UNIVERSIDAD
DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Facultad de Ciencias

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO DE FIN DE GRADO

Técnicas estadísticas aplicadas a minería de texto

Presentado por:
Raquel Enrique Guillén

Tutor:
Úrsula Torres Parejo
Departamento de Estadística e Investigación Operativa

María Dolores Ruiz Jiménez
Departamento de Ciencias de la Computación e Inteligencia Artificial

Curso académico 2021-2022

Técnicas estadísticas aplicadas a minería de texto

Raquel Enrique Guillén

Raquel Enrique Guillén *Técnicas estadísticas aplicadas a minería de texto*.
Trabajo de fin de Grado. Curso académico 2021-2022.

Responsable de tutorización

Úrsula Torres Parejo
Departamento de Estadística e Investigación Operativa

María Dolores Ruiz Jiménez
Departamento de Ciencias de la Computación e Inteligencia Artificial

Doble Grado en Ingeniería Informática y Matemáticas

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Facultad de Ciencias

Universidad de Granada

Declaración de originalidad

D./Dña. Raquel Enrique Guillén

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2021-2022, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 21 de noviembre de 2021

Fdo: Raquel Enrique Guillén

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Agradezco a mis tutoras Úrsula Torres Parejo y María Dolores Ruiz Jiménez por proponerme este proyecto y haber estado ayudándome a lo largo del desarrollo del mismo con consejos, pautas y explicaciones que han sido fundamentales para la correcta evolución del trabajo.

También a mis padres por su apoyo incondicional no solo durante la realización de este proyecto, sino también en el transcurso de la carrera, confiando siempre en mí, así como esas personas que me han apoyado y ayudado emocionalmente con su comprensión a lo largo de los años.

A mis compañeros de clase que han sido un soporte indispensable durante la carrera, ayudándome y haciendo que un camino nada fácil sea más ameno, junto a sus consejos para la realización del presente trabajo.

Summary

In recent years data mining has increasingly become an important discipline in computer science from being considered a subprocess of Knowledge Discovery from Databases. It is also supported by many international conferences throughout the decade that highlight this field. Data that is processed in data mining has a fixed structure, however, nowadays there is fundamental information to study that is not structured such as texts, documents, etc. Here is where text mining (TM) arises, due to the need of analyzing texts from which we can extract useful information that could still be unknown without a correct processing and treatment.

Furthermore, social networks are platforms that have been arising in the last few years. They are used for different reasons: some people would have fun with them, others would use it to communicate with people, at work, and many others to look for information. During this process, people are constantly exposing their ideas on different and many scopes, generating data. It can be collected to study and obtain information that could be unknown at the beginning, showing behaviour patterns that might be useful for decision making or just to study. There is a wide variety of areas where text mining can be applied such as biotechnology, medicine, biomedicine, security, etc. There are also many techniques that may be used: summary, information extraction, classification, among others.

This work pretends to apply different mathematical and mining techniques on text such as clustering, association rules, classification using support vector machines and sentiment analysis. It is important to highlight that the data used in this project has been taken from the famous social network Twitter. Tweets from four spanish politicians (Pedro Sánchez, Pablo Casado, Inés Arrimadas and Pablo Iglesias) that belong to different parties are the subject of study to apply the different techniques and get conclusions from them. Since it is textual data, it needs to be processed in a different way than how structured data has to. Textual data must go through different processes of cleaning, tokenization, removal of stop words or stemming. The stop words are terms that do not have a meaning by themselves. They can vary depending on the area of the study, but usually they are pronouns, determinants, adverbs, etc. that represent "noise" for the useful information processing. This whole process is carried out in chapter 4 as an exploratory study along with a deeper study of the data. We obtained interesting results with the user that represents Pablo Casado. He is the politician that uses the most words and different ones among the rest of politicians. Nevertheless, when it comes to study the frequency of the terms that they use, there are two winner words by far: "sánchez" and "government". Contextualizing this, Casado is the president of a party that is in the political opposition, so we can say that he mentions the president and the government quite often. In this chapter, it is also interesting to see the results obtained in the bigrams and trigrams, since they show how the studied politicians frequently mention facts in the past in their tweets.

When clustering was applied in section 4.4, we could see how heterogeneous the extracted data is. This is guessed from the complexity to obtain the optimal number k of clusters after using three different methods: elbow method, Silhouette coefficient and Gap method.

It is interesting to observe that when we applied association rules in section 4.5, frequent

item sets that came up where similar to the most frequent terms that showed up in bigrams and trigrams in the exploratory study. However, none of them appeared in sets that are related to rules with good metrics (support, lift, etc.) that would represent real patterns of behaviour. Here it is shown exactly what is expected: association rules study the co-occurrence of item sets and it does not need to exist a relationship between words that appear one right next to another, and their occurrence in the same document (tweet in our study).

Applying the classification with support vector machines in subsection 4.6.1, we decided to separate the politicians into two groups to classify their tweets considering the author and using the linear model of this type of classification. In one hand, Inés Arrimadas and Pablo Casado were put together, and on the other hand, Pedro Sánchez was grouped with Pablo Iglesias. This decision was taken because of the, a priori, similarities between the politicians. After training and using the model, we obtained that it was harder to classify the tweets from Arrimadas and Casado, getting an error of 10.44% against 8.72% that resulted from the classification of the other two politicians. This could mean that Arrimadas and Casado post tweets that are more similar than the tweets of Sánchez and Iglesias. It is an interesting result since Inés and Pablo Casado are the presidents of parties that are in the opposition and the other two people formed a government coalition in the past.

The last technique computed is the sentiment analysis in subsection 4.6.2. The objective is to study the sentiments and polarity of the tweets by classifying the collected terms into eight different emotions. For this task, a term classification lexicon is used and the results obtained show a high "trust" in the tweets of all the politicians. A evolution of the polarity along the time can be seen in this study as well. It shows how three out of the four politicians behave almost similarly in the evolutionary graph, but Pablo Iglesias starts with a positivity close to (0.5) that falls to -1 from the first third of the normalized axis.

Talking about the structure of this memory, we can divide it in five different chapters. Firstly, chapter 1 includes an introduction and the objectives of this project. In chapter 2, we introduce the data and text mining, as well as their differences and ways to discover knowledge. It includes a section about the different areas where TM can be applied. It also explains different techniques to use in text mining and an introduction to the natural language processing (NLP) and opinion mining (also known as sentiment analysis). The next chapter (chapter 3) gathers the mathematical and theoretical support to apply the four statistical and mining techniques. In chapter 4 we include the execution and results obtained from the code generated in R, the programming language used in this project, as well as an interpretation of them. Finally, the last chapter (chapter 5) provides a conclusion of the study and future works that could be done. Two appendices are also included regarding the cost of the project and Gantt diagrams (Appendix A), and documentation with other information about the software and packages used (Appendix B).

Keywords: text mining, statistical techniques, Twitter, sentiment analysis, clustering, association rules, classification, support vector machines.

Resumen

En los últimos años, la minería de datos se ha convertido cada vez más en una disciplina importante en las ciencias de la computación al ser considerada un subproceso en el *Descubrimiento de Conocimiento a partir de Bases de Datos*. También cuenta con el apoyo de numerosas conferencias internacionales a lo largo de la década que destacan este campo. Los datos que se procesan en minería de datos tienen una estructura fija, sin embargo, hoy en día existe información fundamental a estudiar que no está estructurada, como son los textos, documentos, etc. Aquí es donde surge la minería de texto, por la necesidad de analizar textos de donde podemos extraer información útil que aún podría ser desconocida sin un correcto procesamiento y tratamiento de los mismos.

Por otro lado, cabe mencionar la existencia de plataformas que paulatinamente han ido creciendo con los años, como son las redes sociales. Los usos de estas son de diversa tipología: ocio y diversión, comunicación con personas o en el ámbito de trabajo, búsqueda de información, etc. Durante este proceso, las personas están constantemente exponiendo sus ideas en muchos y diferentes temas, generando datos que pueden ser recogidos para su estudio y obtener información que en principio podría ser desconocida, mostrando patrones de comportamiento que puedan ser estudiados o utilizados en la toma de decisiones. Hay una gran variedad de áreas en las que se puede aplicar la minería de texto. Algunas de ellas son la biotecnología, medicina, biomedicina, seguridad, etc. Existen también numerosas técnicas que se pueden usar dentro de la misma: resúmenes, extracción de la información o clasificación, entre otras.

Este trabajo pretende aplicar diferentes técnicas matemáticas y de minería sobre texto como clústering, reglas de asociación, clasificación con máquinas de vector soporte y análisis de sentimientos. Es importante destacar que los datos usados en este proyecto han sido extraídos a partir de la red social Twitter. Tweets de cuatro políticos españoles (Pedro Sánchez, Pablo Casado, Inés Arrimadas y Pablo Iglesias) pertenecientes a diferentes partidos son el objeto de estudio para aplicar las diferentes técnicas y obtener conclusiones de ellos. Como son datos textuales, estos necesitan procesarse de diferente forma a cómo tiene que hacerse con los estructurados. Los datos en formato texto deben pasar por diferentes procesos de limpieza, tokenización, eliminación de "stop words" o derivación de palabras. Las "stop words" son palabras que no tienen significado por sí mismas. Pueden variar en función al área de estudio, pero normalmente son pronombres, determinantes, adverbios, etc. que representan "ruido" para el procesamiento de la información útil. Todo este proceso se lleva a cabo en el Capítulo 4 como un análisis exploratorio junto con un estudio más profundo de los datos. Se obtuvieron resultados interesantes con el usuario que pertenece a Pablo Casado; es el político que más palabras y términos diferentes usa de entre el resto de usuarios estudiados. Sin embargo, cuando se estudia la frecuencia de términos que utiliza, hay dos claras palabras ganadoras: "sánchez" y "gobierno". Contextualizando esto, Casado es el presidente de un partido que se encuentra en la oposición, así que podemos decir que hace mención al presidente y al gobierno de manera frecuente. En este capítulo, también es interesante ver los resultados obtenidos en los bigramas y trigramas, ya que muestran como los políticos a menudo hacen mención al pasado en sus tweets.

Al aplicar **clústering** en la Sección 4.4, se puede ver la heterogeneidad de los datos extraídos. Esto se deduce por la complejidad de obtener el número óptimo de clústeres k para la agrupación de los datos tras utilizar tres métodos distintos: el criterio del codo, el coeficiente de Silhouette y el método de Gap.

Es interesante observar que cuando se aplicaron las **reglas de asociación** (Sección 4.5), los conjuntos de ítems frecuentes obtenidos se asemejan a los bigramas y trigramas más frecuentes que aparecieron en el análisis exploratorio previo de los datos. Sin embargo, ninguno de ellos aparece en conjuntos pertenecientes a reglas con buenas medidas como soporte o lift que representarían patrones reales de comportamiento. No se deben mezclar los conceptos de reglas de asociación y n-gramas, pues estos últimos requieren que los términos co-ocurrentes estén en orden estricto de adyacencia. Además en este trabajo se han obtenido de diferente manera: mediante tokenización por pares o tríos de términos (para bigramas y trigramas, respectivamente), asegurando el orden de ocurrencia de las palabras, y por otro lado las reglas se generaron haciendo uso del algoritmo apriori tomando ciertos valores mínimos como umbrales para las medidas de soporte y confianza.

Aplicando la **clasificación con máquinas de vector soporte** en la Subsección 4.6.1, se decidió separar los políticos en dos grupos y tratar de clasificar los tweets en función de la autoría, haciendo uso además del modelo lineal de este tipo de clasificación. Por un lado, Inés Arrimadas y Pablo Casado, y por otro Pedro Sánchez con Pablo Iglesias. Esta decisión se tomó por las posibles similitudes entre los políticos. Tras entrenar y utilizar el modelo, obtuvimos que es más difícil clasificar los tweets de Arrimadas y Casado, obteniendo un error del 10.44 % frente al 8.72 % que resultó de la clasificación de los otros dos políticos. Esto puede significar que Arrimadas y Casado publican tweets que son más similares que los de Iglesias y Sánchez. Es un resultado interesante ya que Inés y Pablo Casado presiden partidos que están en la oposición, y los otros dos llegaron a formar una coalición de gobierno.

La última técnica ejecutada es el **análisis de sentimientos** (Subsección 4.6.2). El objetivo es estudiar los sentimientos y polaridad de los tweets clasificando los términos recogidos en ocho emociones diferentes. Para esta tarea, se usa un diccionario para clasificar los términos, y los resultados obtenidos muestran una alta "confianza" en los tweets de todos los políticos. También se puede ver en este estudio la evolución de la polaridad a lo largo del tiempo de los usuarios, donde tres de ellos se comportan de manera pareja (Arrimadas, Casado y Sánchez), pero Iglesias tiene una gráfica distinta en la que parte de una positividad cercana a 0.5 y decae en el primer tercio de la gráfica hasta valores próximos a -1.

Sobre la estructura de esta memoria, podemos dividirla en cinco capítulos distintos. En primer lugar, el Capítulo 1 incluye una introducción y los objetivos del trabajo. El Capítulo 2 introduce la minería de datos y texto, así como sus diferencias y maneras de descubrir conocimiento. Incluye una sección sobre las diferentes áreas donde la minería de texto se puede aplicar, además de mencionar diferentes técnicas que se usan en la misma y una breve introducción al procesamiento del lenguaje natural y la minería de opinión (también conocida como análisis de sentimientos). El Capítulo 3 reúne el soporte teórico-matemático para aplicar las cuatro técnicas estadísticas y de minería de este trabajo. En el Capítulo 4 se incluye la ejecución y los resultados obtenidos tras la generación del código en R, el lenguaje de programación utilizado en este proyecto, junto con una interpretación de los mismos. Por último, el Capítulo 5 ofrece una conclusión del estudio y futuros trabajos que se podrían realizar. Se incluyen dos apéndices con información sobre el coste del proyecto y diagramas de Gantt (Apéndice A), y documentación con otra información del software y paquetes utilizados en el trabajo (Apéndice B).

Palabras clave: minería de texto, técnicas estadísticas, Twitter, análisis de sentimientos, clústering, reglas de asociación, clasificación, máquinas de vector soporte.

1. Introducción

La minería de datos se ha convertido en una disciplina relevante en el ámbito de la informática. La comunidad de investigación comenzó a usarla a finales de los años 80, y fue a principio de los 90 cuando la minería de datos se reconocía como un proceso secundario dentro de lo que se conoce como *Descubrimiento de Conocimiento a partir de Bases de Datos*. La popularidad de este ámbito tuvo un auge considerable a raíz de diferentes conferencias internacionales a lo largo de la década y a día de hoy hay una tendencia global hacia lo que se conoce como *big data* [28]. Además, hoy en día hay mucha información que es fundamental estudiar, pero que no cuenta con una estructura definida, por lo que no es posible aplicarle las técnicas de minería de datos sin un procesamiento previo. Es por ello que surge la minería de texto o *text mining*, una disciplina que es capaz de extraer datos no estructurados de un conjunto de datos grande para su posterior estudio y que ha ido creciendo exponencialmente en la última década por la accesibilidad de información en la web o documentos. Esta se diferencia de la minería de datos en que la información con la que se trabaja no es estructurada y, por tanto, tiene otros tipos de tratamiento de los datos.

Aquí surge la motivación del presente trabajo: analizar textos de los cuales podemos extraer información útil y que, sin el procesamiento y tratamiento adecuado, hubiera permanecido oculta. Para ello, se ha realizado un experimento, donde los datos se han obtenido de la red social Twitter y ha sido necesario un procesamiento de los mismos de una manera diferente a la que se hace con datos estructurados, para posteriormente poder aplicarle técnicas objeto de estudio. Se ha hecho uso del lenguaje de programación R y se ha utilizado el software RStudio para el desarrollo del trabajo y la creación de un cuaderno que recoge el código y resultados obtenidos. Este se puede consultar en el Apéndice B.

En este trabajo se han utilizado una serie de técnicas estadísticas y de minería que se listan a continuación:

1. Clústering. Técnica que se encarga de agrupar términos en diferentes categorías por similitudes.
2. Reglas de asociación. Pretende estudiar la coocurrencia de términos en un mismo documento.
3. Clasificación mediante máquinas de vector soporte. En este trabajo se utiliza para clasificar los tweets en función de su autoría, utilizando un modelo matemático basado en máquinas de vector soporte.
4. Análisis de sentimientos. Se clasifican las palabras en función a 8 emociones, de forma que se estudian las emociones de los usuarios en sus tweets, así como su polaridad y la evolución de los mismos en un período de tiempo.

1.1. Objetivos

Podemos dividir los objetivos del trabajo en generales y específicos.

Generales:

- Ofrecer una visión general de la minería de texto y de la metodología seguida desde el texto no estructurado hasta el texto con estructura y la obtención de información.
- Dominar un lenguaje de programación para la aplicación de dichas técnicas y metodología.
- Profundizar en el estudio de técnicas estadísticas aplicables a texto para clasificación, predicción, agrupamiento, etc.

Específicos:

- Realizar un ejemplo de aplicación sobre datos reales.
- Empleo de técnicas: análisis exploratorio, clústering, reglas de asociación, clasificación con máquinas de vector soporte y análisis de sentimientos.
- Hacer un estudio de los paquetes de R adecuados para los distintos aspectos de la minería de texto y para la aplicación de las técnicas estadísticas consideradas.
- Contextualizar los datos obtenidos en el análisis exploratorio de los datos.
- Mostrar empíricamente la importancia del uso de técnicas estadísticas en minería de texto para descubrir información o patrones de comportamiento que son a priori desconocidos.

1.2. Estructura

Este trabajo se divide en cuatro partes o capítulos principales.

En primer lugar, encontraremos un capítulo introductorio sobre la minería de texto (Capítulo 2) donde se ofrece una definición de la misma, junto con su contexto histórico y las diferencias con la minería de datos. Del mismo modo, se listan algunas de las aplicaciones de la minería de texto, así como una breve sección dedicada al procesamiento del lenguaje natural (PLN) y la minería de opinión o análisis de sentimientos.

Tras haber contextualizado la situación de nuestro estudio, en el Capítulo 3 se ofrece el soporte teórico-matemático de las distintas técnicas estadísticas y de minería que se aplicarán. Cabe mencionar que uno de los libros más referenciados es el de *Data mining and analysis: fundamental concepts and algorithms* ([53]), ya que proporciona un soporte bastante extendido en las técnicas. En primer lugar se trata el **clústering** (Sección 3.1). Por un lado se explica el clústering jerárquico, el cual hace un agrupamiento de los datos de modo que crea una jerarquía, haciendo uso del método de Ward de varianza mínima, y por otro lado se explica el clústering no jerárquico, específicamente el conocido k-medias y los diferentes criterios para determinar el número de clústeres óptimo para este algoritmo, puesto que se tiene que determiar previamente al uso del mismo: el criterio del codo, el método de Silhouette y el método de Gap. Posteriormente se fundamenta lo relacionado con las **reglas de asociación** (Sección 3.2). En esta sección se ofrecen unas definiciones para contextualizar las reglas y determinarlas. Del mismo modo se profundiza en las medidas de soporte, confianza y lift, pues son claves en la creación y evaluación de las reglas, y en el algortimo que se utiliza

en este trabajo: el "apriori". Este busca, en primer lugar, conjuntos de ítems que se consideran frecuentes para después obtener las reglas de asociación. Del mismo modo, para evaluar las reglas obtenidas, se hace uso del test exacto de Fisher, por lo que también se le da soporte teórico. En tercer lugar, se trata la **clasificación mediante máquinas de vector soporte** (Sección 3.3). Detrás de este tipo de clasificación hay un fundamento matemático que tiene relación con los hiperplanos y clasificadores de margen máximo o margen suave, entre otros. Para finalizar, se trata el **análisis de sentimientos** (Sección 3.4). Dentro de este apartado se fundamentan matemáticamente distintas métricas que se utilizan para suavizar curvas. Estas son: loess, media móvil y la transformada discreta de coseno, las cuales se tratan en profundidad para entender cómo actúan en la suavización de curvas y en la posterior experimentación.

En el Capítulo 4 se realiza la parte experimental del trabajo. Para ello, previamente se recopilaron los datos a partir de Twitter, donde es necesario solicitar una cuenta de desarrollador y crear una aplicación necesaria para la posterior extracción de los tweets. En este capítulo, se hace un análisis exploratorio previo a la aplicación de las técnicas recién mencionadas, el cual ofrece información de los datos recogidos que facilita su contextualización. Tras esto, se explican y se muestran los pasos que se han seguido en la ejecución y creación del código realizado con RStudio, así como los resultados de aplicar las distintas técnicas estadísticas y de minería.

Por último, en el Capítulo 5 se muestran las conclusiones del estudio, junto con los trabajos futuros de posibles investigaciones derivados de este proyecto.

Se incluyen también dos apéndices que muestran la información importante del código (Apéndice A) y los datos del software utilizado (Apéndice B).

2. Minería de texto

Si queremos comprender la tecnología de la minería de texto moderna, debemos verla desde el contexto de la historia tecnológica de la que surgió. Estos desarrollos fueron fruto de los numerosos desafíos debido al aumento del volumen de la información textual.

2.1. Definición de minería de texto

La minería de texto o *text mining* es el descubrimiento de información nueva o presuntamente desconocida, mediante la extracción automática de conocimiento a través de diferentes fuentes textuales. Un elemento clave de la minería de texto es la asociación de esta información extraída para crear nuevos hechos o hipótesis, que se explorarán más a fondo mediante medios de experimentación convencionales [21].

La minería de texto es distinta de lo que conocemos como una búsqueda corriente en la Web. En esta, normalmente el usuario busca contenido que ya conoce previamente y este ha sido escrito por otra persona. El mayor problema a tener en cuenta es poder dejar a un lado todo el contenido o material que no es importante, para encontrar lo que sí es realmente relevante. Por ello, el objetivo del *text mining* es descubrir información desconocida hasta el momento, que sea potencialmente útil o dar soporte al experto sobre los resultados esperados en el análisis.

Dos de las limitaciones fundamentales del *text mining* son, en primer lugar, la imposibilidad de escribir programas que interpreten texto durante un largo período de tiempo y, en segundo lugar, que la información que se recoge no está siempre en formato de texto para la utilización de técnicas de minería de textos.

Cabe destacar la importancia de la lingüística computacional y el procesamiento del lenguaje natural (PLN), del cual hablaremos más tarde como soporte para hacer pequeñas subtareas en el análisis de texto.

2.1.1. ¿En qué se diferencia de la minería de datos?

Para saber en qué se diferencian, primero es necesario dar una definición de la minería de datos o *data mining*.

La minería de datos se refiere a la búsqueda de información útil y relevante en bases de datos, así como el rastreo de patrones ocultos y obtención de información predictiva o descriptiva que los expertos podrían no obtener [40].

La diferencia entre la minería de datos y texto es que, en la segunda, los patrones se extraen del texto como lenguaje natural en lugar de hacerlo a partir de bases de datos estructuradas, como ocurre en la primera, ya que las bases de datos están diseñadas para programas que las procesen automáticamente [21]. Sin embargo, cabe destacar la existencia de nuevas técnicas de procesamiento masivo de datos, en los que los datos podrían no estar estructurados.

2.2. Historia de la minería de texto

La evolución de la minería de texto a lo largo de los años no tiene una estructura clara, debido al desorden con el que se ha desarrollado [35].

A medida que pasaban los años, se vio que la obtención de información en formato texto era cada vez más necesaria, de forma que en 1997 se planteó los beneficios de la minería de texto en la inteligencia artificial usada en la Web. Posteriormente, en 1999, el *text mining* tenía la peculiaridad de tener un nombre para reconocerlo pero no fue hasta meses después cuando su práctica tuvo un claro auge.

Podemos materializar lo que comentamos, viendo que entre 2001 y 2009 creció el número de páginas web existentes en un 40 % cada año. Esto provocó la acumulación de datos textuales y, por tanto, información que podría ser procesada con minería de texto.

2.3. Descubrimiento de Conocimiento a partir de Bases de Datos (Knowledge Discovery from Databases, KDD)

Para comenzar a explicar cómo ha evolucionado la obtención de información y cómo funciona la minería de texto, conviene remontarnos a la forma en la cual se empezó a obtener información a partir de bases de datos.

Todo comenzó en los años sesenta, utilizando técnicas de obtención de datos para su posterior estudio, las cuales han ido evolucionando hasta la actualidad. En su comienzo, estas técnicas eran análisis manuales sobre ciertos datos recogidos y almacenados en una base de datos para hacer predicciones u obtener conclusiones [11].

Como comentamos, a día de hoy estas técnicas han cambiado; el tratamiento de estos datos debe ser distinto por su considerable crecimiento en tamaño, así que es necesaria una automatización de estos procesos de análisis y extracción de datos.

De esta forma surgió lo que conocemos en la actualidad como "Descubrimiento de Conocimiento a partir de Bases de Datos"*(Knowledge Discovery from Databases, KDD)*, cuyo proceso se muestra en la Figura 2.1. Entre las técnicas utilizadas podemos destacar la Estadística Descriptiva, la Inferencia Estadística o el Aprendizaje Automático.

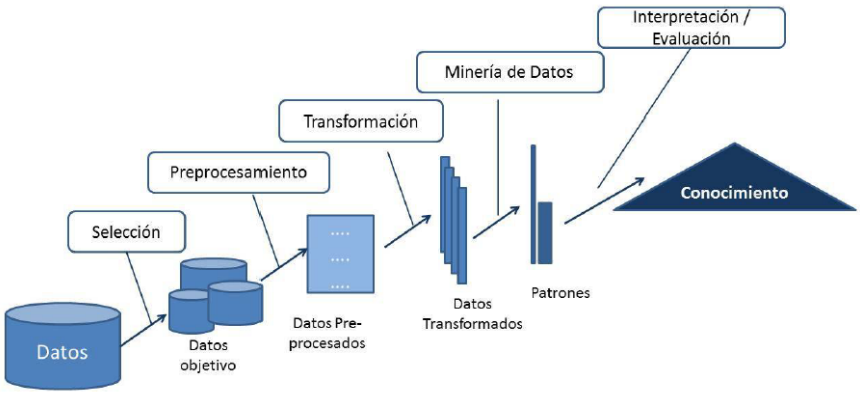


Figura 2.1.: Fases del proceso KDD [11]

Se mencionan y explican ahora los pasos mostrados [13]:

1. Para comenzar, debemos comprender el dominio de aplicación identificando el **objetivo** del proceso del *KDD* desde el punto de vista del cliente, además de la creación o selección de un **conjunto de datos objetivo** sobre el cual obtendremos información. Del mismo modo, se debe hacer un análisis de requisitos en el que se hace un estudio exhaustivo del sistema a desarrollar, así como la definición y aplicación de las técnicas que se van a utilizar.

2. **Limpieza de datos y preprocesamiento.** Para ello, se utilizan operaciones básicas como la eliminación de ruido o selección de estrategias para la gestión de datos perdidos.

3. **Reducción y proyección de los datos.** En función de los objetivos propuestos, debemos elegir características útiles e importantes sobre las cuales podamos basar el estudio, ya que estas harán de representantes. Así mismo, debemos **emparejar** los objetivos establecidos con un método particular de minería de datos (clustering, clasificación, etc.).

4. Se deben elegir el/los algoritmo/s de **minería de datos** para entonces aplicarlos y hacer la búsqueda de patrones de interés representativos.

5. **Interpretación** de los patrones minados, así como la visualización de los mismos.

6. **Consolidación** del conocimiento descubierto. En este paso podemos simplemente reportar los datos, documentarlos, o tenerlos en cuenta para previos y posteriores estudios.

2.3.1. Descubrimiento de Conocimiento en Textos (Knowledge Discovery in Text, KDT)

Como ya se ha comentado anteriormente y se ha podido apreciar, el proceso de la obtención de información a partir de textos difiere del KDD *(Knowledge Discovery from Databases)*, ya que el procesamiento de texto además tiene asociado diversos problemas inherentes del análisis del lenguaje natural como son el significado contextual, jergas, variaciones lingüísticas... Por ello es que procedemos a explicar el proceso KDT (*Knowledge Discovery in Text* o Descubrimiento de Conocimiento en Textos), mostrado en la Figura 2.2 [38]:

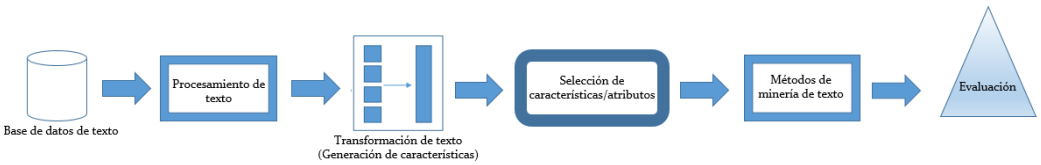


Figura 2.2.: Fases del proceso KDT [38]

1. En la literatura, podemos encontrar una serie de técnicas bastante preestablecidas para el **preprocesamiento** de texto. Entre ellas destacaremos las siguientes:

   a) *Tokenización*: En este paso se borran las comas, espacios, etc. del documento de texto y se divide en las unidades que lo conforman.

   b) Eliminación de *"stop words"* o palabras sin significado. Involucra la eliminación de etiquetas de HTML o XML de las páginas web, a la vez que la eliminación de palabras como *"un"*, *"de"*, etc.

   c) Derivación de palabras o *stemming*. Se utilizan técnicas para encontrar y descubrir las raíces de las palabras para convertirlas en ellas. Un ejemplo podría ser transformar las palabras *viajando*, *viajaremos* en *viajar*.

2. **Transformación de texto/Generación de características.** En este paso se pretende convertir el documento de texto en una bolsa de palabras o en una notación especial de manera que pueda ser usada en tareas de análisis.

3. **Selección de características o atributos.** En esta fase se eliminan las características que son irrelevantes para el propósito del análisis.

4. **Métodos de minería de texto.** En este paso se utilizan técnicas de minería de texto en función a los objetivos del proceso, aunque en muchos casos, podrán ser técnicas adaptadas de la minería de datos.

5. **Evaluación o interpretación de los resultados.** Se pretende evaluar e interpretar los resultados obtenidos en términos de exactitud, precisión, etc.

En general, podríamos sintetizar las principales fases del KDT en *Preprocesamiento, Minería de Texto y Visualización*. Aunque no se debe olvidar que el término *Descubrimiento de Conocimiento en Textos* es más amplio que el de *Minería de texto*, ya que el primero identifica al proceso de descubrimiento completo [11].

2.3.2. Diferencias entre KDD y KDT

En un principio, podríamos pensar que los procesos de descubrimiento de información a partir de bases de datos y a partir de texto podrían ser similares. Sin embargo, como hemos comentado anteriormente, el proceso de KDD es a partir de bases de datos estructuradas, pero en la actualidad, mucha de la información útil y disponible se encuentra en formato texto, de ahí la importancia del *text mining* y el por qué de las diferencias entre ambos procesos de obtención de información, ya que la información en formato textual no está estructurada.

Podemos comenzar viendo que tanto en el KDD como en el KDT se comparte un primer paso en el que se comprende el dominio de la aplicación y se definen qué conceptos son relevantes [11]. Sin embargo, la obtención de la información en el KDT se realiza a través de herramientas de recuperación de información o de forma manual, mientras que en el KDD simplemente se selecciona el/los conjunto/s de datos objetivo.

Como se ha comentado en secciones previas, el KDT necesita de un preprocesamiento de la información para su posterior tratamiento, haciendo uso de la tokenización, eliminación de *stop words*... Estos procesos no son usados en el KDD, sino que se utilizan operaciones como eliminación de ruido, entre otras. Tras este preprocesamiento, el siguiente paso en el KDD es reducir y proyectar los datos para luego elegir los algoritmos que se utilizarán. Por otro lado, en el KDT, se transforma el texto y se generan características para luego seleccionar las relevantes para el análisis.

En último lugar, los dos procesos de descubrimiento de conocimiento vuelven a tener en común los pasos en los que se realizan las tareas correspondientes para llevar a cabo la minería de texto y la minería de datos, siguiendo su posterior interpretación de los resultados obtenidos.

Como podemos observar a lo largo de la comparación comentada, ambos procesos comparten fases, pero vemos como, claramente, el hecho de tener la información en formato textual hace que el tratamiento de la misma tenga que ser diferente al de datos.

