Grado en Matemáticas y Estadística

TRABAJO FIN DE ESTUDIOS

Introducción a la Minería de texto con ayuda de R

Alejandro Lobato Cantos

Sevilla, Octubre de 2023

Índice general

Resumen

En este trabajo se ofrecen unos primeros pasos para introducirse en el campo de la minería de textos, desde una base teórica de las principales tareas hasta una implementación utilizando el software "R". El objetivo principal es intentar dar una visión general de las tareas de las que se encarga esta rama así como representar algunas de sus aplicaciones.

Los resultados y la teoría vienen de una diversidad de fuentes. A la hora de la implementación práctica utilizamos tanto aplicaciones construidas para este trabajo así como una serie de librerías específicas de R creadas para la minería de texto.

Las subramas de la minería de texto que se desarrollan en este trabajo son la reducción de dimensionalidad, el resumen de textos, la clasificación de textos y la extracción de información.

Abstract

In this project we offer some steps towards the learning of text mining, including both some base theory about the main tasks and practical applications using the "R" software. The main purpose is to give an overview of the task text mining takes care of and represent some of its applications.

Both the theory and the results come from a variety of sources. For the practical applications we use both code created for this project as well as "R" libraries that include specific functions developed for text mining.

The sub-branches disscused in this project are dimensionality reduction, text summarization, text classificaction and information extraction.

Índice de figuras

Capítulo 1

Introducción a la minería de texto

La minería de texto es una rama que proviene de la minería de datos cuya característica principal es que se enfoca en obtener información a partir de textos. Los textos empleados pueden tomar distintas formas, desde mensajes cortos como interacciones en redes sociales, mensajes de texto, correos electrónicos o reseñas de productos hasta textos más elaborados y de mayor longitud como libros, páginas web o artículos por ejemplo.

El papel de la minería de texto es el de recaudar información de estas fuentes, las cuales son más difíciles de estudiar con técnicas estándares de minería de datos. Esto se debe a que la información está dispuesta de una forma distinta a la usual y como mínimo hace falta realizar transformaciones para empezar a trabajar. Por ello los procedimientos habituales no son eficaces a la hora de estudiar los textos o es necesario modificarlos para que funcionen dentro de este contexto.

1.1. Tareas de la minería de texto

Dentro de la Minería de texto existe una gran variedad de tareas a realizar. Desde tareas de limpieza y selección, hasta tareas de clasificación, reconocimiento de términos y análisis de sentimientos o ideas presentes en un texto. En este apartado presentaremos algunas de estas tareas y expondremos algunas de los procedimientos empleados para resolverlas.

1.1.1. Reducción de dimensionalidad

La reducción de la dimensionalidad es una técnica de minería de texto que se utiliza para reducir la cantidad de características o variables que se tienen en cuenta en un conjunto de datos, sin perder información relevante. En otras palabras, se trata de simplificar los datos para poder procesarlos y analizarlos de manera más eficiente.

En el contexto de la minería de texto, la reducción de la dimensionalidad se utiliza para manejar conjuntos de datos que contienen un gran número de palabras, términos o características, lo que puede dificultar el procesamiento y análisis de la información, esto es especialmente útil como paso previo a algunos de los procesos anteriormente mencionados como la clasificación o el clustering. La reducción de la dimensionalidad permite representar los datos en un espacio de menor dimensión, lo que facilita el análisis y la visualización de los mismos.

Una de las principales técnicas de reducción de la dimensionalidad es la **indexación semántica latente** (LSI por sus siglas en inglés), la cual consiste en utilizar la descomposición en valores singulares para proyectar palabras con significados y ámbitos similares a un mismo espacio, permitiendo utilizar distintas palabras con significados similares de forma intercambiable. Esto en gran medida soluciona el problema de la polisemia y ayuda a reducir el tamaño de los conjuntos.

Otra de las técnicas más utilizadas es la de los **modelos temáticos**, en ella en lugar de proyectar cada palabra a un espacio semántico, se pretende dar una probabilidad de inclusión en cada uno de los temas. Esto además está expresado en términos probabilísticos haciendo fácil incorporarlo a otros procesos posteriores.

Finalmente la **asignación latente de Dirichlet** nos permite realizar este tipo de asignaciones cuando no tengamos un conjunto de temas de partida.

1.1.2. Extracción de información

La extracción de información es una de las áreas fundamentales de la minería de texto, consiste en encontrar información estructurada partiendo de un texto en el que esta información no lo está. Las dos subtareas fundamentales de las que se encarga son la extracción de relaciones y el reconocimiento de entidades nombradas. La primera consiste en encontrar relaciones semánticas entre distintas entidades mientras que el trabajo de la segunda es encontrar estas entidades, como pueden ser personas, lugares, obras u organizaciones.

Algunas de las aplicaciones más comunes son la biomedicina en la que el problema de la polisemia crea la necesidad de poder realizar búsquedas por entidades de una forma más precisa que buscadores convencionales. Los servicios de inteligencia también utilizan esta información para poder localizar información detallada acerca de nombres de personas relacionados con ciertos eventos bélicos, los objetivos o equipo utilizado. Finalmente, en el mundo de las finanzas, se necesita encontrar de forma rápida y efectiva todas las operaciones de compra-venta relacionada con ciertas organizaciones u empresas para lo cual la extracción de información es de vital importancia.

1.1.2.1. Reconocimiento de entidades nombradas

Dentro del reconocimiento de entidades nombradas podemos encontrar dos formas principales de atacar el problema, una que utiliza reglas definidas previamente y una más cercana al Machine Learning que utiliza modelos estadísticos para intentar resolver el problema.

La aproximación basada en reglas parte de una serie de reglas definidas de antemano. Estas reglas utilizan un patrón que al ser detectado en el texto genera una acción, esta acción suele ser la asignación a un cierto tipo de entidad. Los patrones por otra parte suelen ser una expresión regular que tiene que estar presente en una palabra o conjunto de palabras consecutivas. El problema que tiene la aplicación de reglas es que la generación manual de estas es un proceso costoso ya que requiere una extensa participación de personas especializadas para crearlas. Por otro lado la generación automática de reglas requiere de un conjunto de documentos de entrenamiento donde las palabras han sido ya marcadas con su tipo de entidad. Ambas aproximaciones se han utilizado de forma exitosa en el pasado.

En la aproximación estadística tratamos el problema de la siguiente manera, tenemos un vector de observaciones secuenciadas ($x_1, x_2, ..., x_n$), a cada una de las observaciones x_i de este vector le queremos dar una etiqueta y_i que distinga las palabras que son entidades nombradas de las que no y que además contenga el tipo de entidad que es. Esta etiqueta y_i posiblemente no dependerá únicamente de la observación x_i sino que puede depender de otras observaciones y etiquetas por lo tanto simplemente utilizar métodos de clasificación convencionales puede no dar los mejores resultados.

1.1.2.2. Extracción de relaciones

La extracción de relaciones es una subtarea de la extracción de información que consiste en obtener de un texto pares de entidades y las relaciones semánticas existentes entre ellas. Por ejemplo si tenemos una frase como: "El granadino Federico García Lorca escribió Bodas de Sangre."Deberíamos ser capaces de extraer:

NacioEn(Federico García Lorca,Granada).

CreadorDe(Federico García Lorca, Bodas de sangre).

Las técnicas más simples para resolver este problema consisten en tratar el problema como un problema de clasificación en el que para cada par de entidades dentro de la misma frase intentamos comprobar si pertenece a una serie de relaciones previamente definidas. Dentro de esta froma de trabajar existen distintos métodos, nos centraremos en los métodos del núcleo.

1.1.3. Clasificación

La tarea de la clasificación de textos es parecida a la equivalente en un ámbito general del Machine Learning y algunas de sus técnicas serán similares con adaptaciones. Por ello se podrán aplicar modificaciones de estas técnicas para la clasificación de textos. Previamente antes de la aplicación de los distintos métodos de clasificación es importante realizar un filtro de términos dentro de los textos, eliminando palabras genéricas que están presentes en todos los textos y conservar aquellos términos que aporten más información específica.

Un uso habitual de la clasificación de textos es la organización de documentos, en muchos casos se tiene una colección de documentos muy grande y para poder analizarlo hace falta clarificarlos en una serie de temas; esto se tiene por ejemplo en el ámbito legal debido a la gran cantidad de leyes y otras ordenanzas aprobadas durante los años. Otra aplicación común es la clasificación de correos en Spam o No spam. Finalmente otro uso que se le da a la clasificación es en el ámbito del desarrollo de aplicaciones es el clasificar las peticiones de los usuarios para que lleguen directamente a la persona pertinente sin tener que dedicar el tiempo de una persona a clasificarlas.

Al ser la clasificación un problema ampliamente tratado existen una gran cantidad de métodos de clasificación en minería de texto, cada uno con sus propias ventajas y desventajas. En este trabajo desarrollaremos los siguientes tipos:

Clasificación basada en vecinos cercanos: Este método clasifica los datos de texto en función de la similitud con los ejemplos de entrenamiento. Cada instancia se compara con los ejemplos de entrenamiento y se le asigna la etiqueta de la instancia más cercana. Este método es fácil de implementar y puede ser muy efectivo para clasificar datos similares

a los ejemplos de entrenamiento, pero puede no funcionar bien en casos en los que los datos de prueba son muy diferentes a los ejemplos de entrenamiento.

Clasificación basada en reglas: Este método utiliza reglas definidas por el usuario para categorizar los datos de texto. Por ejemplo, se pueden definir reglas para buscar palabras clave específicas en el texto y asignar una categoría a cada instancia que contenga esas palabras clave. Este método es simple y rápido de implementar, pero puede ser limitado en su capacidad para clasificar datos más complejos o ambiguos.

Clasificación basada en árboles de decisión: Este método utiliza un árbol de decisiones para clasificar los datos de texto. El árbol de decisiones se construye a partir de un conjunto de ejemplos etiquetados y cada nodo del árbol representa una pregunta que se hace sobre las características del texto. A medida que se avanza en el árbol, se llega a una hoja que representa la categoría final del texto. Este método puede ser muy efectivo para clasificar datos complejos, pero puede requerir un conjunto de entrenamiento grande y puede ser propenso a sobreajuste.

Clasificación basada en aprendizaje profundo: Este método utiliza redes neuronales para clasificar los datos de texto. Las redes neuronales son capaces de aprender características complejas del texto y pueden ser muy efectivas para clasificar datos complejos y ambiguos. Sin embargo, este método puede requerir un conjunto de entrenamiento muy grande y puede ser difícil de interpretar.

1.1.4. Resumen de textos

La tarea del resumen de texto consiste en encontrar la información más relevante de un texto y expresarla utilizando frases del mismo que nos den un significado general del mismo. En la práctica las técnicas de resumen suelen tener dos pasos. En un primer paso se realiza un proceso de asignación de puntuaciones o indicadores a las distintas frases del texto, las frases que contengan contenido más específico y menos parecido del resto tendrán una mayor puntuación ya que será más importante incluirlas en el resumen, aquellas frases que contengan información repetida deberían obtener una menor puntuación. En un segundo paso se utilizan estas puntuaciones generadas en el primer paso para seleccionar las frases que mejor representan el texto en su totalidad.

Algunos de los usos más habituales es en el resumen de textos de noticias para encontrar la información más relevante de forma ágil. Otra posible aplicación es el análisis de contratos, en este caso puede detectar cláusulas más arriesgadas o permite comparar condiciones dentro de estos. Existen muchos más casos de campos en los que los resúmenes no son generados de forma general y la automatización de estos puede ahorrar tiempo a la hora de entender si el documento puede tener contenido de interés como en el caso de los e-mails, la investigación de patentes o el análisis de redes sociales.

1.1.4.1. Puntuación de frases

Hay una gran variedad de técnicas de representación de temas, expondremos algunas de ellas.

Palabras temáticas: La idea de esta técnica es medir como de raras son cada una de las palabras del texto en comparación a un gran conjunto de textos. Utilizando test de verosimilitud y la distribución de Bernouilli calculamos un estadístico que nos diga como

de improbable es encontrar ciertas palabras en los textos, estas palabras con un mayor valor del estadístico serán las que consideremos temáticas. La puntuación de las frases vendrá dada por la cantidad de palabras temáticas que contengan o la proporción de estas, si usamos la cantidad tenderemos a puntuar más las frases más largas.

Método Frecuentista: Esta interpretación es una modifiación de usar palabras temáticas donde en lugar de dar pesos binarios a las palabras (0 o 1) da un valor continuo en función de como de probable es cada palabra en el texto, en teoría esto nos aporta más información y nos ayuda a medir de una manera más precisa qué palabras y frases nos aportan más información sobre el texto.

Modelos Bayesianos de Temas: Este modelo es el más avanzado y sofisticado de los propuestos. En el modelo se obtienen las probabilidades de aparición de una palabra para distintos conjuntos; el idioma en general, uno para todo el conjunto de documentos y uno para cada documento a resumir.

1.1.4.2. Selección de frases para el resumen

Este proceso consiste en añadir las frases con mejor puntuación hasta llegar a la extensión requerida para el resumen. También se suele aplicar algún tipo de medida para que todas las frases incluidas no sean muy parecidas entre ellas, mejorando así la calidad de la información aportada.

Veremos dos métodos para la selección de frases. El método de **máxima relevancia marginal**, que en cada paso toma la frase que tiene una mejor puntuación pero las penaliza si se parecen a algunas ya incluidas. Este método tiene un problema y es que si una de las primeras frases es muy larga pero muy relevante e incluye parte de información poco útil; puede limitar el espacio sobrante del resumen haciendo difícil la elección de más frases que completen la información.

El **segundo método** es encontrar una solución global que maximize la información, minimizando la repetición y limitando el espacio máximo. En general encontrar esta solución es un problema computacionalmente muy costoso (NP), por lo que se requiere buscar soluciones aproximadas.

1.1.5. Técnicas de agrupamiento

El clustering es una técnica de minería de texto que consiste en agrupar los documentos en conjuntos o clusters, de acuerdo a su similitud. Los documentos que se encuentran en el mismo cluster comparten características comunes y pueden ser considerados como pertenecientes a una misma categoría o tema. El objetivo del clustering es identificar grupos o patrones de información dentro de los datos, para así poder hacer análisis y tomar decisiones informadas.

Existen varios métodos de clustering en la minería de texto, pero los más comunes son el **clustering jerárquico** y el **clustering no jerárquico**. El clustering jerárquico es un método que se basa en la construcción de una jerarquía de clusters, donde cada cluster contiene subclusters más pequeños. En cambio, el clustering no jerárquico es un método que agrupa los documentos en un número predeterminado de clusters, sin construir una jerarquía.

Uno de los métodos no jerárquicos más comunes es el de los **k-medioides**. Este método consiste en tomar unos documentos iniciales del total (medioides) y crear clusters mandando cada documento al clúster definido por el medioide más cercano. En cada iteración, el algoritmo sustituye uno de los medioides por otro documento del conjunto total y se recalculan los clusters, parando cuando no haya ningún cambio que mejore una cierta función objetivo. Este algoritmo presenta dos problemas particulares para el caso de la minería de texto. El primero es que es un algoritmo algo lento ya que requiere muchas iteraciones para converger por lo que, en el caso de la minería de texto, ralentiza aún más el proceso. El otro es que muchos documentos no comparten suficientes palabras por lo que es difícil encontrar documentos que aglutinen a un conjunto de documentos y sirva como medioide.

El otro método más común, el de las **k-medias** funciona algo mejor en este caso, ya que converge con una velocidad mucho mayor que el previamente nombrado, aún así no carece de inconvenientes. El principal es que el resultado depende mucho de las semillas iniciales tomadas haciendo que la replicabilidad no sea especialmente buena. Además por como se construyen el centroide en el método de k-medias, vamos a encontrar palabras de muchos clusters distintos por lo que los cálculos posteriores a la primera iteración serán más lentos.

1.1.6. Análisis de sentimientos

El análisis de sentimientos es una técnica de procesamiento de lenguaje natural que permite identificar y clasificar las emociones, opiniones y actitudes expresadas en un texto. Esta técnica se utiliza en diversos campos, como la investigación de mercado, la gestión de reputación online, la atención al cliente y la política.

El análisis de sentimientos se basa en la idea de que las palabras y las frases utilizadas en un texto pueden indicar la presencia de ciertas emociones o actitudes. Por ejemplo, si un texto contiene palabras como “feliz”, “alegre” o “divertido”, es probable que se trate de un texto que expresa emociones positivas. Por otro lado, si un texto contiene palabras como “triste”, “decepcionado” o “enojado”, es probable que se trate de un texto que expresa emociones negativas.

Para realizar el análisis de sentimientos, se utilizan algoritmos de aprendizaje automático que pueden clasificar un texto en una de varias categorías, como positivo, negativo o neutro. Estos algoritmos utilizan diferentes técnicas para identificar y evaluar las emociones y actitudes expresadas en el texto, como el análisis léxico, el análisis sintáctico y el análisis semántico.

El **análisis léxico** se basa en la creación de un diccionario de palabras que se utilizan para expresar emociones y actitudes. Este diccionario se utiliza para asignar una puntuación a cada palabra del texto, en función de su asociación con ciertas emociones o actitudes. Por ejemplo, la palabra “amor” puede tener una puntuación alta en la categoría de emociones positivas, mientras que la palabra “odio” puede tener una puntuación alta en la categoría de emociones negativas.

El **análisis sintáctico** se basa en la identificación de patrones gramaticales que se utilizan para expresar emociones y actitudes. Por ejemplo, las oraciones que contienen un verbo en pasado pueden indicar que el autor está expresando emociones negativas o nostalgia.

El **análisis semántico** se basa en la identificación de la relación entre las palabras y las emociones o actitudes que se expresan en el texto. Por ejemplo, si un texto contiene la frase "estoy muy contento con mi nuevo trabajo", el análisis semántico puede identificar la relación entre la palabra "contento" y la emoción positiva que se está expresando.

