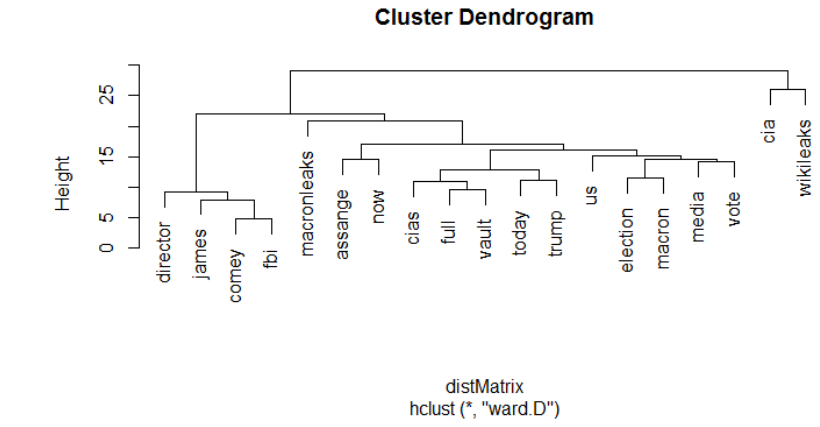
$$D(\mathcal{C}_l, \mathcal{C}_{ij}) = |m_l - m_{ij}|^2$$

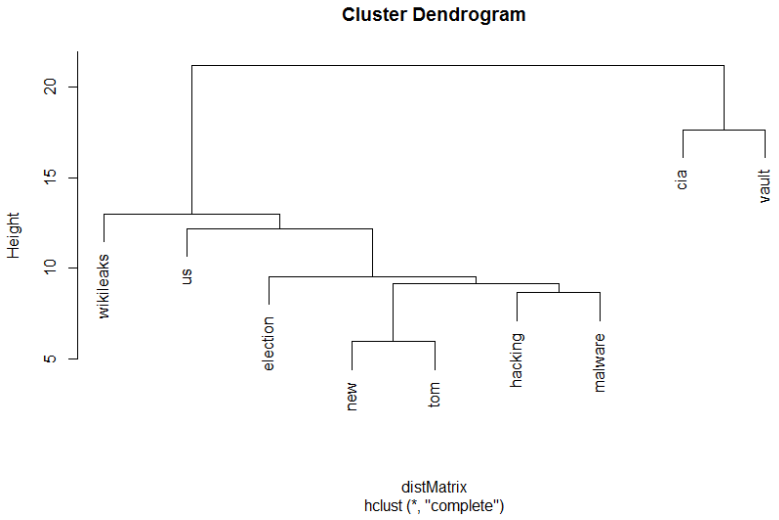
7. Median (WPGMC)= Es similiar a la anterior, siendo la distancia definida como:

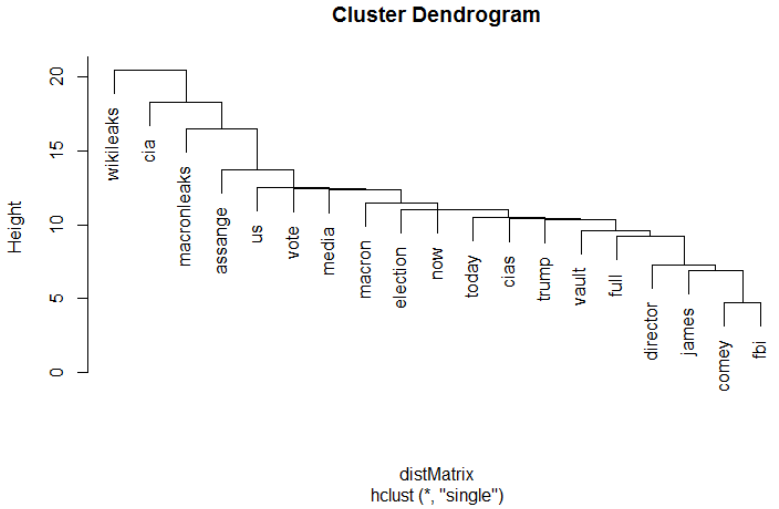
$$D(\mathcal{C}_l, \mathcal{C}_{ij}) = \frac{1}{2} D(\mathcal{C}_l, \mathcal{C}_i) + \frac{1}{2} D(\mathcal{C}_l, \mathcal{C}_j) - \frac{1}{4} D(\mathcal{C}_i, \mathcal{C}_j)$$

