Técnicas Estadísticas en Minería de Textos

Grado en Matemáticas
Departamento de Estadística e Investigación Operativa

Dirigido por José Luis Pino Mejías

Elena Auxiliadora Doctor Bracho

Junio 2018

Índice general

Resumen

El objetivo de este trabajo es realizar una revisión de las bases teóricas de las técnicas estadísticas aplicables a la minería de textos, la programación en R de las principales técnicas y la aplicación a conjuntos de datos extraídos mediante el uso de las librerías específicas para ello.

En los primeros cuatro capítulos, se introducen nociones introductorias relativas a la minería de textos, así como la relación de este campo con la minería de datos, un breve recorrido histórico por su desarrollo y algunos ejemplos de aplicaciones.

En el quinto capítulo, se explican las diferentes etapas que conforman un proceso de minería de textos, mientras que, en el sexto, nos centramos en las diferentes técnicas.

Posteriormente, se aplican algunos de los métodos de los capítulos anteriores a través del software R. En concreto, nos centramos en tres de las técnicas de clasificación y la Asignación Latente de Dirichlet.

Por último, se hace una breve reflexión acerca de los objetivos a los que se enfrenta la minería de textos en su desarrollo futuro, basándonos en la situación en la que se encuentra actualmente.

Abstract

The objective of this work is to review the theoretical bases of the statistical techniques applicable to text mining, the R programming of the main techniques and the application to extracted data sets through the use of specific libraries for this purpose.

In the first four chapters, introductory notions related to text mining are introduced, as well as the relationship of this field with data mining, a brief historical overview of its development and some application examples.

In the fifth chapter, the different stages that make up a text mining process are explained, while, in the sixth, we focus on the different techniques.

Subsequently, some of the methods of the previous chapters are applied through the R software. Specifically, we focus on three of the classification techniques and the Latent Dirichlet Allocation.

Finally, a brief reflection is made about the objectives that text mining faces in its future development, based on the situation in which it is currently.

Capítulo 1

Introducción

La minería de datos es un campo que ha avanzado rápidamente en los últimos años debido a los importantes avances en tecnología de hardware y software. Este avance es especialmente notable en el caso de los datos en forma de texto, en el que el desarrollo de plataformas hardware y software para la web y las redes sociales ha permitido el almacenamiento de grandes cantidades de datos. La web provee de una cantidad enorme de contenido textual creado por un gran número de diferentes usuarios y que es fácil de almacenar y procesar. Este incremento en la cantidad de datos textuales disponibles ha despertado el interés en desarrollar métodos y algoritmos que permitan detectar patrones interesantes en los datos textuales.

La minería de textos es considerada como una de las áreas con mayor potencial dentro de la minería de datos, debido a que la manera habitual de almacenar información es en forma de textos. La minería de textos es, sin embargo, una disciplina mucho más compleja que la minería de datos, que trata con bases de datos estructurados. Esta tarea, por el contrario, requiere tratar con datos textuales que son propiamente desestructurados y más confusos.

Mientras que para el manejo de datos estructurados se suele recurrir a sistemas de bases de datos, para el manejo de los datos textuales es frecuente el uso buscadores, que permiten al usuario encontrar información útil a partir de palabras clave. La mejora de la efectividad y la eficiencia de los motores de búsqueda ha sido, por tanto, un tema de interés central en el campo de recuperación de la información.

Podemos definir la minería de textos como el proceso de extracción de patrones y conocimientos de interés y no triviales a partir de textos no estructurados. Esto es, la información descubierta no debe estar presente de forma explícita en los documentos de texto ya que, en ese caso, no se trataría de un descubrimiento. La investigación en el campo de recuperación de la información, sin embargo, se ha centrado tradicionalmente en la facilitación del acceso a la información más que en el análisis de la información en busca de patrones, el objetivo principal de la minería de textos. El objetivo del acceso a la información es conectar la información apropiada con el usuario apropiado y en el momento apropiado, enfatizando menos en el procesamiento o la transformación de la información textual. Existen, aún así, muchas aplicaciones de la minería de textos en

las que el principal objetivo es descubrir y analizar patrones interesantes, así como la detección de tendencias y outliers. Es, por tanto, un campo multidisciplinar, que incluye conocimientos de recuperación de la información, análisis de textos, extracción de la información, clustering, categorización, aprendizaje de máquinas, etc.