En el siguiente capítulo veremos con más detalles los fundamentos teóricas de las áreas descritas en este capítulo e introduciremos el desarrollo teórico de algunos de estos métodos.

Capítulo 2

Marco teórico

En esta sección detallaremos algunos de los procedimientos más comunes en cada una de las áreas mencionadas de la minería de texto. Ofreceremos un esquema general de los procesos utilizados así como métodos específicos utilizados para los cálculos y una breve base teórica previa donde sea necesario.

2.1. Reducción de dimensionalidad

La reducción de dimensionalidad es una técnica utilizada en la minería de texto para abordar el desafío de trabajar con datos de alta dimensionalidad.

La reducción de dimensionalidad tiene como objetivo disminuir la cantidad de características o dimensiones en el conjunto de datos, al tiempo que se mantiene la información relevante y significativa. Esto se logra mediante técnicas que seleccionan un subconjunto de características importantes o transforman los datos originales a un espacio de menor dimensión. Existen muchas técnicas asociadas a la reducción de dimensionalidad pero aquí trataremos solo 2, el LSI (Latent Semantic Indexing) y el LDA (Latent Dirichlet Allocation).

La reducción de dimensionalidad en la minería de texto ofrece varios beneficios, como la reducción del costo computacional, la eliminación de ruido o características irrelevantes, la visualización de datos en espacios de menor dimensión y la mejora del rendimiento de los algoritmos de clasificación o agrupamiento.

2.1.1. Indexación Semántica Latente (LSI)

El LSI es un método de indexación automática que consiste en proyectar los textos y documentos de una matriz término-documento en un espacio de menor dimensión. Para ello utiliza la descomposición en valores singulares y utiliza estos vectores para construir una matriz aproximada de menor rango con la que es más sencillo trabajar.

Definición 2.1.1 Sea X la matriz término-documento de dimensión WxM de un corpus donde, la d-ésima columna $\mathbf{X}_d$ representa el documento d y la v-ésima fila $\mathbf{T}_v$ representa al término v. Sea la descomposición en valores singulares de X:

$$X = U\Sigma V^T,$$

donde las matrices U y V son ortonormales y Σ es la matriz diagonal:

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & ... & \\ & & \sigma_{min(W,M)} \end{bmatrix}$$

Donde los términos $\sigma_1, \sigma_2, ..., \sigma_{min(W,M)}$ son los valores singulares de la matriz X. Sin pérdida de generalidad, asumimos que los valores singulares están ordenados de mayor a menor.

Definimos la matriz término-documento aproximada $\hat{X}$ de rango K como:

$$\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T = \begin{bmatrix} \hat{U}_1 & ... & \hat{U}_K \end{bmatrix} \begin{bmatrix} \hat{\sigma}_1 & & \\ & ... & \\ & & \hat{\sigma}_K \end{bmatrix} \begin{bmatrix} \hat{V}_1^T & ... & \hat{V}_K^T \end{bmatrix}$$

Esta matriz se puede obtener mediante una descomposición parcial manteniendo los vectores singulares correspondientes a los K mayores valores singulares[23].

Esta descomposición nos permite representar documentos y vectores en un espacio K-dimensional. Para un documento $\hat{\mathbf{X}}_d$ se tiene la relación:

$$\mathbf{X}_d = \hat{U}\hat{\Sigma}\hat{\mathbf{X}}_d$$

Por otro lado también tenemos la siguiente relación para los términos:

$$\mathbf{T}_v = \hat{U}\hat{\Sigma}\hat{\mathbf{T}}_v$$

A partir de estas relaciones podemos obtener las proyecciones de documentos y términos en el espacio semántico creado[6].

Con estas aproximaciones $\hat{\mathbf{X}}_d$ y $\hat{\mathbf{T}}_v$ podemos calcular similaridades entre términos y/o documentos a través de la similaridad cosenoidal.

2.1.2. Análisis Latente de Dirichlet (LDA)

El LDA es una técnica probabilística utilizada para modelar los distintos temas o grupos que aparecen en un documento. Esta técnica se basa en distribuciones multinomiales y de Dirichlet para dar probabilidades de cada tema según las frecuencias de las palabras[1].

2.1.2.1. Conceptos previos sobre distribuciones

Definición 2.1.2 La distribución de Dirichlet de orden K con vector de parámetros $\mathbf{Y}$ tiene una función de densidad dada por :

$$D(\mathbf{X};\mathbf{Y}) = \frac{\Gamma(\sum_{i=1}^{K} y_i)}{\prod_{i=1}^{K}\Gamma(y_i)} \prod_{i=1}^{K} x_i^{y_i-1}$$

Por otro lado tenemos la distribución multinomial:

Definición 2.1.3 Dado un experimento multinomial con K casos y probabilidades y_k para cada caso, la función de probabilidad viene dada por:

$$M(\mathbf{X};\mathbf{Y}) = \frac{n!}{x_1!...x_k!} \prod_{i=1}^{K} y_i^{x_i}$$

De donde obtenemos que

$$M(\mathbf{X};\mathbf{Y}) \propto \prod_{i=1}^{K} y_i^{x_i}$$

Ahora veamos la relación que hay entre estas[22]:

Teorema 2.1.1 La distribución de Dirichlet es la distribución conjugada de la distribución multinomial.

Demostración. Nuestra distribución a posteriori sería:

$$P(p|x) \propto P(x|p) * P(p)$$

donde $P(x|p)$ es la distribución multinomial y $P(p)$ es la función a priori, en nuestro caso la distribución de Dirichlet. Entonces tenemos que:

$$P(p|x) \propto \frac{n!}{x_1!...x_k!} \prod_{i=1}^{K} p_i^{x_i} * \frac{\Gamma(\sum_{i=1}^{K} y_i)}{\prod_{i=1}^{K}\Gamma(y_i)} \prod_{i=1}^{K} p_i^{y_i-1} \propto \ \propto \prod_{i=1}^{K} p_i^{x_i} * \prod_{i=1}^{K} p_i^{y_i-1} = \prod_{i=1}^{K} p_i^{x_i+y_i-1},$$

Lo cual es proporcional a una distribución de Dirichlet de parámetros $x_i + y_i$ por lo tanto tenemos que la función a priori y la posteriori pertenecen a la misma familia por lo que tenemos lo que queríamos probar. ■

Por lo tanto la distribución multinomial es la conjugada de la de Dirichlet.

Definición 2.1.4 Un proceso generativo es un algoritmo que describe como se se selecciona una respuesta.

2.1.2.2. Modelo LDA

Tras exponer estas relaciones entre las distribuciones pasamos a describir como funciona el método LDA. Este es un modelo generativo en el que se asume un proceso generativo para cada documento d descrito de la siguiente forma[4]:

1.Se elige una distribución multinomial θ a partir de una distribución Dirichlet de parámetro α.

$$\theta_i \sim \mathcal{D}(\alpha); p(\theta_i|\alpha) = \frac{\Gamma(W\alpha)}{[\Gamma(\alpha)]^W} \prod_{v=1}^{W} \theta_{iv}^{\alpha-1}$$

2.Se elige una distribución multinomial Φ a partir de una distribución Dirichlet de parámetro β.

$$\Phi_i \sim \mathcal{D}(\beta); p(\Phi_i|\beta) = \frac{\Gamma(W\beta)}{[\Gamma(\beta)]^W} \prod_{v=1}^{W} \phi_{iv}^{\beta-1}$$

3. Para cada palabra del documento w_n:

- Se elige un tema $z_n \sim$ Multinomial(θ).

$$z_{dn} \sim \mathcal{M}(\theta|\alpha); p(z_{dn} = i|\theta_d) = \theta_{di}$$

- Se elige una palabra w_n obtenida a través de la distribución de palabras condicionado al tema elegido para esa palabra.

$$w_{dn} \sim \mathcal{M}(\phi_{z_{dn}}); p(w_{dn} = v|z_{dn} = i, \phi_i) = \phi_{iv}$$

Por tanto el modelo nos permite encontrar patrones de co-ocurrencia entre las palabras y los temas. Es decir, si, por ejemplo, tenemos que para un término v y un tema i la probabilidad $p(w = v|z = i)$ es alta; entonces si nos encontramos con un nuevo documento que contiene la palabra v tendrá una probabilidad aumentada de haber sido generado por el tema i. Esto a su vez aumenta la probabilidad de que los términos presentes en el documento hallan sido generados por el tema i.

2.1.2.3. Verosimilitud y estimación de parámetros

Como nuestro objetivo con el LDA es el de encontrar un modelo que represente como se generan los documentos de un cierto corpus es necesario encontrar un conjunto de parámetros que optimizan la probabilidad de generar los documentos, para evaluar lo bueno que es un conjunto de parámetros la medida que se suele utilizar es su probabilidad empírica que es una expresión de verosimilitud.

Definición 2.1.5 Probabilidad Empírica

Dado un modelo LDA se define su probabilidad empírica $\mathcal{L}$ como:

$$\mathcal{L} = \prod_{d=1}^{M} \prod_{n=1}^{N} p(w_{dn}|z_{dn}, \Phi)p(z_{dn}|\theta_d)p(\theta_d|\alpha)p(\Phi|\beta) = \phi_{zw}\theta_{dz}\frac{\Gamma(W\beta)}{[\Gamma(\beta)]^W} \prod_{v=1}^{W} \phi_v^{\beta-1}$$

El problema con esta verosimilitud es que no se puede optimizar de forma directa ya que la asignación de temas para cada palabra z_{dn} no es observable, vemos los términos pero no el tema que se le asignó según el proceso. Por tanto se plantea otra forma de estimar las distribuciones Φ y θ. Esta forma es el muestreo de Gibbs.

2.1.2.4. Muestreo de Gibbs

El objetivo de este procedimiento es hallar un estimador de la distribución de temas dentro de un documento y la distribución de palabras dentro de un tema, para aplicar el algoritmo necesitamos fijar los parámetros α y β así como el número de temas T. También necesitaremos la expresión de la probabilidad condicionada $P(Z_i|\mathbf{Z}_{-1}, |\mathbf{w})$, esta se puede obtener a partir de las distribuciones mencionadas previamente [10], obtenemos:

$$P(z_i = j|\mathbf{z}_{-i},\ \mathbf{w}) \propto \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + W\beta} \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,\cdot}^{(d_i)} + T\alpha}, [1]$$

donde α y β representan los parámetros de las distribuciones Dirichlet a priori, W y T el número de palabras y de temas distintos respectivamente y $n_{-i,j}^{(x_i)}$ es el recuento de las ocurrencias de x para el tema j en todos los elementos de la iteración menos el i-ésimo.

Es decir, para cada palabra generada la probabilidad de su tema, condicionado al resto de asignaciones y a la palabra, es proporcional al producto de estos dos cocientes. El primero representa la probabilidad de la palabra w_i bajo el tema j mientras que el segundo representa la probabilidad del tema j en el documento d_i. Estos recuentos son los únicos elementos necesarios para aplicar el algoritmo por lo que se puede ejecutar de forma eficiente.

El algoritmo funciona de la siguiente manera. A las variables z_i se le dan un valor inicial de forma aleatoria dentro de la lista de temas $1, 2, ..., T$. En cada iteración del proceso a cada variable z_i se le asigna un tema a través de la función de probabilidad [1]. Después de suficientes iteraciones se registran los valores de las z_i y los utilizamos para estimar las siguientes distribuciones:

$$\hat{\theta}_j^{(w)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + W\beta}$$

Este es el estimador de la probabilidad de generar la palabra w dentro del tema j.

$$\hat{\theta}_j^{(d)} = \frac{n_j^{(d)} + \alpha}{n_\cdot^{(d)} + T\alpha}$$

Este es el estimador de la probabilidad de obtener el tema j en el documento d. Estos dos estimadores son los que nos dan la interación entre tema y documento y palabra documento lo cual nos permite asignar a cada palabra su tema más probable y por consiguiente el tema principal de cada documento.

2.2. Resumen de texto

El objetivo del resumen de texto es construir de forma automática el resumen de un texto a partir del mismo. Dentro de las técnicas de resumen de texto nos centraremos en las técnicas de extracción, que son aquellas que utilizan las propias palabras y frases presentes en el texto para elaborar el resumen. Existen otras técnicas que fueron creadas para un resumen más abstracto pero no las discutiremos.

Dentro de las técnicas de resumen extractivas podemos encontrar tres fases, una primera fase en la que se le da un valor a las palabras del texto según lo representativas que son dentro del texto. Una segunda en la que a partir de estas palabras se le asigna un valor a cada una de las frases del texto y finalmente una tercera en la que a partir de estas puntuaciones se seleccionan frases del texto para crear el resumen.

Las formas de puntuar las frases en la segunda fase suele estar estrechamente ligada a la forma de puntuar las palabras por lo que discutiremos ambas fases a la vez en la siguiente sección.

2.2.1. Representación de los temas

Dentro de esta sección veremos varios métodos para obtener una valoración de que palabras aportan mayor significado dentro de un texto.

2.2.1.1. Palabras temáticas

Esta técnica consiste en hallar palabras altamente descriptivas del texto a través del test del ratio de log-verosimilitud. Para esta técnica es necesario una recopilación de textos para poder compararlo con el texto a resumir.

Estas palabras, las cuales se denominan palabras temáticas son aquellas que son poco frecuentes en el corpus pero son muy frecuentes en el texto, utilizando el test podemos hallar que palabras son temáticas y cuales no.

Una vez halladas estas palabras, las frases obtendrán una puntuación según el número de palabras temáticas que contengan o según la proporción de estas. Utilizar la cantidad total premia más las frases más largas mientras que utilizar la proporción premia la concentración de estas palabras.

Para aplicar el test necesitaremos de algunas definiciones previas.

Definición 2.2.1 Definición

Definimos como palabra una sucesión finita de caracteres alfabéticos.

Definición 2.2.2 Definición

Definimos como texto una sucesión de palabras $w_1, w_2, ..., w_n$ separadas por espacios.

Definición 2.2.3 Sea B un conjunto de textos y W(B) el conjunto de palabras presentes en ese conjunto se define la matriz de términos y documentos como la matriz de dimensiones |W(B)|x|B| donde las filas representan los textos de B, las columnas las palabras de W(B) y cada elemento (i,j) es el número de veces que aparece la palabra j en el texto i.

Para comprobar que una palabra w es mas frecuente en un texto t usaremos el siguiente contraste de hipótesis:

$$H0 : P(w|t) = P(w|B)$$

$$H1 : P(w|t) \neq P(w|B)$$

donde B es un conjunto de textos $t_1, ..., t_m$.

Teorema 2.2.1 Dado un conjunto de textos D, una palabra w y un texto t rechazaremos el anterior contraste de hipótesis para un nivel de confianza de α si :

$$-2log(\lambda) > \chi^2_{1,1-\alpha}$$

con:

$$\lambda = \frac{(\frac{k}{N})^k(1-\frac{k}{N})^{N-k}}{(\frac{k_1}{n_1})^{k_1}(1-\frac{k_1}{n_1})^{n_1-k_1}(\frac{k_2}{n_2})^{k_2}(1-\frac{k_2}{n_2})^{n_2-k_2}}$$

donde $N = n_1 + n_2$ y $k = k_1 + k_2$ con:

n_1: número de palabras en el texto.

n_2: número de palabras en el resto del corpus.

k_1: apariciones de la palabra w en el texto.

k_2: apariciones de la palabra w en el resto del corpus.

Demostración. La aparición de la palabra w en un texto t se puede modelar como un modelo binomial de $n = |w_i|$ intentos con probabilidad p.

Por tanto la función de verosimilitud para una probabilidad p viene dada por:

$$L(p; n, k) = \binom{N}{k} p^k (1-p)^{N-k}$$

Y el test del ratio de verosimilitud vendrá dado por:

$$\lambda = \frac{max_{H0}\binom{n_1}{k_1}p_1^{k_1}(1-p_1)^{n_1-k_1}\binom{n_2}{k_2}p_2^{k_2}(1-p_2)^{n_2-k_2}}{max_{p1,p2}\binom{n_1}{k_1}p_1^{k_1}(1-p_1)^{n_1-k_1}\binom{n_2}{k_2}p_2^{k_2}(1-p_2)^{n_2-k_2}}$$

Como H0 es equivalente a $p_1 = p_2$ tenemos que:

$$\lambda = \frac{max_p\binom{n_1}{k_1}\binom{n_2}{k_2}p^k(1-p)^{N-k}}{max_{p1,p2}\binom{n_1}{k_1}p_1^{k_1}(1-p_1)^{n_1-k_1}\binom{n_2}{k_2}p_2^{k_2}(1-p_2)^{n_2-k_2}}$$

En la prática este máximo se alcanzará para $p = \frac{k}{N}$,$p_1 = \frac{k_1}{n_1}$ y $p_2 = \frac{k_2}{n_2}$. Con el estadístico λ calculado habrá que calcular $-2log(\lambda)$ que se comporta como una χ^2 de 1 grado de libertad [@dunning1994accurate].

Por lo tanto para el nivel α fijado tendremos que rechazamos la hipótesis nula si $-2log(\lambda) > \chi^2_{1,1-\alpha}$ ■

Las palabras para las cuales se rechace la hipótesis nula serán las que se podrán tomar como palabras temáticas.

2.2.1.2. Peso TF*IDF (frecuencia de términos frecuencia inversa de documento)

Dentro de los métodos frecuentistas encontramos principalmente el de la probabilidad de las palabras y el TD*IDF, hablaremos de este último ya que no requiere una lista previa de palabras de parada (estas son palabras muy comunes que deben eliminarse para poder realizar el análisis).

Definición 2.2.4 Definición

Sea B un conjunto $t_1, t_2, ..., t_D$ de textos y w una palabra, entonces definimos la TD*IDF para un documento t como:

$$TF * IDF_w = c(w)log(\frac{D}{d(w)})$$

donde c(w) es el número de veces que aparece w en el documento t y d(w) es el número de documentos de B donde aparece la palabra w.