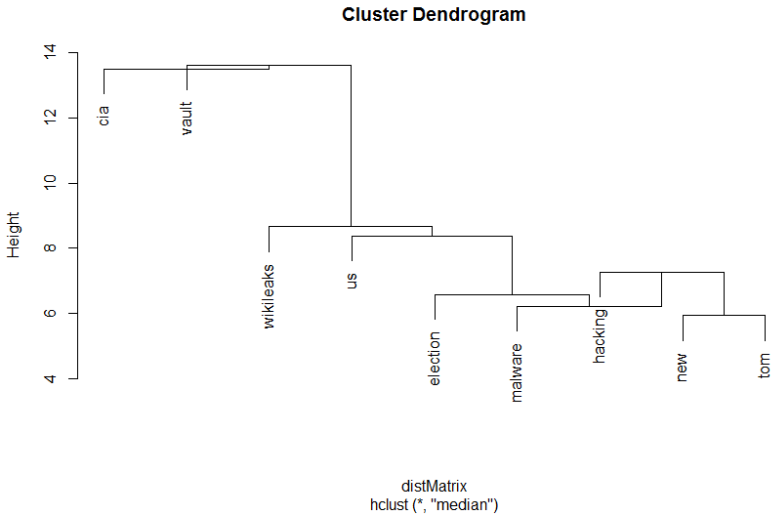
Pasamos entonces a ilustrarlo de nuevo con los datos de la cuenta @wikileaks. Se observa que el método usado se elige en el parámetro method de la función hclust:

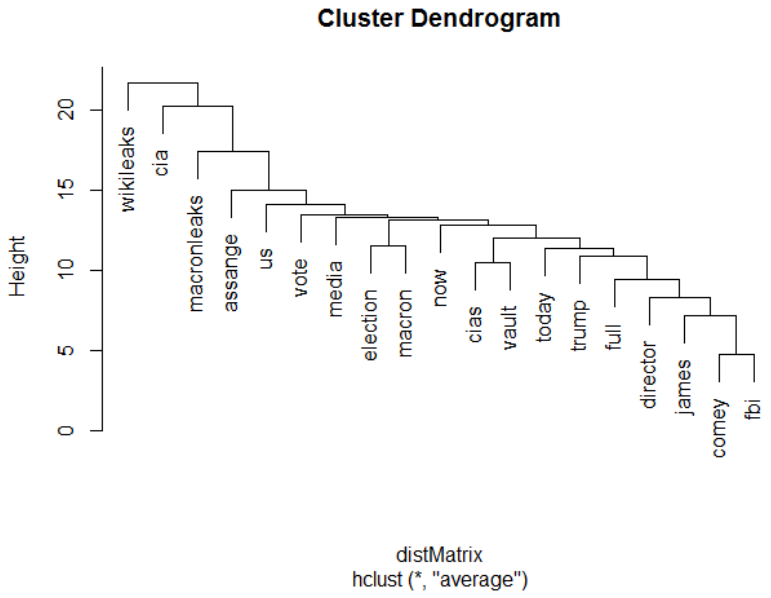
```
> #Creación de grupos
> distMatrix <- dist(scale(m2)) #Distancias entre los términos tipificados
> fit <- hclust(distMatrix, method="ward.D") #Se hacen clusters.
> plot(fit)#Dendrograma
> fitcomp <- hclust(distMatrix, method="complete")
> plot(fitcomp)#Dendrograma
> fitsingle <- hclust(distMatrix, method="single")
> plot(fitsingle)#Dendrograma
> fitmedian <- hclust(distMatrix, method="median")
> plot(fitmedian)#Dendrograma
> fitaver <- hclust(distMatrix, method="average")
> plot(fitaver)#Dendrograma
> fitcentroid <- hclust(distMatrix, method="centroid")
> plot(fitcentroid)#Dendrograma
> fitmc <- hclust(distMatrix, method="mcquitty")
> plot(fitmc)#Dendrograma
```