Existen muchas características clave que diferencian los datos textuales de los datos de otro tipo, como pueden ser los datos cuantitativos. Esto afecta directamente a qué técnicas de minería pueden ser aplicadas a estos datos. La caracteristica más importante de los datos textuales es que son *sparse* y de gran dimensión. Por ejemplo, un corpus puede manejar un vocabulario de alrededor de 100000 palabras, mientras que uno de los documentos de este corpus podría incluir solamente unos pocos cientos de ellas. Así, un corpus puede ser representado como una matriz de tamaño $n \times d$, donde n es el número de documentos y d es el tamaño del vocabulario, en la que el término (i, j) es la frecuencia normalizada de la j-ésima palabra del vocabulario en el i-ésimo documento. Obtenemos entonces una matriz hueca y de gran tamaño, lo que tiene importantes implicaciones a la hora de aplicar las diferentes técnicas.

Por otro lado, los datos textuales pueden ser representados de diferentes formas para su análisis: bolsa de palabras, cadena de palabras, etc. Sin embargo, en la mayoría de sus aplicaciones, resulta deseable representar esta información *semánticamente*, de forma que el análisis y la minería posterior sea más significativo. Los métodos actuales en procesamiento del lenguaje natural, desgraciadamente, no son todavía lo suficientemente robustos para funcionar apropiadamente en textos sin restricciones y generar representaciones semánticas apropiadas para el texto. Asi, la mayoría de las aproximaciones actuales todavía se basan en las representaciones más superficiales y basadas en las palabras, en especial en los métodos de bolsa de palabras, mucho más simples desde un punto de vista algorítmico que los métodos que usan cadenas de palabras. En la representación en bolsa de palabras, sin embargo, se pierde información acerca de la posición de las palabras. En ciertos ámbitos, como el biomédico, y ciertas tareas de minería, como la extracción de información de la web, las técnicas de procesamiento del lenguaje natural están jugando un papel muy importante a la hora de obtener una representación de los textos con un significado más semántico.

En el contexto de las aplicaciones web, los datos textuales están en constante aumento. En este ámbito, los datos aparecen frecuentemente junto con datos multimedia u otro tipo de datos. Por ello, recientemente se han desarrollado nuevas técnicas para la minería conjunta de diferentes tipos de datos. Por ejemplo, en la web encontramos datos textuales e imágenes, frecuentemente conectados unos con otros. Estas conexiones pueden aprovecharse para la mejora del proceso de aprendizaje.

Capítulo 2

La minería de datos

Para el desarrollo de este capítulo se han usado los artículos de Justicia [12] y Valencia [20].

En la actualidad, las empresas y compañías tratan con grandes volúmenes de datos que, sin duda, son una fuente fundamental de información para su desarrollo. Hoy en día es claro que la información es el motor principal de la empresa, y su dependencia, a distintos niveles, de esta información la convierten en un activo estratégico.

La inteligencia de negocios (*business intelligence*) propone la integración de modelos matemáticos y metodologías de análisis para la explotación sistemática de los datos disponibles con el fin de obtener información y conocimiento del negocio, y para el uso de esta información en la toma de decisiones. La minería de datos permite encontrar la información de interés y difícilmente perceptible de entre las enormes cantidades de datos disponibles, detectando relaciones no aparentes o difíciles de detectar sin recurrir a estas herramientas.

Inicialmente, el uso de los sistemas de información estaba orientado al soporte de los procesos básicos de las compañías: ventas, personal, etc. Surgieron posteriormente los sistemas de información orientados a la toma de decisiones, brindando una visión inmediata del estado y las actividades de gestión. El proceso de toma de decisiones a partir del análisis de los datos es el siguiente: en primer lugar, es necesaria la recopilación y el procesamiento de los datos, así como la integración de los mismos en las bases de datos pertinentes. A continuación, estaremos en condiciones de realizar una exploración de los datos para el conocimiento y la descripción de los mismos. Posteriormente, se lleva a cabo la minería de datos, descubriendo información nueva a partir de ellos. Por último, usando técnicas de visualización, se presentan los datos de forma cómoda para su comprensión y se lleva a cabo la toma de decisiones.