2.4. Métodos utilizados en la minería de texto

Los métodos que se nombran a continuación son algunos de los usados en la minería de texto, algunos empleados también en la minería de datos, y estos a su vez se utilizan también en diferentes ámbitos [38] mencionados en la siguiente sección.

- Extracción de la información. Existen numerosas técnicas de extracción de información que buscan e identifican la clave de una frase y la relación con el texto. Para ello, utilizan métodos de unión o relación de patrones.
- Resumen. Se transforma el texto en una versión más corta preservando su información. Pueden ser resúmenes abstractivos o extractivos; este último consiste en seleccionar frases importantes, párrafos, etc. del documento original y concatenarlos en una forma más corta, mientras que el primero intenta desarrollar una comprensión de los conceptos principales para luego expresarlos en lenguaje natural. Para ello, utiliza métodos lingüísticos que examinan e interpretan el texto para encontrar nuevos conceptos que sean los más relevantes del texto original.
- Clasificación. Esta técnica clasifica documentos de texto en clases predefinidas o categorías.
- Clustering. No tiene clases predefinidas o categorías como sí ocurre en la clasificación, pero utilizando medidas de similitud entre diferentes objetos, se encarga de poner los objetos más similares juntos en una misma clase y los menos parejos en otra.
- Vinculación de conceptos. Encuentra documentos relacionados que tienen conceptos en común. El objetivo de este método es proporcionar un método de búsqueda de información en lugar de buscarla como en la recuperación de la información.
- Visualización de la información. Necesitamos una representación visual e interactiva de los datos abstractos para aumentar el uso o la adquisición de conocimiento. Un gobierno podría usar este método para identificar redes terroristas o información sobre crímenes. Podría proporcionar posibles relaciones entre actividades sospechosas para investigar sus conexiones.
- Minado de reglas de asociación. Es una técnica usada para descubrir relaciones entre un conjunto de variables grande dentro de un conjunto de datos.

2.5. Aplicaciones de la minería de texto

Hoy en día, el uso del *text mining* tiene mucho valor a nivel comercial y puede ser aplicado en muchos campos, como los que mencionamos a continuación [38]:

- Biomedicina. Es muy frecuente el uso del análisis de texto para la identificación y clasificación de términos de la biología molecular, así como obtención de información sobre genes y proteínas, polimorfismo genético o relación entre genes y enfermedades no son más que ejemplos de uso en este área.
- Inteligencia competitiva. La minería de texto permite a las empresas organizar y modificar sus estrategias en función de las demandas actuales del mercado, así como de las oportunidades basadas en la información propia recogida por parte de la empresa, el mercado y sus competidores, y gestionar una cantidad considerable de datos para analizarlos y crear un plan.
- Seguridad. Existen paquetes software de minería de texto en venta para propósitos de seguridad, especialmente supervisión y análisis de fuentes de texto on-line como blogs, noticias en Internet, etc. También se utiliza en el estudio de encriptación de texto.
- Gestión de recursos humanos. La minería de texto en este ámbito se usa principalmente para analizar las opiniones del personal, conocer la satisfacción de los empleados, así como leer y almacenar currículums para la selección de nuevo personal.
- Gestión de las relaciones con el cliente. En este ámbito, se utiliza para la redirección de solicitudes automáticamente al servicio apropiado, así como dar respuestas inmediatas a las preguntas más frecuentes.
- Planificación de los recursos de la empresa. Se recogen informes del minado para que así se puedan manejar adecuadamente los problemas y el estado de los recursos, para diseñar acciones futuras y planes que lo gestionen.
- Minería de opinión. Se puede utilizar la minería de texto para hacer un análisis de los sentimientos o un estudio computacional de las opiniones de las personas. Esta técnica se abordará en profundidad a lo largo del presente trabajo.

2.6. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (PLN) trata de extraer una representación completa de un significado a partir de texto libre. Su principal objetivo es diseñar y construir un sistema que analice y entienda el lenguaje natural.

El PLN normalmente hace uso de conceptos lingüísticos como nombres, verbos, adjetivos etc. y estructuras gramaticales. Tiene que tener en cuenta, además, las ambigüedades (de palabras y de estructura gramatical) y las anáforas [25], figura retórica que consiste en la repetición de una o varias palabras al principio de un verso o enunciado, entre muchas otras cosas.

Para hacer lo mencionado, se usan varias representaciones de conocimiento como el léxico de palabras y su significado, así como propiedades y un conjunto de reglas gramaticales, aunque a veces también pueden usar otras fuentes de información como las que ofrecen las ontologías[1] o diccionarios de sinónimos y abreviaciones.

[1] La ontología en el campo de la computación se puede definir como el estudio de las categorías de cosas que existen o pueden existir en un dominio. Algunos autores como T. Gruber la definen como la especificación explícita de la conceptualización.

2.7. Minería de opinión

Con el crecimiento exponencial que han tenido los medios de comunicación sociales, también conocidos como *social media*, las organizaciones e individuos utilizan las opiniones públicas en estos medios de manera cada vez más frecuente para su toma de decisiones [30]. Sin embargo, encontrar y monitorizar sitios donde se pueden extraer opiniones es una tarea difícil debido a la heterogeneidad de las fuentes y los formatos textuales. Además, se une el problema de poder identificar la información que es realmente importante y resumir de manera adecuada las opiniones contenidas en la misma.

Es aquí donde la **minería de opinión** o **análisis de sentimientos** nace y obtiene relevancia, la cual se puede definir como un estudio computacional de las opiniones, valoraciones, actitudes y emociones de las personas sobre entidades, individuos, problemas o eventos.

A lo largo de los años, junto con la evolución de técnicas como el aprendizaje automático en el procesamiento del lenguaje natural o la disponibilidad de conjuntos de datos con los que se pueden entrenar modelos [4], la minería de opinión ha ido incrementando su importancia como aplicación de la minería de texto.

Algunos de los estudios que se pueden hacer dentro de la minería de opinión se listan a continuación [42]:

- Determinación de la polaridad. Se puede estudiar cómo de positiva o negativa es la información recogida en función a los términos que contiene. Estos pueden ser identificados en función a su polaridad tras ponderarlos en un rango numérico, donde si este valor es menor que 0, será negativo, y si es mayor, se considerará positivo.
- Acercamiento basado en *lexicon*. A partir de un diccionario de palabras (lexicon) se pueden clasificar términos cuyo significado está relacionado con estas palabras. Estas pueden ser: alegría, tristeza, angustia, etc.
- Detección de spam de opinión. Con el auge del comercio digital y las reseñas en la web, se produce lo que se conoce como spam de opinión. La mayoría de las técnicas de detección de spam de opinión dependen de tres características relacionadas con una opinión falsa. Estas incluyen: el contenido de la opinión, los metadatos de la opinión y el conocimiento de la vida real sobre el producto.
- Predicción de tasas de mercado y FOREX[2]. Se han realizado diversos estudios en la predicción de mercado tras haber un aumento de los datos financieros con sentimientos de redes sociales como Twitter.

Por lo tanto, podemos decir que el análisis de sentimientos o la minería de opiniones, junto con un análisis de subjetividad, son áreas de investigación interrelacionadas que utilizan diversas técnicas recogidas a partir del procesamiento del lenguaje natural, la recuperación de información y la minería de texto.

[2]Se define FOREX como un mercado de divisas en el cual se pueden realizar acciones de inversión e intercambio de divisas, entre otras cosas [55].

3. Técnicas estadísticas y de minería

A lo largo de este capítulo se van a definir las diferentes técnicas estadísticas y de minería que se aplicarán en este trabajo, junto con su fundamento teórico y matemático para una completa comprensión de las mismas.

3.1. Clústering

El clústering, también llamado clasificación no supervisada y el cual forma parte del análisis exploratorio de datos, tiene como objetivo la separación de un conjunto de datos finito y no etiquetado en un conjunto discreto y también finito de estructuras de datos ocultas, conocidos como clústeres [52].

En este trabajo se describen dos tipos de clústering: jerárquico y no jerárquico. En el primero se hace uso del *método de Ward de varianza mínima* y en el segundo el método conocido como *k-medias*.

3.1.1. Clústering jerárquico

El clústering jerárquico, como su propio nombre indica, se encarga de hacer un agrupamiento de datos formando una jerarquía. Normalmente los resultados de esta técnica se muestran en un dendograma, donde la similitud de los términos vendrá dada por la altura del nodo común que esté más cerca. En ocasiones, esto nos puede ayudar a determinar visualmente el número de clústeres que se adapta mejor a nuestros datos, pero esto no es una tarea sencilla.

Hay dos tipos de técnicas de clústering jerárquico [7]:

- Aglomerativas. Se empieza con tantos clústeres individuales como casos y en los pasos siguientes se unen los pares de clústeres más próximos hasta que sólo quede uno o un número establecido.
- Divisivas. Se empieza con un único clúster que comprenda todos los casos del conjunto de datos y se sigue descomponiendo cada clúster hasta que todos contengan un único caso.

En la Figura 3.1 podemos ver una ilustración de cómo actúan.

Llegados a este punto, dados n puntos de un espacio d-dimensional, el objetivo del clústering jerárquico es crear una secuencia de particiones anidadas que pueden representarse mediante árboles o clústeres jerárquicos, como son los dendogramas. Damos entonces una definición del mismo [53].

Definición 3.1. Dado un conjunto de datos $D = \{x_1, ..., x_n\}$ donde $x_i \in \mathbb{R}^d$, se define el **clústering** $\mathcal{C} = \{C_1, ..., C_k\}$ como una partición de D, esto es, cada clúster es un conjunto de puntos $C_i \subseteq D$, de forma que los clústeres son disjuntos dos a dos ($C_i \cap C_j = \emptyset$ para todo $i \neq j$) y $\bigcup_{i=1}^{k} C_i = D$.

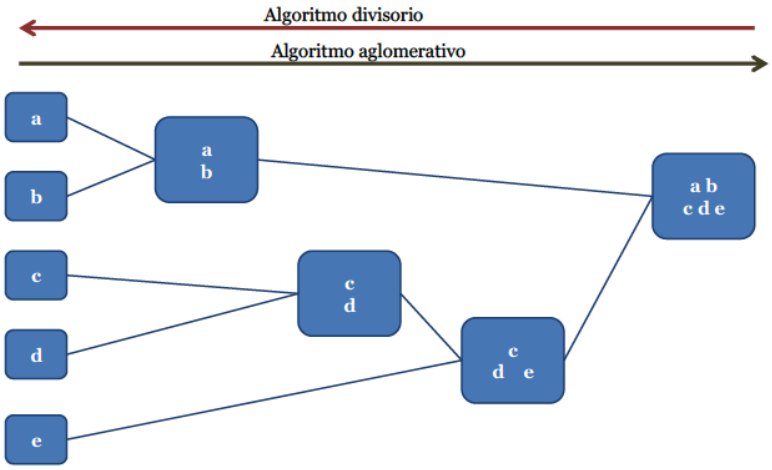


Figura 3.1.: Gráfico de clústering jerárquico divisivo y aglomerativo [44]

Definición 3.2. Se dice que un clústering $\mathcal{A} = \{A_1, ..., A_r\}$ está **anidado** a otro clústering $\mathcal{B} = \{B_1, ..., B_s\}$ si y solo si $r > s$ y para cada clúster $A_i \in \mathcal{A}$ existe un clúster $B_j \in \mathcal{B}$ tal que $A_i \subseteq B_j$.

El clústering jerárquico produce una secuencia de n particiones anidadas $\mathcal{C}_1, ..., \mathcal{C}_n$ desde la partición trivial $\mathcal{C}_1 = \{\{x_1\}, ..., \{x_n\}\}$, donde cada punto está en clústeres distintos, hasta el otro clústering trivial $\mathcal{C}_n = \{\{x_1, ..., x_n\}\}$, en el cual todos los puntos se encuentran en el mismo clúster. En general ocurre que el clústering $\mathcal{C}_{s-1}$ está anidado al clústering $\mathcal{C}_s$.

En este trabajo se hará uso de un método aglomerativo de clústering conocido como *Método de Ward de varianza mínima*.

En el clústering jerárquico aglomerativo, siempre se comienza con cada uno de los n puntos en un clúster separado. Se combinan repetidamente los dos clústeres más cercanos hasta que todos los puntos son miembros del mismo clúster, como se puede observar en el pseudo-código de la Tabla 3.1. Formalmente, dado un conjunto de clústeres $\mathcal{C} = \{C_1, ..., C_m\}$, se busca el par de clústeres más cercanos C_i y C_j, combinándolos en un nuevo clúster $C_{ij} = C_i \cup C_j$. Posteriormente se actualiza el conjunto de clústeres eliminando C_i y C_j, para añadir C_{ij}. Se repite el procedimiento hasta que $\mathcal{C}$ contiene un único clúster. Dado que el número de clústeres decrece en 1 para cada paso, este algoritmo da como resultado una secuencia de clústeres anidados. Si se especifica, se puede parar el proceso de combinación de clústeres cuando haya exactamente k de ellos restantes.

El paso principal en el algoritmo es el de calcular la distancia entre el par más cercano de clústeres. Hay muchas distancias que se pueden utilizar: euclídea, distancia media... Sin embargo, en nuestro estudio al utilizar el método de Ward de varianza mínima, la distancia entre dos clústeres será la distancia de la varianza mínima.

Definición 3.3. Dados dos clústeres disjuntos, se define la **distancia de varianza mínima** entre ellos como la suma de los errores al cuadrado (SSE, *sum of squared errors*) cuando se combinan los dos clústeres. La SSE para un clúster C_i dado viene representada por:

$$SSE_i = \sum_{x \in C_i} \|x - \mu_i\|^2$$

	ALGORITMO CLÚSTERING JERÁRQUICO AGLOMERATIVO
	clusteringAglomerativo(D,k):
1	$\mathcal{C} \leftarrow \{C_i = \{x_i\} \mid x_i \in D\}$ //Cada punto en clústeres separados
2	$\mathcal{A} \leftarrow \{\delta(x_i, x_j) : x_i, x_j \in D\}$ //Se calcula la matriz de distancias
3	Repite:
4	Encuentra el par de clústeres más cercanos $C_i, C_j \in \mathcal{C}$
5	$C_{ij} \leftarrow C_i \cup C_j$ //Se combinan los clústeres
6	$\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_i, C_j\}) \cup \{C_{ij}\}$ //Se actualiza el clústering
7	Se actualiza la matriz de distancias $\mathcal{A}$ para reflejar el nuevo clústering
8	Hasta que $\lvert\mathcal{C}\rvert = k$

Tabla 3.1.: Algoritmo para el clústering jerárquico aglomerativo

donde $\mu_i = \frac{1}{n_i} \sum_{x \in C_i} x$ y $n_i = |C_i|$ denota el número de puntos en el clúster C_i.

Cabe destacar que x es un punto de $\mathbb{R}^d$, así que μ_i será un vector de medias. Esto implica que $||.||$ es la distancia euclídea en d dimensiones. Podemos escribir esta misma ecuación como:

$$SSE_i = \sum_{x \in C_i} \|x - \mu_i\|^2 = \sum_{x \in C_i} x^T x - 2 \sum_{x \in C_i} x^T \mu_i + \sum_{x \in C_i} \mu_i^T \mu_i = (\sum_{x \in C_i} x^T x) - n_i \mu_i^T \mu_i \tag{3.1}$$

Del mismo modo, dado un clústering $\mathcal{C} = \{C_1, ..., C_k\}$, se define su SSE como:

$$SSE = \sum_{i=1}^{k} SSE_i = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Definición 3.4. La medida de Ward define la distancia entre dos clústeres C_i y C_j como el cambio neto en el valor de SSE cuando se combinan en C_{ij}, esto es:

$$\delta(C_i, C_j) = \Delta SSE_{ij} = SSE_{ij} - SSE_i - SSE_j \tag{3.2}$$

Podemos obtener una expresión más simple de esta misma teniendo en cuenta la ecuación (3.1) y (3.2), y dado que $C_{ij} = C_i \cup C_j$ y $C_i \cap C_j = \emptyset$, tenemos $|C_{ij}| = n_{ij} = n_i + n_j$, por lo tanto:

$$\begin{aligned} \delta(C_i, C_j) &= \Delta SSE_{ij} = \\ &= \sum_{z \in C_{ij}} \|z - \mu_{ij}\|^2 - \sum_{x \in C_i} \|x - \mu_i\|^2 - \sum_{y \in C_j} \|y - \mu_j\|^2 \\ &= \sum_{z \in C_{ij}} z^T z - n_{ij} \mu_{ij}^T \mu_{ij} - \sum_{x \in C_i} x^T x + n_i \mu_i^T \mu_i - \sum_{y \in C_j} y^T y + n_j \mu_j^T \mu_j \\ &= n_i \mu_i^T \mu_i + n_j \mu_j^T \mu_j - (n_i + n_j) \mu_{ij}^T \mu_{ij} \end{aligned} \tag{3.3}$$

El último paso es debido a que $\sum_{z \in C_{ij}} z^T z = \sum_{x \in C_i} x^T x + \sum_{y \in C_j} y^T y$. Teniendo en cuenta que $\mu_{ij} = \frac{n_i \mu_i + n_j \mu_j}{n_i + n_j}$, obtenemos:

$$\mu_{ij}^T \mu_{ij} = \frac{1}{(n_i + n_j)^2} \left(n_i^2 \mu_i^T \mu_i + 2 n_i n_j \mu_i^T \mu_j + n_j^2 \mu_j^T \mu_j \right)$$

Uniendo esto con la ecuación (3.3) obtenemos:

$$\begin{aligned}
\delta\left(C_i, C_j\right) &= \Delta \mathrm{SSE}_{ij} = \\
&= n_i \mu_i^{\mathrm{T}} \mu_i + n_j \mu_j^{\mathrm{T}} \mu_j - \frac{1}{\left(n_i + n_j\right)} \left(n_i^2 \mu_i^{\mathrm{T}} \mu_i + 2 n_i n_j \mu_i^{\mathrm{T}} \mu_j + n_j^2 \mu_j^{\mathrm{T}} \mu_j \right) \\
&= \frac{n_i \left(n_i + n_j\right) \mu_i^{\mathrm{T}} \mu_i + n_j \left(n_i + n_j\right) \mu_j^{\mathrm{T}} \mu_j - n_i^2 \mu_i^{\mathrm{T}} \mu_i - 2 n_i n_j \mu_i^{\mathrm{T}} \mu_j - n_j^2 \mu_j^{\mathrm{T}} \mu_j}{n_i + n_j} \\
&= \frac{n_i n_j \left(\mu_i^{\mathrm{T}} \mu_i - 2 \mu_i^{\mathrm{T}} \mu_j + \mu_j^{\mathrm{T}} \mu_j \right)}{n_i + n_j} \\
&= \left(\frac{n_i n_j}{n_i + n_j} \right) \left\| \mu_i - \mu_j \right\|^2
\end{aligned}$$

Por lo tanto, la medida de Ward no es más que una **versión ponderada** de la medida de distancia media, definida como distancia entre las medias o centroides de los dos clústeres, esto es:

$$\delta(\mu_i, \mu_j) = \|\mu_i - \mu_j\|^2$$

La única diferencia es que la medida de Ward pondera la distancia entre las medias mediante la mitad de la media armónica[1] del tamaño de los clústeres.

3.1.2. Clústering no jerárquico

Dado un conjunto de datos con n puntos en un espacio d-dimensional $D = \{x_1, ..., x_n\}$ y un número k de clústeres, el objetivo del clústering no jerárquico es hacer una partición de los datos en k clústeres, formando un clústering $\mathcal{C} = \{C_1, C_2, ..., C_k\}$. Además, para cada clúster C_i existe un punto representativo que lo resume [53]. En nuestro estudio este punto será la media μ_i (también conocido como centroide) de todos los puntos dentro del clúster, esto es:

$$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$$

donde $n_i = |C_i|$ es el número de puntos en el clúster C_i.

Es por ello que vamos a utilizar el algoritmo conocido como **k-medias** o *k-means* en inglés. El nombre de este método viene dado por la representación de cada uno de los clústeres por la media (o media ponderada) de sus puntos, la cual caracteriza a cada grupo y normalmente se encuentra en el centro o en medio de los elementos que lo componen [10].

La idea que hay detrás de este algoritmo consiste en definir clústeres de manera que se minimice la variación total en el interior del clúster (conocida en inglés como *total within-cluster variation*). Esta variación se define como la suma de las distancias euclídeas entre los elementos y el correspondiente centroide [37]:

[1] La media armónica de dos números n_1 y n_2 se define como $\frac{2}{\frac{1}{n_1} + \frac{1}{n_2}} = \frac{2 n_1 n_2}{n_1 + n_2}$

$$W(C_i) = \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

donde x_j es un punto de los datos que pertenece al clúster C_i y μ_i es el valor medio de los puntos pertenecientes al clúster C_i. Cada observación x_j se asigna a un clúster de forma que la suma de los cuadrados (*Sum of Squares, SS*) de la distancia del punto observado al centroide de su clúster sea mínima.

Por otro lado, se define la variación total dentro del clúster (*total within-cluster sum of squares)* como sigue, de forma que se mide la compacidad o bondad del clúster y queremos que este valor sea lo más pequeño posible:

$$tot.withiness = \sum_{i=1}^{k} W(C_i) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2 \tag{3.4}$$

Para explicar el algoritmo, podemos ver la Tabla 3.2 que muestra el pseudo-código del mismo. En él, tras elegir el número de clústeres deseado k, se seleccionan aleatoriamente k objetos del conjunto de datos como centroides iniciales. En segundo lugar, se asigna cada observación al centroide más cercano, basado en la distancia euclídea entre ellos. Para cada uno de los k clústeres se actualiza el centroide del clúster calculando el nuevo valor de la media de todos los datos en el clúster. Posteriormente, y de forma iterativa, se minimiza la variación total dentro del clúster (ecuación (3.4)). Es decir, se repiten los pasos de la asignación al centroide más cercano y la actualización del mismo en el clúster hasta que las asignaciones de clúster dejen de cambiar o se alcance el número máximo de iteraciones. Esta idea se formaliza parando las iteraciones si $\sum_{i=1}^{k} ||\mu_i^t - \mu_i^{t-1}||^2 \leq \epsilon$, donde $\epsilon > 0$ es el umbral de convergencia, t la iteración actual y μ_i^t la media para el clúster C_i en la iteración t. Cabe destacar que, dado que nuestro estudio es con el software R, este utiliza 10 como valor predeterminado para el número máximo de iteraciones.

	ALGORITMO K-MEDIAS
	k-medias(D,k, ϵ):
1	$t = 0$
2	Inicializa aleatoriamente k centroides: $\mu_1^t, \mu_2^t, ..., \mu_k^t \in \mathbb{R}^d$
3	Repite:
4	$t \leftarrow t + 1$
5	$C_j \leftarrow \emptyset$ para todo $j = 1, ..., k$
	//Se asigna clúster
6	Para cada $x_j \in D$:
7	$j* \leftarrow argmin_i\{||x_j - \mu_i||^2\}$ //Asigna x_j al centroide más cercano
8	$C_{j*} \leftarrow C_{j*} \cup \{x_j\}$
	//Se actualiza el centroide
9	Para cada $i = 1$ hasta k:
10	$\mu_i' \leftarrow \frac{1}{\|C_i\|} \sum_{x_j \in C_i} x_j$
11	hasta que $\sum_{i=1}^{k} ||\mu_i^t - \mu_i^{t-1}||^2 \leq \epsilon$

Tabla 3.2.: Algoritmo k-medias. [53]

3.1.3. Criterios para determinar el número de clústeres óptimo

El número k de clústeres tiene que ser predefinido al aplicar k-medias a nuestros datos. Es preferible siempre utilizar un valor que sea óptimo en nuestro estudio, pero esto no siempre es una tarea sencilla. Por eso se recurre a tres de los métodos más populares para buscar este valor: criterio del codo, método de Silhouette y el estadístico de Gap.

3.1.3.1. Criterio del codo

El primer método utilizado en el estudio es el **criterio del codo** o *elbow method*, el cual se encarga de analizar el porcentaje de varianza explicada en función del número de clústeres. La idea es que se debe elegir una cantidad de clústeres para que al agregar otro, el modelado de los datos no sea mucho mejor. Este porcentaje explicado se representa frente al número de clústeres. Los primeros clústeres añadirán bastante información, pero en algún momento la ganancia marginal caerá y dará un ángulo en el gráfico (asemejándose al dibujo de un codo, de ahí el nombre del criterio), de forma que el $\mathcal{K}$ obtenido será el punto de esa caída [9].

El algoritmo que sigue este criterio se puede definir de la siguiente manera [37]:

1. Se ejecuta el algoritmo de k-medias para diferentes valores de k (por ejemplo para $k = 1, ..., 10$).
2. Para cada k, se calcula la variación total dentro del clúster (*total within-cluster sum of squares*, ecuación (3.4)).
3. Se dibuja la curva del *tot.withiness* en función al número de clústeres k.
4. La posición de una curva más pronunciada en el gráfico se considera un indicador del número de clústeres apropiado.

3.1.3.2. Método de Silhouette

El segundo método que se va a utilizar es el conocido como **Silhouette**. Este utiliza la distancia media entre un dato y otros dentro del mismo clúster junto con la distancia entre los diferentes clústeres para medir el resultado final. Esta medida es la que se conoce como *coeficiente de Silhouette individual* (S_i) y se define de la siguiente manera [45]:

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

De forma que a_i representa la media de las distancias entre el dato i y el resto de datos dentro del mismo clúster, y b_i la distancia media del clúster más próximo. El coeficiente se encuentra en el intervalo $[-1, 1]$. Un valor cercano a 1 significa que el dato se corresponde correctamente con su clúster y no con otros, mientras que un valor cercano a -1 significa lo contrario. Si la mayoría de objetos tienen un coeficiente cercano a 1, podríamos decir que el clústering es apropiado.

De forma genérica, se define entonces el **coeficiente de Silhouette** como

$$SC = \frac{1}{n} \sum_{i=1}^{n} S_i$$

donde n representa el número de datos del estudio [54]. Para elegir el mejor valor de k (número de clústeres) se procede como sigue: se aplica el método de k-medias probando distintos valores de k (en nuestro estudio del 1 al 10) y se calcula el coeficiente de Silhouette, de forma que el número óptimo obtenido por este método será el que maximice el coeficiente, pues significará un clústering más apropiado.

3.1.3.3. Método de Gap

El tercer método que se utiliza es aquel que hace uso del estadístico de Gap. Este compara la variación total dentro del clúster para diferentes valores de k con sus valores esperados bajo una distribución de datos nula de referencia (por ejemplo, una distribución sin clústering obvio). El conjunto de datos referente se genera utilizando simulaciones de Montecarlo[2]. Esto es, para cada variable x_i en el conjunto de datos, se calcula su rango $[min(x_i), max(x_i)]$ y se generan valores para los n puntos uniformemente en este intervalo [37].

Para los datos observados y los de referencia, se calcula la variación total dentro del clúster utilizando diferentes valores de k.

Definición 3.5. Sea D_r la suma de las distancias euclídeas por pares para todos los puntos de un clúster C_r. Se define la *suma de cuadrados agrupada dentro del clúster alrededor de las medias del mismo* como:

$$W_k = \sum_{r=1}^{k} \frac{D_r}{|C_r|}$$

La idea es estandarizar el gráfico de $log(W_k)$ mediante la comparación del mismo con su esperanza bajo una distribución de referencia nula apropiada para los datos. La estimación del número óptimo de clústeres es, por tanto, el valor de k para el cual el $log(W_k)$ cae más por debajo de esta curva de referencia. Por lo tanto definimos [47]:

Definición 3.6. Dado k se define el estadístico de Gap como:

$$Gap_n(k) = E_n^*\{log(W_k)\} - log(W_k)$$

donde E_n^* representa la esperanza bajo un tamaño de muestra n de la distribución de referencia.

Nuestra estimación $\hat{k}$ será el valor que maximice $Gap_n(k)$ tras tener en cuenta la distribución de ejemplo. Cabe resaltar que esta estimación es muy general, aplicable a cualquier método de clústering y medida de distancia.

En nuestro caso, vamos a considerar la distribución de referencia de la siguiente manera, ya que es la más simple:

(a) Se genera cada variable de referencia de manera uniforme sobre el rango de los valores observados para esa variable.

Para la implementación del cómputo de este estadístico, se estima $E_n^*\{log(W_k)\}$ mediante la media de B copias, cada una de las cuales se calcula a partir de una muestra de Monte Carlo $X_1^*, X_2^*, ..., X_n^*$ extraída de nuestra distribución de referencia. Por último, se necesita evaluar la distribución muestral del estadístico de Gap.

[2] El método de Montecarlo permite resolver problemas matemáticos mediante la simulación de variables aleatorias. El fundamento del mismo y de la simulación son los números aleatorios: tablas de números aleatorios, generadores de números aleatorios o números pseudoaleatorios [12]

Definición 3.7. Sea $sd(k)$ la desviación típica de las B réplicas de Monte Carlo de $log(W_k^*)$. Si contabilizamos el error de simulación en $E_n^*\{log(W_k)\}$, obtenemos como resultado la cantidad denominada como s_k:

$$s_k = \sqrt{(1 + \frac{1}{B})sd(k)}$$

Usando esto, se elige el tamaño del clúster $\hat{k}$ de forma que sea el k mas pequeño tal que $Gap(k) \geq Gap(k+1) - s_{k+1}$.

El algoritmo para el cálculo del estadístico se explica en los siguientes pasos:

1. Se agrupan los datos observados en un clúster, variando el número de clústeres entre $k = 1, ..., K$ para entonces calcular el correspondiente W_k.

2. Se generan B conjuntos de datos de referencia utilizando lo comentado en (a) y se agrupan cada uno de ellos dando medidas de la dispersión dentro del clúster $W_{kb}^*, b = 1, 2, ..., B; k = 1, 2, ..., K$. Se calcula el estadístico de Gap estimado:

$$Gap(k) = (1/B) \sum_{b=1}^{B} log(W_{kb}^*) - log(W_k)$$

3. Sea $\bar{l} = (1/B) \sum_{b=1}^{B} log(W_{kb}^*)$, se calcula entonces la desviación típica

$$sd_k = \sqrt{(1/B) \sum_{b=1}^{B} log(W_{kb}^* - \bar{l})^2}$$

   y se define $s_k = sd_k \sqrt{1 + 1/B}$.

4. Se elige el número de clústeres como $\hat{k}$ = el menor k tal que $Gap(k) \geq Gap(k+1) - s_{k+1}$

En la Figura 3.2 se muestra un ejemplo utilizando el clústering K-medias. En la figura (a) podemos ver el conjunto de datos, el cual cae en dos clústeres distintos, en (b) se muestra la función de suma de cuadrados dentro del clúster W_k, en (c) las funciones $log(W_k)$ y $E_n^*\{log(W_k)\}$ (las cuales aparecen en el gráfico con puntos representados por O y E, respectivamente) y por último en (d) la curva de Gap junto con barras de error estándar de ± 1. Esta última tiene un máximo claro en $\hat{k} = 2$.