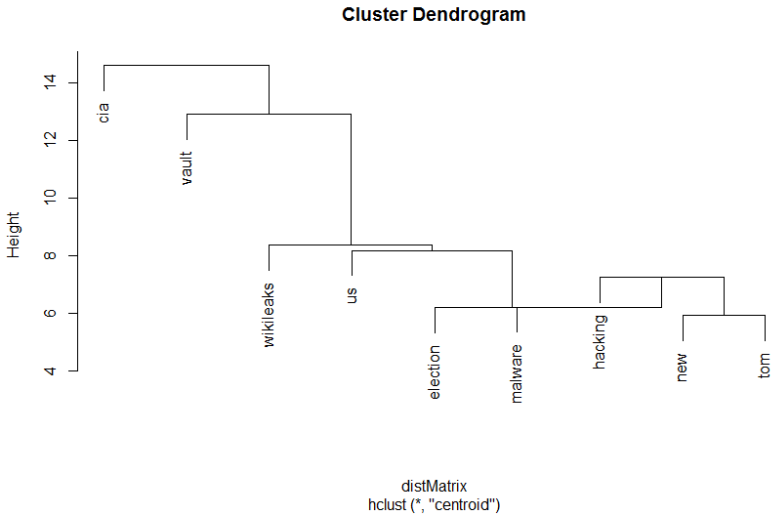

Cluster Dendrogram
Height
0
5
15
25
director
james
comey
fbi
macronleaks
assange
now
cias
full
vault
today
trump
us
election
macron
media
vote
cia
wikileaks
distMatrix
hclust (*, "ward.D")

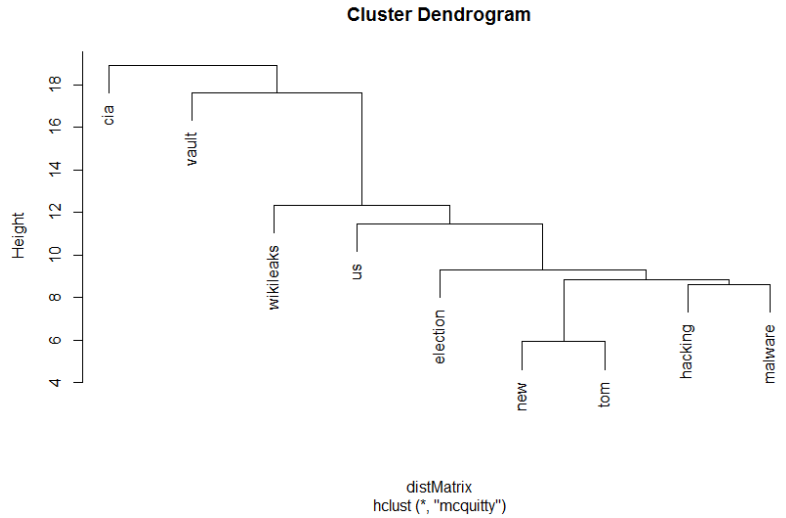

Cluster Dendrogram
Height
5
10
15
20
wikileaks
us
election
new
tom
hacking
malware
cia
vault
distMatrix
hclust (*, "complete")


Cluster Dendrogram
Height
0
5
10
15
20
wikileaks
cia
macronleaks
assange
us
vote
media
macron
election
now
today
cias
trump
vault
full
director
james
comey
fbi
distMatrix
hclust (*, "single")