Normalmente emplearemos un conjunto de documentos que sean de una temática parecida para que las palabras aparezcan en repetidas ocasiones y el análisis sea más fiable. Además esta medida penaliza las palabras muy comunes que no aportan significado específico.

Esto hará que las palabras que sean muy comunes en D tendrán TF*IDF cercano a 0 porque si $d(w) \simeq D$ entonces $log(\frac{D}{d(w)}) \simeq 0$ y por tanto $TF * IDF \simeq 0$. Por tanto esto haría que las palabras muy comunes sean ignoradas al dar puntuación a la frase a la que pertenecen mientras que las palabras muy poco comunes tendrán un valor muy alto si aparecen al menos una vez en el texto.

2.2.2. Selección de frases

Una vez se tienen puntuadas las frases se pasa a la elección donde se suele elegir frase a frase y se van añadiendo según una longitud máxima establecida[5]. También una vez elegida una frase es importante penalizar otras frases que contengan la misma información para que el resumen pueda incluir la mayor cantidad de información distinta pertinente. Veremos dos métodos de selección.

2.2.2.1. Máxima Relevancia Marginal (MMR)

Empecemos definiendo la MMR [15].

Definición 2.2.5 Sea j una frase. La puntuación según el MMR en el momento t de la frase j será:

$$m_j^t = \lambda S(s_j, D) - (1 - \lambda)max_{e \in E_t} R(s_j, e)$$

donde $\lambda \in [0, 1]$ es el peso entre lo significativa que es la frase y la similaridad que tiene con las frases ya elegidas. $S(s_j, D)$ es lo significativa que es la frase j para el

documento D. E_t es el conjunto de frases ya escogidas y $R(s_j, e)$ es la similaridad entre la frase s_j y la frase e.

Definición 2.2.6 Definición

Sean dos vectores A, B se define la similaridad cosenoidal como:

$$cos\theta = \frac{A \cdot B}{|A||B|}$$

Para aplicar esto a 2 frases S_A,S_B tomaremos el conjunto D formado por cada una de las distintas palabras presentes entre ambos vectores y crearemos los vectores A y B tales que sus elementos son el número de veces que aparecen cada una de las palabras de D.

Aplicando la similaridad cosenoidal de esta manera como función $R(s_j, e)$ obtenemos una forma rápida y efectiva de medir la similaridad de frases. La aplicación de esta medida en la práctica se verá con el algoritmo utilizado en R posteriormente en este trabajo.

2.2.2.2. Resumen por selección global

Aunque el método anterior es efectivo a veces puede tener problemas al elegir como primera frase una larga e informativa pero que no permitirá incluir otras frases en el resumen. Por este tipo de casos existe esta opción en la que se plantea como un problema de optimización con algunas restricciones como la minimización de la repetición o maximizar la información.

El problema con esta aproximación es que, en general, este es un problema de dificultad NP por lo que hará falta hallar soluciones aproximadas pero aún así obtienen resultados incluso mejores que los otros métodos [8].

2.3. Clasificación de textos

La clasificación de textos comparte procedimientos con los problemas de clasificación de la minería de datos convencional pero encuentra una gran diferencia y es el gran número de variables que contiene un texto. Esto es así ya que en una de las aproximaciones más comunes se representan los textos como un conjunto de palabras con las frecuencias asociadas a cada documento.

Como las variables a considerar es el número de apariciones de cada una de las palabras presenten en el conjunto de los textos a clasificar es fácil llegar a tener cientos de miles de palabras a priori antes de los preprocesamientos, por ello encontraremos dos fases. Una primera donde se seleccionan las palabras que utilizaremos como variables y una segunda fase donde se emplearán métodos de clasificación estándar para crear modelos y poder aplicarlo a textos que no vengan marcados.

2.3.1. Selección de características

Las técnicas más comunes dentro de la selección de características son el “Stemming” y la eliminación de palabras de parada. El stemming consiste en agrupar distintas palabras

que tienen la misma raíz en una sola, es decir distintos tiempos verbales, géneros, singular, plural, etc; se agruparían dentro del mismo término.

La eliminación de palabras muy comunes también es un proceso usado habitualmente ya que no suelen arrojar ningún tipo de información acerca del documento.

Estas dos técnicas a pesar de no ser exclusivas de la clasificación son de suma importancia a la hora de reducir la dimensión e independientemente de la técnica que apliquemos estas también deberían ser aplicadas.

2.3.1.1. Índice de Gini

Definición 2.3.1 Sea B un conjunto de textos divididos en n clases y sea $p_i(w)$ la probabilidad de un texto t de pertenecer a la clase i condicionada a que contiene la palabra w.

Se define el índice de Gini para la palabra w como:

$$G(w) = \sum_{i=1}^{n} p_i(w)^2$$

Proposición 2.3.1

$$G_i(w) \in (1/n, 1)$$

Demostración. Veamos que dados $a_1, ..., a_n \in [0, +\infty)$ $a_1^2 + ... + a_n^2 \leq (a_1 + ... + a_n)^2$. Esto se tiene de forma inmediata desarrollando la parte derecha de la expresión ya que:

$$(a_1 + ... + a_n)^2 = a_1^2 + ... + a_n^2 + 2\sum_{i \neq j} a_i a_j$$

Por tanto la igualdad se da solo si $\sum_{i \neq j} a_i a_j = 0$ lo cual se tiene solo si al menos n-1 de los números a_i son iguales a 0. Ahora si buscamos el máximo de la suma de los a_i^2 restringiendo su suma obtenemos que:

$$\max_{\sum_i a_i = c} a_1^2 + ... + a_n^2 = c^2$$

Porque $a_1^2 + ... + a_n^2 \leq (a_1 + ... + a_n)^2 = c^2$ Por tanto como:

$$\sum_i p_i(w) = 1$$

tenemos que $max(G(w)) = 1$. Por otro lado veamos que suponiendo que $a_1 + ... + a_n = c$ la suma:

$$\sum_{i=1}^{n} a_i^2$$

alcanza su mínimo cuando $a_1 = a_2 = ... = a_n = c/n$. Sea d=c/n entonces:

$$\sum_{i=1}^{n} a_i^2 = \sum_{i=1}^{n} (d + a_i - d)^2 = nd^2 + \sum_{i=1}^{n} (a_i - d)^2 + 2\sum_{i=1}^{n} d(a_i - d)$$

veamos que $2\sum_{i=1}^{n} d(a_i - d) = 0$.

$$2\sum_{i=1}^{n} d(a_i - d) = 2d(\sum_{i=1}^{n}(a_i) - nd) = 2d(nd - nd) = 0$$

Por tanto como la expresión $nd^2 + \sum_{i=1}^{n}(a_i - d)^2$ se minimiza tomando $a_i = d$ hemos probado que el mínimo se alcanza cuando todos los a_i valen lo mismo. Aplicando el caso a los coeficientes de Gini:

$$\text{mín}(G(w)) = \sum_{i=1}^{n} \frac{1}{n^2} = \frac{n}{n^2} = \frac{1}{n}$$

■

Esto también nos permite ver que cuando los valores de $p_i(w)$ sean muy similares tendremos un índice de Gini más bajo mientras que si una palabra aparece principalmente en una clase de documentos el índice de Gini será cercano a 1[18].

Un problema que nos podemos encontrar con este método es que la distribución global puede estar sesgada, es decir, que el número de documentos de cada clase no sea uniforme. Para solucionar esto se puede construir una versión normalizada del índice para solventar este problema.

Definición 2.3.2 Sean $P_1, ..., P_k$ la distribución global de documentos por clase, definimos las probabilidades normalizadas $p'_i(w)$ de la siguiente forma:

$$p'_i(w) = \frac{p_i(w)/P_i}{\sum_{j=1}^{k} p_j(w)/P_j}$$

El índice de Gini normalizado será entonces:

$$G'(w) = \sum_{i=1}^{k} p'_i(w)^2$$

Una vez tengamos el índice de Gini de todas las palabras podemos quedarnos con aquellas cuyo índice de Gini sea mayor, ya que son las que tienen más poder discriminatorio.

2.3.1.2. Método estadístico basado en el Chi_Cuadrado χ^2

Otras aproximación utilizada implica medir la falta de independencia entre un texto t y una palabra w. Una forma utilizada es a través de tablas de contingencia y el estadístico chi-cuadrado[17].Para ello usaremos la fórmula de Pearson para la χ^2:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

donde χ^2 es el estadístico que se aproxima a una distribución χ^2

O_i es el número de observaciones de i

E_i es el valor teórico esperado de i

n es el número de celdas de la tabla de contingencia

En este caso[19], para cada una de las combinaciones de clase y palabra creamos la tabla de contingencia:

	Texto t	Resto de textos
Contiene w	A	B
No contiene w	C	D

Calculamos los valores esperados teóricos de la siguiente manera:

$$E_{i,j} = \frac{totalColumna_i * totalFila_j}{totalTabla}$$

Al ser una tabla de contigencia 2x2 obtenemos que los grados de libertad de la χ^2 son gl=(ncol-1)(nrow-1)=1*1=1. En la práctica tendremos que calcular los valores A,B,C,D a partir del conjunto de textos de entrenamiento del que dispongamos para cada par de texto y palabra.

Para seleccionar las palabras que utilizaremos en la siguiente fase podemos tomar para cada palabra la media de sus estadísticos χ^2 y quedarnos con aquellas cuya estadístico medio esté por encima del valor deseado de significación de la distribución.

2.3.1.3. Medida de información mutua

Esta aproximación emplea métodos de la teoría de la información para proveer un modelo de información mutua entre las palabras y las clases. Se utilizará la co-ocurrencia de la clase i con la palabra w para hallar una medida de información.

Definición 2.3.3 Sea F(w) la fracción de documentos que incluyen la palabra w, sea P_i la probabilidad global de la clase i, $p_i(w)$ la probabilidad condicionada de la clase i dado que el documento incluye la palabra w entonces definimos la información mutua como:

$$M_i(w) = log(\frac{F(w) \cdot p_i(w)}{F(w) \cdot P_i}) = log(\frac{p_i(w)}{P_i})$$

Si $M_i(w) < 0$ tendremos que existe una correlación negativa entre la clase y la palabra mientras que en el caso contrario tendremos una correlación positiva. Esta definición proviene de que si la clase i y la palabra w fueran independientes se esperaría que tuvieran una co-ocurrencia de $P_i \cdot F(w)$; mientras que la co-ocurrencia real es $F(w) \cdot p_i(w)$, como en la práctica estos valores pueden diferir hemos definido la información mutua en base al ratio entre estos valores.

Para la selección de clases es necesario medir estas informaciones para cada palabra y no solo por clase. Por tanto, la puntuación para cada una de las palabras vendrá dada o bien por la media o por bien el máximo de sus medidas de información mutua por clase. El máximo será especialmente útil cuando queramos valores altos de correlación.

2.3.2. Métodos de clasificación

Como hemos comentado anteriormente los métodos convencionales de clasificación funcionan para los textos una vez se le aplican algunos de los preprocesamientos que hemos

descrito en la anterior sección, no obstante mencionaremos brevemente algunos que han sido utilizados en este ámbito.

2.3.2.1. Vecinos más cercanos (KNN)

Este método consiste en asignar a un nuevo documento sin clasificar la clase más común entre sus k textos más cercanos. Una distancia útil es la similaridad cosenoidal aunque también se pueden utilizar la distancia euclídea u otra medida sobre la matriz de términos documentos obtenida tras el preprocesamiento.

La clasificación por vecinos más cercanos se ha usado junto al preprocesamiento por medida de información mutua [11] con éxito. Aquí es importante aplicar alguna técnica de preprocesamiento si no el gran número de palabras comunes en la mayoría de textos crearía mucho ruido reduciendo la efectividad.

2.3.2.2. Máquinas de vector de soporte

Esta técnica consistente en hallar hiperplanos que separen los elementos de las distintas clases, puede también aplicarse para la clasificación de textos. Una ventaja que ofrece este clasificador es su efectividad en alta dimensionalidad, de hecho en [12] se obtienen evidencias teóricos y prácticas de que este clasificador funciona especialmente bien con los textos obteniendo mejores resultados que otros métodos de clasificación.

Los motivos son la robustez en alta dimensionalidad, un nivel bajo de variables poco importantes y vectores con un gran número de ceros.

2.3.2.3. Árboles de decisión

Los árboles de decisión son un algoritmo popular en la clasificación de textos. Estos árboles son estructuras de flujo de control en forma de árbol, donde cada nodo interno representa una pregunta o prueba sobre una característica del texto, y cada rama representa una posible respuesta o resultado de esa pregunta. Suelen ser la presencia o no de una palabra en el texto.

Una vez construido el árbol, se puede utilizar para clasificar nuevos documentos. Para ello, se sigue el camino desde el nodo raíz hasta una hoja, siguiendo las pruebas realizadas en cada nodo y utilizando las respuestas para tomar decisiones sobre la clasificación final del documento.

Los árboles de decisión en la minería de texto ofrecen varias ventajas. Son fáciles de interpretar y visualizar, lo que permite comprender el proceso de clasificación y tomar decisiones basadas en las reglas extraídas del árbol.

Sin embargo, los árboles de decisión también presentan algunas limitaciones. Pueden ser propensos al sobreajuste si no se controla su crecimiento adecuadamente, lo que puede llevar a un rendimiento deficiente en la clasificación de nuevos datos. Además, en el contexto de la minería de texto, pueden ser sensibles al ruido y a la presencia de palabras irrelevantes o poco informativas.

2.4. Extracción de información

La extracción de información es una de las tareas esenciales de la minería de texto y consiste en la detección automática de entidades y relaciones presentes en un texto. Esta tarea es empleada en varios contextos como la biomedicina, la búsqueda web o el mundo de las finanzas [1] y, aunque existen diferencias en las aplicaciones en cada ámbito nos centraremos en algunas de las más generales.

2.4.1. Reconocimiento de entidades nombradas

El reconocimiento de entidades nombradas (NER, por sus siglas en inglés, Named Entity Recognition), se refiere al proceso de identificar y clasificar entidades significativas en un texto, como nombres de personas, organizaciones, lugares, fechas, cantidades, etc.

Esta es la tarea fundamental dentro del campo de la extracción de información ya que el resto de tareas como el de la extracción de relaciones o eventos depende de un preprocesamiento que incluye el reconocimiento de las entidades.

Hay 2 tipos de aproximaciones, una más simple que se basa en una serie de reglas definidas de antemano que sirven para buscar las entidades de forma automática y una aproximación estadística, que utiliza modelos como cadenas ocultas de Markov o campos aleatorios condicionales para realizar la tarea.

2.4.1.1. Aproximación a partir de reglas

Los métodos basados en reglas para el reconocimiento de entidades nombradas funcionan siguiendo ciertos pasos. En primer lugar, se definen conjuntos de reglas de forma manual o automática. Cada palabra o "token" en el texto se representa mediante características específicas. Luego, el texto se compara con las reglas y se activa una regla si hay una coincidencia. Cada regla consta de un patrón y una acción asociada. El patrón suele ser una expresión regular que se define según las características de los tokens. Cuando el patrón coincide con una secuencia de tokens, se ejecuta la acción correspondiente. La acción puede implicar etiquetar una secuencia de tokens como una entidad, agregar etiquetas de inicio y fin de entidad, o identificar múltiples entidades al mismo tiempo. Por ejemplo, para etiquetar como una entidad de persona cualquier secuencia de tokens de la forma "Don N", donde N es una palabra con la primera letra en mayúscula, se puede definir la siguiente regla:

$$(token = \text{“}Don\text{”} ortografía = PrimeraMayúscula) ß NombrePersona.$$

En esta regla, el lado izquierdo es una expresión regular que coincide con una secuencia de dos tokens donde el primer token es "Don" y el segundo token tiene la característica de tener la primera letra en mayúscula. El lado derecho indica que la secuencia de tokens coincidentes debe etiquetarse como un nombre de persona.

Es posible que una secuencia de tokens coincida con múltiples reglas, y para manejar estos conflictos, se deben establecer políticas para controlar el orden de activación de las reglas. Una opción es ordenar las reglas de antemano para que se verifiquen y activen en secuencia.

La creación manual de reglas para el reconocimiento de entidades nombradas requiere experiencia humana y es un proceso laborioso. Sin embargo, también se han propuesto métodos para aprender automáticamente las reglas. Estos métodos se pueden clasificar en dos enfoques principales: el enfoque de arriba hacia abajo y el enfoque de abajo hacia arriba. Ambos enfoques requieren un conjunto de documentos de entrenamiento donde las entidades nombradas están etiquetadas manualmente. En el enfoque de arriba hacia abajo, se definen reglas generales que pueden cubrir la extracción de muchas instancias de entrenamiento, pero su precisión tiende a ser baja. Luego, el sistema define reglas más específicas de manera iterativa al tomar las intersecciones de las reglas más generales. En el enfoque de abajo hacia arriba, se definen reglas específicas basadas en instancias de entrenamiento que aún no están cubiertas por el conjunto de reglas existente. Luego, estas reglas específicas se generalizan para abarcar más casos.

2.4.2. Aproximación estadística

La mayoría de los métodos estadísticos estarán basados en la etiquetación secuencial la cual trata el texto como una secuencia de palabras a las cuales le intenta asignar una etiqueta que designa el tipo de palabra que es, formalmente:

Definición 2.4.1 Consideremos un texto t como una sucesión de palabras u observaciones $x_1, ..., x_n$ y un conjunto C al que llamaremos sistema de notación que contiene los posibles tipos de entidades. Entonces definimos para cada x_i su etiqueta como su $y_i \in C$ correspondiente.

Por ejemplo un sistema de notación empleado frecuentemente [20], consiste en tres tipos de etiquetas,

B-T: donde B denota que es el inicio de una entidad y T el tipo que es.

I-T: donde I denota que es el interior de una entidad y T el tipo que es.

O: que denota que no se trata de una entidad nombrada.

Normalmente las etiquetas y_i tendrán una fuerte dependencia de su observación x_i pero también dependerán de las etiquetas y observaciones cercanas. Veamos algunos de estos modelos.