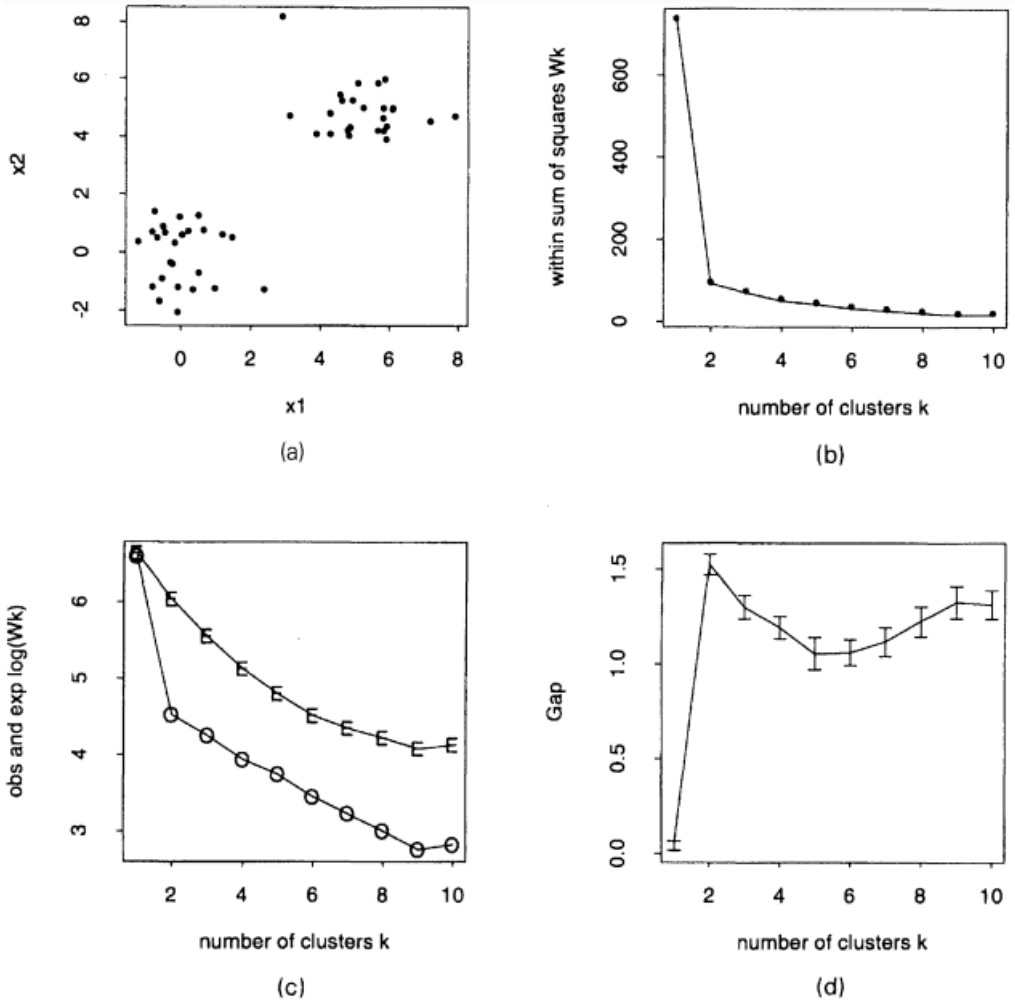


Figura 3.2.: Ejemplo cálculo del estadístico de Gap [47]

3.2. Reglas de asociación

Las reglas de asociación se utilizan para obtener nuevos patrones de objetos/atributos que suelen aparecer juntos a partir de estudiar bases de datos transaccionales. Específicamente, las reglas de asociación se pueden representar mediante implicaciones de la forma $X \rightarrow Y$, donde X e Y son dos conjuntos de ítems disjuntos. De una forma más intuitiva, podemos traducirlo como que si, en función de unos parámetros preestablecidos, encontramos todos los ítems del conjunto X, entonces esperamos encontrar todos los ítems del conjunto Y en un mismo documento. Esto permite establecer relaciones entre variables cualitativas. Cabe destacar que esta relación implica coocurrencia y no causalidad [31]. En este trabajo se hará uso del algoritmo Apriori para la obtención de reglas de asociación, pero es preciso señalar que existen otros algoritmos mejores. Algunos de ellos son el *ECLAT*, *Apriori-TID*, *FP-Growth*, aunque no los vamos a explicar debido al uso del algoritmo Apriori en la sección de experimentación por su simplicidad.

Definición 3.8. Sea $\mathcal{I} = \{i_1, i_2, ..., i_m\}$ un conjunto de elementos llamados ítems. Se denomina **conjunto de ítems** o *itemset* a cualquier conjunto $X \subseteq \mathcal{I}$. Se denota por $\mathcal{I}^{(k)}$ al conjunto de todos los conjuntos de ítems de tamaño k, esto es, los subconjuntos de $\mathcal{I}$ de tamaño k [53].

Definición 3.9. Se define $\mathcal{D}$ como el conjunto de transacciones donde cada transacción T es un conjunto de ítems tal que $T \subseteq \mathcal{I}$. Cada transacción tiene asociada un identificador llamado TID [2].

Definición 3.10. El conjunto de los identificadores de transacciones se le conoce como *tidset* [53]. El tidset para un conjunto de ítems X es el conjunto de todos los TIDs que contienen a X, esto es,

$$t(X) = \{t \in \mathcal{D} | X \text{ está contenido en } t\}$$

Así que podemos definir el soporte de un conjunto de ítems X como:

$$soporte(X) = \frac{|t(X)|}{|\mathcal{D}|}$$

donde $|\mathcal{D}| = N$ representa el número total de transacciones.

Definición 3.11. Se dice que una transacción T contiene un conjunto de ítems $X \subseteq \mathcal{I}$ si $X \subseteq T$.

Definición 3.12. Una **regla de asociación** se define como una expresión de la forma $X \to Y$, donde $X, Y \in \mathcal{I}$ y $X \cap Y = \emptyset$. Al conjunto de ítems a la izquierda de la expresión se le llama **antecedente** y se denomina **consecuente** al conjunto de ítems a la derecha de la regla.

Referenciamos previamente la existencia de unos parámetros que hay que preestablecer para encontrar las relaciones. Se trata del soporte y la confianza. Si tenemos un conjunto de ítems X, su soporte será la fracción de transacciones (tweets en nuestro estudio) que lo incluyen, mientras que la confianza se puede interpretar como la frecuencia con que una transacción que contiene el conjunto de ítems X también contenga el conjunto de ítems Y. Sea N el número de transacciones de la base de datos, podemos escribir las medidas de soporte y confianza como [31]:

$$\begin{aligned} soporte(X \to Y) &= soporte(X \cup Y) = \frac{count(X \cup Y)}{N} \\ conf(X \to Y) &= \frac{soporte(X \cup Y)}{soporte(X)} = \frac{soporte(X \to Y)}{soporte(X)} \end{aligned} \tag{3.5}$$

Observación 3.1. La regla $X \to Y$ se cumple en el conjunto de transacciones D con confianza c si el $c\,\%$ de las transacciones en D que contienen X también contienen a Y. Del mismo modo, la regla $X \to Y$ tiene soporte s en el conjunto de transacciones D si el $s\,\%$ de las transacciones en D contienen $X \cup Y$ [2].

Para obtener las reglas de asociación debemos establecer unos límites mínimos para estos dos parámetros. Además, se pretende en primer lugar encontrar conjuntos de ítems frecuentes que superen este soporte mínimo preestablecido o cuya concurrencia supera un mínimo de transacciones y, en segundo lugar, se pretende obtener las reglas de asociación asociadas a dichos conjuntos de ítems que superen esa confianza preestablecida.

3.2.1. Soporte y confianza

Vamos a estudiar en más profundidad estas dos medidas a partir de limitaciones como *medidas de precisión* y dando propiedades de cada una de ellas.

3.2.1.1. Confianza

La confianza es una **medida de precisión** de una regla. Cualquier medida de precisión MP debe verificar las tres propiedades específicas capaces de separar reglas fuertes de débiles, en el sentido de asignarles altos o bajos valores respectivamente [39]. Estas propiedades son:

P1 Cualquier medida de precisión debe probar la independencia, esto es, $MP(X \to Y) = 0$ cuando $soporte(X \to Y) = soporte(X)soporte(Y)$, aunque se pueden utilizar valores distintos de 0, dependiendo del rango de la medida.

P2 $MP(X \to Y)$ aumenta de forma monótona cuando el $soporte(X \to Y)$ lo hace y otros parámetros se mantienen.

P3 $MP(X \to Y)$ decrece de forma monótona cuando lo hace el $soporte(X)$ (o $soporte(Y)$) y otros parámentros se mantienen.

Sin embargo, vamos a ver como la confianza no verifica todas las propiedades [8]:

Proposición 3.1. *La confianza no verifica la propiedad* ***P1****.*

Demostración. Vamos a dar un contraejemplo. Sea $I_1 = \{i_1, i_2, i_3, i_4\}$ un conjunto de ítems, y R_1 el conjunto mostrado en la tabla A de la Figura 3.3. Las columnas representan los ítems y las filas las transacciones. Si una celda contiene 1, esto significa que el ítem correspondiente está presente en la transacción de la fila. En la tabla B de la misma figura, se muestra el soporte de tres conjuntos de ítems pertenecientes a I_1. Como $soporte(\{i_1\})soporte(\{i_2\}) = 1/3 = soporte(\{i_1, i_2\})$, entonces i_1, i_2 son independientes estadísticamente y por lo tanto la confianza debería ser nula. Sin embargo, $conf(\{i_1\} \to \{i_2\}) = \frac{1/3}{1/2} = \frac{2}{3} \neq 0$. □

Proposición 3.2. *La confianza verifica la propiedad* ***P2****.*

Demostración. Trivial teniendo en cuenta la ecuación (3.5) □

Proposición 3.3. *La confianza verifica la propiedad* ***P3*** *solo para el* $soporte(X)$*.*

Demostración. Es fácil verlo para el $soporte(X)$ según la ecuación (3.5). También es trivial ver que **P3** no se verifica para el $soporte(Y)$ ya que este no aparece siquiera en la ecuación que estamos referenciando, y el $soporte(X \cup Y)$ y $soporte(X)$ se mantienen conforme las condiciones de la propiedad **P3**. □

Con esto hemos demostrado que la confianza no se puede utilizar para estudiar la independencia estadística entre ítems, ya que el soporte del consecuente no influye en ella.

A

i_1	i_2	i_3	i_4
1	0	1	0
0	0	0	1
0	1	1	1
0	1	1	1
1	1	1	1
1	1	1	1

B

Itemset	Support
$\{i_1\}$	1/2
$\{i_2\}$	2/3
$\{i_1, i_2\}$	1/3

Figura 3.3.: Conjunto de ítems, transacciones y soporte para contraejemplo [8].

3.2.1.2. Soporte

En las reglas de asociación se suele pensar que a mayor soporte, mejor conjunto de ítems obtendremos. Sin embargo, hay conjuntos de ítems con un soporte alto que pueden ser engañosos ya que aparecen en muchas de las transacciones y, por lo tanto, cualquier conjunto de ítems (independientemente de su significado) podría ser útil para predecir la presencia de este conjunto de ítems con alto soporte [8].

Un ejemplo de lo que mencionamos lo podemos ver en la tabla A de la Figura 3.3. Cualquier conjunto de ítems que contenga solo a $\{i_1\}$ o $\{i_2\}$ puede predecir al conjunto $\{i_3\}$, ya que cuando aparece $\{i_1\}$, lo hace $\{i_3\}$, y del mismo modo ocurre con $\{i_2\}$. Por consecuente, si también aparecen $\{i_1, i_2\}$, en la tabla podemos observar la ocurrencia de $\{i_3\}$. Además, $conf(\{i_4\} \rightarrow \{i_3\}) = 0.8$, un valor alto, pero no podemos asegurar que estas reglas sean fiables.

Estos problemas pueden resolverse usando medidas de precisión que verifiquen las propiedades **P1, P2** y **P3**, resolviendo los problemas de que la confianza de una regla $X \rightarrow Y$ sea menor o igual que el soporte del consecuente ($soporte(Y)$), lo que conlleva una independencia estadística. En el caso contrario, cuando el soporte de Y es alto y la $conf(X \rightarrow Y) > soporte(Y)$, podemos obtener una buena precisión. Sin embargo, hay poca variabilidad en los datos sobre la presencia de Y que no nos permite estar seguros de la regla obtenida. Por suerte, esta situación puede detectarse comprobando que el $soporte(Y)$ es bajo, aunque no hay ningún método incorporado en la búsqueda de reglas de asociación que incluya esto. Estos problemas nos llevan a tener que recoger muchas más reglas de las que se deberían y es por ello que se podrían utilizar otras medidas que resuelvan este problema, como lo son los factores de certeza de *Shortliffe* y *Buchanan*, que no se contemplan en este trabajo.

3.2.2. Lift

El soporte y la confianza son medidas necesarias para la búsqueda de reglas de asociación. Sin embargo, existe otra medida llamada *lift* que se encarga de valorar cómo de "fiable" es una regla. Podemos definir el lift de una regla como:

$$L(X \to Y) = \frac{conf(X \to Y)}{soporte(Y)} = \frac{soporte(X \to Y)}{soporte(X)soporte(Y)}$$

En otras palabras, el lift mide en qué grado A y B no son independientes [32].
Al calcular su valor tenemos tres posibilidades [22]:

1. Si el lift es mayor que 1, entonces la correlación es positiva.
2. Cuando es menor que 1, la correlación es negativa.
3. Si su valor es 1, entonces la correlación es independiente.

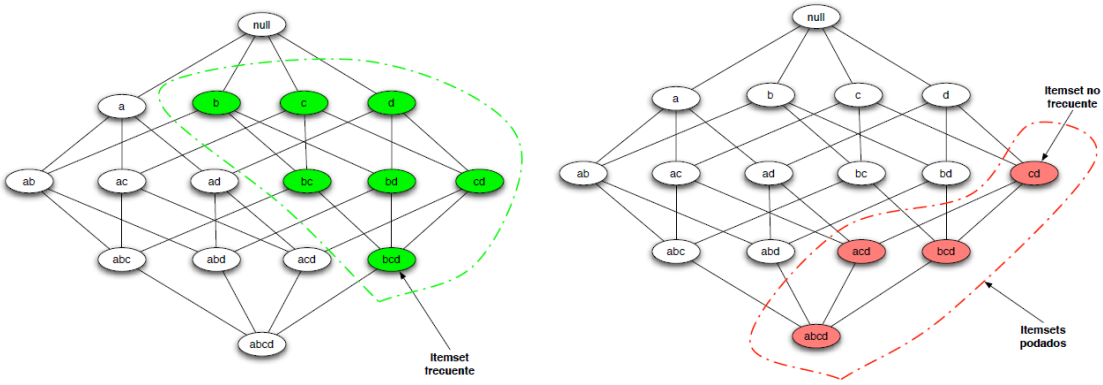
Es decir, un lift mayor que 1 indica una fuerte correlación entre X e Y, mientras que si su valor es 1, esto nos dice que $soporte(X \to Y) = soporte(X \cup Y) = soporte(X)soporte(Y)$. En términos de probabilidad esto significa que la ocurrencia de X y la ocurrencia de Y en la misma transacción son eventos independientes. Por consiguiente, X e Y no están correlados. De esta forma, lift resuelve un problema de impacto del antecedente sobre el consecuente en las reglas de asociación; funcionará para determinar el tipo de impacto de los datos entre sí y, por tanto, clasificará las correlaciones.

3.2.3. Algoritmo Apriori

El algoritmo *Apriori* es uno de los algoritmos más usados que nos permiten generar reglas de asociación entre conjuntos de ítems a partir de los que se consideran frecuentes y fue propuesto por Agrawal et al. en 1993 [1]. La máxima en este algoritmo se basa en que todos los subconjuntos no vacíos de un conjunto de ítems frecuente, también son frecuentes. Por ejemplo, si el conjunto de ítems {b,c} es frecuente, puedo suponer que {b} también lo es (como mínimo aparecerá el mismo número de veces que {b,c}). Del mismo modo, si {c,d} es no frecuente, {b,c,d} tampoco lo será. Esta propiedad se conoce como antimonotonicidad, y permite hacer una poda en el espacio de búsqueda basada en el soporte [31]. Una imagen explicativa de lo comentado se puede observar en la Figura 3.4.

Por otro lado, si tenemos un conjunto de ítems $A = \{X, Y, Z, W\}$, nos podemos preguntar si la confianza de una regla es antimonótona. La respuesta es negativa, pues la confizanza de una regla $XYZ \to W$ puede ser mayor o menor que la confianza de $XY \to W$. Sin embargo, la confianza de las reglas generadas en un mismo conjunto de ítems sí que es antimonótona, ya que se cumple que $conf(XYZ \to W) \geq conf(XY \to ZW) \geq conf(X \to YZW)$[3]. Así que podemos decir que la confianza es **antimonótona** con respecto al número de ítems en la parte derecha de la regla [6].

[3] $XYZ \equiv X \cup Y \cup Z$, $XY \equiv X \cup Y$, $ZW \equiv Z \cup W$ y $YZW \equiv Y \cup Z \cup W$.



(a) Itemsets frecuentes

(b) Itemsets no frecuentes

Figura 3.4.: Explicación itemsets frecuentes y no frecuentes

3.2.3.1. Obtención de conjuntos de ítems frecuentes y reglas de asociación

El algoritmo Apriori genera los conjuntos de ítems frecuentes en base a la siguiente explicación y posterior pseudo-código.

Sea C_k el conjunto de k conjuntos de ítems frecuentes que son candidatos, y F_k el conjunto de conjuntos de ítems frecuentes. Se comienza obteniendo los conjuntos de ítems frecuentes de tamaño 1 ($k = 1$). Iterativamente, se van generando el resto de los k conjuntos de ítems candidatos a partir de los frecuentes de tamaño anterior ($k - 1$), combinándolos entre sí. De estos nuevos candidatos se vuelven a buscar los que son frecuentes en función al soporte. Los que no superan los límites mínimos se podan (antimonotonicidad del soporte). De C_k se selecciona el subconjunto F_k y el proceso termina cuando ya no se generan más k conjuntos de ítems frecuentes. Un pseudo-código se puede observar en la Figura 3.5 [31].

Algoritmo Apriori: Generación de itemsets frecuente

1: $k = 1$;
2: $F_k = \{i | i \in I \wedge \sigma(\{i\}) \geq N \times minSup\}$ ▷ Encontrar itemsets frecuentes
3: **repetir**
4: $k = k + 1$;
5: $C_k = Apriori\_Gen(F_{k-1})$; ▷ Generar itemsets candidatos
6: **para cada** transacción $t \in T$ **hacer**
7: $C_t = subset(C_k, t)$ ▷ itemsets candidatos en t
8: **para cada** itemset $c \in C_t$ **hacer**
9: $\sigma(c) = \sigma(c) + 1$ ▷ Incrementar el contador de soporte
10: **fin para**
11: **fin para**
12: $F_k = \{c | c \in C_k \wedge \sigma(c) \geq N \times minSup\}$
13: **hasta que** $F_k = \emptyset$
14: **Devolver** $\bigcup F_k$

Figura 3.5.: Pseudo-código para la obtención de conjuntos ítems frecuentes [31]

Una vez obtenidos los conjuntos de ítems frecuentes, pasamos a la obtención de las reglas,

cuyo pseudo-código se puede ver en la Figura 3.6. La idea detrás es básicamente la siguiente: para cada conjunto de ítems frecuente X, se obtienen todos los posibles subconjuntos S del mismo, y entonces se crea la regla $(X - S) \rightarrow S$. Si la regla generada no supera la confianza mínima establecida, se descarta. Más específicamente, se empieza haciendo una evaluación de las reglas que tienen un solo ítem en el consecuente y se mantienen las que superan la confianza mínima establecida. Se procede de igual manera con las de tamaño 2 y así sucesivamente. Se comprueba si el tamaño de los consecuentes es menor que el tamaño del conjunto de ítems y se generan los consecuentes de tamaño siguiente. Se prosigue calculando la confianza y si esta supera la mínima establecida, se genera la regla, en caso contrario, se elimina el consecuente.

Algoritmo Apriori: Generación de reglas en Apriori

```
1: para cada k−itemset frecuente f_k, k ≥ 2 hacer
2:     H_1 = {i | i ∈ f_k}                    ▷ consecuentes de un item
3:     ejecutar ap-genrules(f_k, H_1)
4: fin para
```

Algoritmo ap-genrules(f_k, H_m)

```
1: k = |f_k|;                                 ▷ Tamaño del itemset frecuente
2: m = |H_m|                                  ▷ Tamaño del consecuente
3: si k > m + 1 entonces
4:     H_{m+1} =apriori-gen(H_m)
5:     para cada h_{m+1} ∈ H_{m+1} hacer
6:         conf = σ(f_k)/σ(f_k \ h_{m+1})
7:         si conf ≥ minconf entonces
8:             generar la regla (f_k \ h_{m+1}) → h_{m+1}
9:         sino
10:            H_{m+1} = H_{m+1} \ h_{m+1}
11:        fin si
12:    fin para
13:    ejecutar ap-genrules(f_k, H_{m+1})
14: fin si
```

Figura 3.6.: Pseudo-código para la obtención de reglas de asociación [31]

Del mismo modo y llegados a este punto, podemos también obtener lo que se conoce como **reglas maximales**: aquellas generadas por conjuntos de ítems maximales. Un conjunto de ítems es maximal cuando ninguno de sus *superconjuntos* (conjuntos de ítems de tamaño k que contengan al de tamaño k-1) es frecuente.

3.2.3.2. Evaluación con test exacto de Fisher

Contamos con numerosas métricas que podemos utilizar para evaluar las reglas obtenidas. En el presente trabajo se va a calcular el test exacto de Fisher, un test de significancia para ver si las reglas representan patrones reales. Este nos dará como resultado un valor de p, que indicará la probabilidad de obtener una diferencia entre los grupos bajo la hipótesis nula de independencia. Si esta probabilidad es pequeña ($p<0.05$) se rechaza la hipótesis y se asume que las variables no son independientes. En caso contrario, se dirá que no existe evidencia estadística de asociación entre ambas variables. Hay diferentes métodos para calcular el

valor de p y no tienen por qué dar el mismo número. En cualquier caso, los resultados deberían ser evidentes para aceptar o rechazar la hipótesis.

Formalicemos esta idea dando unas nociones previas [33]:

Dada una tabla de contingencia X de dimensiones $r \times c$, sea x_{ij} la entrada de la fila i y columna j. Sea $R_i = \sum_{j=1}^{c} x_{ij}$ la suma de todas las entradas de la fila i, y $C_j = \sum_{i=1}^{r} x_{ij}$ la suma de todas las entradas de la columna j. Se asume que los x_{ij} y sus sumas parciales son enteros no negativos.

Definición 3.13. Sea $\mathcal{T}$ el conjunto de referencia de todas las posibles tablas de contingencia de tamaño $r \times c$ con los mismos totales marginales que X. Esto es:

$$\mathcal{T} = \left\{ Y : Y \text{ es de dimensión } r \times c, R_i = \sum_{j=1}^{c} y_{ij}, C_j = \sum_{i=1}^{r} y_{ij} \right\}$$

Bajo la hipótesis nula de independencia de filas y columnas[4], la probabilidad de observar cualquier $Y \in \mathcal{T}$ se puede expresar como un producto de coeficientes multinomiales:

$$P(Y) = \left(\prod_{j=1}^{c} \frac{C_j!}{y_{1j}! y_{2j}! \ldots y_{rj}!} \right) \Big/ \frac{T!}{R_1! R_2! \ldots R_r!}$$

donde $T = \sum_{i=1}^{r} R_i$.

Definición 3.14. El nivel exacto de significancia o p **valor** asociado a una tabla X observada se define como la suma de probabilidades de todas las tablas en $\mathcal{T}$ que no son más probables que X. Esto es:

$$p = \sum_{Y \in \mathcal{J}} P(Y)$$

donde $\mathcal{J} = \{Y : Y \in \mathcal{T} \text{ y } P(Y) \leq P(X)\}$.

Y solo habría que evaluar p. Cuanto más pequeño es el valor, más fuerte es la evidencia y por tanto rechazaríamos la hipótesis nula. Podemos pasar entonces a definir una tabla de contingencia 2×2 y formalizarlo para reglas de asociación, ya que el test exacto de Fisher se utiliza normalmente para tablas de estas dimensiones.

El test que trabajamos nos dice si una regla $R : X \rightarrow Y$ se produce realmente mediante la comparación de su confianza con cada una de sus generalizaciones $R' : W \rightarrow Y$, donde $W = X \setminus Z, Z \subseteq X$ e incluyendo la regla trivial $\emptyset \rightarrow Y$. Dado un conjunto de datos D, donde W aparece, podemos crear una tabla de contingencia de tamaño 2×2 entre Z y el consecuente Y como en la Tabla 3.3. Los valores de las celdas son[5] [53]:

$a = soporte(W \cup Z \cup Y) = soporte(X \cup Y)$ $\quad$ $b = soporte(W \cup Z \cup \neg Y) = soporte(X \cup \neg Y)$
$c = soporte(W \cup \neg Z \cup Y)$ $\quad$ $d = soporte(W \cup \neg Z \cup \neg Y)$

donde:

- a representa el número de transacciones que contienen a X e Y.
- b denota el número de transacciones que contienen a X pero no a Y.

[4] La hipótesis nula de independencia entre filas y columnas H_0 supone que el elemento de la fila i y la columna j son independientes

[5] Destacar que $\neg Z$ son todos los ítems que no están en el conjunto Z

- c es el número de transacciones que contienen W e Y pero no Z.
- d muestra el número de transacciones que contienen W pero no Z ni Y.

Por otro lado tenemos:

$$\text{marginales de fila: } a+b = soporte(W \cup Z) = soporte(X), \quad c+d = soporte(W \cup \neg Z)$$
$$\text{marginales de columna: } a+c = soporte(W \cup Y), \quad b+d = soporte(W \cup \neg Y)$$

donde los marginales de fila dan la frecuencia de ocurrencia de W con y sin Z, y los marginales de columna especifican el recuento de ocurrencias de W con y sin Y. Por último la suma de todas las celdas es $n = a+b+c+d = soporte(W)$.

W	Y	$\neg Y$	
Z	a	b	$a+b$
$\neg Z$	c	d	$c+d$
	$a+c$	$b+d$	$n = soporte(W)$

Tabla 3.3.: Tabla de contingencia para Z e Y, condicional en $W = X \setminus Z$ [53]

Se procede entonces a calcular el p-valor para esta tabla de contingencia y compararlo con el nivel de significación establecido para el test, 0.05 si se utiliza la significación estándar, de forma que se podría rechazar la hipótesis nula y, por tanto, las reglas obtenidas reflejarían patrones reales.

3.3. Clasificación mediante máquinas de vector soporte

En este apartado se explica cómo construir un modelo de aprendizaje estadístico basado en las máquinas de vector soporte para deducir la autoría de los tweets. Este método se utilizó en su comienzo para la clasificación binaria, pero hoy en día se utiliza para clasificación múltiple y regresión [43].

3.3.1. Hiperplano y Clasificador de margen máximo

Definición 3.15. En un espacio d-dimensional, definimos un **hiperplano** (h) como un subespacio afín de dimensión $d-1$. Además, para un vector de escalares dado $w = (a_1, ..., a_d)$ y un parámetro a_0, este cumple la siguiente ecuación:

$$h(x) = a_0 + a_1x_1 + a_2x_2 + ... + a_dx_d = w^Tx + a_0 = 0 \tag{3.6}$$

Por lo tanto, todos los puntos $x \in \mathbb{R}^d$ que cumplen la ecuación anterior pertenecen al hiperplano.

Cuando un punto no satisfaga la ecuación, será porque la expresión es mayor o menor que 0, ($\neq 0$). Eso significa que el punto dado por el vector estará a un lado u otro del hiperplano, por lo que podemos verlo como divisor de un espacio de dimensión d en dos partes [43].

Supongamos ahora que estamos en una situación en la que podemos hacer uso de un hiperplano para separar perfectamente un conjunto de n datos. Los puntos del espacio que no pertenecen al hiperplano verifican entonces una de las dos ecuaciones:

$$a_0 + a_1x_1 + a_2x_2 + ... + a_dx_d > 0$$
$$a_0 + a_1x_1 + a_2x_2 + ... + a_dx_d < 0$$

De forma que el hiperplano puede actuar como separador de puntos en dos clases en función del lado en el que se encuentren del mismo: $+1$ y -1. Esto es:

$$y_i = \begin{cases} +1 & \text{si } h(x) > 0 \\ -1 & \text{si } h(x) < 0 \end{cases} \quad con\ i = 1, ..., n$$

Así que podemos simplificar esta idea en una ecuación:

$$y_i(a_0 + a_1x_1 + a_2x_2 + ... + a_dx_d) > 0\ con\ i = 1, ..., n$$

Por lo tanto, el clasificador más simple consiste en asignar cada observación a una clase en función del lado del hiperplano en el que se encuentre, siendo esta $+1$ ó -1.

Podemos estudiar la lejanía de una observación del hiperplano y, por tanto, la confianza de la clasificación.

Definición 3.16. Sea un punto $x \in \mathbb{R}^d$ tal que x no pertenece al hiperplano h y x_p es la proyección ortogonal de x en el hiperplano. Sea $r = x - x_p$, podemos escribir x como:

$$x = x_p + r$$

$$x = x_p + r\frac{w}{||w||} \tag{3.7}$$

donde r es la distancia directa del punto x a x_p, esto es, r da la compensación de x desde x_p en términos del vector de peso unitario $\frac{w}{||w||}$. r será positiva si está en la misma dirección que w, y negativa si r está en dirección contraria a w [53].

Podemos ver gráficamente esta idea en la Figura 3.7.

Si ahora utilizamos las ecuaciones (3.6) y (3.7), obtenemos:

$$\begin{aligned} h(x) &= h(x_p + r\frac{w}{||w||}) \\ &= w^T(x_p + r\frac{w}{||w||}) + a_0 \\ &= w^Tx_p + a_0 + r\frac{w^Tw}{||w||} \\ &= h(x_p) + r||w|| \\ &= r||w|| \end{aligned}$$

Usando este resultado, tenemos una expresión para la distancia directa de un punto al hiperplano:

$$r = \frac{h(x)}{||w||}$$

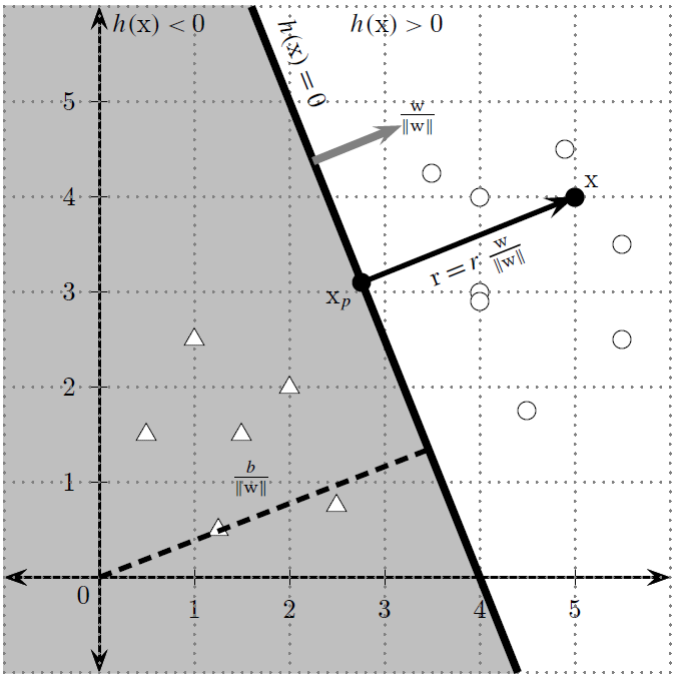


Figura 3.7.: Representación gráfica de la distancia de un punto a un hiperplano [53]

Observación 3.2. Necesitamos que la distancia sea positiva, así que es conveniente multiplicar r por la clase y_i del punto ($\{+1, -1\}$), por lo que la expresión de la distancia de un punto x al hiperplano viene dada por:

$$\delta = y_i r = \frac{y_i h(x)}{||w||}$$

Por otro lado, el número de hiperplanos posibles puede ser infinito, así que se necesita seleccionar uno que sea óptimo. Es aquí donde entra un concepto llamado **hiperplano óptimo de separación** (*maximal margin hyperplane*), el cual es el hiperplano más distante de todas las observaciones del entrenamiento. Para encontrarlo, se calcula la distancia de cada observación al plano, donde la menor de estas distancias (el margen) determina la lejanía del hiperplano con las observaciones. El hiperplano óptimo de separación se define como aquel que tiene mayor margen. Esta idea no se puede aplicar, ya que habría infinitos hiperplanos con los que medir las distancias, es por ello que se recurre a métodos de optimización que no contemplamos en este trabajo.

En la Figura 3.8 podemos observar el hiperplano óptimo de una muestra, de forma que las tres observaciones equidistantes respecto al hiperplano óptimo son los llamados **vectores soporte**. Su nombre se debe a que son vectores d-dimensionales que *soportan* y definen el margen máximo. La importancia de estos vectores reside en la modificación por consecuente del hiperplano óptimo de separación que se genera al aplicar cambios en estos vectores. Formalizamos estas ideas.