Cluster Dendrogram
Height
4
6
8
10
12
14
cia
vault
wikileaks
us
election
malware
hacking
new
tom
distMatrix
hclust (*, "median")


Cluster Dendrogram
Height
0
5
10
15
20
wikileaks
cia
macronleaks
assange
us
vote
media
election
macron
now
cias
vault
today
trump
full
director
james
comey
fbi
distMatrix
hclust (*, "average")


Cluster Dendrogram
Height
4
6
8
10
12
14
cia
vault
wikileaks
us
election
malware
hacking
new
tom
distMatrix
hclust (*, "centroid")



Vemos que el método usado varia significativamente los resultados así como la representación de estos. Sin embargo, los temas en los que se separa el contenido por ramas son parecidos a los que obtuvimos con el k-means.

3.3.2. Algoritmos divisivos

Este método procede de forma inversa al anterior. Se parte de un único cluster que irá siendo dividido hasta llegar a clusters atómicos. Definimos primero el diámetro de un conjunto como la máxima distancia entre dos puntos de dicho conjunto: $diam(\mathcal{C}_i) = max\{d(x_j, x_l)\}\ \forall x_j, x_l \in \mathcal{C}_i$ En resumen el algoritmo se describe como:

- Comenzamos con un cluster $\mathcal{C}_l$ que contenga a todos los puntos, el cual se supone que se dividirá en otros dos $\mathcal{C}_\rangle$ y $\mathcal{C}_j$. Pues bien, en la primera etapa tomaremos $\mathcal{C}_l = \mathcal{C}_i$ y $\mathcal{C}_j$ vacío;
- Para cada uno de los datos $x_m \in \mathcal{C}_i$,
  - Para la primera iteración computamos su distancia promedio a todo el resto de objetos:

$$d(x_m, \mathcal{C}_i\ x_m) = \frac{1}{N_{Ci} - 1} \sum_{x_p \in \mathcal{C}_i} d(x_m, x_p);$$

- Para las iteraciones restantes computamos la diferencia entre la distancia promedio a $\mathcal{C}_\rangle$ y la distancia a $\mathcal{C}_|$:

$$d(x_m, \mathcal{C}_i\, x_m) - d(x_m, \mathcal{C}_j) = \frac{1}{N_{Ci} - 1} \sum_{x_p \in \mathcal{C}_i} d(x_m, x_p) - \frac{1}{N_{Cj} - 1} \sum_{x_q \in \mathcal{C}_j} d(x_m, x_q);$$

- Para la primera iteración movemos los datos con valores máximos a $\mathcal{C}_j$
- Para el resto de iteraciones, si el valor máximo de la ecuación anterior es mayor que cero, movemos los datos con la máxima diferencia a $\mathcal{C}_j$. Repetir el paso 2 b) y 3 b). En caso contrario, se para el algoritmo.

En comparación a los aglomerativos el coste computacional es mucho mayor por lo que no es generalmente muy usado. Al empezar se empieza por considerar las $2^{N-1} - 1$ posibles particiones en dos subconjuntos no vacíos, lo cual ya es bastante pesado de computar. En el caso general este algoritmo tiene complejidad $\mathcal{O}(2^N)$.

Capítulo 4

Análisis de sentimientos

Introduciremos ahora una nueva técnica, muy importante y en relativo auge en la actualidad. Se trata del análisis de sentimientos o minería de opinión. Con esta pretendemos principalmente extraer el sentimiento que subyace en un documento (en nuestra posterior aplicación tomaremos cierto conjunto de tweets). Con este fin se hará uso de herramientas NLP (programación neurolingüística), análisis de texto, lingüística computacional...

La tarea básica será conocer la polaridad de un texto, es decir, si la opinión expresada se considera positiva, negativa o neutral. Además, tambien se puede profundizar intentando clasificar los documentos según los estados emocionales tales como la ira, alegría, tristeza...