La minería de datos es, por tanto, una disciplina fundamental hoy día. Su uso no se limita al ámbito empresarial, sino que tiene aplicaciones tan variadas como la detección de fraude, el análisis de gases o el estudio de la genética humana. Vamos a proceder a definirla más rigurosamente. La minería de datos es el proceso de obtener conocimiento nuevo a partir de grandes cantidades de datos (Han and Kamber, 2000). Es un campo interdisciplinar con contribuciones de muchas áreas, como son la estadística, el

aprendizaje de máquinas, recuperación de la información, reconocimiento de patrones y bioinformática. El objetivo de la minería de datos, por tanto, es el descubrimiento de patrones, perfiles, anomalías y tendencias a través del análisis de los datos.

Las fases de la extracción de conocimiento a partir de bases de datos (*KDD, Knowledge Discovery from Databases*)* son las siguientes (Han and Kamber, 2000):

1. Limpieza de datos, en la que se elimina el ruido y los datos inconsistentes obtenidos de las fuentes externas usando métodos estadísticos.

2. Integración de datos, en la que se combinan las distintas fuentes de datos.

3. Selección de datos, pues los datos almacenados en la base de datos pueden no ser todos de interés.

4. Transformación de los datos a un formato apropiado para la minería, pudiéndose optar por distintos tipos de modelos.

5. Minería de datos, en la que se aplican las técnicas pertinentes, en función de los datos y de la información que se desea extraer, para obtener patrones de datos.

6. Evaluación de los patrones obtenidos en el proceso de minería de datos, identificando los que resultan interesantes y los que no.

7. Representación del conocimiento.

Las principales técnicas en este campo incluyen clasificación y predicción, clustering, detección de outliers, reglas de asociación, análisis secuencial, análisis de series temporales, redes neuronales, árboles de decisión, etc. También son frecuentes técnicas más novedosas como el análisis de redes sociales o el análisis de sentimientos.

Dada la enorme variabilidad de los datos disponibles y de los cuales podemos extraer información de interés, la minería de datos incluye muchas áreas diferentes en función del tipo de datos a analizar. Entre estas áreas se encuentra la minería de textos, en la que los datos son documentos de texto, partiendo por esta razón de información muy poco estructurada.

2.1. Diferencias entre la minería de datos y la minería de textos

La consideración de datos de tipo textual y no de otro tipo de datos estructurados acarrea una serie de problemas que hacen que, dentro de la minería de datos, la minería de textos requiera un tratamiento especial. Estos problemas son:

*Es frecuente identificar el concepto de minería de datos con el proceso completo de KDD: la fase principal identifica al proceso completo

- La falta de estructura de los textos, que carecen de una estructura homogénea procesable de manera automática, incluso en lenguajes semiestructurados como el HTML.
- La heterogeneidad de los documentos, debida a la diversidad de las fuentes externas de las que estos se obtienen, y que pueden ser censos, redes sociales, páginas web, bases de datos, informes empresariales, etc.
- Los diferentes idiomas en los que, con frecuencia, se encuentran los diferentes textos, no solo entre conjuntos de documentos, sino también dentro de una misma colección.
- La dependencia del contexto y del dominio de la aplicación a la hora de analizar el texto, lo que hace necesario el uso de diccionarios, tesaurus u ontologías específicas de dicho contexto.

Debido a todas estas características, aunque podemos situar la minería de textos dentro de la minería de datos, esta requiere de otros ámbitos de conocimiento como la recuperación de la información, la extracción de la información o el procesamiento del lenguaje natural.

KDD	**KDT**
Comprensión del dominio de la aplicación	Definición de los conceptos de interés
Selección de un conjunto de datos objetivo	Obtención de los textos mediantes técnicas de recuperación de la información o de forma manual
Limpieza, pre-procesamiento y transformación de los datos	Descripción de los conceptos y representación y transformación de los textos: eliminación de stopwords, palabras poco relevantes...
Desarrollo de modelos e hipótesis	Identificación de los conceptos de interés en nuestra colección de textos
Minería de datos	Minería de textos
Interpretación y visualización de los resultados	Interpretación de los resultados por un humano

Cuadro 2.1: Comparación de los procesos de KDD y KDT

En general, las fases en un proceso de descubrimiento de conocimiento en textos (*KDT, Knowledge Discovery in Text*) [**] son las mismas que las del descubrimiento de

[**] De nuevo, es frecuente identificar el concepto de minería de textos con el proceso completo de KDT

conocimiento en datos, sin embargo, las características especiales de la minería de textos exigen la revisión del proceso: la falta de estructura hace que las etapas de selección, preprocesamiento y transformación de los datos sean imprescindibles. A pesar de que en capítulos posteriores se desarrolla más detenidamente este proceso, se incluye el cuadro 2.1 (Paralic, 2001) en el que se comparan el proceso de extracción de conocimiento de los datos y de los textos.