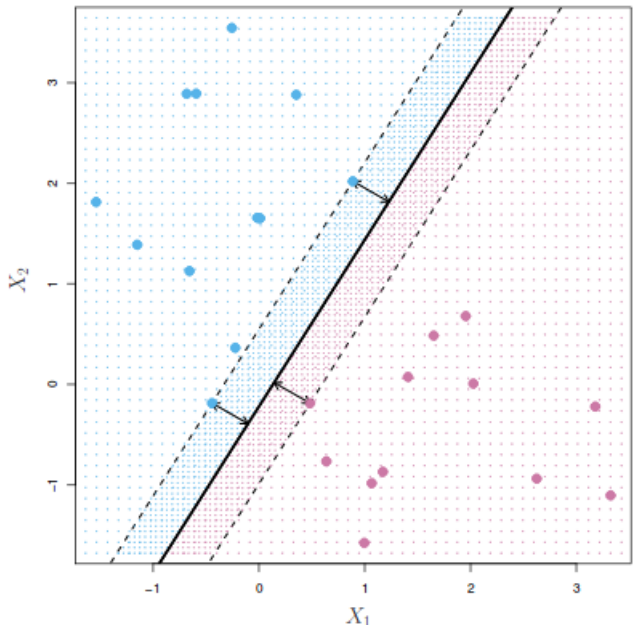


Figura 3.8.: Hiperplano óptimo de una muestra [43]

Definición 3.17. Se define el **margen** de un clasificador lineal como la distancia mínima de un punto $x_i \notin h$ al hiperplano de separación, esto es:

$$\delta^* = \min_{x_i} \left\{ \frac{y_i h(x_i)}{||w||} \right\}$$

Observación 3.3. $\delta_* \neq 0$ ya que $h(x)$ se supone que es un hiperplano de separación.

Definición 3.18. Todos los vectores x que alcancen esta distancia mínima son denominados **vectores soporte**. En otras palabras, un vector soporte x^* es un punto que se encuentra precisamente en el margen del clasificador y satisface la condición:

$$\delta^* = \frac{y^* h(x^*)}{||w||}$$

donde y^* es la clase para x^*. El numerador da la distancia absoluta del vector soporte al hiperplano, mientras que el denominador la convierte en una distancia relativa en términos de w.

Para encontrar el **hiperplano canónico**, este debe cumplir que $y^* h(x^*) = 1$ para un vector soporte x^*, y por tanto el margen viene dado por:

$$\delta^* = \frac{y^* h(x^*)}{||w||} = \frac{1}{||w||}$$

Para cada vector soporte x_i^* con clase y_i^*, tenemos $y_i^* h(x_i^*) = 1$ y para cualquier punto que no es un vector soporte tenemos que $y_i h(x_i) > 1$ ya que, por definición, debe estar más alejado del hiperplano que un vector soporte.

Es en este momento donde se aplicarían las técnicas de optimización para encontrar el hiperplano óptimo con mayor margen.

3.3.2. Clasificador de vector soporte o margen suave

Todo lo explicado en la sección previa ha sido bajo el supuesto de poder separar los datos con un hiperplano, pero la realidad es que esto no ocurre normalmente, por lo que no existiría un hiperplano de separación. Para ello, se extiende el concepto de este hiperplano de manera que permita errores de clasificación bajo un límite establecido. A este nuevo concepto se le denomina **clasificador de vector soporte** (*support vector classifier*) o margen suave (*soft margin*) [43].

El límite que se establece es un hiperparámetro C, el cual controla el número y el rigor de las violaciones del margen e hiperplano que se toleran. Si $C = \infty$, ninguna violación del margen se permitirá y el resultado obtenido será parejo al hiperplano óptimo de separación, el cual sólo se puede obtener si los resultados se pueden separar. Al disminuir el valor de C, más grande es el margen, menos se penalizarán los errores y un mayor número de observaciones estará dentro del margen o al otro lado del hiperplano.

Pasemos entonces a formalizar estas ideas. Para ello, necesitamos hacer uso de una nueva variable ξ_i de forma que:

$$y_i h(x_i) \geq 1 - \xi_i$$

donde $\xi_i \geq 0$ es la variable de holgura para un punto x_i, la cual indica cúanto viola el margen de separación, esto es, el punto puede que esté más cercano al hiperplano que $\frac{1}{||w||}$. Los valores de la variable en cuestión indican tres tipos de puntos:

- Si $\xi_i = 0 \Rightarrow$ el punto correspondiente x_i está al menos a una distancia de $\frac{1}{||w||}$ del hiperplano.
- Si $0 < \xi_i < 1 \Rightarrow$ el punto está entre el margen y correctamente clasificado, esto es, está en el lado correcto del hiperplano.
- Si $\xi_i \geq 1 \Rightarrow$ el punto está mal clasificado y aparece en el lugar incorrecto del hiperplano.

El objetivo entonces es encontrar el hiperplano con margen máximo que del mismo modo minimiza los términos de holgura, esto es:

$$\min_{w,\xi_i} \left\{ \frac{||w||^2}{2} + C \sum_{i=1}^{n} (\xi_i)^k \right\}$$

donde C y k son constantes que penalizan por clasificaciones erróneas. El término $\sum_{i=1}^{n} (\xi_i)^k$ da la pérdida, es decir, una estimación de la desviación del caso separable. El escalar C, el cual se elige empíricamente, es una constante de la regularización que controla la compensación entre maximizar el margen (minimizar $\frac{||w||^2}{2}$) o minimizar la pérdida. Es aquí entonces donde vemos que si $C \to 0$, la pérdida casi desaparece y el objetivo es maximizar el margen. Por otro lado, si $C \to \infty$, el margen deja de tener mucho efecto y la finalidad sería minimizar la pérdida. La constante k mide la forma de pérdida y normalmente se le da los valores 1 ó 2. Cuando $k = 1$, el objetivo es minimizar la suma de las variables de holgura, mientras que si $k = 2$ (también llamado pérdida cuadrática), se pretende minimizar la suma de las variables de holgura cuadráticas [53]. Para ello se utilizan multiplicadores de Lagrange que no veremos en este trabajo.

El problema que tenemos entonces a resolver es la búsqueda del valor óptimo de C, el cual se encuentra mediante un proceso conocido como **validación cruzada** (*cross-validation*), una técnica usada, entre otras cosas, para la evaluación de los resultados de un análisis estadístico y la estimación de la precisión del modelo utilizado [43].

La validación cruzada divide el conjunto de datos D en K partes del mismo tamaño $D_1, D_2, ..., D_K$ llamados *folds*. Cada uno de ellos se trata, a su vez, como el conjunto de prueba y los folds restantes forman el conjunto de entrenamiento $D \setminus D_i = \bigcup_{j \neq i} D_j$. Tras entrenar el modelo M_i sobre $D \setminus D_i$, evaluamos su rendimiento en el conjunto de pruebas D_i para obtener la i-ésima estimación θ_i. El valor esperado de la medida del rendimiento puede estimarse como:

$$\hat{\mu}_\theta = \mathrm{E}[\theta] = \frac{1}{\mathrm{K}} \sum_{i=1}^{\mathrm{K}} \theta_i$$

y su varianza como:

$$\hat{\sigma}_\theta^2 = \frac{1}{\mathrm{K}} \sum_{i=1}^{\mathrm{K}} (\theta_i - \hat{\mu}_\theta)^2$$

En la Tabla 3.4 podemos ver el pseudo-código para la validación cruzada K-fold, también conocido como validación cruzada de k iteraciones. Tras barajar aleatoriamente el conjunto de datos D, se particiona en K folds iguales (excepto quizás el último). Posteriormente, cada fold D_i se usa como el conjunto de test sobre el cual se evalúa el rendimiento θ_i del clasificador M_i entrenado en $D \setminus D_i$. La media y varianza estimada de θ pueden reportarse. La validación cruzada K-fold puede repetirse numerosas veces; al barajar aleatoriamente en el comienzo asegura que los folds son diferentes en cada paso.

	ALGORITMO VALIDACIÓN CRUZADA K-FOLD
	Cross-Validation(K,D):
1	$D \leftarrow$ baraja aleatoriamente D
2	$\{D_1, D_2, ..., D_K\} \leftarrow$ partición de D en K partes iguales
3	para cada $i \in [1, K]$:
4	$\quad M_i \leftarrow$ entrena clasificador en $D \setminus D_i$
5	$\quad \theta_i \leftarrow$ evalúa M_i sobre D_i
6	$\hat{\mu}_\theta = \mathrm{E}[\theta] = \frac{1}{\mathrm{K}} \sum_{i=1}^{\mathrm{K}} \theta_i$
7	$\hat{\sigma}_\theta^2 = \frac{1}{\mathrm{K}} \sum_{i=1}^{\mathrm{K}} (\theta_i - \hat{\mu}_\theta)^2$
8	return $\hat{\mu}_\theta$,$\hat{\sigma}_\theta^2$

Tabla 3.4.: Algoritmo validación cruzada K-fold

Normalmente K toma los valores 5 ó 10. Cuando $K = n$, se denomina validación cruzada *leave-one-out*, donde el conjunto de test comprende un único punto y los datos restantes se usan con propósitos de entrenamiento [53].

3.4. Análisis de sentimientos

En la Sección 2.7 se hizo una introducción al análisis de sentimientos o minería de opinión que recogía algunos de los estudios que se pueden hacer con ella. En este trabajo se pretende hacer un acercamiento basado en un *lexicon* (diccionario de palabras) y determinar la

polaridad de la información recogida. Es por ello que dentro de esta sección se procederá a hacer un estudio de las palabras extraídas, las cuales se clasificarán en función a ocho sentimientos, explicados más a fondo en la Subsección 4.6.2 y se estudiará, además, la polaridad de los términos.

Posteriormente se aplicarán diferentes técnicas haciendo un recuento de palabras con cada emoción, así como una nube de sentimientos. Por último se representa una evolución en la polaridad de los términos a lo largo del tiempo que se muestra en función a tres variables: *loess*, media móvil o *rolling mean* y la transformada discreta de coseno o *syuzhet DCT (Discrete cosine transformation)*, de las cuales se habla en profundidad a continuación. Estas variables son las encargadas de suavizar y normalizar la curva de la evolución a lo largo del tiempo.

3.4.1. Loess smooth

Loess es una estrategia simple para ajustar curvas suaves a datos empíricos. El término "loess" viene del acrónimo en inglés "local regression" y todo el procedimiento es una generalización directa de los métodos tradicionales de mínimos cuadrados para el análisis de datos. Esta estrategia no es paramétrica en el sentido de que la técnica de ajuste no requiere una especificación previa de la relación entre las variables dependientes e independientes. Normalmente se usa como un suavizante de diagramas, pero se puede generalizar a datos multivariantes; existen procedimientos de inferencia para los intervalos de confianza y otros tests estadísticos. Es por ello que loess es particularmente útil en el campo de las elecciones y comportamiento electoral, ya que las teorías a menudo conducen a expectativas de relaciones empíricas no lineales aunque las consideraciones previas ofrecen poca orientación sobre formas precisas de proceder [23].

El procedimiento que tratamos hemos dicho que es no paramétrico. Sin embargo, hay algunos parámetros que deben ser dados antes del ajuste para garantizar que la curva loess pasa por el centro de los puntos de los datos empíricos. El hecho de seleccionar los valores para estos parámetros es un proceso subjetivo, pero las consideraciones que hay que tener en cuenta son sencillas.

3.4.1.1. Parámetro de suavidad α

En un ajuste loess, el parámetro α determina la anchura de la ventana deslizante. Esto es, α da una proporción de las observaciones que se utilizará en cada regresión local. En consecuencia, este parámetro se especifica como un valor entre 0 y 1. En la función *simple_plot* de la librería *syuzhet* que usaremos en la parte experimental, este valor será de 0.1 [24]. La Figura 3.9 muestra el efecto de cambiar el parámetro α. Las cuatro imágenes muestran curvas de loess que se ajustan exactamente a los mismos datos. Los valores de α varían de 0.15 a 1. La curva ajustada se vuelve más suave cuanto mayor es el valor del parámetro. Esto ocurre por dos razones: en primer lugar, las ventanas de ajuste más anchas (mayor α) significan que las observaciones tenderán a anularse entre sí y, por lo tanto, tendrán proporcionalmente menos influencia en las regresiones locales. En segundo lugar, que α tenga valores altos significa que un menor número de observaciones cambiará al pasar de una ventana de ajuste a la siguiente. Ambos factores deberían tender a estabilizar las líneas de regresión local y los valores ajustados, produciendo así una curva de loess más suave [23].

Podemos pensar en la curva de loess como una cadena que se coloca en el rango de los valores de X dentro de los datos. El valor controla la "holgura" de esta cuerda. Si tiene un

valor alto, tirará de ella y estará más apretada, produciendo una curva más recta. Por esta razón, a veces se denomina *parámetro de tensión* en el ajuste loess.

Podemos preguntarnos qué valores de α son los mejores para cada conjunto de datos, y la respuesta es que esta decisión es individual. En la imagen A y B de la Figura 3.9 podemos ver que el valor del parámetro no es muy apropiado. Con un conjunto de valores bajos el ancho de la ventana es muy estrecho y las regresiones locales son muy sensibles a la variación del ruido dentro de los valores de los datos. Esto es lo que produce las curvas onduladas, las cuales difuminan la estructura general de los datos. Por otro lado, las imágenes C y D son muy suaves, pero tampoco pasan por el centro de la nube de puntos; la mayoría de puntos de los datos en la región central de los valores de X caen por debajo de la curva de loess. El problema es que las ventanas amplias producidas por valores altos del parámetro impiden que las regresiones locales se ajusten lo suficiente como para seguir la curva correspondiente sobre los datos. Es por ello que un valor intermedio de α debería proporcionar una buena combinación entre el sobreajuste de las dos primeras imágenes y la falta de ajuste que ocurre en las dos últimas.

Las decisiones sobre el valor adecuado deben tomarse individualmente caso por caso. El objetivo general es producir una curva de loess que sea lo más suave posible, pero que aún capture toda la estructura importante que existe dentro de los datos.

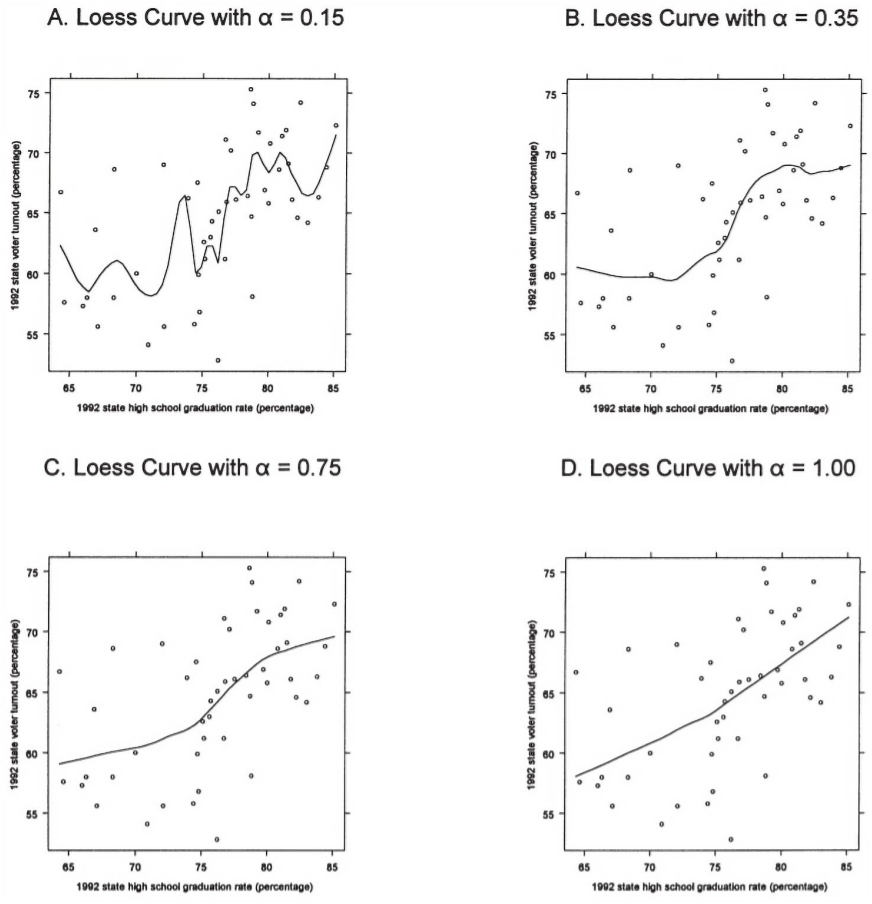


Figura 3.9.: Efecto de α en la curva de loess [23]

3.4.1.2. Grado del polinomio loess λ

El parámetro λ especifica el grado del polinomio que el procedimiento loess ajusta sobre los datos. Si $\lambda = 1$, entonces las ecuaciones lineales son las que se ajustan dentro de cada una de las ventanas. Por otro lado, si $\lambda = 2$ se utilizarán ecuaciones cuadráticas. Estas complican el proceso de ajuste, pero a veces son necesarias para generar una curva suave que siga los datos de una forma aceptable [23].

En la Figura 3.10 podemos ver el efecto del parámetro λ en la curva de loess. En la imagen (A) la curva tiene como parámetros $\lambda = 1, \alpha = 0.5$, mientras que la (B) tiene $\lambda = 2, \alpha = 0.5$. Se puede observar que una ecuación cuadrática local proporciona una curva de loess más precisa en esta situación y para estos datos.

En la práctica, la especificación del parámetro λ suele ser fácil; la decisión se puede tomar solo mediante una inspección visual del diagrama de dispersión. Si la nube de puntos se ajusta a un patrón generalmente monótono (ya sea creciente o decreciente), entonces λ debe establecerse en 1 para un ajuste localmente lineal. Si los datos muestran un patrón no monótono, con mínimos y/o máximos locales, entonces λ debería establecerse como 2 para ecuaciones cuadráticas localmente.

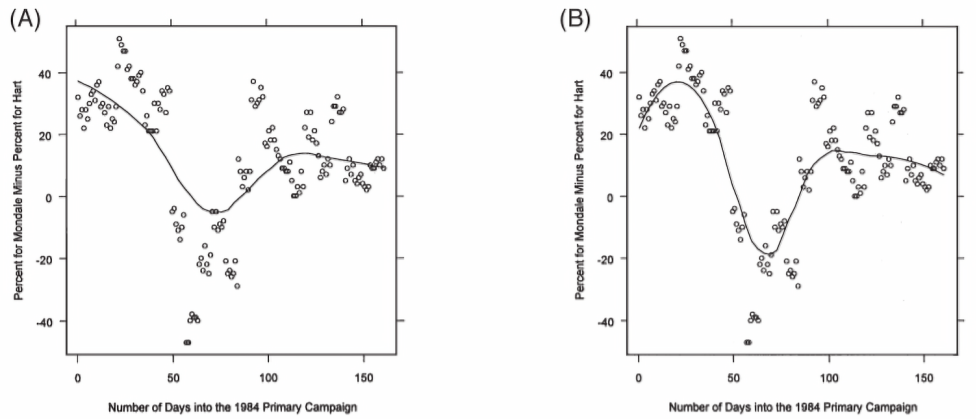


Figura 3.10.: Efecto de λ en la curva de loess [23]

3.4.1.3. Robustez en el loess

En el procedimiento de ajuste loess, este paso de robustez es opcional. Sin embargo, es muy común que esté incluido en el cálculo de las curvas suavizadas a través del loess [23].

El paso de robustez del procedimiento que tratamos minimiza las observaciones que tienen más probabilidades de tener un efecto negativo en las regresiones locales, esto es, aquellas con altos residuos. Tras esto, es más probable que la curva suave realice un seguimiento de las áreas más concentradas de los puntos de datos, en lugar de "perseguir los valores atípicos" en el diagrama de dispersión.

En la Figura 3.11 podemos ver el efecto de este paso en una curva y cómo "ignora" estos valores atípicos como resultado del procedimiento de reducción de importancia a las observaciones con residuos altos. Las líneas continuas muestran la curva tras tener en cuenta la robustez, mientras que las discontinuas muestran la misma omitiendo este paso en el proceso de ajuste. Estas dos curvas tienen como parámetro $\alpha = 0.75$ y $\lambda = 2$.

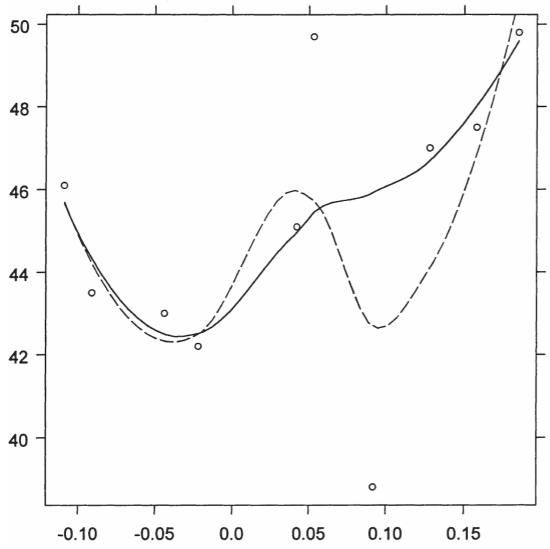


Figura 3.11.: Efecto de la robustez en la curva de loess [23]

Cuando el objetivo es producir un resumen gráfico de los datos puede que sea mejor incluir los pesos de robustez en el procedimiento de ajuste con loess. La única desventaja potencial de una estimación robusta con loess es que los residuos pueden no tener un "buen comportamiento", es decir, pueden no estar distribuidos normalmente con una media nula y varianza constante. Esto es un problema solo si la persona que analiza pretende conseguir más que un simple resumen gráfico. En pocas palabras, un ajuste no robusto con loess puede generar un resumen de los datos que es más complicado de lo que realmente necesita ser. En el peor de los casos se puede producir una representación confusa de la estructura dentro de los datos.

3.4.2. Media móvil

La media móvil se utiliza para analizar los datos de series temporales[6] definidos en una ventana de movimiento, calculando las medias de diferentes subconjuntos del conjunto de datos total. De esta forma se puede crear una curva suavizada de los datos. Dado que implica tomar el promedio del conjunto de datos a lo largo del tiempo, también se denomina media rodante.

En resumen, es un acercamiento matemático para la reducción de variación aleatoria en análisis de parámetros.

Definición 3.19. Sea una serie temporal $x(t), t = 1, 2, ..., N$. Se construye la secuencia de sumas acumulada como:

$$y(t) = \sum_{i=1}^{t} x(i), t = 1, 2, ..., N.$$

Definición 3.20. Se define la **media móvil** o *rolling mean* en una ventana de movimiento

[6]Una serie temporal o cronológica es una sucesión de datos medidos en determinados momentos y ordenados cronológicamente.

como la función $\tilde{y}(t)$ de forma que

$$\tilde{y}(t) = \frac{1}{n} \sum_{k=-\lfloor (n-1)\theta \rfloor}^{\lceil (n-1)(1-\theta) \rceil} y(t-k)$$

donde n es el tamaño de la ventana, $\lfloor x \rfloor$ es el mayor entero menor o igual que x, $\lceil x \rceil$ el menor entero mayor o igual que x, y θ es el parámetro de posición con valor en el intervalo $[0, 1]$ [19].

Por lo tanto, la función de la media móvil considera $\lceil (n-1)(1-\theta) \rceil$ puntos de datos en el pasado y $\lfloor (n-1)\theta \rfloor$ puntos en el futuro.

De esta forma se consigue hacer un promedio de los datos dentro de la ventana de movimiento y crear la cuva suavizada de los mismos. Cabe destacar que hay diferentes tipos de medias móviles: simple, explonencial, ponderada, Hull, KAMA, etc.

3.4.3. Transformada discreta de coseno

Las transformaciones, y en particular las transformaciones integrales, se usan sobre todo para la reducción de la complejidad en problemas matemáticos [41]. Ecuaciones diferenciales e integrales pueden convertirse en ecuaciones algebraicas cuyas soluciones se pueden obtener de una manera más sencilla. El análisis de transformación, aplicado en el procesamiento digital de señales[7], tiene un objetivo similar. La transformada de Fourier, que descompone una señal en sus componentes de frecuencia, y la transformada de Karhunen-Loeve, que descorrelaciona una secuencia de señal, son ejemplos conocidos en el área de procesamiento de estas señales.

Para explicar la transformada discreta de coseno (*Discrete cosine transformation*, DCT) es necesario explicar la transformada de coseno de Fourier.

Definición 3.21. Dada una función $x(t)$ con $-\infty < t < \infty$, se define su transformada de Fourier como:

$$X(\omega) \equiv F[x(t)] = \left(\frac{1}{2\pi}\right)^{1/2} \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt \tag{3.8}$$

donde $i = \sqrt{-1}, \omega = 2\pi f$ es la frecuencia en radianes y f es la frecuencia en hercios.

La función $x(t)$ se puede recuperar por la transformada inversa de Fourier:

$$x(t) \equiv F^{-1}[X(\omega)] = \left(\frac{1}{2\pi}\right)^{1/2} \int_{-\infty}^{\infty} X(\omega) e^{i\omega t} d\omega \tag{3.9}$$

Las ecuaciones (3.8) y (3.9) denotan la transformada y transformada inversa de Fourier para una función $x(t)$. Si definimos esta para $t \geq 0$, podemos construir una nueva función $y(t)$ dada por:

$$y(t) = \begin{cases} x(t), & t \geqslant 0 \\ x(-t), & t < 0 \end{cases}$$

Entonces:

[7] El procesamiento digital se define como el proceso de cambio de una señal digital en un sistema, con la finalidad de resaltar o eliminar diversas características de la misma que tienen cierto significado relevante para una aplicación en concreto [36]

$$
\begin{aligned}
F[y(t)] &= \left(\frac{1}{2\pi}\right)^{1/2} \left(\int_0^\infty x(t)e^{-i\omega t}dt + \int_{-\infty}^0 x(-t)e^{-i\omega t}dt\right) \\
&= \left(\frac{1}{2\pi}\right)^{1/2} \int_0^\infty x(t)\left[e^{-i\omega t} + e^{i\omega t}\right]dt \\
&= \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\cos(\omega t)dt
\end{aligned} \tag{3.10}
$$

Podemos ahora definir la **transformada de coseno de Fourier** de $x(t)$ como:

$$
X_c(\omega) \equiv F_c[x(t)] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\cos(\omega t)dt \tag{3.11}
$$

Teniendo en cuenta que $X_c(\omega)$ es una función impar de ω, podemos aplicar la inversa de Fourier a (3.10) para obtener que si $t \geq 0$, $y(t) = x(t)$ y por tanto:

$$
x(t) \equiv F_c^{-1}\left[X_c(\omega)\right] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty X_c(\omega)\cos(\omega t)d\omega \tag{3.12}
$$

Por lo tanto, las ecuaciones (3.11) y (3.12) definen un par de transformadas de coseno de Fourier.

Podemos ver en (3.11) que la transformada de coseno de Fourier tiene un núcleo dado por:

$$
K_c(\omega, t) = cos(\omega t) \tag{3.13}
$$

Sea $\omega_m = 2\pi m\delta f$ y $t_n = n\delta t$ la frecuencia y el tiempo regular de la muestra, donde δf y δt representan los intervalos de muestra unitarios para la frecuencia y tiempo respectivamente, y m,n son enteros. Podemos escribir entonces la ecuación (3.13) como:

$$
K_c\left(\omega_m, t_n\right) = K_c(2\pi m\delta f, n\delta t) = \cos(2\pi mn\delta f\delta t) = K_c(m, n)
$$

Si escribimos $\delta f\delta t = 1/(2N)$ con N entero, tenemos que:

$$
K_c(m, n) = cos\left(\frac{\pi mn}{N}\right) \tag{3.14}
$$

Esta ecuación (3.14) representa el núcleo del coseno de Fourier discretizado. Podemos representarlo como una matriz de transformación de dimensión $(N+1) \times (N+1)$, la cual la vamos a denotar por $[M]$ y podemos escribir su elemento mn-ésimo como:

$$
[M]_{mn} = cos\left(\frac{\pi mn}{N}\right) \qquad m, n = 0, 1, ..., N
$$

Cuando aplicamos la matriz a un vector columna $x = [x(0), x(1), ..., x(N)]^T$, obtenemos un vector $X = [X(0), X(1), ..., X(N)]^T$ tal que:

$$
X = [M]x,
$$

donde

$$
X(m) = \sum_{n=0}^{N} cos\left(\frac{\pi mn}{N}\right) x(n), \qquad m = 0, 1, ..., N \tag{3.15}
$$

Decimos entonces que el vector x ha sufrido una **transformación discreta**. Esta transformada de coseno en (3.15) fue definida por primera vez en 1980 por Kitajima ([29]) y fue llamada *transformada simétrica de coseno*.

Podemos dar entonces las definiciones de las transformadas discretas de coseno clasificadas como aparece en [41]:

(1) DCT-I:

$$\left[C_{N+1}^{I}\right]_{mn} = \left(\frac{2}{N}\right)^{1/2} \left[k_m k_n \cos\left(\frac{mn\pi}{N}\right)\right] \quad m,n = 0,1,\ldots,N$$

(2) DCT-II:

$$\left[C_{N}^{II}\right]_{mn} = \left(\frac{2}{N}\right)^{1/2} \left[k_m \cos\left(\frac{m\left(n+\frac{1}{2}\right)\pi}{N}\right)\right] \quad m,n = 0,1,\ldots,N-1$$

(3) DCT-III:

$$\left[C_{N}^{III}\right]_{mn} = \left(\frac{2}{N}\right)^{1/2} \left[k_n \cos\left(\frac{\left(m+\frac{1}{2}\right)n\pi}{N}\right)\right] \quad m,n = 0,1,\ldots,N-1$$

(4) DCT-IV:

$$\left[C_{N}^{IV}\right]_{mn} = \left(\frac{2}{N}\right)^{1/2} \left[\cos\left\{\frac{\left(m+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\pi}{N}\right\}\right] \quad m,n = 0,1,\ldots,N-1$$

donde

$$k_j = \begin{cases} 1, & \text{si } j \neq 0 \text{ ó } N \\ \frac{1}{\sqrt{2}}, & \text{si } j = 0 \text{ ó } N \end{cases}$$

Aquí notamos que DCT-II es la transformada discreta de coseno que se nombró por primera vez en [3]. DCT-III es la transpuesta de DC-II, y DCT-IV es una versión de DCT-I. En la Figura 3.12 podemos ver las funciones de base para DCT-II con $N = 16$.

Cabe destacar que no hay acuerdo entre los diferentes autores que definen la transformada de Fourier; difieren en el signo de la exponencial o en si el valor que multiplica la integral es $1/2\pi$ ó $1/\sqrt{2\pi}$, entre otras cosas.

Para concluir, como en este estudio se utiliza la DCT, cabe resaltar una característica de la transformada de Fourier de una función integrable: el Teorema 3.1, tomando como referencia la ecuación (3.8). Previamente, vamos a denotar $L^1(\mathbb{R})$ al espacio de todas las funciones $x : \mathbb{R} \to \mathbb{C}$ tales que $\int_{-\infty}^{\infty} |x(t)|dt < \infty$ y a enunciar el Lema de Riemann-Lebesgue ([48]), cuya demostración queda fuera de este trabajo:

Lema 3.1. ***(de Riemann-Lebesgue)*** *Sea una función $x \in L^1(\mathbb{R})$, entonces*

$$\lim_{\omega\to\pm\infty} X(\omega) \equiv \lim_{\omega\to\pm\infty} \left(\frac{1}{2\pi}\right)^{1/2} \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt = 0$$

Teorema 3.1. *La transformada de Fourier de una función integrable $x \in L^1(\mathbb{R})$ es una función continua, acotada y $\lim_{\omega\to\pm\infty} X(\omega) = 0$*

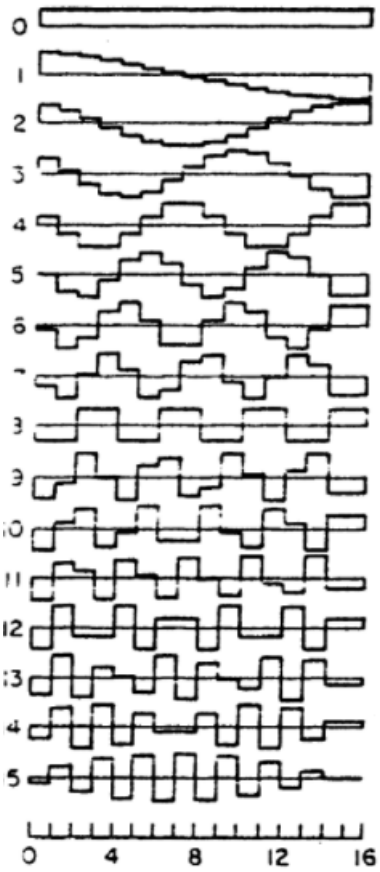


Figura 3.12.: Funciones de base para DCT-II con $N = 16$ [41]

Demostración. La continuidad y acotación son triviales teniendo en cuenta que $|e^{-i\omega t}| = 1$ $\forall \omega, t \in \mathbb{R}$ y por tanto, $X(\omega) \leq \int_{-\infty}^{\infty} |x(t)|dt < \infty$. Podemos asegurar que $\lim_{\omega \to \pm\infty} X(\omega) = 0$ por el Lema 3.1. □

Este resultado nos dice que la transformada de Fourier es una **operación que suaviza las funciones**. Además, esta característica se puede extender fácilmente a la DCT y un ejemplo de su aplicación podrá verse en la Subsección 4.6.2 de la experimentación.