Es necesario comentar el alcance de dichas técnicas pues como es de suponer no son del todo precisas ya que una emoción depende de muchos factores como el contexto del mensaje, el idioma o región del autor, la ironía, sarcasmo... De hecho, ni siquiera los humanos nos ponemos de acuerdo en lo que a juzgar el sentimiento de un documento respecta. Numerosos estudios muestran que el porcentaje de acuerdo es tan sólo de $70 - 80\,\%$. Es por eso que tomaremos muchos de los resultados obtenidos de manera más o menos indicativa, sabiendo que pueden ser erróneos.

Por contextualizar un poco el uso de dichas técnicas dirtemos que es una herramienta imprescindible para conocer las opiniones de los consumidores en torno a ciertos productos necesarias para realizar campañas de marketing adecuacuadas, para hacer predicciones de bolsa o incluso para las prediccio-

nes en intención de voto en elecciones políticas. El equipo de Barack Obama se centró bastante en estos estudios, llegando incluso a impulsar las investigaciones en este campo.

4.1. Análisis de polaridad

En esta sección estaremos interesados en saber si un comentario es positivo o negativo, es decir, en conocer su polaridad. De esta forma no estaremos analizando en sí el sentimiento del mensaje o documento sino que simplemente le asignaremos una puntuación en función de las palabras utilizadas. Esto hace que, por ejemplo, no sea capaz de reconocer una ironía asignandole una puntuación positiva cuando el usuario estaba realizando una crítica.

Dicha puntuación se realizará en base a dos grandes bases de datos, una de palabras que normalmente se utilizan en un sentido positivo y otro para las utilizadas negativamente. Está claro que dichas bases de datos son una parte fundamental del proceso, afectando directamente al resultado. Es por esto que para ilustrarlo con un ejemplo usaremos las listas almacenadas en "positive.RDataz "negative.RData", las cuales están en ingés y son utilizadas para estudios internacionales.

Una vez cargadas estas bases procedemos a puntuar cada uno de los documentos. Se recorrerá cada mensaje palabra a palabra observando las ocurrencias de dichas palabras en las listas de positivas y negativas. Por cada palabra que aparezca en la base de las positivas otorgaremos un punto a la puntuación mientras que para cada una que aparezca en las negativas restaremos un punto a la puntuación.

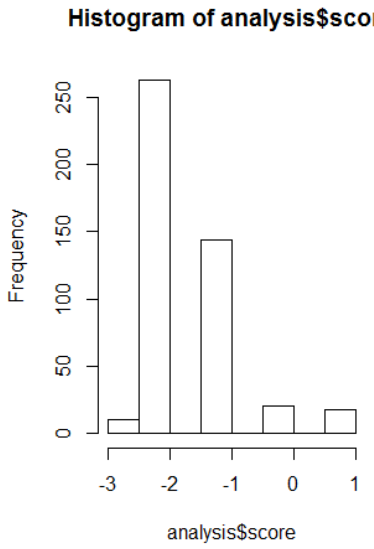
Finalmente obtenemos un análisis del conjunto de mensajes o tweets, análizando la puntuación de cada uno y obteniendo su media, mediana e histograma.