2.4.2.1. Cadenas ocultas de Markov

Empezemos definiendo una cadena oculta de Markov.

Definición 2.4.2 Sea un conjunto de estados Q=1,2,...,N, sea V un cojunto de valores posibles $v_1, ..., v_m$. Sea π_i la probabilidad de que el estado inicial sea Q_i. Sea el conjunto de probabilidades de transición R= r_{ij} :

$$r_{ij} = P(q_t = j | q_{t-1} = i)$$

Y sea S=$s_j(v_k)$ el conjunto de las probabilidades de observación de v_k condicionada al estado j:

$$b_j(v_k) = P(o_t = v_k | q_t = j)$$

Entonces diremos que el Modelo Oculto de Markov es la tupla (Q,V,π,R,S)

En nuestro caso tendremos que el conjunto V hace referencia al conjunto de posibles palabras a observar mientras que Q son los posibles estados que definamos. Por ejemplo uno de los modelos empleados "Nymble"[3], utiliza varios tipos de entidades nombradas y NaN (Not a Name) como posibles estados más un estado "principio de frase" con probabilidad 1 de ser el inicial y un estado "final de frase".

El sistema Nymble modela las probabilidades de la siguiente forma:

- Se modelan las etiquetas según la etiqueta anterior y la palabra anterior.
- Si una palabra es la primera de su entidad nombrada se modela condicionada a la etiqueta actual y la anterior.
- Si una palabra no es la primera de su entidad nombrada se modela según la palabra anterior.

Además el sistema asume que existe una palabra "final" de cada entidad nombrada y modela su probabilidad. Dado un conjunto de frases etiquetadas con entidades nombradas, Nymble realiza una estimación de máxima verosimilitud para encontrar los parámetros del modelo que maximizan $p(X, Y)$, donde X representa todas las frases en los datos de entrenamiento e Y representa sus secuencias de etiquetas verdaderas. Para estimar las probabilidades utilizan las siguientes fórmulas:

$$P(NC|NC_{-1}) = \frac{c(NC, NC{-}1, w_{-1})}{c(NC{-}1, w_{-1})}$$

Para la probabilidad de una etiqueta.

$$P(w_{primera}|NC, NC_{-1}) = \frac{c(NC, NC{-}1, w_{primera})}{c(NC{-}1, NC)}$$

Para la probabilidad de la primera palabra de una entidad nombrada.

$$P(w|w_{-1}, NC) = \frac{c(NC, w, w_{-1})}{c(NC, w_{-1})}$$

Para la probabilidad de una palabra intermedia de una entidad nombrada.

Donde NC denota una clase de entidad y c() es el conteo de casos de esa combinación en el corpus de entrenamiento.

Finalmente durante la predicción Nymble utiliza los parámetros del modelo para encontrar el vector y que maximiza la probabiliad de $p(x, y)$ para el vector de palabras x dado.

2.4.2.2. Campos Aleatorios Condicionales

Este método considera que las etiquetas y_i no solo dependen de las etiquetas anteriores sino también de las futuras.

Definición 2.4.3 Sea $\mathbf{x} = (x_1, ..., x_n)$ el vector secuencia de las palabras y sea $\mathbf{y} = (y_1, ..., y_n)$ el vector de las etiquetas. Según el modelo de campos aleatorios condicionales la $p(\mathbf{x}|\mathbf{y})$ será:

$$p(\mathbf{x}|\mathbf{y}) = \frac{1}{Z(\mathbf{x})} exp(\sum_i \sum_j \lambda_j f_j(y_i, y_{i-1}, x, i)),$$

donde $Z(\mathbf{x})$ es un factor de normalización de todas las posibles secuencias de etiquetas:

$$Z(\mathbf{x}) = \sum_{y'} exp(\sum_i \sum_j \lambda_j f_j(y'_i, y'_{i-1}, x, i))$$

Para entrenar este modelo de nuevo utilizamos máxima verosimilitud para obtener los parámetros que maximizen p($\mathbf{Y}|\mathbf{X}$). Este modelo ha sido utilizado para obtener tasas de acierto de 84 % en textos en inglés[16].

En el siguiente capítulo utilizaremos el software "R" para implementar alguno de los procesos y tareas que hemos visto en este capítulo.

Capítulo 3

Aplicación de la minería de texto en R

En este apartado vamos a poner en uso algunos de los métodos mencionados en el capítulo anterior viendo como estos métodos pueden ser implementados en R. Algunos métodos los implementaremos sin utilizar ninguna librería específica, empleando herramientas comunes para ver como se pude construir desde la base; pero otros los desarrollaremos utilizando librerías específicas que tienen funciones capaces de facilitar mucho la implementación.

La versión de R utilizada es R version 4.2.1 (2022-06-23 ucrt), mientras que el equipo donde se ha ejecutado el código cuenta con un procesador AMD Ryzen 3 2200G with Radeon Vega y con 8 GB de memoria RAM.

3.1. Resumen de texto

Nuestro objetivo en esta sección es obtener resúmenes automáticos de noticias, para ello utilizaremos el método del estadístico TF*IDF para asignar los pesos a las palabras y la máxima relevancia marginal para elegir las frases. Veamos los pasos.

En primer lugar el conjunto de datos empleados para resumir son 50 artículos sobre adquisiciones de empresas, el conjunto de datos lo hemos obtenido de la librería "tm"[**?**].

Estos documentos son relativamente cortos pero cuentan con suficientes frases como para que tenga sentido el resumen de los mismos.

La función "TermDocumentMatrix" de la librería tm[**?**] nos permite obtener la matriz de documentos-artículos a partir de la lista de artículos y asignarle a cada palabra su TF*IDF. También nos permite realizar algunas funciones de limpieza de datos como la eliminación de palabra de parada, signos de puntuación y realizar "stemming".

	10	110	12	44	45	68	96
abil	0.0000000	0.0168473	0.000000	0	0.0000000	0.0000000	0
accept	0.0000000	0.0000000	0.000000	0	0.0000000	0.0000000	0
access	0.0000000	0.0000000	0.000000	0	0.0000000	0.0000000	0
accompani	0.0000000	0.0168473	0.000000	0	0.0000000	0.0000000	0
accord	0.0000000	0.0138623	0.000000	0	0.0000000	0.0000000	0
acquir	0.0111674	0.0000000	0.000000	0	0.0065846	0.0423009	0
acquisit	0.0000000	0.0065207	0.087377	0	0.0000000	0.0000000	0
acr	0.0000000	0.0000000	0.000000	0	0.0000000	0.0000000	0
act	0.0000000	0.0000000	0.000000	0	0.0000000	0.0000000	0
action	0.0000000	0.0000000	0.000000	0	0.0000000	0.0000000	0

Aquí podemos ver un extracto de la matriz término-documento donde cada fila corresponde a una palabra (en este caso se puede ver que el stemming ha dejado las raíces de algunas de ellas) y las columnas corresponden a los documentos nombrados a partir de su ID.

Una vez que tenemos esta matriz podemos definir varias funciones que nos permiten iterar el proceso y aplicarlo rápidamente a los 50 documentos mediante el siguiente bucle:

El texto original sería este:

```
## Computer Terminal Systems Inc said
## it has completed the sale of 200,000 shares of its common
## stock, and warrants to acquire an additional one mln shares, to
## <Sedio N.V.> of Lugano, Switzerland for 50,000 dlrs.
##     The company said the warrants are exercisable for five
## years at a purchase price of .125 dlrs per share.
##     Computer Terminal said Sedio also has the right to buy
## additional shares and increase its total holdings up to 40 pct
## of the Computer Terminal's outstanding common stock under
## certain circumstances involving change of control at the
## company.
##     The company said if the conditions occur the warrants would
## be exercisable at a price equal to 75 pct of its common stock's
## market price at the time, not to exceed 1.50 dlrs per share.
##     Computer Terminal also said it sold the technolgy rights to
## its Dot Matrix impact technology, including any future
## improvements, to <Woodco Inc> of Houston, Tex. for 200,000
## dlrs. But, it said it would continue to be the exclusive
## worldwide licensee of the technology for Woodco.
##     The company said the moves were part of its reorganization
## plan and would help pay current operation costs and ensure
## product delivery.
##     Computer Terminal makes computer generated labels, forms,
## tags and ticket printers and terminals.
##  Reuter
```

Mientras que el resumen sería este:

The company said if the conditions occur the warrants would be exercisable at a price

equal to 75 pct of its common stock's market price at the time, not to exceed 1.50 dlrs per share. Computer Terminal also said it sold the technolgy rights to its Dot Matrix impact technology, including any future improvements, to of Houston, Tex. for 200,000 dlrs. Reuter Computer Terminal makes computer generated labels, forms, tags and ticket printers and terminals. Computer Terminal Systems Inc said it has completed the sale of 200,000 shares of its common stock, and warrants to acquire an additional one mln shares, to <Sedio N.V.> of Lugano, Switzerland for 50,000 dlrs.

Como podemos ver en este ejemplo se mantiene la información sobre la compañía de venta, el precio y la venta de los derechos de su tecnología.

También este código nos permitiría realizar resúmenes más largos ajustando el parámetro de longitud mínima del resumen además de que podríamos darle un peso distinto a la redundancia de las frases ajustando el parámetro λ.

Para realizar una comparación más detallada aplicaremos esto mismo a unas pocas noticias en español.

El texto es el siguiente:

La multiplicación y el agravamiento de las crisis actuales están ejerciendo una presión todavía mayor sobre la salud mental de las personas y los servicios disponibles para apoyarlas. La Organización Mundial de la Salud (OMS) está trabajando codo con codo con los países para lograr este objetivo. En Chile, el Gobierno ha asumido el reto de que nadie vuelva a tener que afrontar solo sus necesidades de salud mental. Pensando en este objetivo, mediante la estrategia Construyendo Salud Mental se está fortaleciendo el liderazgo en este ámbito en todos los sectores, mejorando la prestación de servicios y el apoyo en emergencias y reforzando datos, pruebas e investigaciones en la materia.

El resumen generado es el siguiente:

La multiplicación y el agravamiento de las crisis actuales están ejerciendo una presión todavía mayor sobre la salud mental de las personas y los servicios disponibles para apoyarlas. Cada vez más personas padecen los efectos persistentes de la pandemia de covid, el repunte de las emergencias relacionadas con el clima y las consecuencias constantes de conflictos y desplazamientos de muchas regiones del mundo. Mientras tanto, sigue habiendo estigmatización y discriminación contra las personas con problemas de salud mental y discapacidades psicosociales en nuestras escuelas, lugares de trabajo y comunidades.

Es de esperar que la falta de la atención y el apoyo de calidad necesarios para hacer frente a las necesidades de salud mental aumente como consecuencia de la cantidad de personas que viven con una afección de salud mental —hasta mil millones de personas (uno de cada ocho de nosotros)— y de la persistente y tradicional falta de inversión en servicios de salud en ese campo: las consecuencias para la salud, la felicidad y el bienestar de millones de personas son fáciles de predecir.

La pandemia de covid expuso las vulnerabilidades de los sistemas de salud mental a nivel mundial y agravó problemas presentes en ese momento y reveló otros nuevos. En Chile, como en muchos países, las repercusiones de la pandemia en la salud mental han sido significativas. El aislamiento, la incertidumbre y la perturbación de la vida cotidiana afectaron a personas y comunidades, lo que puso de manifiesto la importancia de contar con sistemas de apoyo resilientes.

Teniendo en cuenta todo lo que ha sucedido en este siglo, tenemos que transformar la concepción y el modo de actuación sobre la salud mental, con miras a mejorarla. Necesitamos cambiar nuestras actitudes para dar prioridad a la salud mental como parte integral de nuestra salud y bienestar, así como en cuanto derecho humano básico y factor contribuyente fundamental para la salud pública, el bienestar social y el desarrollo sostenible.

Debemos fortalecer la prestación de servicios de atención de salud mental para que todo el espectro de necesidades al respecto quede cubierto por una red comunitaria de apoyo y servicios accesibles, asequibles y de calidad.

La Organización Mundial de la Salud (OMS) está trabajando codo con codo con los países para lograr este objetivo. La Iniciativa Especial de la OMS sobre Salud Mental es un buen ejemplo de cómo el aumento de la capacidad en el nivel de la atención primaria de salud puede mejorar el acceso a servicios para las personas que más los necesitan. Gracias a la Iniciativa Especial, en los nueve países participantes se ha ampliado el acceso a servicios locales de salud mental, inexistentes hasta el momento, a 40 millones de personas desde 2019.

Aunque el sector de la salud tiene mucho que aportar, no puede trabajar solo. Como expusieron los Ministros en la Cumbre Mundial sobre Salud Mental organizada la semana pasada por el Gobierno de la Argentina, la transformación de la atención de la salud mental requiere un planteamiento pangubernamental y pansocial de promoción, protección y atención de la misma.

También debemos reorganizar los entornos que influyen en la salud mental, de manera que se reduzcan los riesgos y se fortalezcan los factores de protección para que todas las personas, independientemente de quiénes sean, tengan las mismas oportunidades de prosperar y alcanzar el nivel más alto posible de bienestar.

En Chile, el Gobierno ha asumido el reto de que nadie vuelva a tener que afrontar solo sus necesidades de salud mental. Pensando en este objetivo, mediante la estrategia Construyendo Salud Mental se está fortaleciendo el liderazgo en este ámbito en todos los sectores, mejorando la prestación de servicios y el apoyo en emergencias y reforzando datos, pruebas e investigaciones en la materia. Un factor fundamental es la integración de los servicios de salud mental en la atención primaria y en los centros comunitarios, lo que posibilita su abordaje integral, junto los servicios sociales a nivel municipal.

También tiene como objetivo la prevención del suicidio, que es un importante problema de salud pública dada su alta carga entre los jóvenes de todo el mundo: es la segunda causa de muerte entre las personas de 15 a 24 años.

Las estrategias ambiciosas encaminadas a proteger y mejorar la salud mental requieren nuevos niveles de liderazgo y compromiso político, así como asignaciones mucho más sólidas de recursos en los sectores de la salud y en otros sectores. Como se ha visto en Chile, por ejemplo, la atención y el compromiso a largo plazo con la salud mental de toda la población pueden generar beneficios reales y considerables a lo largo del tiempo.

Hay, sin embargo, demasiados países en los que todavía es necesario tomar más medidas para que las personas reciban la atención adecuada y de calidad que necesitan.

Tras celebrar este martes el Día Mundial de la Salud Mental, centrado su defensa como derecho humano, es importante recordar el elevado número de personas que siguen

sufriendo coerción, malos tratos y descuido en los servicios de salud mental y a los que se les niega el derecho a opinar sobre su tratamiento. Tenemos la responsabilidad colectiva de que, en el marco del tratamiento y los servicios de esa salud, se respeten los derechos humanos de las personas y se apoye su recuperación.

Para ayudar a los países en este sentido, la OMS y la Oficina del Alto Comisionado de las Naciones Unidas para los Derechos Humanos acaban de publicar nuevas orientaciones transformadoras sobre salud mental, derechos humanos y legislación, que ayudarán a los países a que sus leyes y políticas sobre este ámbito de la salud estén en consonancia con las normas internacionales de derechos humanos.

Hacemos un llamamiento a las personas y comunidades para que reconozcan la salud mental como un derecho humano universal, mejoren sus conocimientos al respecto, sean conscientes del problema y actúen en la promoción y protección de la salud mental de todas las personas, y a los gobiernos para que tomen las medidas necesarias con las que lograr que todas las personas puedan alcanzar el nivel más alto posible de salud mental.

Este resumen vemos que contiene información relevante acerca de los temas pero también incluye información muy específica que no es tan relevante, como la frase acerca de la situación particular de Chile, así como un poco de repetición de la información y que no presenta alguna de las frases que contienen información importante como el penúltimo párrafo del texto.

3.2. Clasificación

El problema de clasificación que trataremos es la clasificación automática en una serie de temas preestablecidos de leyes a partir del nombre de la ley. Para ello trabajaremos con un conjunto de textos proveniente del paquete "RTextTools"[**?**] que contiene el nombre de 4449 leyes aprobadas por el congreso de EEUU así como la clase a la que corresponden. Nuestro objetivo será utilizar 4000 de estos documentos como conjunto de entrenamiento y los otros 449 para comprobar la precisión de los modelos generados.

	ID	cong	billnum	h_or_sen	major	text
11	11	107	4509	HR	18	To suspend temporarily the duty on
12	12	107	4510	HR	16	To amend chapter 171 of title 28,
13	13	107	4511	HR	18	To suspend temporarily the duty on
14	14	107	4512	HR	12	To amend the Internal Revenue Code
15	15	107	4513	HR	2	To strengthen the authority of the
16	16	107	4514	HR	3	To authorize the Secretary of Vete
17	17	107	4515	HR	3	To amend title XVIII of the Social
18	18	107	4516	HR	18	To suspend temporarily the duty on
19	19	107	4517	HR	18	To suspend temporarily the duty on
20	20	107	4518	HR	18	To suspend temporarily the duty on
21	21	107	4519	HR	18	To suspend temporarily the duty on

Las columnas más importantes de esta tabla son "major" que hace referencia a la clase del documento y "text" que hace referencia al texto del mismo.

En esta sección analizaremos los resultados obtenidos aplicando el preprocesado por el método del estadístico χ^2. Posteriormente utilizaremos el método de los vecinos más

cercanos y máquinas de vector de soporte (SVM) para realizar la clasificación.

En un siguiente paso compararemos los resultados con los obtenidos con las funciones definidas en el paquete "RTextTools"[**?**] que nos proporciona una aplicación más inmediata y ágil para la clasificación de textos.

Aquí podemos ver un extracto de las 10 filas de ese conjunto de datos:

Primero creamos la matriz término-documento como en la sección anterior, la utilizaremos para realizar los cálculos de los estadísticos χ^2 para cada pareja de documento y palabra.