4. Experimentación

En este capítulo se pretende aplicar las técnicas estadísticas y de minería abordadas en el capítulo anterior, así como la explicación del proceso de extracción de los datos y un previo análisis exploratorio de los mismos, utilizando el lenguaje de programación R.

4.1. Obtención de datos

En el trabajo presente se van a extraer los datos de una red social llamada *Twitter*, la cual nos permite extraer los conocidos *tweets* a partir de diferentes cuentas registradas y las características de la propia red social como *hashtags*, etc. Los tweets son mensajes publicados en esta red social que contienen texto, fotos, GIF o vídeo. Los hashtags son un conjunto de caracteres precedidos por el símbolo #.

En nuestro caso, tras hacer una previa investigación de los usuarios que pueden ser objetos interesantes de estudio junto con el número de tweets que se podían extraer de cada uno, se ha decidido hacer la extracción de los datos a partir de cuatro cuentas pertenecientes a figuras políticas españolas de distintos partidos políticos como lo son el actual presidente Pedro Sánchez, Pablo Iglesias, Inés Arrimadas y Pablo Casado.

Para realizar la obtención de los datos se han seguido los pasos ofrecidos en *R project*[1] y en la página oficial de la red social[2], donde tenemos que solicitar en primer lugar una cuenta de desarrollador. Una vez obtenida, accedemos a la sección de aplicaciones y creamos una nueva. Posteriormente, necesitamos un método de autorización para poder crear nuestro *token*. Existen dos posibles métodos, pero en este caso se va a utilizar la autenticación basada en el navegador, la cual hace uso de las dos claves de la API: la pública y la privada. Con ello, creamos nuestro propio *token* y pasamos a la extracción de datos.

Para proceder, se ha creado una función llamada *extraccion_tweets* que se puede encontrar en el cuaderno del Apéndice B, la cual recibe tres argumentos:

- *cuenta*: cuenta de la red social de la que se extraerán los tweets.
- *tweetsmax*: número máximo de tweets a extraer. Por defecto 3200.
- *output_file_name*: nombre del fichero de salida. En caso de no especificarse se crea un nombre por defecto.

Cabe destacar una característica del número máximo de tweets a extraer. Antiguamente, Twitter sólo dejaba extraer 200 tweets como máximo al llamar a la función encargada, por lo que si queríamos obtener más, debíamos hacerlo de forma reiterativa llamando a la función tantas veces como quisiéramos para extraer el número de tweets deseados. Sin embargo, a día de hoy, esta función es capaz de extraer hasta 3200 tweets siendo llamada una sola vez. Esta es la razón por la que se ha elegido este número como máximo de tweets a extraer y se comprueba que *tweetsmax* esté en el intervalo [1,3200].

[1] https://cran.r-project.org/web/packages/rtweet/vignettes/auth.html
[2] https://developer.twitter.com/

Como se ha comentado anteriormente, se ha hecho un análisis comparando cuántos tweets se podían extraer de diferentes figuras políticas, partidos, personas... hasta que se ha decidido optar por estos cuatro políticos españoles, ya que fueron los mejores en función al número de tweets que se pudieron extraer junto con el interés de estudiar estas cuentas. De este modo, se muestra en la Tabla 4.1 cada usuario y el número de tweets extraídos para cada uno, así como la fecha del primer y último tweet que se ha podido recoger, para registrar el periodo en el que estamos estudiando a cada político.

Usuario	Número de tweets extraídos	Fecha inicio y fin de datos recogidos
@InesArrimadas	1678	05/11/2019 – 17/08/2021
@pablocasado_	2201	14/01/2020 – 16/08/2021
@PabloIglesias	2049	22/08/2019 – 04/05/2021
@sanchezcastejon	1734	04/02/2020 – 15/08/2021

Tabla 4.1.: Usuarios y tweets recogidos

4.2. Limpieza y tokenización

Como se ha comentado en las fases de la minería de texto, una vez que tenemos nuestros datos recogidos en forma textual, tenemos que someterlos a un proceso de limpieza para su posterior tokenización y estudio. Para ello, se ha creado una función llamada *limpia_y_tokeniza*, la cual se puede encontrar en el cuaderno del Apéndice B que se encarga de aplicar los siguientes cambios en los datos, en el orden mostrado:

1. Convierte todo el texto en minúsculas.
2. Suprime enlaces.
3. Suprime iconos/emojis.
4. Suprime signos de puntuación.
5. Suprime números.
6. Elimina espacios en blanco sobrantes.
7. Tokeniza el texto.
8. Elimina tokens con longitud 1.

Cabe destacar que esta función ha sido modificada a lo largo del estudio, ya que aparecían nuevos problemas. Entre ellos, al utilizar texto en español y funciones de paquetes creados para hacer estudios en inglés, había determinados signos y términos, como por ejemplo "€" que no eran eliminados, por lo que se ha tenido que optar por eliminarlos uno a uno a medida que se iban encontrando en el estudio. Otro problema que surgió fue la eliminación de algunos iconos o emojis, ya que en R no se encontró ninguna función que se encargase de este trabajo, por lo que se optó por buscar la codificación de los mismos y eliminarlos siguiendo el patrón, el cual, a saber, eran del tipo `<U[\^ ]*>`. Otro problema que se encontró fue al hacer la derivación de palabras, ya que por el mismo problema de los paquetes, daba

unos resultados no deseados en nuestro estudio. Por ejemplo, palabras como “todos“ y “facturas“ las convertía en “tod“ y “factur“ respectivamente, por lo que se optó a no aplicar la derivación.

Se procede entonces a la aplicación de esta función en todos los tweets recogidos de cada usuario, añadiendo una columna a la que se ha llamado *texto_tokenizado*, la cual contiene, para cada tweet, un vector con las palabras que lo componen, con los cambios comentados en la lista anterior. Se presenta en la tabla Tabla 4.2 una muestra de algunos tweets junto con la limpieza y tokenización del texto.

Texto	Texto tokenizado
Atento también a la situación del #IFAzuébar, en Castellón(...)	c(“atento“, “también“, “la“, “situación“, “del“, “ifazuébar“, “en“, “castellón“...)
Pendientes de la evolución en las próximas horas del fuego (...)	c(“pendientes“, “de“, “la“, “evolución“, “en“, “las“, “próximas“, “horas“, “del“, “fuego“...)
Mi solidaridad y la de todo el pueblo español con Haití (...) tras este terrible suceso. https://t.co/579x11H5xu	c(“mi“, “solidaridad“, “la“, “de“, “todo“, “el“, “pueblo“, “español“, “con“, “haití“, (...) “tras“, “este“, “terrible“, “suceso“)

Tabla 4.2.: Tabla de tweets tokenizados

4.3. Análisis exploratorio

Tras haber hecho la limpieza y tokenización de los datos, nos encontramos con que cada objeto de estudio es un vector de palabras tal y como se muestra en la Tabla 4.3, de forma que no es lo que se conoce como *tidy data*, en el cual una observación es una fila. Por ello, para tener esta estructura deseada de los datos, hacemos uso de un procedimiento conocido como *unnest*, el cual se encarga de expandir esta lista de palabras, de manera que tengamos una fila para cada palabra tokenizada, tal y como se muestra en la Tabla 4.4.

texto_tokenizado <list>
<chr [36]>
<chr [42]>
<chr [48]>
<chr [43]>

Tabla 4.3.: Antes de aplicar unnest

usuario	fecha	tweet_id	token <chr>
sanchezcastejon	2021-08-15 07:48:58	1426813150653997058	atento
sanchezcastejon	2021-08-15 07:48:58	1426813150653997058	también
sanchezcastejon	2021-08-15 07:48:58	1426813150653997058	la
sanchezcastejon	2021-08-15 07:48:58	1426813150653997058	situación

Tabla 4.4.: Tras aplicar unnest

Teniendo los datos de esta forma, podemos proceder a aplicar filtros, sumatorios y representaciones de los datos de una manera más sencilla.

4.3.1. Distribución temporal de tweets

Como inicio de esta sección del análisis exploratorio, puede ser interesante ver un gráfico de la distribución temporal de tweets de cada usuario (Figura 4.1), ya que comienzan su actividad en la red social en distintos puntos temporales, y actúan de distinta manera a lo largo del tiempo. Se puede contrastar con la Tabla 4.1 y ver que las fechas de los tweets recogidos se ven reflejadas en el gráfico.

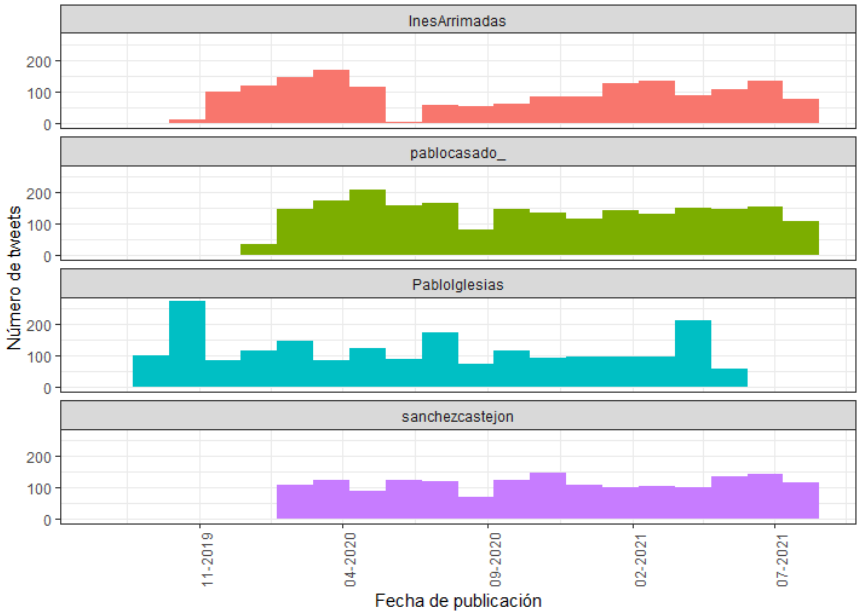


Figura 4.1.: Distribución temporal de los tweets

De la misma forma, podemos superponer en un gráfico la actividad de cada usuario a lo largo del tiempo, obteniendo la imagen de la Figura 4.2, dividida por meses.

Podemos observar que Pablo Casado y Pedro Sánchez tienen una actividad similar en el periodo estudiado, mientras que la actividad de Pablo Iglesias es irregular a lo largo del tiempo. Aludiendo al gráfico, el primer pico del ex-vicepresidente del gobierno se sitúa sobre el mes de octubre-noviembre de 2019, justo cuando fueron las últimas elecciones generales de España, por lo que el usuario podría haber hecho uso de la red social en tiempos de campaña antes de las elecciones. Del mismo modo ocurre con el pico en el mes de abril de 2021, ya que fueron las elecciones de Madrid y este usuario se presentó como candidato. El segundo pico podría deberse a que en ese mes de julio del 2020 se habló en España sobre el caso "Dina". Con respecto a Inés Arrimadas, también podríamos decir que tiene una actividad similar a los dos primeros políticos comentados, salvo por una anomalía, y es que dejó de publicar tweets en junio de 2020, esto podría deberse a que la política dio a luz a finales de mayo del mismo año y cesó su actividad en Twitter.

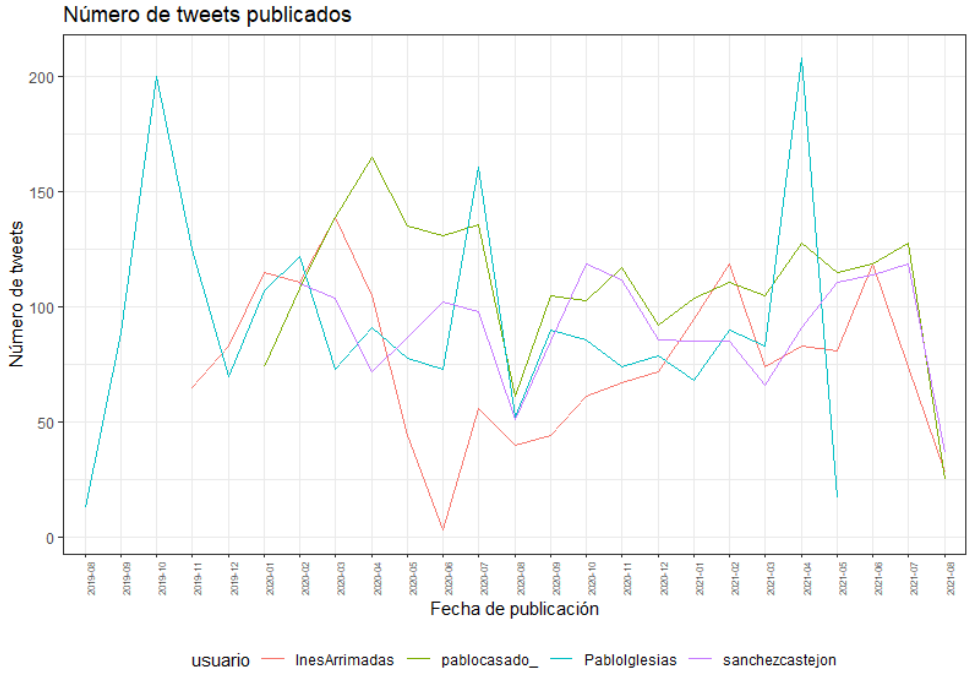


Figura 4.2.: Análisis temporal de tweets publicados

4.3.2. Distribución de palabras

Proseguimos el análisis exploratorio haciendo un estudio de la frecuencia y distribución de palabras publicadas de cada usuario.

4.3.2.1. Total de palabras utilizadas por cada usuario

En esta sección se muestra en primer lugar una tabla con el número de palabras total utilizadas por cada usuario (ver Tabla 4.5), viendo como este número es similar entre usuarios, a excepción de Pablo Casado, el cual se diferencia del resto considerablemente, con hasta 20.000 palabras.

Usuario	Total de palabras
@InesArrimadas	60499
@pablocasado_	87365
@PabloIglesias	59271
@sanchezcastejon	67830

Tabla 4.5.: Usuarios y total de palabras utilizadas

En segundo lugar se muestra un gráfico de barras (Figura 4.3) con esta misma información para mejor visualización de los datos.

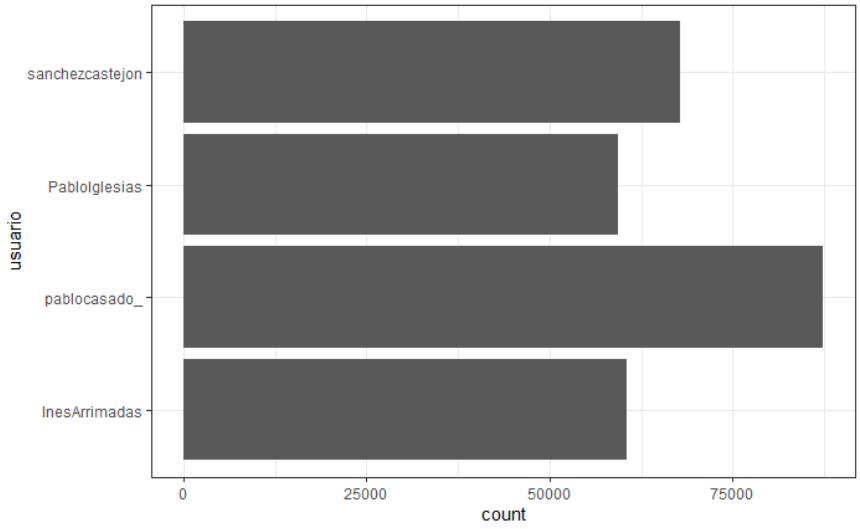


Figura 4.3.: Número total de palabras por usuario

4.3.2.2. Palabras distintas utilizadas por cada usuario

Es interesante añadir al estudio cuántas de las palabras utilizadas de entre las cifras mostradas anteriormente son distintas para cada usuario. La Tabla 4.6 recoge el número de palabras distintas empleadas por cada político. Este mismo recuento puede verse de forma gráfica en la Figura 4.4.

Usuario	Palabras distintas
@InesArrimadas	7999
@pablocasado_	10092
@PabloIglesias	9498
@sanchezcastejon	8949

Tabla 4.6.: Usuarios y palabras distintas utilizadas

Podemos observar como Pablo Casado sigue siendo el usuario que más palabras distintas utiliza en sus tweets, aunque esta vez no destaca tan claramente como lo hizo en la sección anterior, ya que le sigue de cerca Pablo Iglesias con 9498 palabras distintas.

4.3.2.3. Distribución de las palabras en los tweets por usuario

En este apartado se representan las longitudes de los tweets por usuario en un gráfico de cajas y bigotes para ver la distribución de los datos (Figura 4.5).

Destacan los resultados de Pablo Iglesias, ya que la longitud media de sus tweets es menor que para el resto de usuarios estudiados, pero la desviación típica es la más alta. En cuanto al resto de usuarios, se podría decir que la longitud de sus tweets es similar, al igual que se puede observar en el gráfico como, en el rango, la longitud máxima de todos los usuarios es similar, con un grado de dispersión parejo. Sin embargo, si lo medimos con el rango intercuartílico, Pablo Iglesias pasaría a ser el político con mayor grado de dispersión.

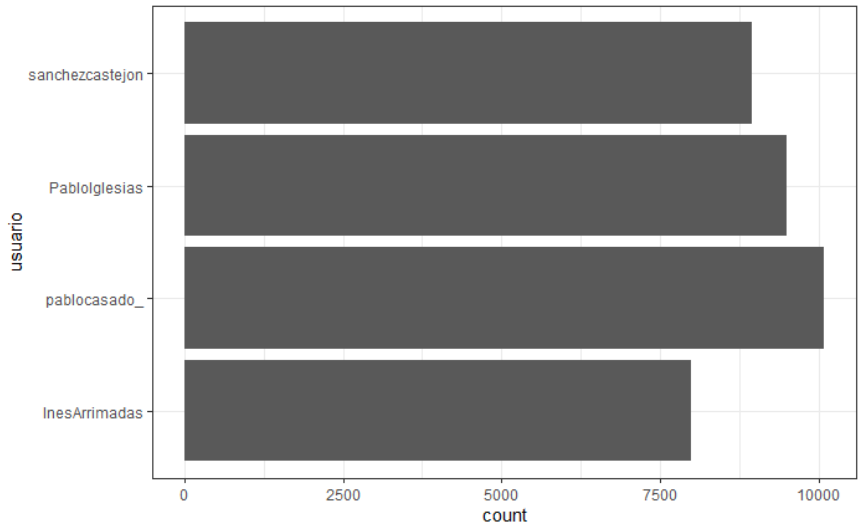


Figura 4.4.: Palabras distintas por usuario

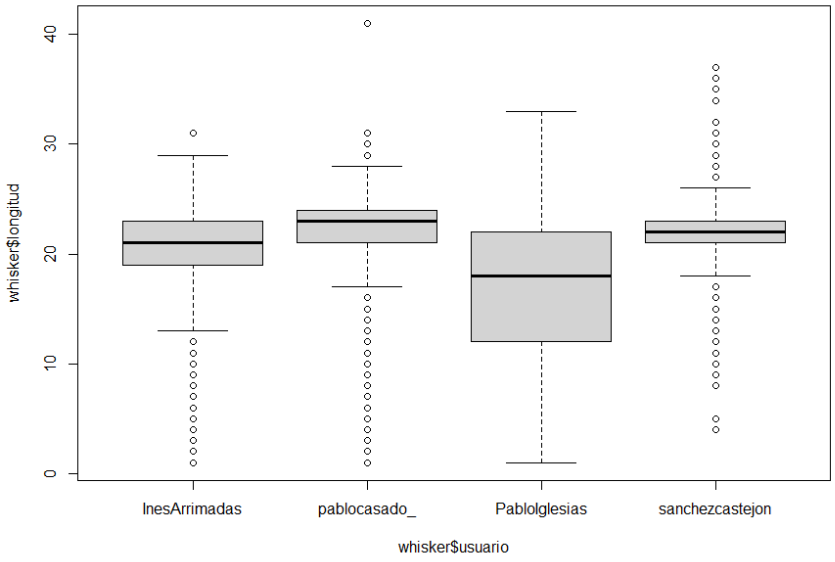


Figura 4.5.: Gráfico de cajas y bigotes para palabras en tweets

4.3.2.4. Palabras más utilizadas por usuario y stop words

Llegados a este punto, podemos pasar a estudiar cuáles son las palabras que más utilizan en sus tweets estos usuarios. Al hacer el recuento, nos damos cuenta de que las palabras más usadas son palabras que carecen de significado, esto es, determinantes, pronombres... (ver cuaderno del Apéndice B). Es en este momento donde debemos hacer uso de las anteriormente mencionadas *stop words*, palabras que recogeremos en una lista y eliminaremos pues no aportan información al estudio.

El idioma utilizado en el texto estudiado es el español, por lo que tendríamos que buscar algún paquete que nos permitiese eliminar estas palabras directamente o, en su defecto, crear un diccionario o lista de palabras a eliminar del presente trabajo. En este caso, el paquete *tm* nos proporciona una lista de stop words almacenada en el fichero *stopwords.txt* que aparece en el cuaderno del Apéndice B, la cual utilizaremos en el estudio.

Tras eliminar estas palabras, se vuelven a buscar cuáles son las palabras más utilizadas por usuario y vienen recogidas en la Figura 4.6.

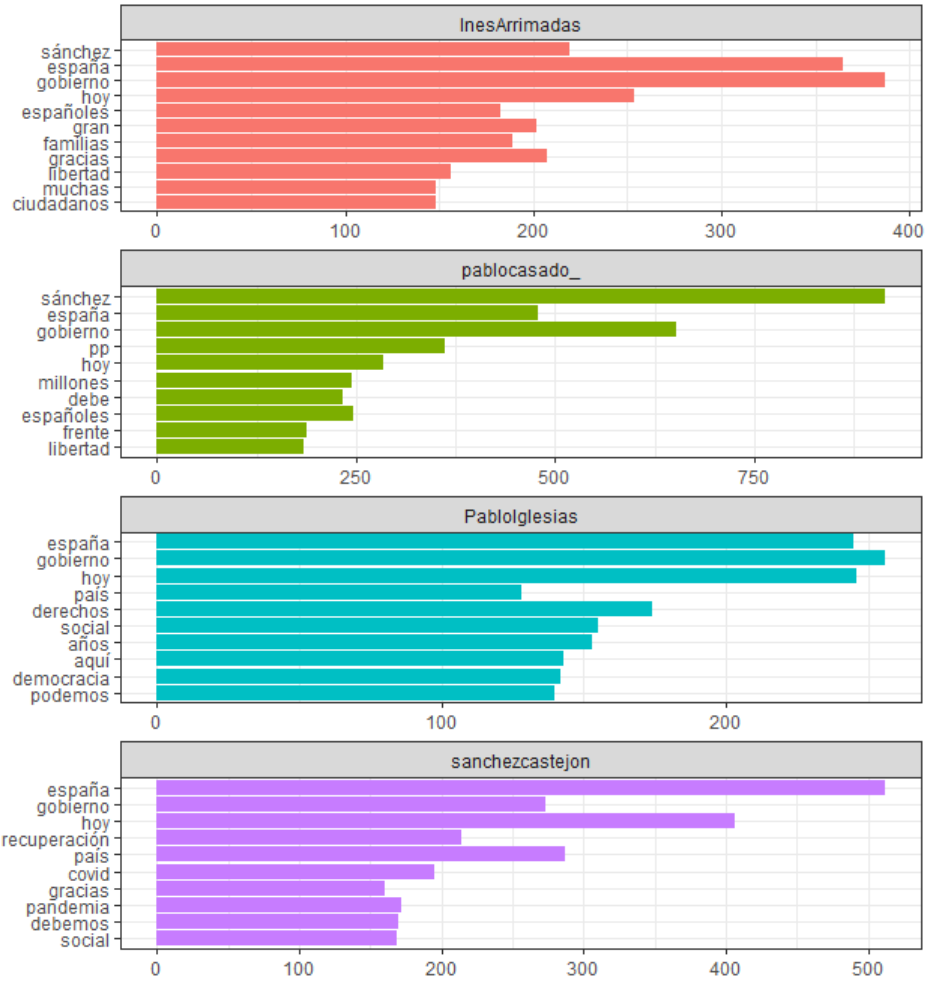


Figura 4.6.: Palabras más utilizadas por usuario

Comenzando por Pedro Sánchez, no sorprenden en absoluto sus resultados. Al ser el actual presidente de España, cobra sentido que dentro de sus palabras más utilizadas se encuentren "españa", "hoy", "gobierno", "país", "recuperación" o "covid". En cuanto a Pablo Iglesias, ex-miembro del partido Unidas Podemos, un partido con ideologías izquierdistas, y ex-vicepresidente del gobierno, tampoco sorprende que comparta palabras con Pedro Sánchez y algunas de las más utilizadas en sus tweets sean "derechos", "social", "democracia" o "podemos".

Es interesante cuanto menos observar que Pablo Casado, habiendo visto en los estudios anteriores que era el usuario que utilizaba más palabras y distintas, tenga como palabra predominante "sánchez". Podemos encontrar otras palabras que tienen sentido, como "pp", ya que son las siglas del partido al que este político pertenece, el Partido Popular. En cuanto a Inés Arrimadas, también conviene comentar que aparece, al igual que Casado, la palabra "sánchez" como una de las más usadas. Estos resultados cobran sentido si ponemos en contexto político a estos usuarios: Inés Arrimadas pertenece al partido político llamado Ciudadanos, el cual, junto al partido de Casado y otros más, forman parte de la oposición actual en España, por lo que podemos ver que en muchos de sus tweets se le menciona, probablemente para criticar su mala gestión o reprochar acciones.

También conviene comentar que los cuatro políticos comparten las palabras "españa", "gobierno" y "hoy", lo cual tiene sentido pues son figuras políticas del país.

4.3.2.5. Word clouds

Una forma de representar los resultados recién obtenidos es, por ejemplo, haciendo uso de las llamadas nubes de palabras o "Word clouds", un gráfico en el que las palabras más importantes son de mayor tamaño, y en función de la misma van disminuyendo. Para ello hacemos uso del paquete *wordcloud* que nos permitirá visualizar los resultados de otra manera, y de una función propia que muestre al usuario, junto con su nube de palabras, aplicando la restricción de que en cada una de ellas no habrá más de 200 palabras, ya que no nos interesa mostrar tantas en el gráfico, ni caben en el mismo. Ver Figura 4.7.

4.3.2.6. N-gramas. Bigramas, trigramas y representación

Puede ser interesante estudiar si existe alguna relación entre las palabras de los tweets con las que estamos trabajando. Más adelante, dentro de las técnicas aplicadas, aplicaremos las reglas de asociación, técnica que se encarga de estudiar qué palabra o conjunto de palabras tienden a coexistir en los mismos documentos (tweets). En este caso, vamos a utilizar los n-gramas para estudiar qué palabras tienden a aparecer inmediatamente después de otras.

En secciones anteriores, se comentó el uso del proceso conocido como *unnest* para tokenizar por palabras, pero también podemos utilizarlo para tokenizar en secuencias de palabras consecutivas, lo que se conoce como n-gramas. Al estudiar con qué frecuencia una palabra va seguida de otra distinta, podemos construir un modelo de relaciones entre ellas.

Podemos iniciarnos creando lo que se conoce como *bigramas* para examinar qué **dos** palabras son consecutivas y su frecuencia es alta. Para ello hacemos uso del mismo proceso *unnest*, pero en este caso pasando por argumento la opción de n-grama con n=2. Esta estructura creada en una nueva columna no es más que una variación del formato de texto en cada tweet.

Tras mostrar los bigramas más comunes, se puede observar en el cuaderno del Apéndice B que los resultados no aportan información, pues son resultados como "https t.co", "de la",

Word cloud de Inés Arrimadas

Word cloud de Pablo Casado

Word cloud de Pablo Iglesias

Word cloud de Pedro Sánchez

Figura 4.7.: Word clouds de los distintos políticos

"a la", etc. Nos vemos obligados por lo tanto a crear una función que se encargue de simplemente limpiar los datos, tal y como se hizo anteriormente. Esta función se ha llamado *limpiar* y se puede encontrar en el cuaderno mencionado. De igual manera, tenemos que volver a hacer uso de las ya comentadas "stop words" para obtener resultados que no carezcan de sentido y volver a obtener el bigrama. No hacemos uso de la función *limpia_y_tokeniza*, pues como su propio nombre indica, tokeniza los datos y no es lo que buscamos en esta sección, solo limpiarlos.

Realizada esta limpieza, hacemos uso de nuevo de las nubes de palabras para representar los bigramas (Figura 4.8) junto con un diagrama de barras (Figura 4.9).

De nuevo podemos darle sentido a estos resultados obtenidos, ya que se han recogido los datos a partir de cuatro figuras políticas del país. Bigramas como "sánchez debe" no sorprenden ya que en estudios anteriores hemos mostrado información sobre algunos políticos los cuales tenían la palabra "sánchez" como una de las más usadas en la red social. Al tener en cuenta que estos dos usuarios son parte de la oposición, cobra sentido que mencionen a Pedro Sánchez seguido de la palabra "debe", pues cuestionan las actuaciones del mismo y ofrecen alternativas que el presidente debería seguir.

Por estar en un contexto político español, tampoco asombran bigramas como "fondos europeos", "toda españa", "servicios públicos"... También cabe destacar los dos bigramas que más han aparecido: "hace años" y "muchas gracias". Este último nos ofrece información sobre un uso que podrían darle los políticos a esta red social y es, en efecto, para agradecer públicamente a ciertas personas o actuaciones. Del mismo modo, podemos observar que se hace una clara referencia al pasado constantemente, ya que la expresión "hace años" es repetida con frecuencia por parte de los usuarios.

Del mismo modo explicado anteriormente se procede a obtener los n-gramas con n=3,

Figura 4.8.: Word cloud de los bigramas

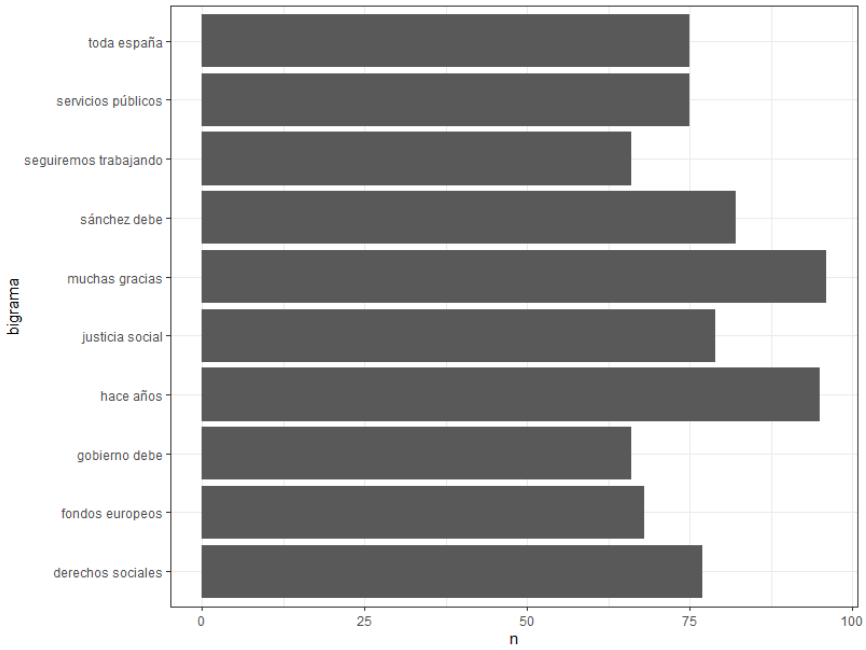


Figura 4.9.: Diagrama de barras de los bigramas más frecuentes

llamados *trigramas*. Estos son, de nuevo, secuencias de tres palabras que aparecen juntas y en el orden mostrado. Aquí, si estudiamos los trigramas más frecuentes sin eliminar las stop words, obtenemos que los más frecuentes son los mostrados en la Tabla 4.7. Algunos de estos resultados son interesantes, como por ejemplo "estado de alarma", haciendo referencia al régimen excepcional que se declaró debido al COVID-19. En función del tipo de estudio que se quiera hacer, se podrían dejar los datos con las stop words o sin ellas.