4.1.1. Aplicación

Con motivo de la polémica que causa la corrupción de dirigentes políticos en España nos proponemos estudiar la opinión de usuarios angloparlantes respecto del tema. Recogeremos los mensajes que contienen las palabras "Spainz çorruption", es español España y corrupción y analizaremos los resultados de polaridad.

```
> #spcorruption<-searchTwitter("Spain+corruption",n=3000, lang="en")
> #save(file="spcorruption.RData",spcorruption)
> load("spcorruption.RData")
> length(spcorruption)

[1] 455

> tweets.text = sapply(spcorruption, function(t)t$getText())
> pos = scan("positive.txt", what="character",comment.char=";")
> neg = scan("negative.txt", what="character",comment.char=";")
> score.sentiment =
+   function(sentences, pos.words, neg.words, .progress="none")
+ {
+   library(plyr)
+   library(stringr)
+   # we want a simple array of scores back,
+   #so we use "l" + "a" + "ply" = laply:
+   scores = laply(sentences, function(sentence, pos.words, neg.words) {
+     # clean up sentences with R's regex-driven global substitute, gsub():
+     sentence = gsub("[[:punct:]]", '', sentence)
+     sentence = gsub("[[:cntrl:]]", '' , sentence)
+     sentence = gsub("\\d+", '', sentence)
+     # and convert to lower case:
+     sentence = tolower(sentence)
+
+     # split into words. str_split is in the stringr package
+     word.list = str_split(sentence, "\\s+")
+     # sometimes a list() is one level of hierarchy too much
+     words = unlist(word.list)
+
+     # compare our words to the dictionaries of positive & negative terms
```

```
+     pos.matches = match(words, pos.words)
+     neg.matches = match(words, neg.words)
+     # match() returns the position of the matched term or NA
+     # we just want a TRUE/FALSE:
+     pos.matches = !is.na(pos.matches)
+     neg.matches = !is.na(neg.matches)
+
+     # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
+     score = sum(pos.matches)-sum(neg.matches)
+
+     return(score)
+   }, pos.words, neg.words, .progress=.progress )
+
+   scores.df = data.frame(score=scores, text=sentences)
+   return(scores.df)
+ }
> analysis = score.sentiment(tweets.text, pos, neg, .progress="none")
> table(analysis$score)


 -3  -2  -1   0   1
 10 263 144  20  18


> mean(analysis$score)


[1] -1.498901


> median(analysis$score)


[1] -2


> hist(analysis$score)
```



Con esto vemos que los mensajes relativos a la corrupcion de España no han sido nada positivos. Esto se debe al gran número de casos que sufrieron las instituciones públicas en los últimos años. Se comprueba entonces que ha habido una difusión en medios internacionales de dichos problemas pues hemos tomado tweets en inglés. Resultando una media de -1.4 aproximadamente y observando dicho histrograma definitivamente deberiamos plantearnos un problema de opinión pública aquí.

4.2. Algoritmos de clasificación

Tal y como se ha dicho anteriormente los métodos de clasificación tienen gran importancia en lo que a análisis de sentiemientos respecta. Es por esto explicaremos la base matemáticas de tres de los algoritmos de clasificación más importantes hasta el momento.

Introduciremos después una aplicación de estos y una propuesta para estudios posteriores.

4.2.1. Clasificador Naives-Bayes

Constituye una técnica supervisada en la cual a partir de un conjunto de datos de entrenamiento lograremos realizar predicciones para nuevas entradas de datos test.
Se trata de un método de clasificación basado en uno de los teoremas más famosos del matemático Thomas Bayes. Recordamos que este se utiliza para calcular la probabilidad de algo en base a probabilidades anteriormente calculadas. Dicho teorema dice que la probabilidad de que ocurra el suceso A dada la ocurrencia del suceso B se determina como sigue:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4.2.1}$$

El término naives que usualmente se incluye en el nombre del método hace referencia al término ïngenuo". Esto se debe a que estamos suponiendo la independencia entre las variables predictoras, entre algunas otras hipótesis simplificadoras.

Para el caso concreto del análisis de sentimientos dicho teorema nos será de verdadera ayuda incluyendo cada documento d dentro de la clase C donde se maximice $P(d|C)$. Es decir, incluimos cada mensaje en la clase que más probabilidad tenga de incluirlo.

4.2.2. Support Vector Machine

Las SVMs, en español máquinas de vectores soporte, son un conjunto de algoritmos de aprendizaje supervisado[4]. Tienen su origen en el estudio del aprendizaje estadístico que desarrollaba Vladimir Vapnik y su equipo de los laboratorios $AT\&T$ sobre los años 90 del siglo pasado. En la actualidad son usadas para todo tipo de problemas multivariantes como la regresión, sin embargo, fueron originalmente pensadas para clasificación binaria.