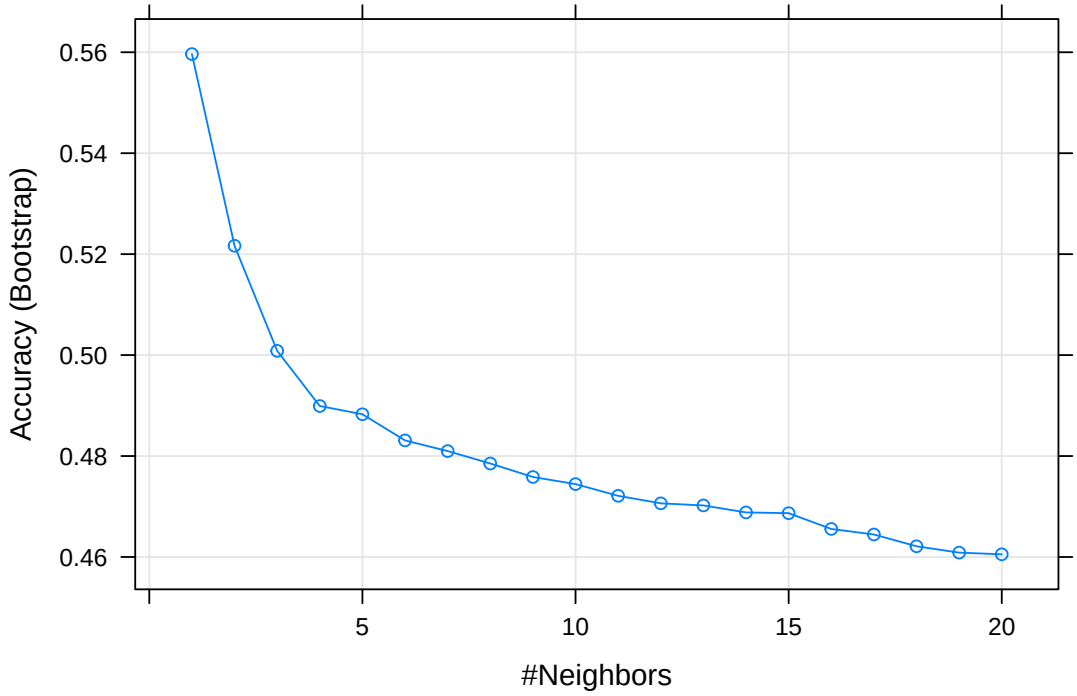
	1	2	3	4	5	6	7
abandon	0.04005	0.02015	0.15980	0.02802	0.06074	0.05126	0.04631
abaya	0.04005	0.02015	0.15980	0.02802	0.06074	0.05126	0.04631
abduct	0.12021	0.06049	0.47963	0.08410	0.18232	0.15386	0.13900
abductor	0.04005	0.02015	0.15980	0.02802	0.06074	0.05126	0.04631
abel	0.04005	0.02015	0.15980	0.02802	0.06074	0.05126	0.04631
abet	0.04005	0.02015	0.15980	0.02802	0.06074	0.05126	0.04631
abil	0.72401	1.19737	0.10803	0.50652	1.09802	0.92664	0.05465
abl	0.08012	0.04032	2.21075	0.05605	0.12151	0.10255	0.09264
abm	0.08012	0.04032	0.31967	0.05605	0.12151	0.10255	0.09264
aboard	0.04005	0.02015	0.15980	0.02802	0.06074	0.05126	0.04631

Aquí podemos ver un extracto de la matriz donde cada celda $c_{i,j}$ representa el estadístico χ^2 de independencia entre la palabra i y la clase j. Utilizando esta matriz obtenemos la lista de palabras que usaremos para la clasificación, nos quedaremos solamente con aquellas cuyo estadístico medio para todas las clases sea mayor que el percentil 99 de una χ^2 de un grado de libertad. Este es un extracto de la lista de palabras:

```
##  [1] "duti"        "temporarili"
##  [3] "suspend"     "educ"
##  [5] "energi"      "health"
##  [7] "privat"      "social"
##  [9] "xviii"       "medicar"
```

Ahora necesitamos crear los conjuntos de entrenamiento y testeo para generar y evaluar los distintos modelos. Para ello nos quedaremos solo con las palabras que se encuentren en la lista.

Primero probaremos a realizar la clasificación mediante vecinos más cercanos, en particular comprobaremos el número de vecinos entre 1 y 20 que da mejores resultados. Una vez que lo tengamos seleccionado lo usaremos para realizar la clasificación.



Aquí podemos ver que el número óptimo de vecinos a considerar es 1. Obtenemos esta conclusión ya que la precisión empeora si consideramos un número mayor de vecinos.

Finalmente realizamos la predicción utilizando el modelo utilizado sobre el conjunto de testeo.

```
##       Accuracy          Kappa
##   6.325167e-01   6.010909e-01
##  AccuracyLower  AccuracyUpper
##   5.860518e-01   6.772230e-01
##   AccuracyNull AccuracyPValue
##   1.469933e-01  9.726935e-122
##  McnemarPValue
##            NaN
```

Ahora veremos los resultados obtenidos por el SVM:

```
##       Accuracy          Kappa
##   6.770601e-01   6.481419e-01
##  AccuracyLower  AccuracyUpper
##   6.316339e-01   7.201345e-01
##   AccuracyNull AccuracyPValue
##   1.469933e-01  1.428762e-142
##  McnemarPValue
##            NaN
```

Comparando las precisiones podemos observar que para este caso el SVM obtiene una mayor precisión con 67,7 % de acierto frente al 62,36 % del KNN. Esta precisión no es un mal resultado para este nivel de refinamiento e información. Para conseguir mejores

resultados usaremos las herramientas que nos proporciona la librería de R "RTextTools"[**?**].

Emplearemos una serie de modelos de clasificación que ofrece la librería así como las funciones de preprocesamiento "create_matrix" y "create_container". Que nos ayudan con un preprocesamiento rápido y dividiendo el conjunto de datos inicial en entrenamiento y evaluación.

```
set.seed(2023)
doc_matrix=create_matrix(USCongress$text, language="english",
                         removeNumbers=TRUE,stemWords=TRUE,
                         removeSparseTerms=.998)
container=create_container(doc_matrix, USCongress$major, trainSize=1:4000,
                           testSize=4001:4449, virgin=FALSE)
```

Emplearemos una serie de algoritmos ya implementados en el paquete para generar distintos modelos y compararemos su precisión. Aquí podemos ver un resumen de los resultados así como el código utilizado. Podemos ver que es bastante conciso dado que el paquete tiene funciones avanzadas que permiten realizar la clasificación de forma sencilla.

```
SVM <- train_model(container,"SVM")
SLDA <- train_model(container,"SLDA")
BOOSTING <- train_model(container,"BOOSTING")
BAGGING <- train_model(container,"BAGGING")
RF <- train_model(container,"RF")
NNET <- train_model(container,"NNET")
TREE <- train_model(container,"TREE")

SVM_CLASSIFY <- classify_model(container, SVM)
SLDA_CLASSIFY <- classify_model(container, SLDA)
BOOSTING_CLASSIFY <- classify_model(container, BOOSTING)
BAGGING_CLASSIFY <- classify_model(container, BAGGING)
RF_CLASSIFY <- classify_model(container, RF)
NNET_CLASSIFY <- classify_model(container, NNET)
TREE_CLASSIFY <- classify_model(container, TREE)

analytics <- create_analytics(container,
                              cbind(SVM_CLASSIFY, SLDA_CLASSIFY,
                                    BOOSTING_CLASSIFY, BAGGING_CLASSIFY,
                                    RF_CLASSIFY, NNET_CLASSIFY, TREE_CLASSIFY))
summary(analytics)
```

```
## ENSEMBLE SUMMARY
##
##        n-ENSEMBLE COVERAGE
## n >= 1                1.00
## n >= 2                1.00
## n >= 3                0.92
## n >= 4                0.76
## n >= 5                0.53
```

```
## n >= 6                0.29
## n >= 7                0.13
##        n-ENSEMBLE RECALL
## n >= 1              0.74
## n >= 2              0.74
## n >= 3              0.78
## n >= 4              0.84
## n >= 5              0.88
## n >= 6              0.94
## n >= 7              0.98
##
##
## ALGORITHM PERFORMANCE
##
##        SVM_PRECISION
##               0.6390
##           SVM_RECALL
##               0.6370
##           SVM_FSCORE
##               0.6280
##       SLDA_PRECISION
##               0.6320
##          SLDA_RECALL
##               0.6360
##          SLDA_FSCORE
##               0.6250
## LOGITBOOST_PRECISION
##               0.5855
##    LOGITBOOST_RECALL
##               0.6055
##    LOGITBOOST_FSCORE
##               0.5755
##    BAGGING_PRECISION
##               0.5800
##       BAGGING_RECALL
##               0.3915
##       BAGGING_FSCORE
##               0.4025
##    FORESTS_PRECISION
##               0.6715
##       FORESTS_RECALL
##               0.6215
##       FORESTS_FSCORE
##               0.6245
##       TREE_PRECISION
##               0.1635
##          TREE_RECALL
##               0.1970
```

```
##          TREE_FSCORE
##               0.1540
##   NNETWORK_PRECISION
##               0.0990
##      NNETWORK_RECALL
##               0.1260
##      NNETWORK_FSCORE
##               0.0860
```

Podemos ver en la tabla que el resultado proporcionado por el método de árboles aleatorios es mejor que el que teníamos llegando hasta una precisión del 67,15 %, como indica la salida "forests_precision". Si lo comparamos con los resultados anteriores vemos que no hay una gran diferencia de nivel de precisión indicando que el preprocesamiento realizado a partir de la Chi Cuadrado ha sido tan efectivo como el que realiza la librería.

3.3. Reducción de dimensionalidad.

3.3.1. Modelado de temas

Vamos a ver un ejemplo del libro "Text mining in practice with R"[14] donde queremos agrupar las distintas noticias del periódico "The Guardian" según el tema que tratan y el sentimiento general presente en estos artículos. El objetivo es crear un "treemap" que ilustre esta información y los artículos utilizados son aquellos que hablan sobre Pakistán. Para ello vamos a emplear el análisis latente de Dirichlet (LDA) con el muestreo de Gibbs. Este modelo nos permite encontrar temas comunes en los artículos y posteriormente realizaremos la asignación de temas.

Aplicamos la función lexicalize que nos crea el corpus y el vocabulario presente en los documentos. De esta manera conseguimos el conjunto de términos presentes dentro del conjunto de documentos.

```
documents <- lexicalize(articles)

wc <- word.counts(documents$documents,
                  documents$vocab)
doc.length<- document.lengths(documents$documents)
summary(documents)
```

```
##           Length Class  Mode
## documents     50  -none- list
## vocab      10515  -none- character
```

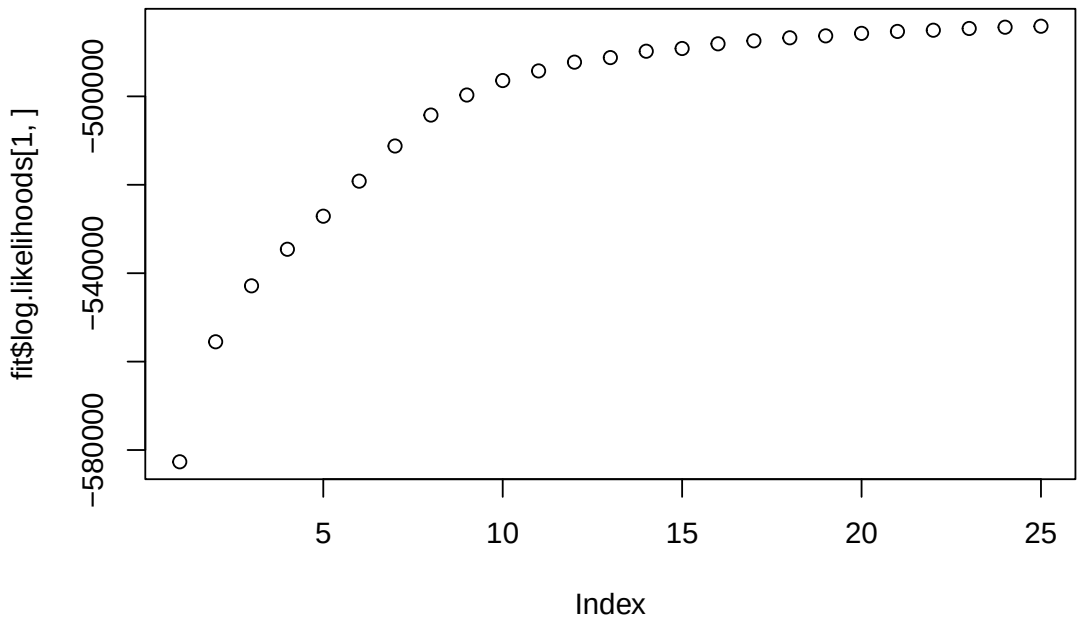
```
documents$vocab[1:20]
```

```
##  [1] "pm"           "win"
##  [3] "series"       "puts"
##  [5] "two"          "play"
##  [7] "truth"        "wasnt"
##  [9] "highest"      "quality"
## [11] "game"         "james"
```

```
## [13] "vinces"       "score"
## [15] "truly"        "noteworthy"
## [17] "performances" "came"
## [19] "afridi"       "bat"
```

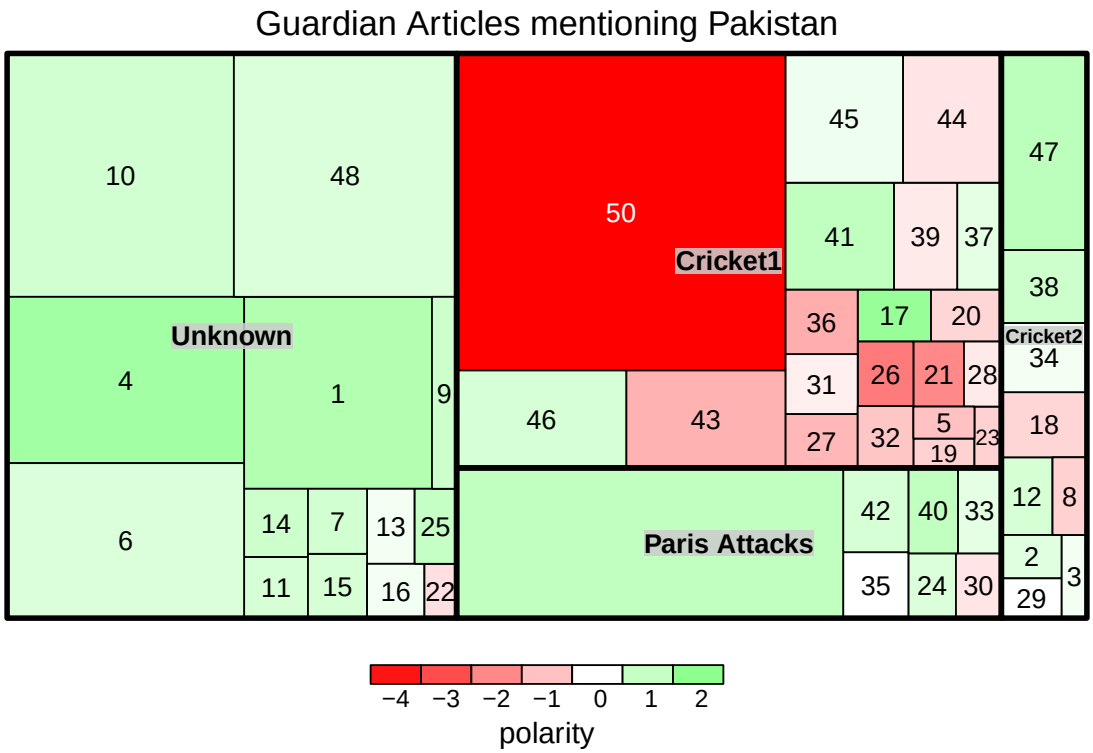
Hemos obtenido una lista de los distintos 10525 términos en 50 documentos. Ahora emplearemos el muestreo de Gibbs para generar el modelo LDA que nos relaciona los temas subyacentes con los términos. Para ello necesitaremos asignar unos parámetros a priori para nuestras distribuciones, así como fijar el número de temas a detectar y el número de iteraciones del algoritmo. También mediremos como mejora la bondad de ajuste según aumentan las iteraciones del muestreo de Gibbs a través de la log-verosimilitud.

```
k <- 4 # numero de temas a identificar
num.iter <- 25 # numero de repeticiones de muestreos
alpha <- 0.02 #parámetro de la distribución Dirichlet a priori
eta <- 0.02   #parámetro de la distribución multinomial a priori
set.seed(1234)
fit <- lda.collapsed.gibbs.sampler(documents =
                          documents$documents, K = k,
                          vocab = documents$vocab,
                          num.iterations = num.iter, alpha = alpha,
                          eta = eta,initial = NULL, burnin = 0,
                          compute.log.likelihood =
                                      TRUE)
plot(fit$log.likelihoods[1,])
```



Aquí podemos observar como mejora la log-verosimilitud según añadimos iteraciones al proceso. Con el modelo que hemos creado podemos asignar a cada documento el tema

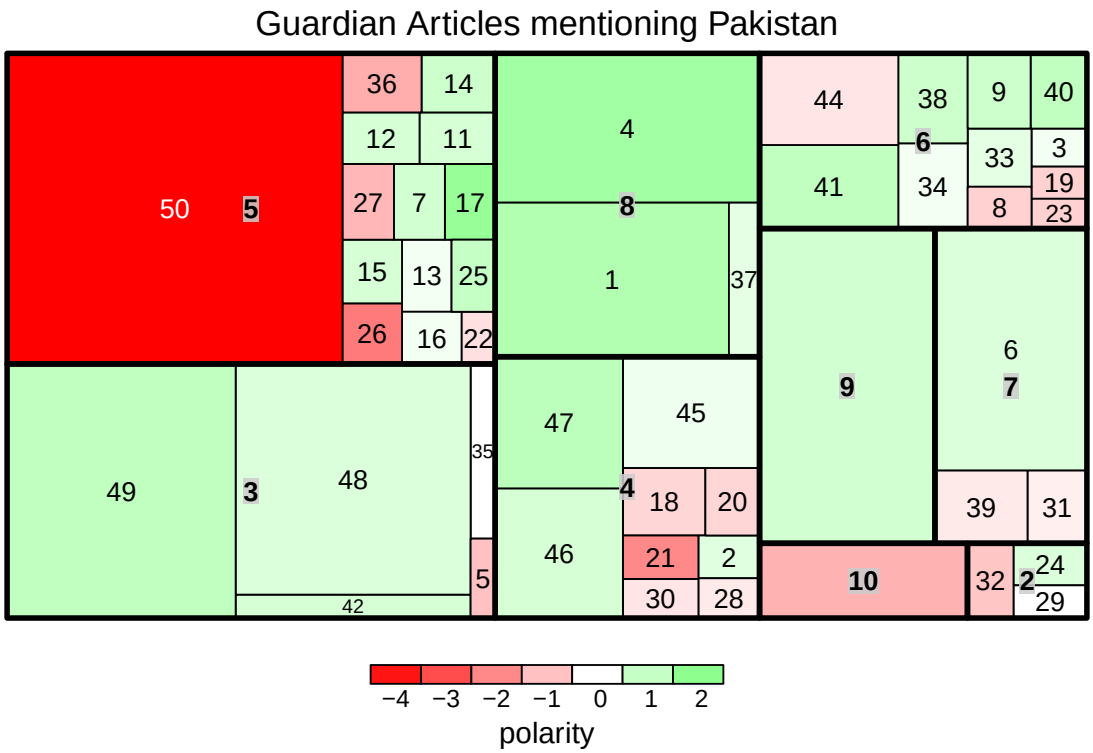
que se ajuste más a él. El modelo asigna un tema a cada palabra y para cada documento contaremos el número de palabras de cada tema que contiene y le asignaremos el que aparezca más.