Trigrama	n
a todos los	154
estado de alarma	144
el gobierno de	128
todo mi apoyo	114
en el congreso	107

Tabla 4.7.: Trigramas con stop words

Podemos visualizar los resultados tras eliminar las stop words con una nube de palabras (Figura 4.10) y un gráfico de barras (Figura 4.11).

Figura 4.10.: Word cloud de los trigramas

Siguen apareciendo resultados que apoyan a toda la información obtenida hasta el momento. Ejemplo de ello es la aparición del trigrama "hoy hace años", lo que nos dice que esa mención al pasado suele ser por aniversarios de ciertos hechos o menciones a la antigua organización terrorista ETA: "años eta asesinó". Vemos un conjunto de trigramas como son "seguiremos trabajando juntos", "debemos trabajar juntos", "debemos seguir trabajando". Esto nos aporta información sobre el uso de los políticos de la red social: llamar al pueblo

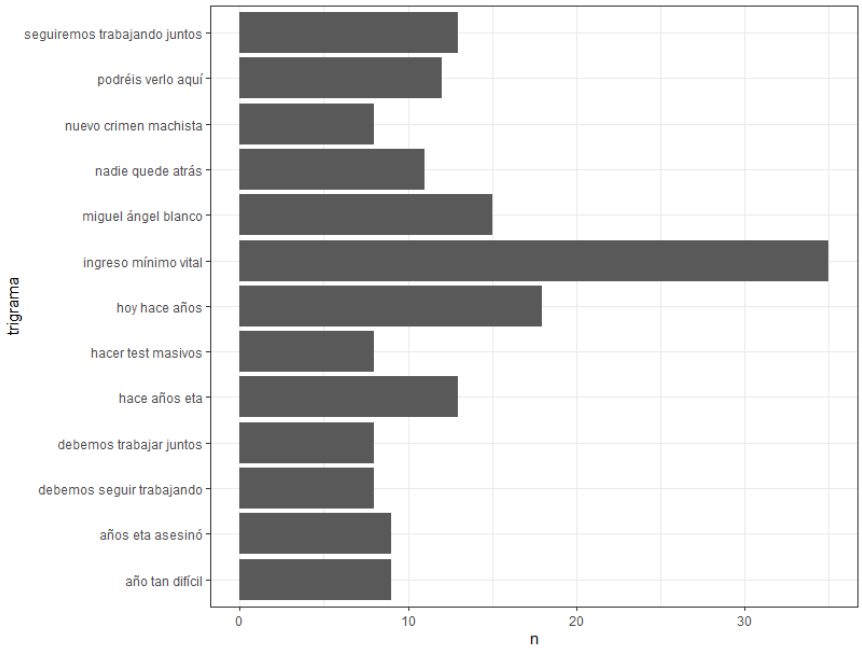


Figura 4.11.: Diagrama de barras de los Trigramas más frecuentes

para actuar unido.

Por otro lado, trigramas como "año tan difícil" y "hacer test masivos" podrían estar relacionados dada la situación epidémica que vivimos, junto con otros temas de índole internacional. Por último, dado el contexto de los datos, que el trigrama más frecuente sea "ingreso mínimo vital" es racional pues es un tema que los políticos han abordado con frecuencia, sobre todo a raíz de la pandemia producida por el COVID-19.

4.3.3. Frecuencia de términos y frecuencia inversa de documentos

Pasamos ahora a estudiar la importancia de ciertos términos en función a la frecuencia con la que aparecen. Sin embargo, puede ser que los resultados obtenidos solo a partir de la frecuencia absoluta no sean tan relevantes en el ámbito que lo estamos estudiando, por lo que se recurre posteriormente al método de la frecuencia inversa del documento, en este caso del tweet. Esta técnica es muy usada en minería de texto.

4.3.3.1. Frecuencia de términos

Para calcular la frecuencia de cada término en nuestro estudio, tenemos que buscar primero el número de veces que aparece cada término por tweet, esto es, en cada documento, para luego añadir una columna con el total de términos existentes en el tweet y acabar calculando la frecuencia. Vamos a llamar a este número tf (term frequency) por sus siglas en inglés.

$$tf = \frac{n_t}{L_d}$$

Donde n_t hace referencia al número de términos y L_d a la longitud del tweet o documento. Un breve ejemplo de estos resultados podría ser el mostrado a continuación.

Tenemos el siguiente tweet original (Tabla 4.8):

Usuario	Identificador del tweet	Texto
PabloIglesias	1164445168646467584	@feminoacid Respiro fuertecito <U+0001F607>

Tabla 4.8.: Ejemplo tweet original

Donde *@feminoacid* es un usuario de Twitter y *<U+0001F607>* es un emoticono de una cara sonriente con un halo. Tras aplicarle la limpieza y tokenización de los datos, obtenemos la modificación del texto mostrada en la Tabla 4.9. De forma que aplicándole la fórmula mencionada, pasamos a calcular el tf de cada término (Tabla 4.10). Esto significa que en un tweet limpio y tokenizado, como es el mostrado, de 3 palabras y todas distintas, todos los términos tendrán un $tf = 0.333$, tal y como se muestra en la tabla. Se puede ver el código en el cuaderno del Apéndice B.

Usuario	Identificador del tweet	Texto
PabloIglesias	1164445168646467584	feminoacid
PabloIglesias	1164445168646467584	respiro
PabloIglesias	1164445168646467584	fuertecito

Tabla 4.9.: Ejemplo tweet tokenizado

Identificador del tweet	Palabra	n	Longitud documento	tf
1164445168646467584	feminoacid	1	3	0.333
1164445168646467584	fuertecito	1	3	0.333
1164445168646467584	respiro	1	3	0.333

Tabla 4.10.: Ejemplo de resultados tf de un tweet

4.3.3.2. Frecuencia inversa de documentos

Para la obtención de la frecuencia inversa de documentos, se calcula el número total de documentos (tweets) y se busca el número de documentos en los que aparece cada término para posteriormente calcular la frecuencia inversa. Se denota *idf* (inverse document frequency) por su nombre en inglés.

$$idf = ln\left(\frac{n_{documentos}}{n_{documentos\_con\_el\_término}}\right)$$

Donde $n_{documentos}$ representa el número de documentos totales (tweets) y $n_{documentos\_con\_el\_término}$ representa el número de documentos que contienen al término.

Se calcula el número total de documentos, en este caso 7596. Y, por ejemplo, se calcula en cuántos documentos aparece la palabra "españa", 1525, por lo que se puede concluir que el idf de la palabra es $ln\left(\frac{7596}{1525}\right) = 1,6$. Un informe más detallado de los resultados se puede encontrar en el cuaderno del Apéndice B.

Uniendo lo comentado en estos dos apartados, podríamos obtener un valor para cada palabra al que vamos a llamar *tfidf*, pues será la multiplicación de ambos valores, número que se podría utilizar en los mismos estudios que se han realizado a lo largo del análisis exploratorio, ya que además, el $tfidf$ es un parámetro correctivo cuyo objetivo es compensar la alta frecuencia de palabras como artículos o similares, para dar más relevancia a las palabras realmente significativas.

$$tfidf = tf \cdot idf$$

Tomando como ejemplo el mostrado en la sección anterior, vemos que aunque las palabras tengan la misma frecuencia de términos, "respiro" aparece en otro tweet, por lo que su idf será distinto y, por tanto, su $tfidf$ (ver Tabla 4.11).

Palabra	n	Longitud tweet	tf	Número de tweets	idf	tfidf
feminoacid	1	3	0.333	1	8.935245	2.9784151
fuertecito	1	3	0.333	1	8.935245	2.9784151
respiro	1	3	0.333	2	8.242098	2.7473661

Tabla 4.11.: Ejemplo de resultados tfidf de términos

Como se ha comentado, podríamos usar este valor para estudiar, por ejemplo, las palabras significativas más utilizadas, entre otras cosas.

4.4. Clústering

Tras hacer el análisis exploratorio de los datos, pasamos a aplicar algunas de las técnicas estadísticas y de minería más empleadas en texto, comenzando por el clústering. En ella, se pretende agrupar conjuntos de objetos que no están etiquetados en un mismo subconjunto, llamado clúster.

Para proceder, necesitamos una representación numérica de los datos en forma de lo que se conoce como *term document matrix* (una matriz que describe la frecuencia de términos que aparecen en una colección de documentos), y nos encontramos con que tenemos 21585 términos y 152077 documentos. Es necesario reducir este tamaño, ya que la capacidad de cómputo que tenemos es limitada y no vamos a obtener resultados interesantes. Para ello, se reduce el parámetro *sparsity*, el cual toma valores en el intervalo [0,1] y mide la dispersión de los datos. Cuanto más se vaya alejando de 1, menos términos irán apareciendo, pues será más restrictivo y recogerá menos datos. Nos quedamos con un valor de 0.999, recogiendo 103 términos.

Llegados a este punto es necesario que nuestros datos estén en un formato de matriz para su posterior análisis. Se muestra en la Figura 4.12 algunas de las palabras recogidas con una frecuencia de aparición mayor o igual a 300.

Para finalizar, podemos encontrar dos tipos de clústering: jerárquico y no jerárquico, que son los que vamos a abordar con más profundidad a continuación.

4.4.1. Clústering jerárquico de palabras

En nuestro trabajo, vamos a hacer uso de la función *dist()*, la cual calcula y devuelve la matriz de distancia calculada utilizando la medida de distancia especificada para calcular

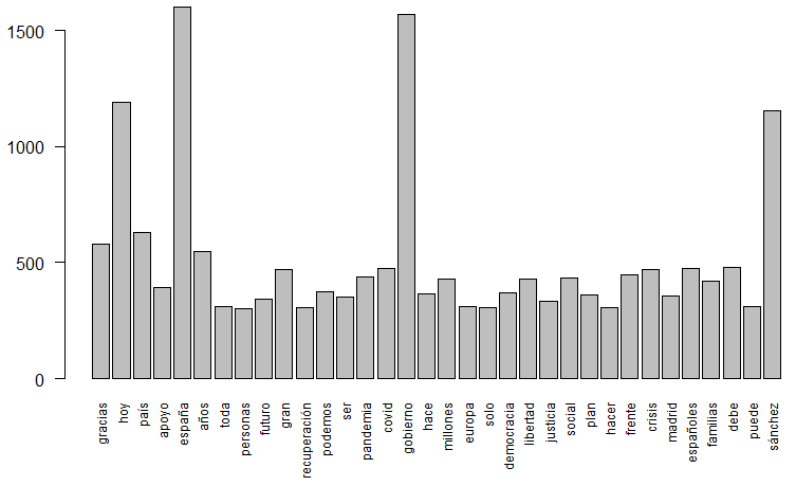


Figura 4.12.: Diagrama de barras de palabras con frecuencia $\geq$ 300

las distancias entre las filas de una matriz de datos, es decir, para calcular la distancia entre palabras en cada documento. Por defecto calcula la distancia euclídea, pero podría utilizar otras. Si la distancia es alta, significará que esas dos palabras no deberían estar en el mismo clúster. Del modo opuesto ocurre si la distancia es pequeña.

Hay varios métodos que podríamos utilizar para hacer el clústering jerárquico, pero en nuestro caso haremos uso del llamado "Método de Ward de varianza mínima", un método divisivo ya descrito en la Def. 3.3 en el cual se utiliza el error de la suma de los cuadrados o la varianza, intentando minimizarlos para agrupar términos. Obtenemos el resultado que vemos en la Figura 4.13.

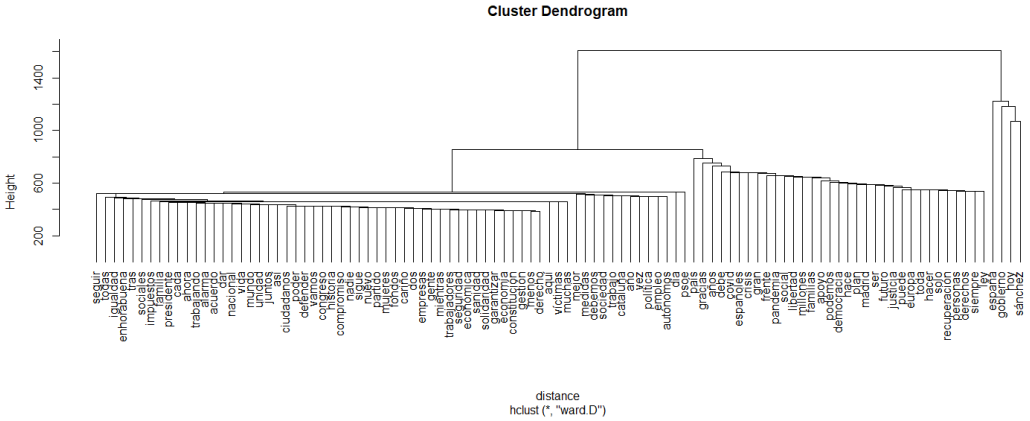


Figura 4.13.: Dendograma

Visualmente, quizás podríamos establecer que con 4 clústeres se pueden separar los datos de la manera mostrada en la Figura 4.14. En el primer clúster comenzando por la izquierda

se recogen palabras con usos de índoles políticos. Se ha visto en secciones anteriores que los usuarios utilizan Twitter para hacer referencias al pasado, para dar la enhorabuena... por lo que se recogen términos de estas temáticas junto con otras de tipo económico, entre otras. En segundo lugar se recogen un grupo de palabras usadas por los políticos que giran en torno al gobierno actual y su actuación, de ahí la aparición de palabras como "psoe", "política", "empleo", "medidas"... En el tercer clúster se recogen palabras cuya relación (en nuestro conexto) es el de ciertos problemas o frentes del país, actuales y antiguos. Esto explica que palabras como "covid", "pandemia" o "crisis" se encuentren en este clúster. Por último, vemos palabras como "sánchez", "gobierno", "hoy" y "españa". Términos aparecidos a lo largo del análisis exploratorio con características en común, como ser palabras muy usadas por los políticos estudiados.

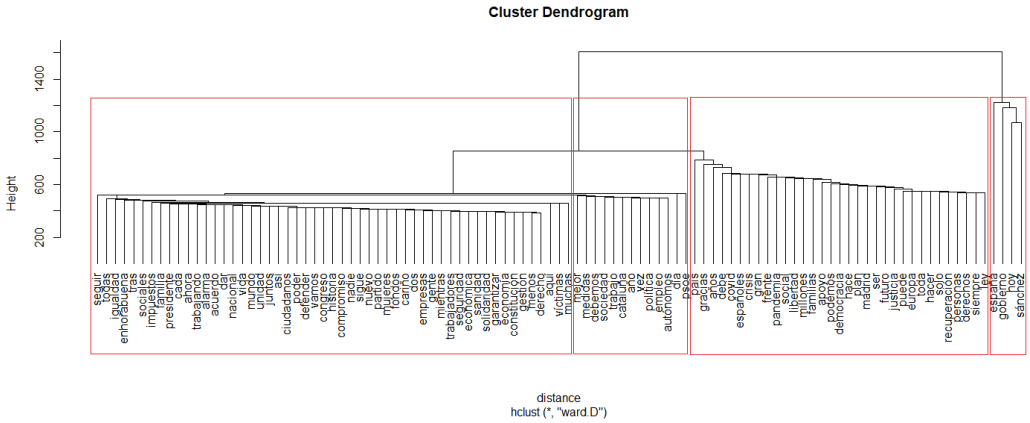


Figura 4.14.: Dendograma con 4 clústeres

4.4.2. Clústering no jerárquico: k-medias

En el presente trabajo vamos a hacer uso del algoritmo llamado k-medias, el cual es el más conocido para este tipo de clústering y pretende clasificar los datos en k clústeres, donde k se especifica previamente o se determina su valor óptimo durante el proceso.

Vamos a evaluar los resultados de este algoritmo teniendo en cuenta dos resultados del código como salida en R:

- El resultado de dividir la suma de los cuadrados entre grupos entre la suma total de cuadrados (lo que aparece en el código como between_SS / total_SS). Si este valor es alto, significará que las distancias o diferencias entre clústeres son altas, lo que significa que los clústeres están separados entre ellos.
- El resultado de la suma de cuadrados dentro de cada clúster (*within cluster sum of squares by cluster*). Queremos que estos resultados sean pequeños, pues conllevará que la variabilidad de los elementos dentro de los clústeres es baja, lo que significa que los elementos en cada clúster son cercanos.

Como se ha comentado, el número de clústeres k debe establecerse previamente a la ejecución del algoritmo. Esto puede ser una tarea difícil, por lo que en el presente trabajo se ha hecho uso de tres métodos que nos proporcionan un número óptimo para esta variable.

Vamos a utilizar entonces en primer lugar el **criterio del codo** (elbow method) explicado en la Subsubsección 3.1.3.1. La salida obtenida se puede observar en la Figura 4.15, donde podríamos elegir k=5 como número óptimo de clústeres.

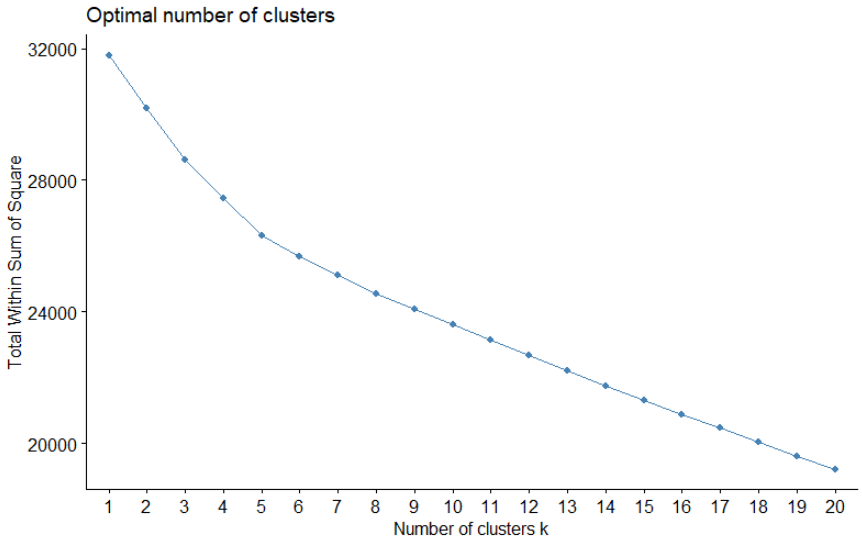


Figura 4.15.: Criterio del codo

Procedamos con más métodos para hacer una elección correcta de esta variable.

En segundo lugar, aplicamos el método **Silhouette** explicado en la Subsubsección 3.1.3.2 a nuestros datos y obtenemos el resultado mostrado en la Figura 4.16.

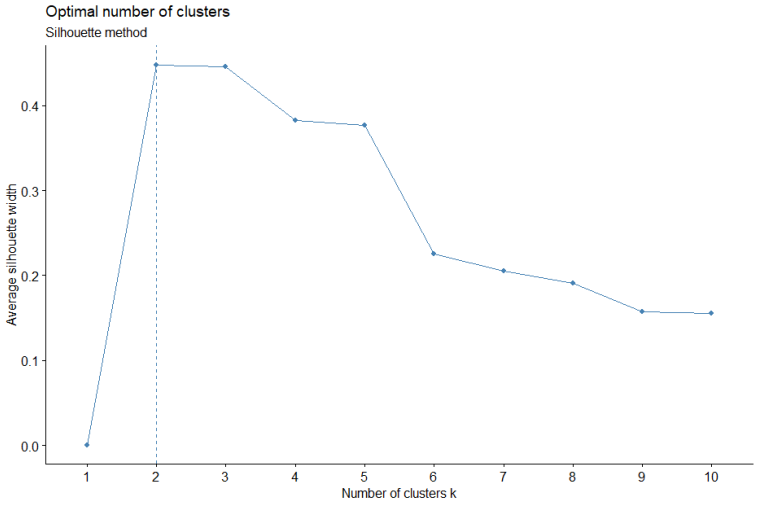


Figura 4.16.: Método de Silhouette

El gráfico nos muestra que ahora k debería ser 2 para tener un clústering óptimo, un valor distinto al obtenido en el método anterior.

Vamos a utilizar un tercer método, conocido como **método de Gap (brecha)** abordado en

profundidad en la Subsubsección 3.1.3.3. Este método estadístico calcula, en primer lugar, el número óptimo de clústeres, obteniendo la salida del método de k-medias, para luego comparar el cambio en la dispersión dentro de los clústeres y obtener un número de clústeres apropiado. Obtenemos la Figura 4.17 como resultado.

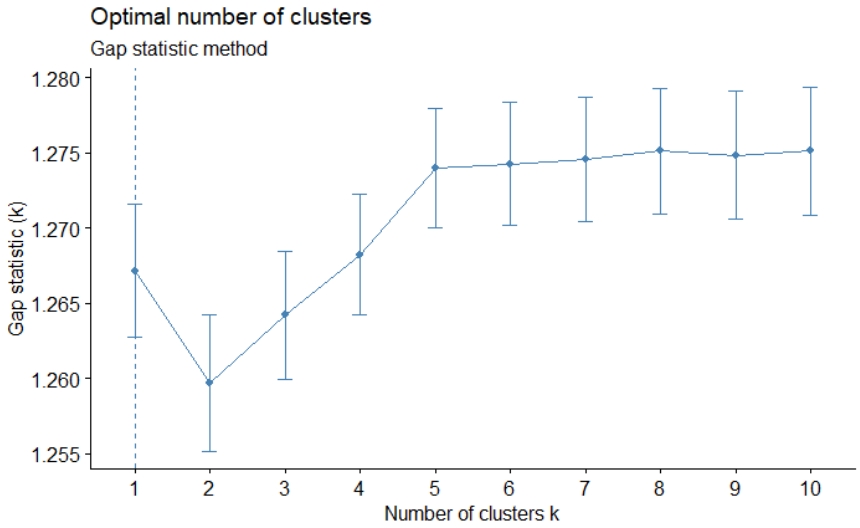


Figura 4.17.: Método de Gap

El cual nos da como número óptimo k=1. Sin embargo, podemos ver en la imagen referenciada que para k=4, el valor del estadístico que calcula es similar. Por otro lado, el método nos ofrece visualmente el valor que obtiene el estadístico en función del número de clústeres, donde el valor máximo nos dará el número óptimo para k, y vemos que a partir de k=5 se estabiliza el valor de este estadístico.

Recogiendo la información obtenida por estos tres métodos, podríamos decir que el número óptimo de clústeres se encuentra entre 1 y 5. Por ello, se muestra en la Tabla 4.12 una comparativa donde se muestran k y los valores mencionados al principio de la sección (Within cluster sum of squares by cluster y between_SS / total_SS).

k	Suma de cuadrados dentro del clúster	between_SS / total_SS
1	31775.48	2.085518*e*-09 %
2	0.00, 30188.11	5.258256 %
3	0.00, 0.00, 28632.67	10.97628 %
4	0.00, 0.00, 27451.71, 0.00	15.75044 %
5	26305.58, 0.00, 0.00, 0.00, 0.00	20.79369 %

Tabla 4.12.: Comparativa resultados para diferentes valores de k

Los datos reflejados en la columna "Suma de cuadrados dentro del clúster" se puede deber a la heterogeneidad de los datos, por lo que resulta difícil agruparlos de una manera correcta.

Por los resultados obtenidos en los diferentes métodos, vemos que si seguimos aumentando k, obtendremos mejores resultados, pero con una variabilidad entre ellos más baja,

por lo que podríamos elegir k=5 como número óptimo de clústeres en nuestro estudio. Sin embargo, quizás sería mejor elegir el número de clústeres en función a la interpretación que podamos y queramos hacer de los datos. En la Figura 4.14 se muestra una separación de los datos con características en común, comentadas en la sección previa, que divide al conjunto de términos. Por consiguiente, podríamos elegir k=4 como número óptimo de clústeres basándonos en la interpretación de los datos.

Está claro que en nuestro estudio no hay un valor claramente óptimo como número de clústeres, lo que refleja una heterogeneidad en el conjunto de los datos.

4.5. Reglas de asociación

Las reglas de asociación se representan con implicaciones de la forma $X \rightarrow Y$, donde X e Y son conjuntos de ítems disjuntos. En esta sección de la experimentación se pretende, por tanto, encontrar estas relaciones de coocurrencia entre los distintos conjuntos de ítems. Para ello, se hace uso de la elección de ciertos parámetros como el soporte y la confianza para la obtención de resultados.

Para la generación de las reglas de asociación se hace uso del paquete *arules* de manera que podemos obtener las transacciones con las que trabajaremos (tweets en nuestro estudio). Para conseguirlo, exportamos los datos en un archivo *.csv* y lo leemos haciendo uso de la función *read.transactions()* del mismo paquete, consiguiendo una matriz de dimensiones 7595x21670 y cuyo tamaño se representa en la Figura 4.18. Posteriormente, como nos interesa estudiar qué palabras "aparecen juntas", se restringen las transacciones a aquellas que tengan al menos 2 palabras, y la dimensión de nuestra matriz reduce a 7504x21670 cuya representación se muestra en la Figura 4.19.

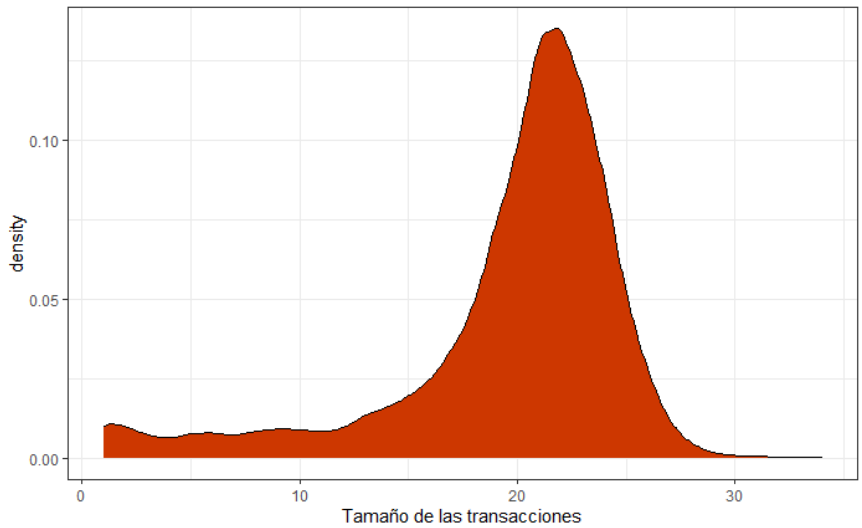


Figura 4.18.: Tamaño de las transacciones

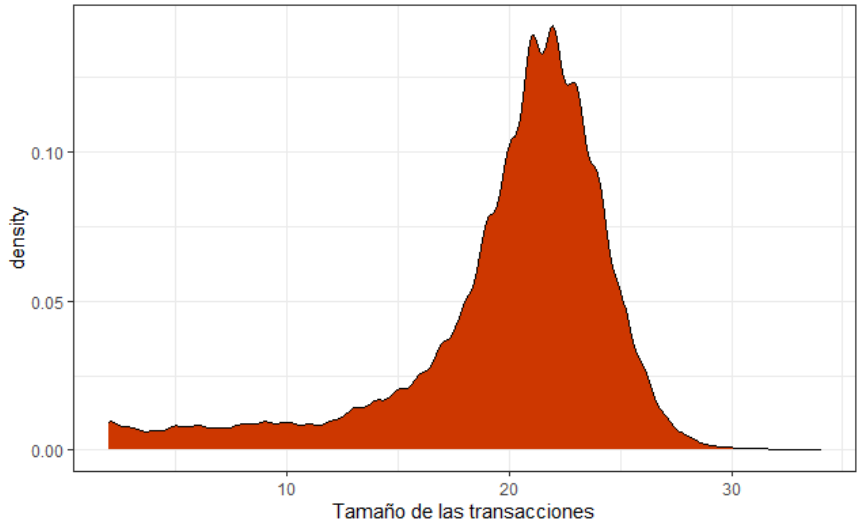


Figura 4.19.: Nuevo tamaño de las transacciones

4.5.1. Soporte y confianza

En nuestro estudio tenemos que considerar que los datos son de grandes dimensiones, por lo que el soporte tendrá que ser pequeño al esperar que cada evento sea "poco común". Por esta razón se han ido realizando pruebas para ver qué valores daban un resultado adecuado para nuestro estudio, y se ha decidido elegir 0.003 para el sorporte mínimo, lo que significa que se supone frecuente cuando aparece al menos 22 veces. Del mismo modo se ha elegido una confianza mínima del 85 %.

4.5.2. Búsqueda de conjuntos de ítems frecuentes

En el paquete usado durante esta sección encontramos la función *apriori()*, la cual se utilizará para la obtención de los conjuntos de ítems frecuentes y, posteriormente, las reglas de asociación.

Estableciendo el valor del soporte mínimo como 0.003, obtenemos la salida mostrada en la Figura 4.20 tras la obtención de los conjuntos de ítems frecuentes, donde podemos ver que hay un total de 2638 conjuntos que superan el soporte mínimo establecido, y muchos de ellos están formados por 1 y 2 ítems (1198 y 1351, respectivamente). De igual manera, podemos ver que algunos de los ítems más frecuentes son palabras que ya aparecieron en el análisis exploratorio de los datos, como son "españa", "gobierno" o "sánchez". También podemos observar cómo la media del soporte de los conjuntos de ítems frecuentes que superan el soporte mínimo es de 0.007253, teniendo un máximo de 0.203225, una cifra que aceptamos en el estudio.

```
set of 2638 itemsets

most frequent items:
  españa gobierno  sánchez      hoy     país  (Other)
     244      237      178      145       61     3304

element (itemset/transaction) length distribution:sizes
   1    2    3    4
1198 1351   87    2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.00    2.00    1.58    2.00    4.00

summary of quality measures:
    support         transIdenticalToItemsets     count
 Min.   :0.003065   Min.   :0.000e+00        Min.   :  23.00
 1st Qu.:0.003598   1st Qu.:0.000e+00        1st Qu.:  27.00
 Median :0.004531   Median :0.000e+00        Median :  34.00
 Mean   :0.007253   Mean   :8.133e-06        Mean   :  54.43
 3rd Qu.:0.007329   3rd Qu.:0.000e+00        3rd Qu.:  55.00
 Max.   :0.203225   Max.   :1.066e-03        Max.   :1525.00

includes transaction ID lists: FALSE
```

Figura 4.20.: Salida en R de la obtención de conjuntos de ítems frecuentes

4.5.3. Obtención de reglas

Se vuelve a hacer uso de la función *apriori()* para la obtención de las reglas, de forma que se generan 51 reglas de asociación, cuyo resumen se puede ver en la Figura 4.21. En ella además podemos observar que hay 23 reglas con un único ítem en el antecedente y consecuente. El lift mínimo de las reglas generadas es de 15 ($>> 1$), por lo que puede ocurrir que se repitan estos patrones obtenidos y reflejen comportamientos reales.

```
set of 51 rules

rule length distribution (lhs + rhs):sizes
 2  3  4
23 23  5

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   3.000   2.647   3.000   4.000

summary of quality measures:
    support         confidence        coverage            lift            count
 Min.   :0.003065   Min.   :0.8519   Min.   :0.003065   Min.   : 15.29   Min.   :23.00
 1st Qu.:0.003198   1st Qu.:0.8929   1st Qu.:0.003465   1st Qu.: 31.61   1st Qu.:24.00
 Median :0.003865   Median :0.9375   Median :0.004131   Median : 55.78   Median :29.00
 Mean   :0.004228   Mean   :0.9369   Mean   :0.004539   Mean   : 77.18   Mean   :31.73
 3rd Qu.:0.004531   3rd Qu.:0.9718   3rd Qu.:0.004797   3rd Qu.:101.12   3rd Qu.:34.00
 Max.   :0.009728   Max.   :1.0000   Max.   :0.011327   Max.   :266.81   Max.   :73.00
```

Figura 4.21.: Salida en R de la obtención de las reglas de asociación

Pasamos entonces a extraer las reglas maximales definidas en la Subsubsección 3.2.3.1. Hacemos uso de la función *is.maximal()* para obtener un total de 36 reglas maximales. Se muestran en la Tabla 4.13 las 5 reglas maximales con mayor lift. Para ver más resultados consultar el cuaderno del Apéndice B.