El objetivo principal de las SVMs es encontrar un hiperplano que separe de forma óptima el conjunto de datos a tratar en dos grupos. Dicho concepto de separación óptima consiste en buscar el hiperplano que tenga la máxima distancia con los puntos que estén más cerca de él mismo. Es por esto que es común la denominación homóloga de clasificador de margen máximo.

Recordamos que en un espacio p-dimensional un hiperplano se define por la ecuación:

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0 \tag{4.2.2}$$

Supongamos que tenemos una matriz de datos, a los cuales se les suele denominar como de entrenamiento, $\mathbb{X}$ de dimensiones $n \times p$ consistente en n observaciones p-dimensionales,

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \cdots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix} \tag{4.2.3}$$

las cuales pertenecen a alguna de dos clases. Es decir, le asignamos una etiqueta de clase $y_1, \cdots, y_n \in \{-1, 1\}$, donde -1 representa una clase y 1 otra. De esta forma:

$$\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} > 0 \ si \ y_i = 1$$

y

$$\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} < 0 \ si \ y_i = -1$$

Por lo tanto $\forall i = 1, \cdots, p$ se tiene que:

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > 0 \tag{4.2.4}$$

En base a esto clasificaremos una nueva observación $x^* = (x_1^*, \cdots, x_p^*)^T$ según el signo de la función $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$. Cabe destacar que cuanto más cercano sea el valor de $f(x^*)$ a 0 más posible es que estemos realizando una mala clasificación. Es por esto que se suele utilizar tambien la magnitud de esta función como indicadora de la probabilidad de estar realizando una buena clasificación.

Nos planteamos entonces la construcción de dicho hiperplano, teniendo que determinar los parámetros. Estos son resultado del problema de optimización:

$$\max_{\beta_0, \cdots, \beta_p} M$$

$$sujeto \ a \ \sum_{i=0}^{p} \beta_i^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > M$$

4.2.3. Clasificador de máxima entropía

A diferencia dekl método de Naives -Bayes no presupondremos independencia entre las características o términos. La idea básica es que la probabilidad de que un documento pertenezca a una clase debe maximizar la entropía de clasificación, no introduciendo un sesgo en el sistema que estemos analizando.

De esta forma lo primero que se hará es definir f como una función binaria que nos indicará si en cierto documento d ocurre una cierta clase c y si está presente un cierto término w. Es decir,

$$f(d,c) = \begin{cases} 1 & c_i = c, w_k \in d \\ 0 & c.c. \end{cases}$$

Así, la probabilidad de que un documento pertenezca a cierta clase viene dada por:

$$P(c_j|d) = \frac{1}{Z(d)} exp(\sum_i \lambda_i f(d, c_j))$$

donde $Z(d)$ es una constante de normalización que se obtiene al sumar sobre todos los $P(c_j|d)$ para todas las clases c_j y λ_i debe satisfacer las condiciones del sistema donde las características observadas para un valor observado en el universo sea igual a la esperada en el conjunto de entrenamiento que le proporcionemos al algoritmo.

Al igual que para Naives-Bayes incluiremos el documento d en la clase c que maximice $P(c_j|d)$.

Aplicación

Una opción interesante que se podía utilizar para el análisis de sentimientos es mediante la utilización del paquete sentiment de R creado por Timothy Jurka. Véase [10], en el que se dice .Este paquete utiliza un clasificado naive Bayes entrenado sobre un léxico de emociones y otro para la subjetividad de

los textos..."

La función que nos interesa de este paquete es la de $classify\_emotion$, la cual permite identificar cualquier mensaje en alguno de los siguientes sentimientos:enfado, disgusto, miedo, alegría, tristeza y sorpresa.