Aquí podemos ver los 4 temas principales, artículos sobre dos partidos de Cricket distintos, los ataques en París y una miscelánea, viendo los artículos podemos darle nombre a estos temas obteniendo este diagrama. En el "treemap" creado por la función del mismo nombre podemos observar que cada polígono numerado representa un artículo siendo su tamaño proporcional a su longitud; por otro lado los colores representan los sentimientos presentes en el artículo los valores rojo representan un sentimiento más negativo. Los 4 temas corresponden a los polígonos con el borde grueso.

Se puede observar como el Cricket siendo un tema más ligero genera sentimientos más positivos mientras que es casi unánime el tono negativo en los ataques de París. Esta división de los sentimientos nos ayuda a asegurarnos de que la asignación de temas ha sido correcta por lo que el procedimiento ha cumplido su objetivo.

Podemos ahora también comprobar que obtendríamos si añadimos más temas, en este caso probaremos con 10, utilizaremos el mismo procedimiento que antes pero cambiando ese parámetro.



Aquí al añadir demasiados temas perdemos un poco lo que significa cada uno por lo que el análisis nos da peor información a pesar de haber añadido más temas. En este caso un proceso que se puede realizar es aplicar un algoritmo de selección del número óptimo de temas para no hacerlo de forma arbitraria; en nuestro caso estamos tomando el ejemplo de un libro por lo que sabíamos que 4 era el número correcto pero en un caso real no lo sabremos a priori.

3.3.2. Reducción de dimensionalidad y programas electorales

El objetivo de esta sección es utilizar el LSI para obtener la cercanía de los programas políticos en cuanto a los términos utilizados. Para ello usaremos los programas disponibles de las elecciones generales en castellano. El uso del LSI nos permitirá obtener un espacio de baja dimensión para realizar comparaciones más precisas y visuales que si usáramos la matriz término-documento entera, ya que esta tiene demasiadas filas y columnas.

En primer lugar realizamos un proceso previo de lectura de los pdf y limpieza del texto, que nos permite obtener los programas como cadenas de palabras. Una vez que tenemos los programas como texto crearemos la matriz término-documento.

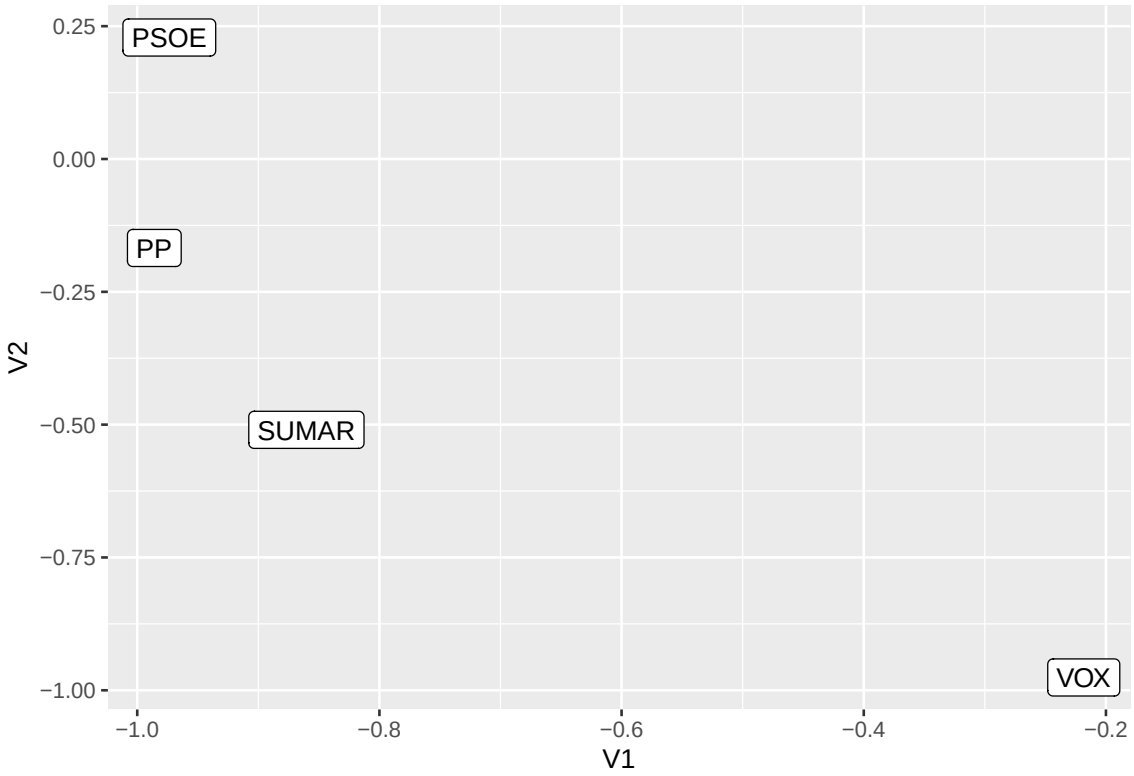
```
library(pdftools)
library(lsa)
library(stringr)
library(matlib)
setwd("C:/Users/alelo/Desktop/TFG/MemoriaTFE/Programas")
programas_pdf = list.files(pattern = "pdf$")

programas_pdf = programas_pdf[-c(1,2,3,4,8)]
```

```
programas_texto=lapply(programas_pdf,pdf_text)

n_progs=length(programas_pdf)
vector_programas=numeric(n_progs)
for( i in 1:n_progs){
  vector_programas[i]=gsub("\\\n", " ", programas_texto[[i]])%>%
    paste0(collapse = " ")%>%
    str_squish()
  vector_programas[i]=gsub("[^[:alnum:][:space:]]","",vector_programas[i])
}
programas_pdf=gsub(".pdf","",programas_pdf)
data_programas=cbind(vector_programas,programas_pdf)%>%
  as.data.frame()


TDM=TermDocumentMatrix(data_programas$vector_programas,
                       control = list(removePunctuation = TRUE,
                                      stopwords = FALSE,
                                      tolower = TRUE,
                                      stemming = FALSE,
                                      removeNumbers = TRUE,
                                      bounds = list(global = c(1, Inf))))%>%
  weightTfIdf(normalize = TRUE)
TDM$dimnames$Docs=programas_pdf
a=as.matrix(TDM)
set.seed(1237)
filas=nrow(a)
kable(a[sample(nrow(a),10),])
```

	PP	PSOE	SUMAR	VOX
avalado	0.00e+00	2.85e-05	0.0000144	0.0000000
interesan	0.00e+00	0.00e+00	0.0000000	0.0001013
eléctrico	7.93e-05	5.32e-05	0.0001014	0.0000000
mínima	0.00e+00	2.85e-05	0.0000431	0.0000000
penaliza	0.00e+00	0.00e+00	0.0000144	0.0000507
perdón	4.78e-05	0.00e+00	0.0000000	0.0000507
respaldar	0.00e+00	0.00e+00	0.0001150	0.0000000
activismo	0.00e+00	0.00e+00	0.0000288	0.0000000
apuestas	0.00e+00	5.90e-06	0.0000119	0.0000210
limitan	0.00e+00	1.42e-05	0.0000144	0.0000000

Este es un pequeño fragmento de la matriz de términos documento, en este caso con una ponderación TF-IDF. Podemos ver que hay muchos valores iguales a 0 ya que muchos términos no suelen aparecer en todos los programas.

Ahora con esta matriz de 4 columnas y 16106 filas utilizaremos el método LSA para obtener una matriz proyectada 4x2 que represente la anterior. Utilizaremos las proyecciones en las 2 nuevas coordenadas para representar la similitud entre el lenguaje empleado por cada partido.



Aquí podemos observar como Vox está separado completamente de los otros 3 partidos, habiendo distancias más pequeñas entre los otros 3. También podemos comprobar uno de los problemas del LSA y es que más allá de la cercanía, las 2 coordenadas extraídas no son fáciles de interpretar ya que no siguen ningún tipo de criterio fijado a priori.

Si repetimos este procedimiento añadiendo más partidos podemos proyectarlos en un espacio 3D y observar los resultados que obtenemos.

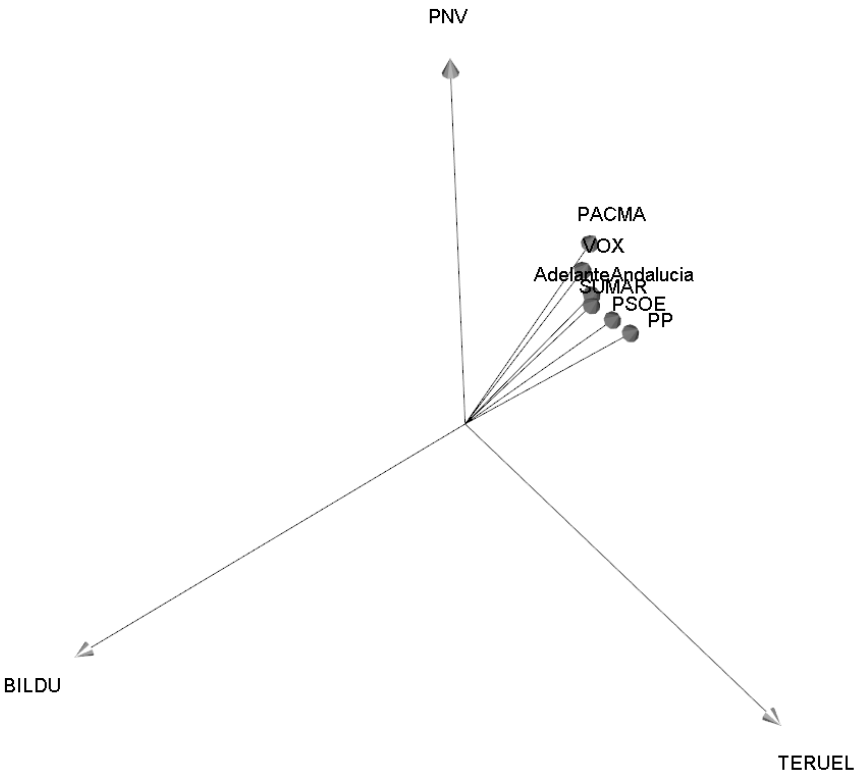


Figura 3.1: Programas de 9 partidos en 3D.

Aquí podemos ver como los programas de PNV, BILDU y Teruel Existe son completamente distintos a los otros analizados y parecen formar unos ejes. De entre ellos Bildu es completamente ortogonal a los demás.

Si quitamos estos 3 partidos que difieren del resto el resultado que obtenemos es el siguiente.

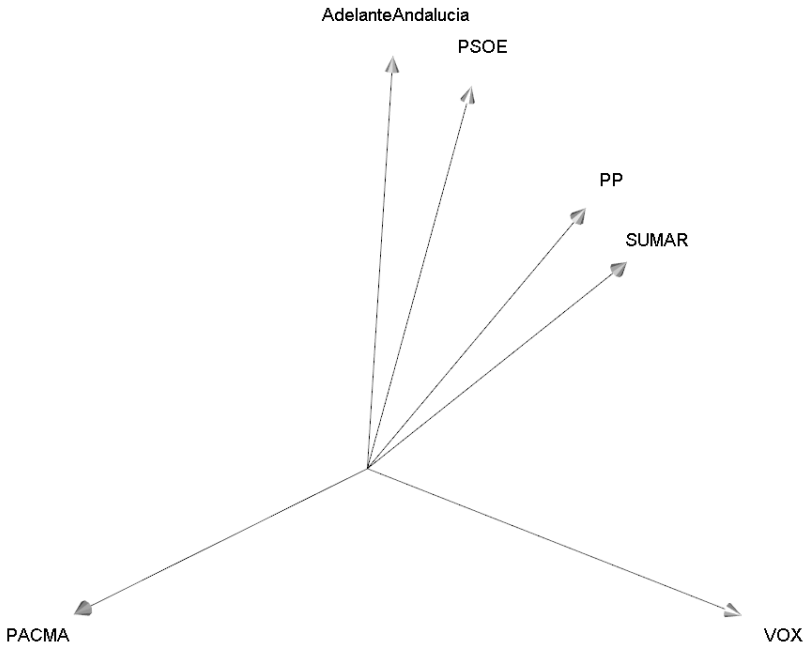


Figura 3.2: Programas de 6 partidos en 3D.

En este otro podemos observar que Pacma se presenta de forma completamente perpendicular al resto de los otros partidos mientras que los demás parecen estar presentes en un gradiente entre los programas de Vox y Adelante Andalucía, siendo más cercanos a este último.

3.4. Extracción de información

3.4.1. Reconocimiento de entidades nombradas

En esta sección nuestro objetivo será reconocer automáticamente las entidades de tipo persona, lugar u organización de un texto.

Vamos a empezar tomando las funciones que nos aporta el paquete openNLP[**?**] de R,este paquete contiene varias herramientas de procesamiento del lenguaje natural pero nos centraremos en la que nos permite reconocer los tipos de entidades.

En particular utilizaremos un modelo de máxima entropía.

```
#Anotadores de openNLP
person_ann = Maxent_Entity_Annotator(kind = "person")
location_ann = Maxent_Entity_Annotator(kind = "location")
organization_ann = Maxent_Entity_Annotator(kind = "organization")
word_ann = Maxent_Word_Token_Annotator()
sent_ann = Maxent_Sent_Token_Annotator()
```

Con estos modelos ya creados necesitaremos un texto para intentar reconocer sus

entidades. Utilizamos uno acerca de adquisición de empresas que sabemos que contiene nombres, lugares y organizaciones.

```
##  [1] "Which"     "countries" "have"
##  [4] "rejected"  "the"       "merger"
##  [7] "?"         "As"        "of"
## [10] "right"
```

Aquí podemos observar las primeras 10 palabras del texto ya separadas. Este vector de palabras lo usaremos como argumento para el modelo que tenemos y nos devolverá aquellas palabras que constituyan alguna de las entidades que buscamos.

Finalmente procederemos a obtener las entidades del texto.

La lista de los nombres extraída es: XboX Live, Infinity Ward, Bobby Kotick

United States, Australia

La lista de las organizaciones extraída es: Federal Trade Commission, FTC, New York Times, District Court, BNN Bloomberg, Microsoft, Markets Authority, CMA, Activision, Blizzard, SEC

Como podemos comprobar ha extraído una lista completa de los lugares y organizaciones mencionados. Podemos destacar de forma especial que la lista de nombres no solo incluye los nombres propios de personas "Bobby Kotick" en este caso, sino que también incluye de un servicio como es Xbox Live. También podemos considerar que comete un error aquí ya que "Inifinty Ward" es un estudio de videojuegos los que se podría considerar una organización por lo que está reconociendo que es una entidad pero no es capaz de clasificarla de forma precisa.

3.4.2. Extracción de relaciones

Nuestro objetivo en esta sección será en primer lugar encontrar los términos más utilizados de una serie de libros y en segunda instancia encontrar las palabras que tienen una relación entre ellas. Utilizaremos una aproximación frecuentista basada en el TF-IDF para hallar estas relaciones y finalmente utilizaremos la metodología LDA visto en la sección anterior para observar el uso del lenguaje de estas obras [21].

Las obras que utilizaremos son "El Quijote" de Miguel de Cervantes,"Los Pazos de Ulloa" de Emilia Pardo Bazán, "El árbol de la ciencia" de Pío Baroja y "El esperpento" de Ramón María del Valle Inclán. Los libros son extraídos de la librería gutenbergr[**?**] que a su vez los obtiene del repositorio del mismo nombre. Las novelas estarán en Castellano.