Regla maximal	Lift
{castilla} ⇒ {león}	266.80889
{ganaderos} ⇒ {agricultores}	192.41026
{paz, pésame} ⇒ {descanse}	171.22222
{mínimo, vital} ⇒ {ingreso}	159.65957
{cariño, seres} ⇒ {queridos}	156.33333

Tabla 4.13.: Reglas maximales con mayor lift

4.5.4. Evaluación con test exacto de Fisher

Como se ha comentado en el capítulo anterior, existen numerosas métricas para evaluar las reglas de asociación obtenidas. En nuestro trabajo hacemos uso del test exacto de Fisher desarrollado en la Subsubsección 3.2.3.2. Para ello, llamamos a la función *interestMeasure()*, dándole como argumento *measure=fishersExactTest*. Obtenemos como salida el resumen de la evaluación los datos mostrado en la Tabla 4.14.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000e+00	0.000e+00	0.000e+00	2.232e-27	0.000e+00	1.139e-25

Tabla 4.14.: Salida Test exacto de Fisher

Esta tabla recoge los 51 p-valores obtenidos al evaluar las correspondientes reglas generadas, mostrando el valor más pequeño, el mayor, cuartiles, mediana y media de los mismos. Estos son pequeños y menores que 0.05, por lo que es muy probable que las reglas reflejen patrones reales.

4.5.5. Visualización

Para la visualización de los resultados obtenidos, hacemos uso del paquete *arulesViz* que nos permite representar las reglas de asociación de numerosas formas en función a diferentes parámetros. Se van a utilizar cinco visualizaciones que representan los resultados:

- Dos gráficos de dispersión, uno donde el color representa el lift (Figura 4.22) y otro donde el color representa el número de ítems (Figura 4.23).
- Un gráfico de coordenadas paralelas, en el que cada regla queda representada por una flecha que va de izquierda a derecha relacionando las palabras (Figura 4.24).
- Un gráfico que muestra por un lado el conjunto de ítems del antecedente (eje horizontal) y el conjunto de ítems del consecuente (eje vertical). Este gráfico relaciona estos dos conjuntos de ítems con un círculo coloreado en función al lift, y de tamaño en función al soporte de la regla (Figura 4.25).
- En la Figura 4.26, las reglas vienen representadas por vértices conectados a elementos mediante flechas. Estos vértices tienen un tamaño en función al soporte, y un color en función al lift. Los ítems del antecedente están conectados con flechas que apuntan al vértice que representa la regla, y los ítems del consecuente tienen una flecha que apunta a sí mismo.

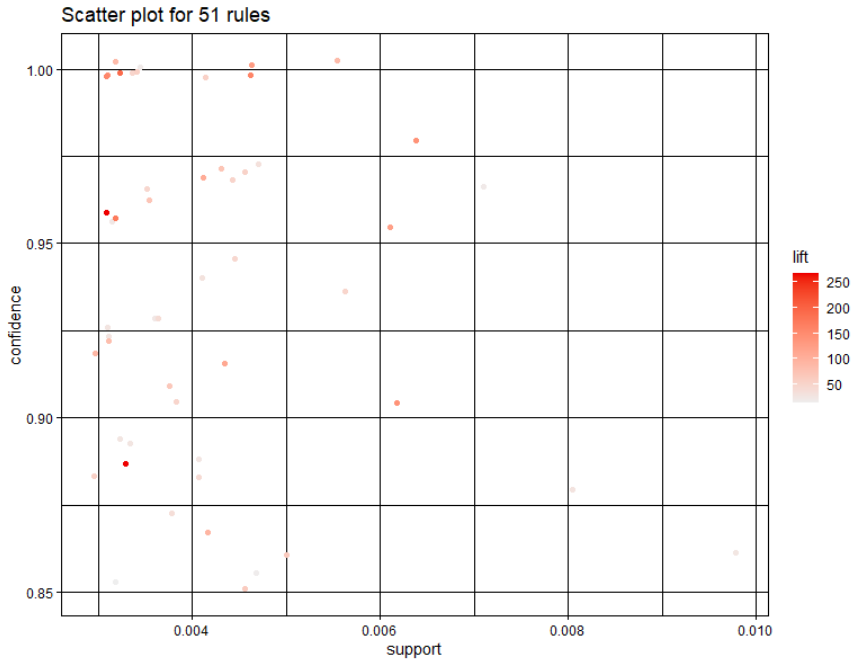


Figura 4.22.: Gráfico de dispersión coloreado en función del lift

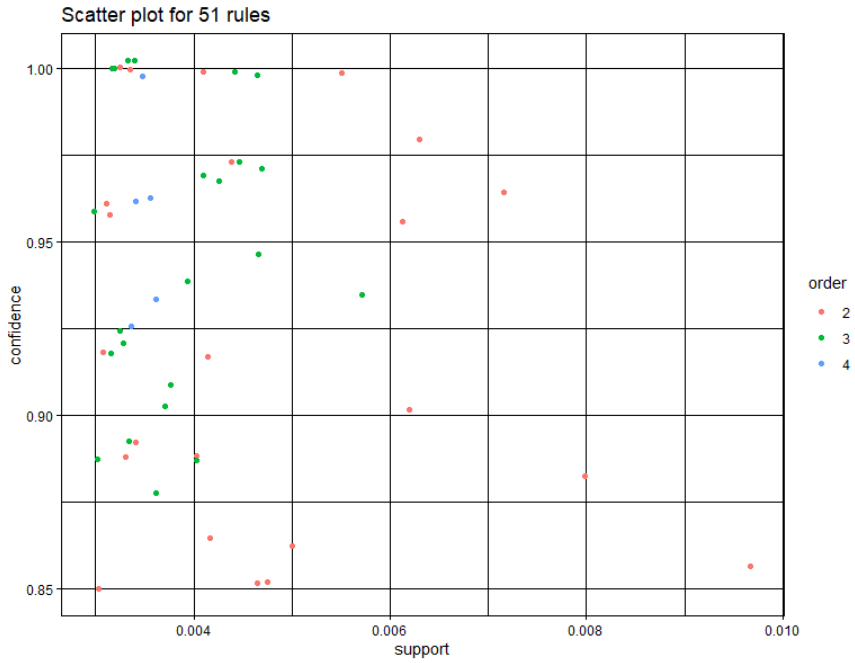


Figura 4.23.: Gráfico de dispersión coloreado en función del número de ítems

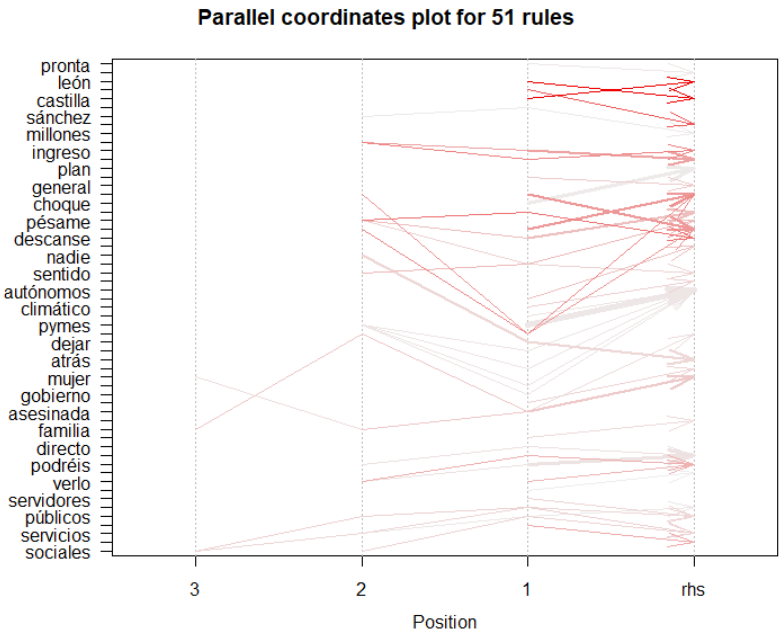


Figura 4.24.: Gráfico de coordenadas paralelas

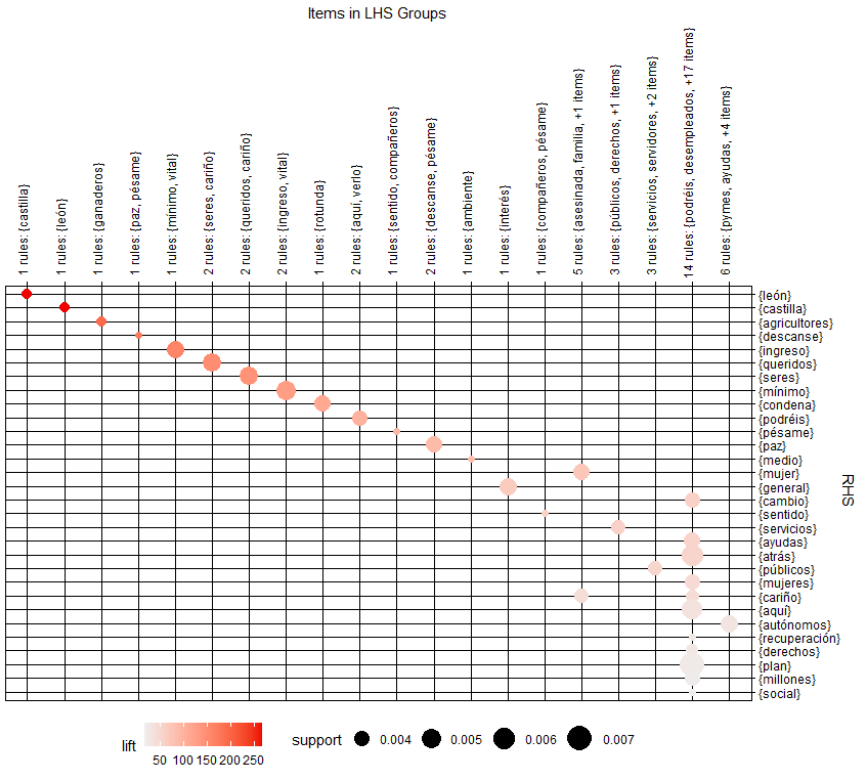


Figura 4.25.: Gráfico de visualización basado en una matriz

Figura 4.26.: Gráfico con etiquetas y flechas

4.6. Clasificación

En esta sección se procede a hacer dos tipos de clasificación. En primer lugar, se pretende clasificar los tweets en función de su autoría, haciendo uso del modelo de aprendizaje estadístico basado en **máquinas de vector soporte**. Por otro lado, se hará una clasificación de palabras haciendo uso del **análisis de sentimientos** estudiando, entre otras cosas, la polaridad y las emociones.

4.6.1. Clasificación de tweets

En este apartado se pretende construir un modelo de aprendizaje estadístico basado en máquinas de vector soporte para deducir la autoría de los tweets. Se aplicará por un lado a los tweets de Pablo Casado e Inés Arrimadas, y por otro lado a los de Pedro Sánchez y Pablo Iglesias.

El fundamento de las máquinas de vector soporte es lo que se conoce como el clasificador de margen máximo (maximal margin classifier) explicado en la Subsección 3.3.1. Otras bases primordiales son el concepto de hiperplano del cual se habló en profundidad en el capítulo anterior, junto con otros conceptos como clasificador de vector soporte o margen suave.

4.6.1.1. Separación de datos

En todo modelo de aprendizaje estadístico conviene separar los datos en entrenamiento y test para evaluar la capacidad del modelo. En nuestro caso, vamos a elegir un 20 % de los tweets aleatoriamente como test y el 80 % restante para el entrenamiento. Se crea la matriz con la que se trabajará, haciendo uso del paquete *quanteda* y funciones como *dfm* y *dfm_tfidf*. Se procede entonces a limpiar y tokenizar los documentos de entrenamiento, así como la eliminación de las "stop words". Además, se eliminan las palabras que aparecen en menos de 5 documentos para eliminar datos que no son relevantes, quedándonos con 3103 tweets para el caso de Arrimadas y Casado, y 3026 para Iglesias y Sánchez. Del mismo modo se procede con el conjunto de documentos de test, obteniendo 776 tweets para Arrimadas y Casado, y 757 para Iglesias y Sánchez. Se puede consultar el código en el cuaderno del Apéndice B.

4.6.1.2. Modelo SVM lineal

Para la experimentación del método, se va a hacer uso del paquete *e1071* y la función *svm()*, donde el argumento *cost* determina el hiperparámetro C explicado en la Subsección 3.3.2 y *kernel="linear"* determina el clasificador de vector soporte explicado en la misma subsección.

Tras entrenar el modelo con un valor de $C = 1$, se utiliza el mismo para predecir la autoría de los tweets del test. Obtenemos como matriz de confusión la mostrada en la Tabla 4.15 para el caso de Arrimadas y Casado, y en la Tabla 4.16 la matriz de Iglesias y Sánchez.

	predicho	
observado	**Inés Arrimadas**	**Pablo Casado**
Inés Arrimadas	63	274
Pablo Casado	3	436

Tabla 4.15.: Matriz de confusión tweets Arrimadas y Casado con C=1

	predicho	
observado	**Pablo Iglesias**	**Pedro Sánchez**
Pablo Iglesias	378	19
Pedro Sánchez	68	292

Tabla 4.16.: Matriz de confusión tweets Iglesias y Sánchez con C=1

Obtenemos entonces 277 clasificaciones incorrectas y un porcentaje de error del 35.7 % en el primer caso, y 87 clasificaciones incorrectas con un error del 11.49 % para los otros dos políticos. Es por ello que tenemos que optimizar el hiperparámetro C. Recurrimos a la validación cruzada explicada en la Subsección 3.3.2 para obtener el valor óptimo haciendo uso de la función *tune()* que se encarga de comparar valores, y los que se utilizarán son 0.1, 0.5, 1, 2.5, 5, 6, 7, 8 y 10.

Podemos ver en la Figura 4.27 el rendimiento del método para el caso de Inés Arrimadas y Pablo Casado, de forma que al estudiar los mejores valores para el parámetro del coste, obtenemos $C = 7$. Al utilizar ese valor en el modelo obtenemos un 10.44 % de error de clasificación, lo que significa que 81 clasificaciones se hicieron incorrectamente (ver Tabla 4.17). Se puede consultar el código y detalles de la salida en el cuaderno del Apéndice B.

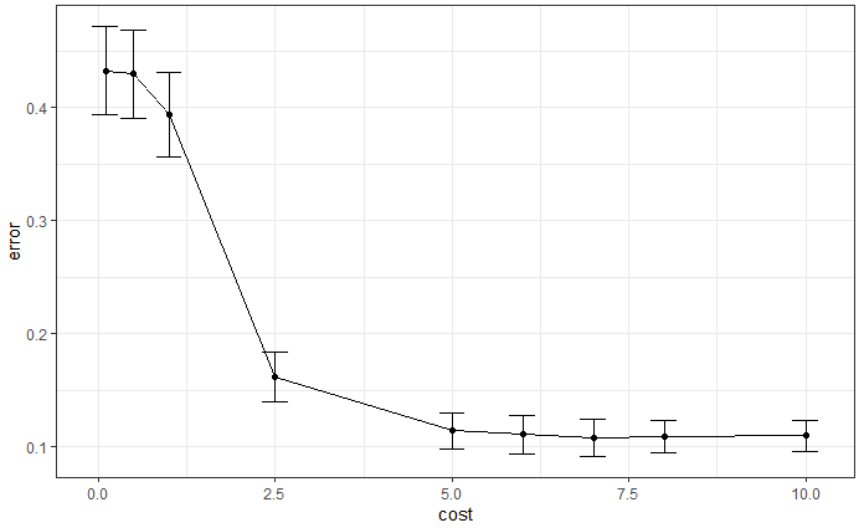


Figura 4.27.: Rendimiento del método al clasificar los tweets de Arrimadas y Casado

	predicho	
observado	**Inés Arrimadas**	**Pablo Casado**
Inés Arrimadas	289	48
Pablo Casado	33	406

Tabla 4.17.: Matriz de confusión tweets Arrimadas y Casado con C=7

La Figura 4.28 nos muestra el rendimiento para Pedro Sánchez y Pablo Iglesias dando como información los mejores valores para el hiperparámetro, en este caso $C = 10$. Obtenemos

entonces un porcentaje de error del 8.72 % y 66 clasificaciones incorrectas (ver Tabla 4.18). Para más detalle consultar el cuaderno del Apéndice B.

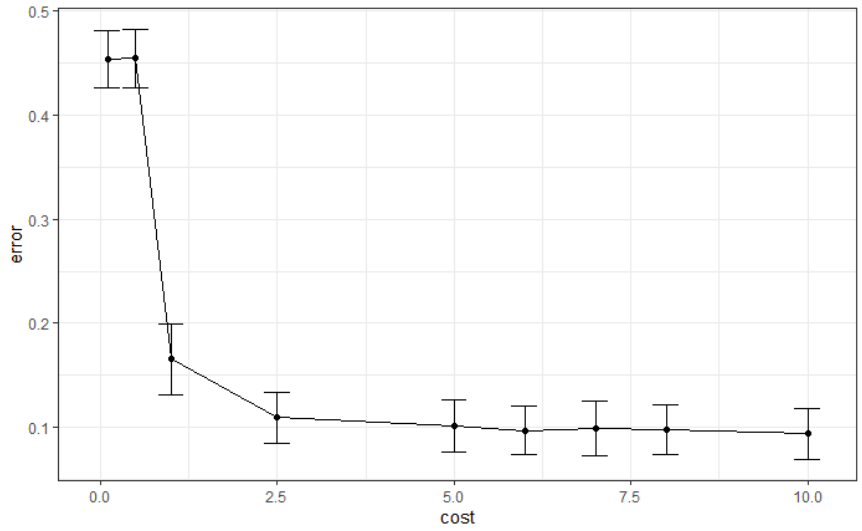


Figura 4.28.: Rendimiento del método al clasificar los tweets de Iglesias y Sánchez

	predicho	
observado	**Pablo Iglesias**	**Pedro Sánchez**
Pablo Iglesias	356	41
Pedro Sánchez	25	335

Tabla 4.18.: Matriz de confusión tweets Iglesias y Sánchez con C=10

Esto se puede interpretar de manera que los tweets de Sánchez e Iglesias son más distintos entre ellos que los otros dos políticos, pues el modelo tiene más dificultades a la hora de hacer una clasificación de los tweets en función a la autoría en el caso de Inés Arrimadas y Pablo Casado.

Podemos comparar los datos obtenidos a lo largo de esta sección, y se observa que en el caso del presidente del gobierno e Iglesias, a pesar de utilizar $C = 10$, que es el número más alto con el que se ha probado el modelo, se podrían clasificar los tweets en función a su autoría de una manera más sencilla, ya que simplemente entrenando el modelo con un valor del coste de 1, obtuvimos un error de clasificación del 11.49 %. De igual manera, en la Figura 4.27 y Figura 4.28 se puede ver cómo hay una clara caída del error para dos momentos distintos en función del coste, produciéndose en la segunda figura mucho antes que en la primera.

4.6.2. Clasificación de palabras

En esta sección se pretende hacer una clasificación de palabras a partir del análisis de sentimientos. Para desarrollarlo, contamos con varios paquetes que contienen diccionarios de sentimientos que nos ayudan a clasificar las palabras de distintas formas. En nuestro caso, y

como el estudio se va a realizar con palabras en español, se va a utilizar el paquete *syuzhet*, el cual tiene cuatro diccionarios de sentimientos: *NRC, Stanford, Afinn y Bing*. El primero ofrece un diccionario en nuestro idioma y permite clasificar en función a la polaridad de una palabra, así como en otras 8 emociones. Estas son: *enfado, anticipación, disgusto, miedo, alegría, tristeza, sorpresa* y *confianza*. Hay que tener en cuenta que se trabaja con traducciones automáticas de términos, por lo que el estudio está sujeto a fallos por la subjetividad del lenguaje y traducciones. Cabe mencionar que los resultados obtenidos en R serán en inglés, a pesar de utilizar un diccionario en español, ya que la salida no está traducida.

Recogemos los tweets que tenemos limpios, tokenizados y sin *stop words* haciendo uso de la función *get_nrc_sentiment()*. Se muestra en la Figura 4.29 un resumen de las frecuencias con que se asocia cada sentimiento a las palabras de los tweets. En ella podemos ver que en media se asocian más sentimientos a las palabras de los tweets clasificados como positivos (0.1306) que negativos (0.09354). También es interesante observar valores como la mediana o los cuartiles, los cuales son nulos. Esto podría deberse a que haya pocas palabras que tengan asignada una emoción en el diccionario.

```
     anger          anticipation       disgust           fear             joy        
 Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.00000   Median :0.00000   Median :0.00000   Median :0.0000   Median :0.0000  
 Mean   :0.03446   Mean   :0.03831   Mean   :0.01782   Mean   :0.0557   Mean   :0.0363  
 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.0000  
 Max.   :5.00000   Max.   :3.00000   Max.   :6.00000   Max.   :5.0000   Max.   :5.0000  
    sadness          surprise          trust          negative         positive       
 Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000  
 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000  
 Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000   Median :0.0000  
 Mean   :0.04296   Mean   :0.02402   Mean   :0.07995   Mean   :0.09354   Mean   :0.1306  
 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000  
 Max.   :7.00000   Max.   :2.00000   Max.   :3.00000   Max.   :7.00000   Max.   :5.0000  
```

Figura 4.29.: Resumen de los sentimientos representados en los tweets

Podemos visualizar en un gráfico de barras las emociones de todas las palabras recogidas para ver cuáles son las más frecuentes en nuestros datos (Figura 4.30). Para calcular la altura de cada barra, partimos de una matriz donde las columnas son las emociones y las filas representan términos de forma que las celdas contienen el número de veces que la palabra ha sido clasificada como un sentimiento. Se procede entonces a sumar los valores de la columna de la emoción correspondiente, para después sumar el resultado de todas las emociones y finalmente dividir cada suma de cada emoción entre el total. De igual manera, se obtiene la misma representación de forma individual para cada político estudiado (Figura 4.31).

En la Figura 4.32 podemos estudiar los resultados obtenidos entre ellos comparando el número de palabras clasificadas según la emoción. Vemos que Pablo Casado ha tenido más palabras clasificadas en las distintas emociones que el resto de políticos. Esto se apoya en los resultados obtenidos en el análisis exploratorio, donde se mostraba que él era el usuario que utiliza más palabras distintas, por lo que tiene sentido ver que más palabras han sido clasificadas en las ocho emociones. Por otro lado, cabe destacar que los cuatro políticos tienen su barra más alta en la "confianza". Esto cobra sentido pues los usuarios estudiados son figuras políticas. Si tomasemos otro tipo de usuarios como estudio, quizás no tendrían una confianza tan alta, ya que los políticos quieren transmitir confianza a la población. También es interesante ver como Casado tiene más palabras clasificadas como miedo que Sánchez como confianza. Por último, otro dato interesante es ver que Pedro Sánchez y Pablo Iglesias son los usuarios con menos palabras clasificadas como "disgusto". Contextualizando

este resultado, Sánchez es el actual presidente del país e Iglesias fue el vicepresidente, por lo que la referencia a situaciones malas o incómodas es más baja que los otros dos políticos.

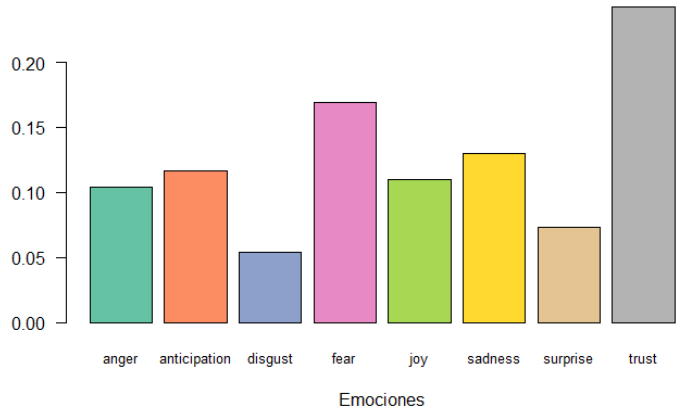


Figura 4.30.: Emociones de todas las palabras

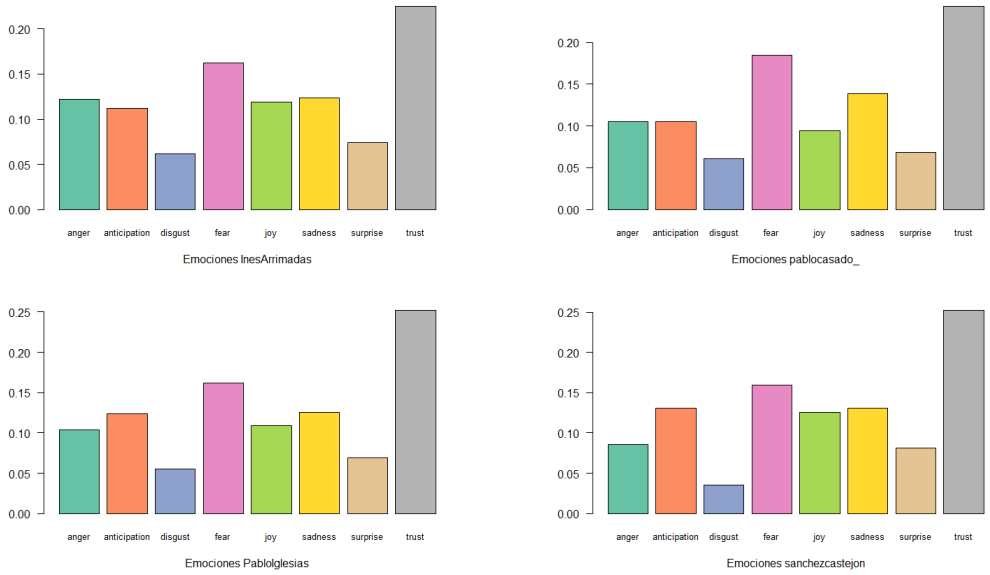


Figura 4.31.: Emociones de las palabras de los distintos políticos

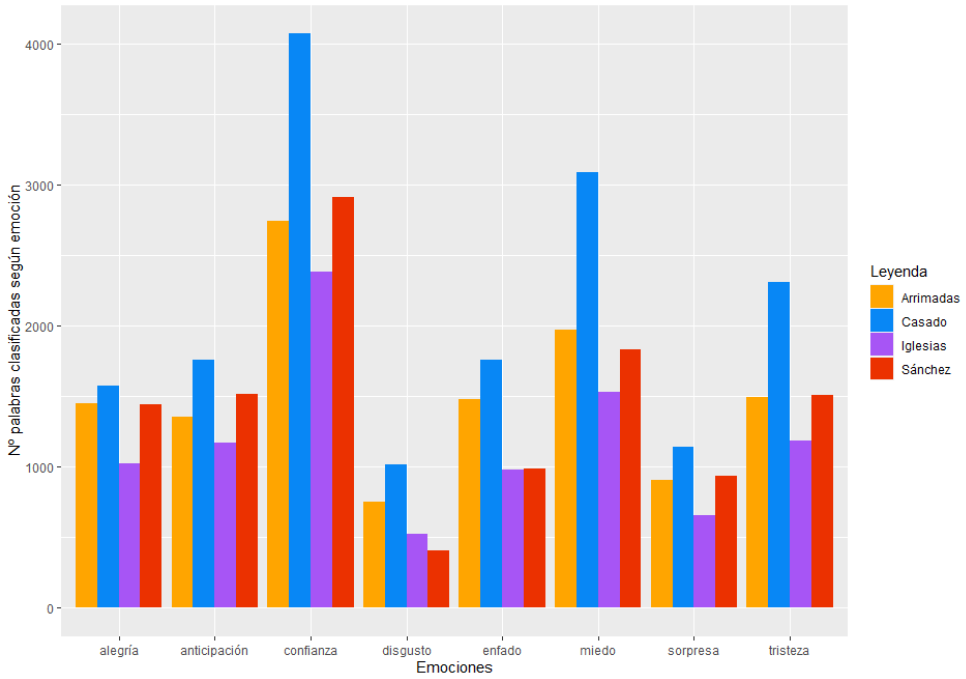


Figura 4.32.: Emociones conjuntas

4.6.2.1. Recuento de palabras con cada emoción

Puede ser interesante conocer qué palabras son las más usadas en función de la emoción. Vamos a recoger, por ejemplo, cuántas palabras se clasificaron como "alegría" y "tristeza": 172 y 270, respectivamente para Pablo Casado, 155 y 223 para Inés Arrimadas, 162 y 211 para Pablo Iglesias y 147 y 169 para Pedro Sánchez. En la Tabla 4.19 se muestran las 5 palabras que más aparecieron y se clasificaron en estas dos emociones para cada político, junto con el número de veces que aparecieron y fueron clasificadas en alegría y tristeza. Podrían haber incongruencias al clasificar palabras en las distintas emociones, ya que se utilizan palabras con una traducción literal, lo que puede acarrear incoherencias en los resultados. El código y más palabras pueden verse en el cuaderno del Apéndice B.

4.6.2.2. Nube de sentimientos

El objetivo de este apartado es crear una nube parecida a la *word cloud* mencionada anteriormente, pero en este caso tendremos las palabras posicionadas en función de cada uno de los sentimientos, colocados en el borde de la nube, y su tamaño de nuevo viene dado por la frecuencia de la misma. Para ello se recogen todas las palabras de cada político en vectores distintos y se hace uso de la función *iconv* del paquete *wordcloud*, la cual convierte un vector de caracteres entre codificaciones. En este caso se tuvo que utilizar ya que había errores con letras tildadas y caracteres como ñ. Posteriormente se crea un corpus de palabras para luego transformarlo en una matriz donde las columnas son las 8 emociones y las filas son los términos, de forma que representa cúantas veces se ha clasificado cierto término como cierto sentimiento.

	tristeza		alegría	
	palabra	**recuento**	**palabra**	**recuento**
Pablo Casado	crisis	159	libertad	184
	pandemia	149	independencia	47
	peor	64	crear	38
	dejar	53	especial	36
	evitar	42	igualdad	36
Inés Arrimadas	crisis	110	libertad	156
	pandemia	86	igualdad	82
	lucha	47	trabajo	76
	violencia	29	orgullo	44
	terrible	28	abrazo	36
Pablo Iglesias	crisis	104	trabajo	68
	compromiso	54	acuerdo	61
	violencia	34	libertad	42
	pandemia	33	igualdad	33
	lucha	26	defensa	26
Pedro Sánchez	pandemia	172	igualdad	90
	compromiso	102	trabajo	84
	crisis	97	acuerdo	80
	emergencia	70	avanzar	80
	lucha	55	progreso	50

Tabla 4.19.: Top 5 palabras más utilizadas clasificadas en tristeza y alegría para cada político

Un ejemplo de lo explicado se puede observar en la Tabla 4.20, donde *anticip.* se refiere al sentimiento de anticipación y las celdas representan el número de veces que el término de la fila ha sido clasificado como la emoción de la respectiva columna. Para más ejemplos se puede consultar el cuaderno del Apéndice B.

	tristeza	**alegría**	**enfado**	**miedo**	**anticip.**	**disgusto**	**confianza**	**sorpresa**
abandonado	5	0	5	5	0	0	0	0
abandonar	7	0	0	7	0	0	0	0
abandono	10	0	10	10	0	0	0	10

Tabla 4.20.: Ejemplo de matriz de sentimientos

Llegados a esta punto, hacemos uso de otra función del paquete *wordcloud* llamada *comparison.cloud*. Esta es la que utilizaremos para la representación de la nube de sentimientos. Este proceso se aplica a todos los políticos, y los resultados se muestran en las imágenes de la Figura 4.33. Es interesante ver como la palabra *pandemia* aparece en todas las imágenes, así como *gobierno*. La aparición de estos términos cobra sentido ya que estudiamos figuras políticas del país y hemos sufrido las consecuencias del conocido *SARS-CoV-2* en los últimos años.