Sin embargo, desde el 2014 esta librería no está disponible por requerimiento de su mantenedor. Después de buscar por nuevos paquetes y librerías no se encontró ninguno en el que hubiese una función homóloga, solamente encontramos funciones para el análisis de polaridad. Es por esto que se propone para estudios posteriores el desarrollo o aplicación de funciones que realicen una verdadera clasificación en sentimientos de un mensaje.

Capítulo 5

Conclusiones

En la presente memoria hemos visto diversas herramientas matemáticas que permiten el análisis de redes sociales en general y Twitter en particular, así en el capítulo 2, de análisis de frecuencia, nos planteamos dotar de una nueva estructura al conjunto de datos extraidos. Hicimos uso de la matriz de ocurrencia, clásica herramienta matemática, basada en lo que llamamos un corpus. En nuestra aplicación a los tweets de Trump usamos como corpus el conjunto de palabras que conformaban los tweets. Sin embargo podemos pensar en usar otro conjunto, el cual pueda ayudarnos a sacar conclusiones más claras para estudios más específicos.

El wordcloud que se hace posteriormente no hace más que obtener las palabras más frecuentes a partir de la matriz de ocurrencias y representarla. Además de ser una forma interesante de representación puede ser combinada con las técnicas de clasificación explicadas en el último capítulo. Se recomienda leer [10] para verlo.

En el capítulo 3, de clustering, cabe destacar la importancia que tiene la distancia, y por tanto la métrica, en las aplicaciones. En la aplicación del algoritmo de k-medoids podemos ver distintos resultados cambiando el parámetro metric en la función pamk, $> pamResult < -pamk(m3, metric =$ "$manhattan$"), véase [11]. En la aplicación del clustering jerárquico aglomerativo, para la creación de dendrogramas, tambien es clave cómo definamos la distancia. Se ve reflejado en las figuras de los dendrogramas.

Por último hacer hincapié en la importancia de los métodos de clasifi-

cación explicados en el último capítulo. El Support Vector Machine es una técnica de aprendizaje supervisado y por lo tanto hace uso de un conjunto de datos de entrenamiento. El hecho de que haya una gran cantidad de datos en las redes sociales que usar como conjunto de entrenamiento y que haya posibilidad de acceder a ellos fácilmente hace posible realizar predicciones con más seguridad.

Bibliografía

[1] Yanchang Zao, *R and Data Mining: Examples and Case Studies* http://www.RDataMining.com, 2013

[2] Yanchang Zhao, Yonghua Cen, *Data mining applications with R* http://www.RDataMining.com, 2013

[3] Rui Xu, Donald C. Wunsch, *Clustering* John Wiley and Sons, 2009

[4] Jénifer Sánchez Gallego, *Aportaciones del entorno de computación estadística R al análisis de redes sociales"* Universidad Granada, Trabajo Fin de Máster

[5] [CRAN], *Package cluster* https://cran.rstudio.com/web/packages/cluster/cluster.pdf, 2017

[6] [CRAN], *Package wordcloud* https://cran.rstudio.com/web/packages/wordcloud/wordcloud.pdf, 2015

[7] [CRAN], *Package igraph* https://cran.rstudio.com/web/packages/igraph/igraph.pdf, 2015

[8] [CRAN], *Package twitteR* https://cran.rstudio.com/web/packages/twitteR/twitteR.pdf, 2016

[9] Arkaitz Zubiaga, *Real-Time Classification of Twitter Trends* JOURNAL OF THE ASSOCIATION FOR INFORMATION SCIENCE AND TECHNOLOGY, 66(3):462-473, 2015

[10] Javier Mateo Fernández, *Análisis de contenidos en Social Media: Clasificación de mensajes e identificación de influyentes en el Banco Central Europeo (BCE).]* Universidad Complutense de Madrid, TFM.

[11] [CRAN], *Package fpc* https://cran.rstudio.com/web/packages/fpc/fpc.pdf, 2015