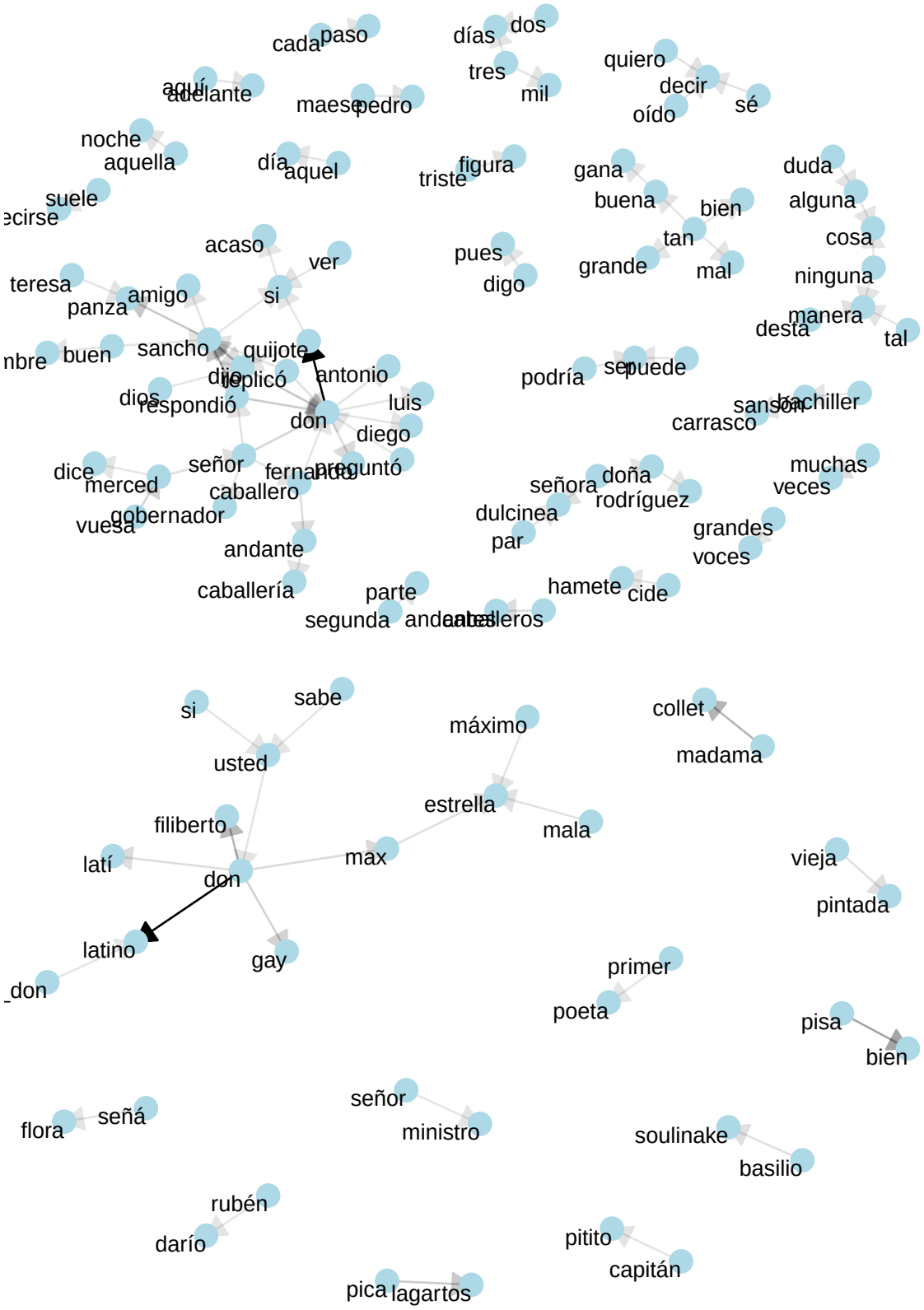
gutenberg_id	text
2000	Al libro de don Quijote de la Mancha
2000	Que trata de la condición y ejercicio del famoso
2000	hidalgo don Quijote de la Mancha
2000	Que trata de la primera salida que de su tierra hizo
2000	el ingenioso don Quijote
2000	Donde se cuenta la graciosa manera que tuvo don
2000	Quijote en armarse caballero
2000	De lo que le sucedió a nuestro caballero cuando salió
2000	de la venta
2000	Donde se prosigue la narración de la desgracia de
2000	nuestro caballero

Aquí vemos un extracto del formato en el que obtenemos la tabla, la primera columna es el ID correspondiente al "El Quijote" mientras que la segunda columna contiene renglones pertenecientes al libro.

Utilizando tidytext crearemos dos funciones. La primera cuenta la frecuencia de cada par de términos consecutivos a los que llamaremos bigramas y la segunda que nos permite representar en un gráfico estas relaciones.

```
count_bigrams <- function(dataset) {
  dataset %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
    filter(!is.na(bigram))%>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
    filter(!word1 %in% stopwords("es"),
           !word2 %in% stopwords("es")) %>%
    count(word1, word2, sort = TRUE)
}

set.seed(1234)
visualize_bigrams <- function(bigrams) {

  a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

  bigrams %>%
    graph_from_data_frame() %>%
    ggraph(layout = "fr") +
    geom_edge_link(aes(edge_alpha = n), show.legend = FALSE, arrow = a) +
    geom_node_point(color = "lightblue", size = 5) +
    geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
    theme_void()
}
```

Finalmente presentamos los grafos de asociación de términos para cada uno de los libros. cada nodo representa una palabra y las flechas representan que el nodo origen precede al nodo destino. Además la frecuencia está representada mediante el color de la flecha, cuanto más opaca es la flecha mayor frecuencia.

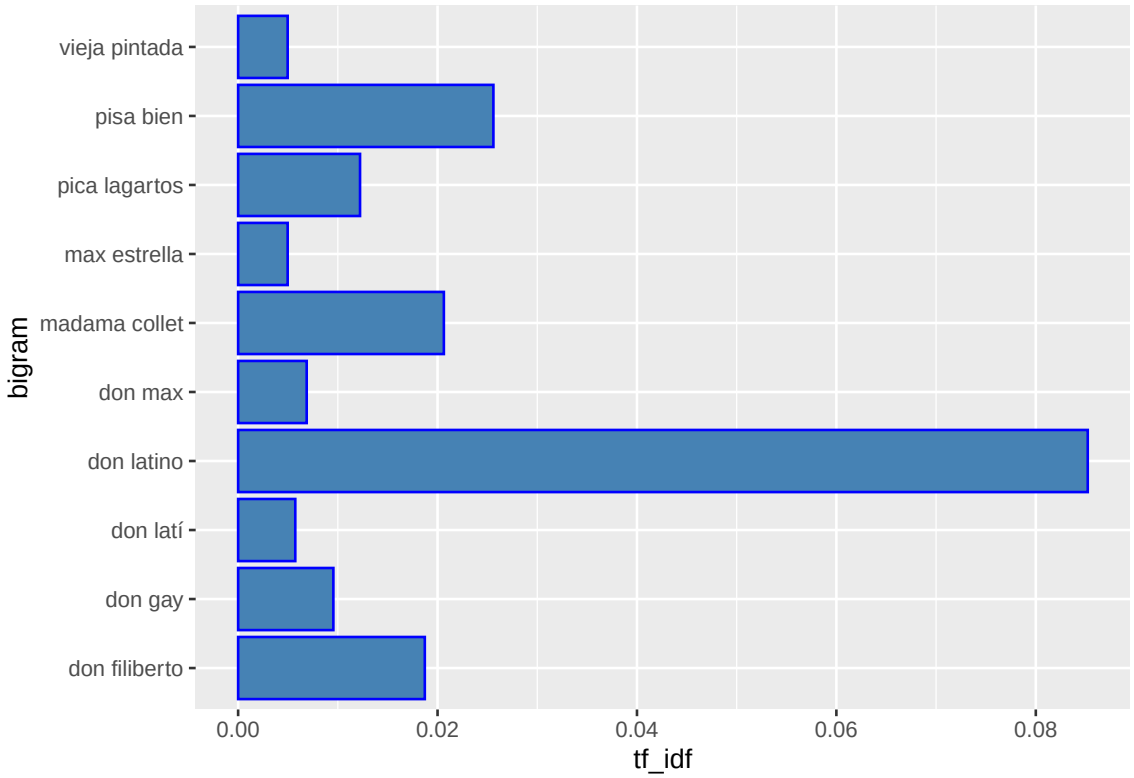


Como podemos ver las relaciones más comunes son “Don” seguido del nombre de alguno de los personajes principales de las novelas. Para poder ver el orden de forma más precisa representaremos los bigramas de cada libro del más al menos frecuente según el TF_IDF.

book	bigram	n	tf	idf	tf_idf
68745	don latino	223	0.0614664	1.3862944	0.0852105
68745	pisa bien	67	0.0184675	1.3862944	0.0256014
2000	don quijote	2061	0.0350880	0.6931472	0.0243212
68745	madama collet	54	0.0148842	1.3862944	0.0206339
68745	don filiberto	49	0.0135061	1.3862944	0.0187234
68745	pica lagartos	32	0.0088203	1.3862944	0.0122275
68745	don gay	25	0.0068908	1.3862944	0.0095527
2000	respondió sancho	303	0.0051585	1.3862944	0.0071512
2000	dijo don	296	0.0050393	1.3862944	0.0069860
68745	don max	18	0.0049614	1.3862944	0.0068780
2000	sancho panza	281	0.0047840	1.3862944	0.0066320
2000	respondió don	265	0.0045116	1.3862944	0.0062543
18005	don eugenio	9	0.0041744	1.3862944	0.0057869
60464	hermano juan	7	0.0041667	1.3862944	0.0057762
68745	don latí	15	0.0041345	1.3862944	0.0057316
2000	dijo sancho	236	0.0040178	1.3862944	0.0055699
68745	max estrella	13	0.0035832	1.3862944	0.0049674
68745	vieja pintada	13	0.0035832	1.3862944	0.0049674
68745	basilio soulinake	12	0.0033076	1.3862944	0.0045853
2000	vuesa merced	180	0.0030645	1.3862944	0.0042482

Aquí podemos ver los 20 bigramas con mejor frecuencia relativa, los más frecuentes son los nombres de personajes en general y en particular los del esperpento. Al tratarse de una obra de teatro cada intervención de los personajes recoge su nombre por lo que aumenta la proporción con la que aparecen. Ausente de esto queda su coprotagonista, Max Estrella, ya que sus intervenciones vienen marcadas simplemente por "Max". Por otro lado también vemos que las combinaciones de un verbo que indica el habla junto a un nombre de personaje son comunes dado que aparecen con mucha frecuencia en los diálogos de las novelas.

Podemos representar los términos de cada obra por separado obteniendo el siguiente resultado:



Aquí observamos de nuevo que las combinaciones más comunes son los honoríficos seguidos del nombre de un personaje así como verbos que indican el habla seguidos por personajes. La única combinación que rompe este patrón en El Quijote es "caballeros andantes" al ser un bigrama bastante inusual en general pero muy repetido en esta obra.

Si realizamos un análisis con las otras obras podemos observar que tenemos un caso similar en el resto de obras, teniendo expresiones como "dos cuartos" en "El Árbol de la ciencia" y "pisa bien" en "el Esperpento" las excepciones.

3.5. Conclusiones

En este capítulo hemos tratado con cuatro áreas de la minería de textos. Hemos empezando tratando con R el "resumen de texto".Se puede decir que tiene métodos sólidos y la implementación de la teoría es sencilla y eficiente, pero necesita de conjuntos de textos en los idiomas específicos para poder realizar su labor de forma efectiva. Para tener buenos métodos para textos en español en R sería necesaria la elaboración de un repositorio adecuado.

En segundo lugar hemos visto los métodos de clasificación. Estos consiguen buenos resultados con precisiones de hasta un 70 %, pero los tiempos de ejecución son bastante altos comparados con el resto de las áreas. Tanto el método basado en la χ^2 implementado en este trabajo como las comparaciones realizadas mediante "RTextTools" tardan un tiempo considerable para obtener resultados con un conjunto de 4000 textos y 50000 palabras distintas.

En tercer lugar hemos trabajado con la reducción de dimensionalidad. Hemos hallado buenos resultados pero al mismo tiempo no hemos profundizado en la selección del número de temas con el LDA, este podría ser un aspecto en el que avanzar. Por otro lado el LSI

hemos logrado encontrar similitudes en los programas pero la evaluación de esta misma es un tema más complicado por el hecho de falta de clasificación previa. Un posible análisis interesante sería correlar la distancia según el LSI con la longitud de los programas ya que los programas más cortos parecen tender a separarse de los demás al tener menos términos distintos.

En cuarto y último lugar hemos tratado con la extracción de información. En el reconocimiento de entidades nombradas hemos utilizado un modelo ya creado para obtener resultados bastante precisos, una vía de avance sería implementar en R el método "Nymble" descrito en la sección de teoría. De esta forma también se podrían expandir el número de tipos de entidades que se estudian.

La extracción de relaciones realizada ha servido para ilustrar los patrones de ocurrencia más comunes pero no se ha implementado parte de la base teórica para encontrar más tipos de relaciones específicas. Una buena forma de avanzar en este aspecto sería el estudio del paquete "x.ent" [**?**] el cual contiene funciones útiles para la extracción de información.

Sobre otros aspectos de la minería de texto se podrían añadir aplicaciones y una base teórica sobre el análisis de sentimientos en textos. En este trabajo se ve una pequeña aplicación dentro del LDA pero el tema es suficientemente rico como para que tuviese su propia sección. De este tema además existe una amplia bibliografía tanto práctica como teórica [2],[9].

Finalmente hay aplicaciones específicas a temas como la biomedicina o las redes sociales que también tienen un estudio muy interesante pero que no hemos llegado a ver en este trabajo.

Apéndice A

Apéndice: Scripts de R

Este apéndice incluye el código completo utilizado para la generación de los resultados, extractos del mismo se encuentran en el capítulo 3 pero aquí aparece la secuencia entera con sus comentarios.

A.1. Resumen de textos

```
## Cargamos los paquetes que vamos a utilizar
library(readr)
library(tidyr)
library(tm)
library(dplyr)
library(ds4psy)
library(lsa)
library(tidytext)
#Realizamos una lectura de los datos que se encuentran en una carpeta
#del sistema, realizamos un muestreo ya que el repositorio cuenta con
#demasiados documentos y no los necesitamos en este caso. Nos quedamos
#con archivos de todos los temas disponibles. Además los dividimos en
#entrenamiento y testeo.
vector_temas=list.files(
  "C:/Users/alelo/Desktop/TFG/Reuters21578-Apte-90Cat/test")
lista_articulos=c()
set.seed(1997)
for (i in vector_temas){
  ruta_tema=paste0(
    "C:/Users/alelo/Desktop/TFG/Reuters21578-Apte-90Cat/test/",i)
  articulos=list.files(ruta_tema)
  l=length(articulos)*1
  elegidos=sample(articulos,size = l, replace = FALSE)
  if(0<length(elegidos)){
  for (j in 1:length(elegidos)){
    siguiente_archivo=paste0(ruta_tema,"/",elegidos[j])
    siguiente_fila=c(read_file(siguiente_archivo),elegidos[j],i)
```

```
    lista_articulos=rbind(lista_articulos,siguiente_fila)
  }
  }
}

lista_articulos_test=lista_articulos%>%as.data.frame()
rownames(lista_articulos_test)=1:nrow(lista_articulos)
colnames(lista_articulos_test)=c("Articulo","CODART","Tema")
#

lista_articulos=c()
for (i in vector_temas){
  ruta_tema=paste0(
    "C:/Users/alelo/Desktop/TFG/Reuters21578-Apte-90Cat/training/",i)
  articulos=list.files(ruta_tema)
  l=length(articulos)*0.4
  elegidos=sample(articulos,size = l, replace = FALSE)
  if(0<length(elegidos)){
    for (j in 1:length(elegidos)){
      siguiente_archivo=paste0(ruta_tema,"/",elegidos[j])
      siguiente_fila=c(read_file(siguiente_archivo),elegidos[j],i)
      lista_articulos=rbind(lista_articulos,siguiente_fila)
    }
  }
}

lista_articulos_training=lista_articulos%>%as.data.frame()
rownames(lista_articulos_training)=1:nrow(lista_articulos)
colnames(lista_articulos_training)=c("Articulo","CODART","Tema")


lista_articulos=rbind(lista_articulos_training,lista_articulos_test)




############################################################

#Creamos la matriz término documento y realizamos una
#limpieza de texto básica.
TDM =TermDocumentMatrix(lista_articulos$Articulo,
                        control = list(removePunctuation = TRUE,
                                       stopwords = TRUE,
                                       tolower = FALSE,
                                       stemming = FALSE,
                                       removeNumbers = TRUE,
                                       bounds = list(global = c(1, Inf))))
```

```
TFM_training=textmatrix(list.files("/Reuters21578-Apte-90Cat/test/acq/"),
                        stemming=FALSE, language="english",
            minWordLength=2, maxWordLength=FALSE, minDocFreq=1,
            maxDocFreq=FALSE, minGlobFreq=FALSE, maxGlobFreq=FALSE,
            stopwords=NULL, vocabulary=NULL, phrases=NULL,
            removeXML=FALSE, removeNumbers=TRUE)
TDM$dimnames$Docs=lista_articulos$CODART
## Numero de documentos
n=nrow(lista_articulos)
TDM_training=TDM[,c(1:nrow(lista_articulos_training))]
TDM_test=TDM[,
     c((nrow(lista_articulos_training)+1):(nrow(lista_articulos)-3600))]
lsa_trainig=lsa(TDM_training,150)
temp=tidy(TDM_test)%>%
  mutate(term=as.factor(term))%>%
  mutate(document=as.factor(document))%>%
  pivot_wider(names_from = term,values_from =count)
temp=as.matrix(TDM_test)
resultados=as.matrix(t(fold_in(temp,lsa_trainig)))
as.textmatrix(lsa_trainig)
```