Nube de emociones de Inés Arrimadas

Nube de emociones de Pablo Casado

Nube de emociones de Pablo Iglesias

Nube de emociones de Pedro Sánchez

Figura 4.33.: Nube de emociones de los distintos políticos

4.6.2.3. Evolución de emociones

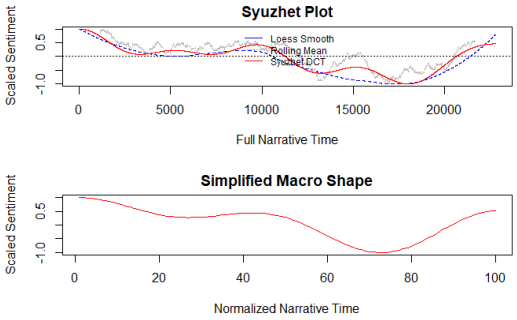
Se puede hacer un estudio evolutivo de las palabras utilizadas por cada usuario en función de su polaridad. Para ello, habría que normalizar los datos que tenemos ya que una palabra se considera positiva cuando su valor se acerca a 1, y negativa cuando lo hace a -1, considerando el 0 como neutro. Esto se calcula multiplicando las cifras de la columna de valores negativos por -1 y sumando el valor de la columna de valores positivos, esto es:

```
valor_sentimientos <- (sentimientos$negative * (-1)) + sentimientos$positive
```

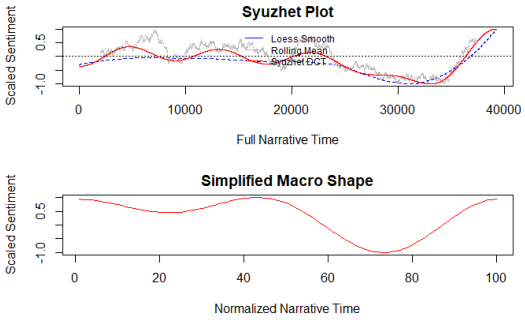
Del mismo modo, para poder hacer una comparativa de las distintas evoluciones de los políticos, es necesario hacer un filtro de los tweets para que estos sean los recogidos en un mismo intervalo temporal. La Tabla 4.1 muestra las fechas de inicio y fin de datos recogidos

por políticos, así que se ha elegido la fecha de inicio más tardía y la fecha final más temprana. De esta manera, los tweets con los que se trabajará serán desde el 04/02/2020 hasta el 04/05/2021.

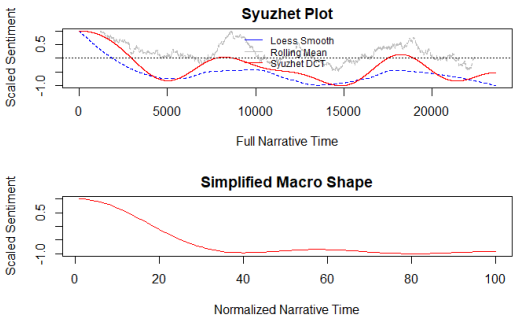
Se procede entonces a representar la evolución de los sentimientos haciendo uso de la función *simple_plot* dentro del paquete utilizado en esta sección. Esta utiliza un vector de sentimientos y aplica tres medidas para suavizar curvas: loess, media móvil y la transformada discreta de coseno (DCT), las cuales fueron explicadas en la Subsección 3.4.1, Subsección 3.4.2 y Subsección 3.4.3, respectivamente. Esta función da como resultado dos imágenes; en la primera se muestran estas tres medidas en el mismo gráfico, y la segunda imagen muestra solo la transformada discreta de coseno, pero en un eje temporal normalizado. La forma de la DCT es idéntica en ambos gráficos. En la Figura 4.34 podemos ver la evolución de la polaridad de los sentimientos de cada político estudiado a lo largo del tiempo. Podemos observar como las gráficas de Inés Arrimadas, Pablo Casado y Pedro Sánchez son ligeramente parecidas, obteniendo unos picos de positividad en un intervalo de $[40, 60]$, siendo estos máximos para Casado y Sánchez, ya que Arrimadas tiene su punto máximo de positividad al principio de la recolección de los datos, como ocurre con Pablo Iglesias. La gráfica de este político es desigual al resto, ya que tiene una clara decadencia en la polaridad de sus tweets, manteniéndose negativa a lo largo de la evolución. Cabe resaltar que ninguno de los políticos presenta una polaridad en sus tweets mayor que 0.5, aunque en el resumen vimos que los tweets eran más positivos que negativos.



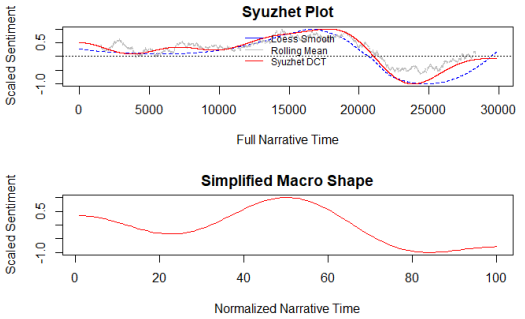
Evolución de Inés Arrimadas



Evolución de Pablo Casado



Evolución de Pablo Iglesias



Evolución de Pedro Sánchez

Figura 4.34.: Evolución de los distintos políticos

5. Conclusiones y trabajos futuros

Para comenzar esta sección, cabe comentar algunas de las dificultades que se han dado a la hora de la realización del trabajo, pues no se ha tratado con minería de datos ni minería de texto anteriormente, así que se ha tenido que hacer un estudio profundo desde cero sobre estas dos áreas, cómo se deben tratar los datos en forma textual y las diferentes técnicas estadísticas y de minería que se pueden aplicar, para así elegir las que podrían ser más interesantes para este proyecto. Dentro de esta investigación sobre el tema, ha sido necesario hacer una comprensión de los datos y resultados obtenidos tras aplicar las reglas para poder contextualizarlos y dotarlos de sentido. Del mismo modo, tampoco se había trabajado anteriormente con el lenguaje de programación R, por lo que un estudio del mismo ha sido necesario previo al inicio del presente trabajo.

Al concluir el trabajo y recapitular lo estudiado y los resultados obtenidos, podemos sacar las conclusiones que se discuten a continuación.

El objetivo de la minería de texto es hacer un estudio de datos que están en formato texto, de forma que se descubre información que era desconocida o no se podía conocer. En nuestro caso, se decidió tratar con texto proveniente de la red social Twitter, específicamente de usuarios que corresponden con cuatro políticos españoles de diferentes partidos, consiguiendo el primer objetivo específico del presente trabajo: realizar un ejemplo de aplicación sobre datos reales. A saber, estos políticos son Pedro Sánchez, Pablo Iglesias, Inés Arrimadas y Pablo Casado. Este estudio se hizo para ver si se mostraban patrones o comportamientos interesantes al aplicar las distintas técnicas estadísticas y de minería como clústering, reglas de asociación, clasificación con máquinas de vector soporte y análisis de sentimientos, además de un análisis exploratorio previo. Efectivamente a lo largo del trabajo se han visto patrones de comportamiento de los políticos que a priori no se conocían o no se podían conocer.

Para llevar a cabo la aplicación de las técnicas, se hizo una búsqueda a través de los diferentes paquetes en R que son adecuados para el estudio y la minería de texto.

En el análisis exploratorio previo a la aplicación de las técnicas podemos destacar, por ejemplo, los resultados obtenidos en relación a Pablo Casado, ya que es el usuario que más palabras utiliza y también el que más palabras distintas usa en sus tweets. Sin embargo, al estudiar las palabras más frecuentes tras haber eliminado las "stop words", tiene dos palabras con una frecuencia muy alta de uso que destacan sobre el resto de palabras y usuarios: "sánchez" y "gobierno". También es interesante ver cómo en el bigrama y trigrama aparecen cojuntos de palabras en las que se hace alusión al pasado.

Por un lado, el clústering mostró la heterogeneidad de los datos, ya que buscar el número óptimo de clústeres no fue una tarea sencilla. En la obtención de reglas de asociación recogimos conjuntos de ítems frecuentes que aparecieron previamente en el análisis exploratorio como bigramas o trigramas con más frecuencia, pero sin embargo, las reglas obtenidas con esos términos no tienen valores de lift o soporte elevados. Esto muestra como, efectivamente, los n-gramas y las reglas de asociación no tienen por qué coincidir. Esto es debido a principalmente dos motivos: por un lado, los bigramas y trigramas son conjuntos de términos que co-ocurren en orden estricto de adyacencia en los tweets, y por otro lado, el método de

obtención de los mismos ha sido distinto.

A la hora de hacer una clasificación de los tweets y hacer uso del modelo lineal con máquinas de vector soporte, se eligieron por un lado a Inés Arrimadas junto con Pablo Casado y, por otro lado, a Pedro Sánchez y Pablo Iglesias para intentar clasificar los tweets por autoría. La razón de esta separación y distribución de los usuarios es que son los políticos que podrían tener más características en común entre ellos y, por tanto, más difícil podría ser "adivinar" la autoría de los tweets. Para Inés y Pablo Casado, tras entrenar el modelo, se obtiene un error del 10.44 % frente a un 8.72 % en el caso de Sánchez e Iglesias. Esto nos muestra que, en pequeña medida, los tweets de Casado y Arrimadas son más parejos que los de los otros dos políticos, ya que el modelo tiene más dificultades y errores a la hora de clasificarlos. Este dato es interesante, pues por un lado Sánchez e Iglesias fueron compañeros en el gobierno y, por el otro, Arrimadas y Casado forman parte de la oposición.

Por último, en el análisis de sentimientos se pretendía estudiar los tweets en función a diferentes emociones, cumpliéndose que los políticos mostraron una alta "confianza" en los mismos. Esto cobra sentido pues los usuarios a estudiar pertenecen a figuras políticas del país y lo que pretenden es dar una imagen y sentimiento de confiaza hacia el pueblo. Del mismo modo se ve en la evolución de la polaridad como, con una pequeña diferencia, tres de los cuatro políticos se comportan de manera parecida a lo largo del gráfico evolutivo, pero Pablo Iglesias parte de una positividad cercana a 0.5 que decae hasta -1 a partir del primer tercio del eje normalizado.

La aplicación de estas técnicas muestran empíricamente la importancia del uso de las mismas en la minería de texto para descubrir información o patrones de comportamiento que son, a priori, desconocidos. Del mismo modo, hemos podido contextualizar los resultados obtenidos a lo largo del estudio, cumpliendo así todos los objetivos propuestos en la Sección 1.1.

Como trabajos futuros, se podrían implementar librerías en R que permitiera hacer diversos tipos de análisis de sentimientos en otros idiomas, como el español. Esto es porque actualmente estamos limitados a la utilización de unas pocas librerías o paquetes como el que se ha usado en este trabajo para el análisis (*syuzhet*). Esto evitaría problemas como la traducción literal de términos y clasificación errónea de los mismos en las distintas emociones, aparte de que se podrían hacer estudios distintos sobre sentimientos como el que ofrece el diccionario *AFINN*, que asigna palabras con una puntuación en el rango de $[-5, 5]$ en función de la polaridad. Aquí, un valor negativo indica sentimiento negativo y puntuaciones positivas indican un sentimiento positivo. Este diccionario es en inglés y la ventaja con respecto al utilizado en el proyecto es que se puede hacer un estudio más detallado sobre la polaridad de los datos. Del mismo modo, se podría añadir una función que recoja "stop words" en español, y otra para la limpieza de los datos; en este trabajo nos hemos visto obligadas a crear una función propia, pues la limpieza que se hacía era incorrecta, ya que en inglés no se hace uso de signos de puntuación como "¿", entre otros. Otro problema viene en la derivación de palabras; no hemos podido aplicarla por la misma razón que mencionamos, ocurrían incongruencias como, por ejemplo, la transformación de la palabra "todas" en "tod", lo cual no era conveniente para nuestro estudio. Podría entonces desarrollarse alguna función que se encargue de hacer una derivación de palabras correcta, siendo el castellano la lengua utilizada.

Por otro lado, en el paquete *syuzhet* hacemos uso del diccionario *NCR* para la clasificación de términos en función a ocho emociones. Sin embargo, la salida de estos resultados siguen siendo en inglés. La propuesta que se ofrece, por tanto, es hacer una traducción en el *output* para que se pueda hacer un estudio 100 % en la lengua que utilizamos y los resultados sean

más visuales y coherentes.

Para finalizar, otro trabajo futuro podría ser la aplicación de otras técnicas de minería como la elaboración de resúmenes, donde se reduce la longitud y el detalle de un documento mientras se retienen los puntos más importantes y el significado general [16].

A. Estimación de costes y planificación

Como parte de este trabajo se presenta a continuación una estimación detallada de los posibles costes que acarrean el presente trabajo. Vamos a establecer un precio por hora de 30€ y se va a dividir el trabajo en 3 grupos:

- Estudio previo en el área de la minería de texto y datos, así como las posibles técnicas y campos aplicativos.
- Implementación de las técnicas estadísticas y el análisis exploratorio previo.
- Composición de la memoria recogiendo lo ejecutado y resultados obtenidos.

Se pueden ver entonces en la Tabla A.1 el desglose de lo comentado, así como el coste total del proyecto.

Concepto	**Cantidad(horas)**	**Precio(€/h)**	**Coste total(€)**
Estudio de la minería de datos	10	30	300
Estudio de la minería de texto	15	30	450
Estudio de técnicas y ámbitos de aplicación	50	30	1500
Extracción de información	30	30	900
Análisis exploratorio	60	30	1800
Clústering	65	30	1950
Reglas de asociación	45	30	1350
Clasificación con máquinas de vector soporte	55	30	1650
Análisis de sentimientos	30	30	900
Redacción de la memoria	70	30	2100
Correcciones	20	30	600
Total	**450**	**30**	**13500**

Tabla A.1.: Tabla de costes

Por otro lado, al comienzo del trabajo de hizo un planteamiento de la forma de trabajo a lo largo del tiempo, mostrado en la Figura A.1. En este, se pretendía escribir el código generado en R de las técnicas a la misma vez que se redactaba el soporte matemático de las mismas.

Sin embargo, el desarrollo del proyecto no ha sido como el que se propuso, sino que ha sido escalado, tal y como se puede ver en la Figura A.2, aparte de haber una ausencia en el desarrollo del mismo durante las primeras semanas de junio. En primer lugar se creó y documentó la parte experimental del trabajo, para posteriormente hacer el soporte matemático de las técnicas estadísticas y de minería aplicadas, lo que acarreó un tiempo de correcciones de la memoria más alto.

Diagrama de Gantt propuesto

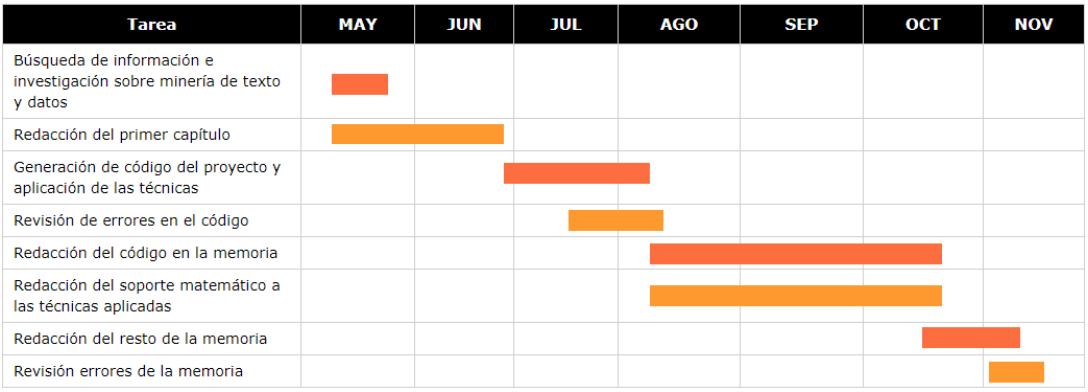


Figura A.1.: Diagrama de Gantt propuesto

Diagrama de Gantt real

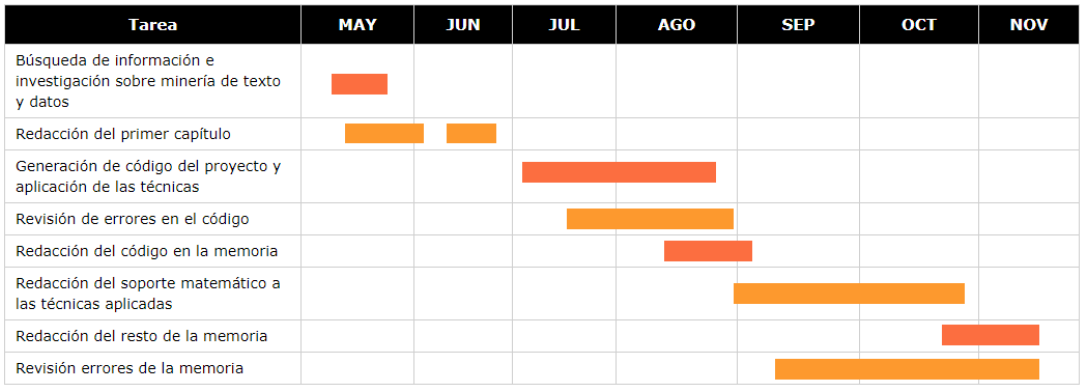


Figura A.2.: Diagrama de Gantt real

B. Software

B.1. Archivos del proyecto

Podemos encontrar el proyecto en la plataforma *Github*, de forma que:

- El cuaderno con el código y el desarrollo producido se encuentra en el fichero **notebook.Rmd**. Es un documento en Markdown R con fragmentos que se pueden ejecutar de forma independiente e interactiva, con la salida visible inmediatamente debajo de la entrada. Del mismo modo, para leer este cuaderno se proporciona el archivo **notebook.nb.html**, el cual muestra el desarrollo del proyecto de una forma clara.
- La carpeta **images** contienen todas las imágenes utilizadas en el cuaderno.
- El fichero **stopwords.txt** recoge las palabras eliminadas al ser consideradas stop words.
- Los ficheros **datos_tweets_@usuario.csv** incluyen los tweets recogidos de la red social para cada usuario estudiado.
- **tweets_AR.csv** abarca los tweets recogidos para su tratamiento en las reglas de asociación.

B.2. Lenguaje

Para la realización de este trabajo se ha utilizado íntegramente el software de desarrollo *RStudio 1.3.1093* para utilizar el lenguaje de programación *R*.

B.3. Paquetes utilizados

Todos los paquetes que se han utilizado en este trabajo están implementados en R. Estos son:

- *tidyverse*: conjunto de paquetes que comparten estructuras de datos y gramática. Algunos de los paquetes que contiene son equivalentes a: *ggplot2*, *dplyr*, *tidyr*, *tibble*, *readr*, entre otros. Abarca las tareas que se repiten en la mayoría de proyectos de ciencia de datos: importación de datos, ordenación, manipulación, visualización y programación [49].
- *tidyr*: herramientas para crear datos ordenados (*tidy data*), donde cada columna es una variable, cada fila es una observación y cada celda contiene un único valor. Contiene herramientas para cambiar la forma y jerarquía (procesos *nest* y *unnest*) de un conjunto de datos, convirtiendo listas anidadas en *data frames*. También incluye herramientas para trabajar con valores perdidos [50].

- *knitr*: Inspirado por *Sweave*, una función en R que permite la integración de código en este lenguaje en LaTeX (crear reportes dinámicos que pueden ser actualizados automáticamente si los datos o el análisis cambian), el paquete knitr fue diseñado para ser un motor transparente para la generación de informes dinámicos con R, y combinar características en otros paquetes complementarios en un solo paquete [51].
- *tidytext*: este paquete se utiliza para la minería de texto haciendo uso de los principios de datos ordenados [46].
- *lubridate*: paquete en R que se encarga de trabajar con datos de fecha y hora [18].
- *tm*: paquete que se utiliza en la minería de texto en R. La estructura principal para la gestión de documentos en este paquete es llamado *corpus*, que representa una colección de documentos de texto [14].
- *rtweet*: se implementó para hacer llamadas, recopilar y organizar datos de Twitter [27].
- *twitteR*: proporciona una interfaz para la API web de Twitter [17].
- *wordcloud*: se utiliza para crear nubes de palabras, visualizar diferencias y similitudes entre documentos y evitar el *overplotting* en diagramas de dispersión con texto [15].
- *RColorBrewer*: proporciona paletas de colores para los gráficos.
- *Rcpp*: paquete disponible en el CRAN (Comprehensive R Archive Network) que permite emplear código C o C++ en proyectos R.
- *factoextra*: funciones para extraer y visualizar la salida de análisis de datos multivariantes. También contiene funciones para simplificar los pasos del análisis de clústering y visualización de datos de una forma clara [26].
- *arules*: proporciona la infraestructura para representar, manipular y analizar patrones y datos de transacciones (conjuntos de ítems frecuentes y reglas de asociación). También proporciona implementaciones en C de los algoritmos de minería de asociaciones como Apriori [20].
- *arulesViz*: extiende el paquete *arules* para representar gráficamente las reglas de asociación.
- *quanteda*: paquete R que proporciona un flujo de trabajo integral y un conjunto de herramientas para tareas de procesamiento del lenguaje natural, como la gestión de corpus, la tokenización, el análisis y la visualización [5].
- *e1071*: ofrece funciones para análisis de clases latentes, transformada de Fourier de tiempo corto, agrupamiento difuso, máquinas de vectores de soporte, cálculo de la ruta más corta... [34].
- *syuzhet*: extrae el sentimiento del texto utilizando una variedad de diccionarios de sentimientos [24].

Bibliografía

Las referencias se listan por orden alfabético. Aquellas referencias con más de un autor están ordenadas de acuerdo con el primer autor.

[1] Agrawal, R., Imieliński, T., y Swami, A. Mining association rules between sets of items in large databases. En *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, páginas 207–216, 1993. [Citado en pág. 25]

[2] Agrawal, R., Srikant, R., et al. Fast algorithms for mining association rules. En *Proc. 20th int. conf. very large data bases, VLDB*, volumen 1215, páginas 487–499. Citeseer, 1994. [Citado en pág. 22]

[3] Ahmed, N., Natarajan, T., y Rao, K. R. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974. [Citado en pág. 41]

[4] Bakshi, R. K., Kaur, N., Kaur, R., y Kaur, G. Opinion mining and sentiment analysis. En *2016 3rd international conference on computing for sustainable global development (INDIACom)*, páginas 452–455. IEEE, 2016. [Citado en pág. 11]

[5] Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., y Matsuo, A. quanteda: An r package for the quantitative analysis of textual data. *Journal of Open Source Software*, 3(30):774, 2018. [Citado en pág. 86]

[6] Berzal, F. Reglas de asociación. Departamento de Ciencias de la Computación e IA, Universidad de Granada, 2016. [Citado en pág. 25]

[7] Berzal, F. Clustering jerarquico. *Universidad de Granada. Disponible en línea: http://elvex. ugr. es/idbis/dm/slides/42 %20Clustering último acceso octubre 2021*, 2017. [Citado en pág. 13]

[8] Berzal, F., Blanco, I., Sánchez, D., y Vila, M.-A. Measuring the accuracy and interest of association rules: A new framework. *Intelligent Data Analysis*, 6(3):221–235, 2002. [Citado en págs. 23 and 24]

[9] Bholowalia, P. y Kumar, A. Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9):17–24, 2014. [Citado en pág. 18]

[10] Cambronero, C. G. y Moreno, I. G. Algoritmos de aprendizaje: knn & kmeans. *Inteligencia en Redes de Comunicación, Universidad Carlos III de Madrid*, 23, 2006. [Citado en pág. 16]

[11] de la Torre, J., del Consuelo, M., et al. *Nuevas técnicas de minería de textos: Aplicaciones*. Tesis Doctoral, Universidad de Granada, 2017. [Citado en págs. 6 and 8]

[12] Faulín, J. y Juan, Á. A. Simulación de monte carlo con excel. 2005. [Citado en pág. 19]

[13] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., et al. Knowledge discovery and data mining: Towards a unifying framework. En *KDD*, volumen 96, páginas 82–88, 1996. [Citado en pág. 6]

[14] Feinerer, I. Introduction to the tm package text mining in R. *Accesible en línea: http://cran. r-project. org/web/packages/tm/vignettes/tm. pdf último acceso octubre 2021*, 2013. [Citado en pág. 86]

[15] Fellows, I., Fellows, M. I., Rcpp, L., y Rcpp, L. Package 'wordcloud'. *R Package, Maintainer Ian and Rcpp, Linking To and Rcpp*, 2018. [Citado en pág. 86]

[16] Gaikwad, S. V., Chaugule, A., y Patil, P. Text mining methods and techniques. *International Journal of Computer Applications*, 85(17), 2014. [Citado en pág. 81]

[17] Gentry, J., Gentry, M. J., RSQLite, S., y Artistic, R. L. Package 'twitter'. *R package version*, 1(9), 2016. [Citado en pág. 86]

[18] Grolemund, G. y Wickham, H. Dates and times made easy with lubridate. *Journal of statistical software*, 40(1):1–25, 2011. [Citado en pág. 86]

[19] Gu, G.-F., Zhou, W.-X., et al. Detrending moving average algorithm for multifractals. *Physical Review E*, 82(1):011136, 2010. [Citado en pág. 39]

[20] Hahsler, M., Buchta, C., Gruen, B., Hornik, K., Johnson, I., Borgelt, C., y Hahsler, M. M. Package 'arules', 2021. [Citado en pág. 86]

[21] Hearst, M. What is text mining. *SIMS, UC Berkeley*, 5, 2003. [Citado en pág. 5]

[22] Hussein, N., Alashqur, A., y Sowan, B. Using the interestingness measure lift to generate association rules. *Journal of Advanced Computer Science & Technology*, 4(1):156, 2015. [Citado en pág. 25]

[23] Jacoby, W. G. Loess: a nonparametric, graphical tool for depicting relationships between variables. *Electoral Studies*, 19(4):577–613, 2000. [Citado en págs. 35, 36, 37, and 38]

[24] Jockers, M. Package 'syuzhet'. *URL: https://cran. r-project. org/web/packages/syuzhet, último acceso octubre 2021*, 2017. [Citado en págs. 35 and 86]

[25] Kao, A. y Poteet, S. R. *Natural language processing and text mining*. Springer Science & Business Media, 2007. [Citado en pág. 10]

[26] Kassambara, A. y Mundt, F. Package 'factoextra'. *Extract and visualize the results of multivariate data analyses*, 76, 2017. [Citado en pág. 86]

[27] Kearney, M. W. y Kearney, M. M. W. Package 'rtweet'. *URL: https://cran. r-project. org/web/packages/rtweet/rtweet. pdf último acceso octubre 2021*, 2016. [Citado en pág. 86]

[28] Khatri, M. D. y Dhande, S. History and current and future trends of data mining techniques. *IJARCSMS International Journal of Advance Research in Computer Science and Management Studies*, 2(3), 2014. [Citado en pág. 1]

[29] Kitajima, H. A symmetric cosine transform. *IEEE Transactions on Computers*, 29(04):317–323, 1980. [Citado en pág. 41]

[30] Liu, B. y Zhang, L. A survey of opinion mining and sentiment analysis. En *Mining text data*, páginas 415–463. Springer, 2012. [Citado en pág. 11]

[31] Martínez, C. G. Reglas de asociación. RPubs, https://rpubs.com/Cristina_Gil/Reglas_Asociacion último acceso octubre 2021, 2020. [Citado en págs. 21, 22, 25, 26, and 27]

[32] McNicholas, P. D., Murphy, T. B., y O'Regan, M. Standardising the lift of an association rule. *Computational Statistics & Data Analysis*, 52(10):4712–4721, 2008. [Citado en pág. 25]

[33] Mehta, C. R. y Patel, N. R. A network algorithm for performing fisher's exact test in r × c contingency tables. *Journal of the American Statistical Association*, 78(382):427–434, 1983. [Citado en pág. 28]

[34] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C., Lin, C.-C., y Meyer, M. D. Package 'e1071'. *The R Journal*, 2019. [Citado en pág. 86]

[35] Miner, G., Elder IV, J., Fast, A., Hill, T., Nisbet, R., y Delen, D. *Practical text mining and statistical analysis for non-structured text data applications*. Academic Press, 2012. [Citado en pág. 6]

[36] Moya, J. P. A. Procesamiento digital de señales. *Instituto Tecnológico de Costa Rica*, 2011. [Citado en pág. 39]

[37] of Cincinnati Business Analytics students, U. K-means Cluster Analysis. Github, https://uc-r.github.io/kmeans_clustering último acceso octubre 2021. [Citado en págs. 16, 18, and 19]

[38] Patel, F. N. y Soni, N. R. Text mining: A brief survey. *International Journal of Advanced Computer Research*, 2(4):243, 2012. [Citado en págs. 7 and 9]

[39] Piatetsky-Shapiro, G. Discovery, analysis, and presentation of strong rules. *Knowledge discovery in databases*, páginas 229–238, 1991. [Citado en pág. 23]

[40] Pujari, A. K. *Data mining techniques*. Universities press, 2001. [Citado en pág. 5]

[41] Rao, K. R. y Yip, P. *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014. [Citado en págs. 39, 41, and 42]

[42] Ravi, K. y Ravi, V. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-based systems*, 89:14–46, 2015. [Citado en pág. 11]

[43] Rodrigo, J. A. Máquinas de Vector Soporte (Support Vector Machines, SVMs). https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines último acceso septiembre 2021, 2017. [Citado en págs. 29, 32, 33, and 34]

[44] Ruiz Ruiz, P. Clustering. http://www.pabloruizruiz10.com/resources/Curso-Machine-Learning-Esp/5—Aprendizaje-No-Supervisado/Intro-Clustering.html último acceso noviembre 2021. [Citado en pág. 14]

[45] Shi, C., Wei, B., Wei, S., Wang, W., Liu, H., y Liu, J. A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):1–16, 2021. [Citado en pág. 18]

[46] Silge, J. y Robinson, D. tidytext: Text mining and analysis using tidy data principles in r. *Journal of Open Source Software*, 1(3):37, 2016. [Citado en pág. 86]

[47] Tibshirani, R., Walther, G., y Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001. [Citado en págs. 19 and 21]

[48] Weinberger, H. F. *Ecuaciones diferenciales en derivadas parciales*. Reverte, 2012. [Citado en pág. 41]

[49] Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., et al. Welcome to the tidyverse. *Journal of open source software*, 4(43):1686, 2019. [Citado en pág. 85]

[50] Wickham, H. y Wickham, M. H. Package 'tidyr'. *Easily Tidy Data with spread and gather() Functions*, 2017. [Citado en pág. 85]

[51] Xie, Y. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, 2017. [Citado en pág. 86]

[52] Xu, R. y Wunsch, D. *Clustering*, volumen 10. John Wiley & Sons, 2008. [Citado en pág. 13]

[53] Zaki, M. J., Meira Jr, W., y Meira, W. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014. [Citado en págs. 2, 13, 16, 17, 22, 28, 29, 30, 31, 33, and 34]

[54] Zhou, H. B. y Gao, J. T. Automatic method for determining cluster number based on silhouette coefficient. En *Advanced Materials Research*, volumen 951, páginas 227–230. Trans Tech Publ, 2014. [Citado en pág. 19]

[55] Zuluaga Blanco, D. C., Vanegas Baquero, G. F. Y., et al. Qué es el mercado forex y cómo se proyecta en colombia para el 2008. B.S. thesis, Universidad de La Sabana. [Citado en pág. 11]