A.2. Clasificación de textos

A.2.1. Script clasificación RTextTools

```
# Creamos la matriz término documento y realizamos una
# selección de palabras
doc_matrix <- create_matrix(USCongress$text, language="english",
                            removeNumbers=TRUE,
                            stemWords=TRUE, removeSparseTerms=.998)
container <- create_container(doc_matrix,
                              USCongress$major, trainSize=1:4000,
                              testSize=4001:4449, virgin=FALSE)
#Entrenamos los distintos modelos
SVM <- train_model(container,"SVM")
SLDA <- train_model(container,"SLDA")
BOOSTING <- train_model(container,"BOOSTING")
BAGGING <- train_model(container,"BAGGING")
RF <- train_model(container,"RF")
NNET <- train_model(container,"NNET")
TREE <- train_model(container,"TREE")
#realizamos la clasificación a aprtir de los modelos
SVM_CLASSIFY <- classify_model(container, SVM)
SLDA_CLASSIFY <- classify_model(container, SLDA)
BOOSTING_CLASSIFY <- classify_model(container, BOOSTING)
BAGGING_CLASSIFY <- classify_model(container, BAGGING)
```

```
RF_CLASSIFY <- classify_model(container, RF)
NNET_CLASSIFY <- classify_model(container, NNET)
TREE_CLASSIFY <- classify_model(container, TREE)
#Obtenemos un resumen de los resultados
analytics =create_analytics(container,
                            cbind(SVM_CLASSIFY, SLDA_CLASSIFY,
                                BOOSTING_CLASSIFY, BAGGING_CLASSIFY,
                                RF_CLASSIFY, NNET_CLASSIFY, TREE_CLASSIFY))
summary(analytics)
```

A.2.2. Script clasificación χ^2

```
#Cargamos los paquetes
library(RTextTools)
library(readr)
library(tidyr)
library(dplyr)
library(tm)
library(caret)
library(e1071)
#Cargamos los datos y los dividimos en entrenamiento y prueba
data("USCongress")
data_train=USCongress[1:4000,]
data_test=USCongress[4001:4449,]

#Creamos la matriz término documento con y sin los datos de prueba
TDM= TermDocumentMatrix(data_train$text,
                        control = list(removePunctuation = TRUE,
                                       stopwords = TRUE,
                                       tolower = TRUE,
                                       stemming = TRUE,
                                       removeNumbers = TRUE,
                                       bounds = list(global = c(1, Inf))))
TDM_ampliada=TermDocumentMatrix(USCongress$text,
                                control = list(removePunctuation = TRUE,
                                       stopwords = TRUE,
                                       tolower = TRUE,
                                       stemming = TRUE,
                                       removeNumbers = TRUE,
                                       bounds = list(global = c(1, Inf))))

#Definimos las funciones que obtienen los elementos
#para nuestra tabla de contingencia
clases=sort(unique(data_train$major))

A=function(clase, palabra){
  vector_clases=which(data_train$major==clase)
```

```
  return(length(TDM[c(palabra),c(vector_clases)]$i))
}
B=function(clase,palabra){
  vector_clases=which(data_train$major!=clase)
  return(length(TDM[c(palabra),c(vector_clases)]$i))
}

C=function(clase,palabra){
  docs_clases=length(which(data_train$major==clase))
  return(docs_clases-A(clase,palabra))
}

D=function(clase, palabra){
  docs_noc_clase=length(which(data_train$major!=clase))
  return(docs_noc_clase-B(clase,palabra))
}

O_i_j=function(clase,palabra){
  return(matrix(ncol=2,nrow=2,data=c(A(clase,palabra),B(clase,palabra),
                                     C(clase,palabra),D(clase,palabra))))
}
E_i_j=function(clase,palabra){
  a=A(clase,palabra)
  b=B(clase,palabra)
  c=C(clase,palabra)
  d=D(clase,palabra)
  N=a+b+c+d
  e_1_1=(a+c)*(a+b)/N
  e_1_2=(a+b)*(b+d)/N
  e_2_1=(a+c)*(c+d)/N
  e_2_2=(b+d)*(c+d)/N
  return(matrix(ncol=2,nrow=2,data=c(e_1_1,e_1_2,e_2_1,e_2_2)))
}

chi_sq_clase_palabra=function(clase, palabra){
  O=O_i_j(clase,palabra)
  E=E_i_j(clase,palabra)
  chi=0
  for(i in 1:2){
    for (j in 1:2){
      sumando=((O[i,j]-E[i,j])^2)/E[i,j]
      chi=chi+sumando
    }
  }
  return(chi)
}
# Calculamos los estadísticos de chi-cuadrado para cada par de
# clases y términos.Ordenamos los resultados según el estadístico.
```

```
m=nrow(TDM)
n=length(clases)
matriz_chis=matrix(data=NA,nrow=n,ncol=m)
for(i in 1:n){
for(j in 1:m){
  matriz_chis[i,j]=chi_sq_clase_palabra(clases[i],TDM$dimnames$Terms[j])
}
}
matriz_chis=as.data.frame(t(matriz_chis))
colnames(matriz_chis)=clases
rownames(matriz_chis)=TDM$dimnames$Terms
matriz_chis%>%
  mutate(mean=rowMeans(.))%>%
  arrange(desc(mean))->matriz_ordenada
#Nos quedamos con las palabras que lleguen al umbral establecido

palabras_comparacion=matriz_ordenada%>%
  filter(mean>qchisq(0.99,1))%>%
  rownames()
matriz_filtrada=TDM[c(palabras_comparacion),c(1:4000)]%>%
  as.matrix()%>%
  t()%>%
  as.data.frame()%>%
  mutate(clase=as.factor(data_train$major))->data_model_train

matriz_filtrada_2=TDM_ampliada[c(palabras_comparacion),c(4001:4449)]%>%
  as.matrix()%>%
  t()%>%
  as.data.frame()%>%
  mutate(clase=as.factor(data_test$major))->data_model_test
# Elegimos el mejor número de puntos para el algoritmo KNN

grid <- expand.grid(.k=seq(1,20,by=1))
trainControl <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"
variables=ncol(data_model_train)-1
fit.knn <- train(data_model_train[,1:variables],
                 as.factor(data_model_train[,variables+1]), method="knn",
                 tuneGrid=grid,metric = metric)
knn.k2 <- fit.knn$bestTune
print(fit.knn)
plot(fit.knn)
# Realizamos la predicción para knn y svm
prediccion_knn <- predict(fit.knn, newdata = data_model_test)
cf_knn <- confusionMatrix(prediccion_knn, data_model_test$clase)
print(cf_knn)

svm=svm(clase~.,data=data_model_train)
```

```
prediccion_svm=predict(svm,data_model_test)
#Obtenemos los resultados de la precisión de la predicción
cf_SVM=confusionMatrix(prediccion_svm, data_model_test$clase)
print(cf_SVM)
```

A.3. Reducción de dimensionalidad e identificación de temas

A.3.1. LDA

```
library(tm)
library(qdap)
library(lda)
library(dplyr)
library(pbapply)
library(LDAvis)
library(treemap)
library(car)
options(stringsAsFactors = F)
# Cargamos los artículos
text<-read.csv(
  'Guardian_articles_11_14_2015_12_1_2015.csv')
# Limpiamos los textos
articles <- iconv(text$body, "latin1", "ASCII", sub="")
articles <- gsub('http\\S+\\s*', '', articles)
articles <- bracketX(articles,bracket='all')
articles <- gsub("[[:punct:]]", "",articles)
articles <- removeNumbers(articles)
articles <- tolower(articles)
removeWords(articles,c(stopwords('en'),
                                  'pakistan','gmt','england'))
# Limpiamos los espacios
blank.removal<-function(x){
  x<-unlist(strsplit(x,' '))
  x<-subset(x,nchar(x)>0)
  x<-paste(x,collapse=' ')
}
articles<-pblapply(articles,blank.removal)

# Aplicamos la funcion lexicalize del paquete lda
# para crear el corpus y el vocabulario
documents <- lexicalize(articles)

wc <- word.counts(documents$documents,
                  documents$vocab)
doc.length<- document.lengths(documents$documents)
```

```
# Utilizamos la lda con muestreo gibbs para crear el modelo
k <- 4 # numero de temas a identificar
num.iter <- 25 # numerod e repeticiones de muestreos
alpha <- 0.02
eta <- 0.02
set.seed(1234)
fit = lda.collapsed.gibbs.sampler(documents =
                                    documents$documents, K = k,
                                  vocab = documents$vocab,
                                   num.iterations = num.iter,
                                  alpha = alpha, eta = eta,
                                   initial = NULL, burnin = 0,
                                  compute.log.likelihood = TRUE)
# Aquí podemos ver que según iteramos mejora la verosimilitud
plot(fit$log.likelihoods[1,])

# calculamos la distribución de temas en los documentos
theta <- t(pbapply(fit$document_sums + alpha, 2,
                   function(x) x/sum(x)))
phi <- t(pbapply(t(fit$topics) + eta, 2, function(x)
  x/sum(x)))

# Convertimos los artículos en JSON ya que la función que
# sumaremos lo requiere

article.json <- createJSON(phi = phi,theta = theta,
                           doc.length = doc.length, vocab =
                             documents$vocab,
                           term.frequency = as.vector(wc))
# funcion que asigna el tema a cada documento
doc.assignment<-function(x){
  x<-table(x)%>%
  as.matrix()%>%
  t()%>%
  max.col()
}

assignments<-unlist(pblapply(fit$assignments,doc.assignment))
# Al concoer los temas de antemano podemos darle el nombre
assignments<-recode(assignments, "1='Cricket1';
2='Paris Attacks'; 3='Cricket2';4='Unknown'")

# Creamos un treemap para visualizar las clases de los articulos y
# la polaridad de sentimientos con la funcion polarity
article.ref<-seq(1:nrow(text))
article.pol<-polarity(articles)[[1]][3]
article.tree.df<-cbind(article.ref,
```

```
                        article.pol,doc.length,assignments)
treemap(article.tree.df,index=c("assignments",'article.ref'),
        vSize="doc.length",vColor="polarity", type="value",
        title="Guardian Articles mentioning Pakistan",
        palette=c("red","white","green"))
```

A.3.2. LSI en programas electorales

```
library(pdftools)
library(lsa)
library(stringr)
library(matlib)

#Obtenemos los programas en pdf del directorio
programas_pdf = list.files(pattern = "pdf$")

programas_pdf = programas_pdf[-c(1,2,3,4,8)]

programas_texto=lapply(programas_pdf,pdf_text)

n_progs=length(programas_pdf)
## Realizamos la limpieza de texto
vector_programas=numeric(n_progs)
for( i in 1:n_progs){
  vector_programas[i]=gsub("\\\n", " ", programas_texto[[i]])%>%
    paste0(collapse = " ")%>%
    str_squish()
  vector_programas[i]=gsub("[^[:alnum:][:space:]]","",vector_programas[i])
}
programas_pdf=gsub(".pdf","",programas_pdf)
data_programas=cbind(vector_programas,programas_pdf)%>%
  as.data.frame()

# Creamos la matriz término documento
TDM=TermDocumentMatrix(data_programas$vector_programas,
                       control = list(removePunctuation = TRUE,
                                      stopwords = FALSE,
                                      tolower = TRUE,
                                      stemming = FALSE,
                                      removeNumbers = TRUE,
                                      bounds = list(global = c(1, Inf))))%>%
  weightTfIdf(normalize = TRUE)
TDM$dimnames$Docs=programas_pdf
a=as.matrix(TDM)
set.seed(1237)
filas=nrow(a)
```

```
# Realizamos el LSA
lsa=lsa(TDM,dims=dimcalc_share())

# Proyectamos cada programa a un vector de dimensión 3

V=lsa$dk%>%as.data.frame()
for(i in 1:n_progs){
  V[i,]=V[i,]/sqrt(sum(V[i,]^2))
}

ggplot(V, aes(V1,V2,label=rownames(V)))+
  geom_label()
```

A.4. Extracción de información

```
#Carga de librerías
library(openxlsx)
library(rJava)
library(NLP)
library(openNLP)
library(RWeka)
library(openNLPmodels.en)
library(coreNLP)
library(readr)

#Anotadores de openNLP
person_ann = Maxent_Entity_Annotator(kind = "person")
location_ann = Maxent_Entity_Annotator(kind = "location")
organization_ann = Maxent_Entity_Annotator(kind = "organization")
word_ann = Maxent_Word_Token_Annotator()
sent_ann = Maxent_Sent_Token_Annotator()

#Obtención del texto
setwd("C:/Users/alelo/Desktop/TFG")
text=gsub("\\\n", " ",read_file("NER_Blizzard.txt"))

#Transfromación del texto en string
text <- paste(text, collapse = " ")

text<- as.String(text)

pipeline = list(sent_ann,
                word_ann,
                person_ann,
                location_ann,
                organization_ann)
```

```
text_annotations = NLP::annotate(text, pipeline)
text_doc = AnnotatedPlainTextDocument(text, text_annotations)
words(text_doc)[1:10]
#Creación de la función a aplicar
entities = function(doc, kind) {
  s = doc$content
  a = annotation(doc)
  if(hasArg(kind)) {
    k <- sapply(a$features, "[[", "kind")
    s[a[k == kind]]
  } else {
    s[a[a$type == "entity"]]
  }
}
#Se realiza el reconocimiento
nombres=entities(text_doc,kind="person")
print(paste0(c("La lista de los nombres extraída es: ")
             ,paste0( nombres,collapse=", ")))
lugares=entities(text_doc,kind="location")
print(paste0(c("La lista de los lugares extraída es: "),
             paste0( lugares,collapse=", ")))
organizaciones=unique(entities(text_doc,kind="organization"))
print(paste0(c("La lista de las organizaciones extraída es: ")
             ,paste0( organizaciones,collapse=", ")))
```

Bibliografía

[1] Aggarwal, Charu C y Aggarwal, Charu C (2015). *Mining text data*. Springer.

[2] Alaminos-Fernández, Antonio Francisco (2023). «Introducción a la minería de texto y análisis de sentimiento con R».

[3] Bikel, Daniel M; Miller, Scott; Schwartz, Richard y Weischedel, Ralph (1998). «Nymble: a high-performance learning name-finder». *arXiv preprint cmp-lg/9803003*.

[4] Blei, David M; Ng, Andrew Y y Jordan, Michael I (2003). «Latent dirichlet allocation». *Journal of machine Learning research*, **3(Jan)**, pp. 993–1022.

[5] Carbonell, Jaime y Goldstein, Jade (1998). «The use of MMR, diversity-based reranking for reordering documents and producing summaries». En: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 335–336.

[6] Deerwester, Scott; Dumais, Susan T; Furnas, George W; Landauer, Thomas K y Harshman, Richard (1990). «Indexing by latent semantic analysis». *Journal of the American society for information science*, **41(6)**, pp. 391–407.

[7] Dunning, Ted (1994). «Accurate methods for the statistics of surprise and coincidence». *Computational linguistics*, **19(1)**, pp. 61–74.

[8] Filatova, Elena y Hatzivassiloglou, Vasileios (2004). «A formal model for information selection in multi-sentence text extraction».

[9] Flores-Ruiz, David; Elizondo-Salto, Adolfo y Barroso-González, María de la O. (2021). «Using Social Media in Tourist Sentiment Analysis: A Case Study of Andalusia during the Covid-19 Pandemic». *Sustainability*, **13(7)**. ISSN 2071-1050. doi: 10.3390/su13073836.
https://www.mdpi.com/2071-1050/13/7/3836.

[10] Griffiths, Thomas L y Steyvers, Mark (2004). «Finding scientific topics». *Proceedings of the National academy of Sciences*, **101(suppl_1)**, pp. 5228–5235.

[11] Han, Eui-Hong; Karypis, George y Kumar, Vipin (2001). «Text categorization using weight adjusted k-nearest neighbor classification». En: *Advances in Knowledge Discovery and Data Mining: 5th Pacific-Asia Conference, PAKDD 2001 Hong Kong, China, April 16–18, 2001 Proceedings 5*, pp. 53–65. Springer.

[12] Joachims, Thorsten (1998). «Text categorization with support vector machines:

Learning with many relevant features». En: *European conference on machine learning*, pp. 137–142. Springer.

[13] Jordan, Michael I (2010). «The conjugate prior for the normal distribution». *Lecture notes on Stat260: Bayesian Modeling and Inference*.

[14] Kwartler, Ted (2017). *Text mining in practice with R*. John Wiley & Sons.

[15] Mao, Yuning; Qu, Yanru; Xie, Yiqing; Ren, Xiang y Han, Jiawei (2020). «Multi-document Summarization with Maximal Marginal Relevance-guided Reinforcement Learning».

[16] McCallum, Andrew y Li, Wei (2003). «Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons».

[17] Oakes, Michael; Gaaizauskas, Robert; Fowkes, Helene; Jonsson, Anna; Wan, Vincent y Beaulieu, Micheline (2001). «A method based on the chi-square test for document classification». En: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 440–441.

[18] Park, Heum; Kwon, Soonho y Kwon, Hyuk-Chul (2010). «Complete gini-index text (git) feature-selection algorithm for text classification». En: *The 2nd international conference on software engineering and data mining*, pp. 366–371. IEEE.

[19] Plackett, Robin L (1983). «Karl Pearson and the chi-squared test». *International statistical review/revue internationale de statistique*, pp. 59–72.

[20] Ramshaw, Lance A y Marcus, Mitchell P (1999). «Text chunking using transformation-based learning». En: *Natural language processing using very large corpora*, pp. 157–176. Springer.

[21] Silge, Julia y Robinson, David (2017). *Text mining with R: A tidy approach.* " O'Reilly Media, Inc.".

[22] Storkey, AJ (2020). «Machine Learning and Pattern Recognition: Note on Dirichlet Multinomial».

[23] Van Huffel, Sabine (1990). «Partial singular value decomposition algorithm». *Journal of computational and applied mathematics*, **33(1)**, pp. 105–112.

[24] Xu, Zhao; Yu, Kai; Tresp, Volker; Xu, Xiaowei y Wang, Jizhi (2003). «Representative sampling for text classification using support vector machines». En: *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14–16, 2003. Proceedings 25*, pp. 393–407. Springer.