Capítulo 3

Historia de la minería de textos

En el desarrollo de este capítulo se ha recurrido al libro de Miner et al. [14].

Los avances tempranos en el campo de la minería de textos fueron motivados por la necesidad de catalogar documentos de texto. Sin embargo, pronto la investigación se centró en la extracción de datos de los textos mediante técnicas de procesamiento del lenguaje natural (NLP). Hoy en día, la necesidad de extraer información de los textos continúa motivando externamente el desarrollo en este campo. Por otro lado, la voluntad de desarrollar una "máquina inteligente" lo motiva internamente: la capacidad de entender y obtener conocimiento de un texto es clave en las investigaciones enfocadas al desarrollo de la inteligencia artificial para crear máquinas que simulen lo más fielmente posible la forma de pensar del cerebro humano.

En 2001, Internet daba acceso a aproximadamente 10 millones de páginas web, lo que supone alrededor de 100 terabytes. En 2009, ese volumen de datos había ascendido a 150 billones de páginas web, aproximadamente 1500 terabytes, lo que supone una tasa de crecimiento del 40 % anual más o menos. A pesar de ello, no se incluyen entre estas las páginas web corporativas ocultas por los firewalls. En Internet podemos encontrar textos de muy diversa índole (páginas personales, páginas corporativas públicas, de noticias, libros electrónicos o tesis doctorales). No es de extrañar, por tanto, que uno de los sectores dentro de la alta tecnología con mayor crecimiento sea el desarrollo de los motores de búsqueda. Por supuesto, este crecimiento no es lineal, y la tasa actual puede no mantenerse, pero la acumulación de textos existente en forma de páginas web supone un ejemplo de la velocidad con la que estamos acumulando datos en forma de texto. Actualmente, factores determinantes para esta acumulación son la creciente industria de los e-books y la conversión de obras en papel a formato electrónico.

En el origen de la minería de textos, la investigación estaba motivada por el creciente volumen de información textual que se estaba generando en el mundo. El reto más importante, por lo tanto, era el acceso a una información concreta que estaba enterrada entre la enorme acumulación de documentos de texto. Así, las actividades que se desarrollaban en este campo consistían básicamente en resumir la información de la que se disponía para reducir el cuerpo de texto a un tamaño más manejable, y en la posterior clasificación de forma lógica de esa información resumida. Los avances en este sentido se

realizaron básicamente desde tres áreas: las ciencias de la información, el procesamiento del lenguaje natural, y los estudios para resumir y clasificar textos en las bibliotecas.

3.1. La minería de textos moderna

Mientras que inicialmente las actividades se centraban en la recuperación y el resumen de la información, los avances posteriores se enfocaron en la extracción de la información, una serie de pasos ordenados diseñados para extraer términos, atributos de los términos, datos, y eventos (Sanger y Feldman, 2007). En este sentido se han desarrollado poderosos métodos desde el año 2000, sin embargo, podemos considerar que llevan realizándose actividades de extracción de información desde 1987 con las Message Understanding Conferences (MUCs). Las primeras MUCs se llevaron a cabo por el Naval Ocean Systems Center (NOSC) y el Defense Advanced Research Projects Agency (DARRPA) en 1987 y 1988 para analizar mensajes militares, en concreto los procedentes de las operaciones de la flota naval. Las MUCs han tenido una gran influencia, que se ve reflejada en los muchos conceptos procedentes de estas conferencias que se han integrado posteriormente en la extracción de información en minería de textos.

Otro aspecto fundamental en el progreso de la minería de textos ha sido, sin duda, el desarrollo de la minería web (*web scraping*). En el año 1997, en una presentación de la *International Conference on Tools with Artificial Intelligence*, Rob Cooley se preguntó si habría algún beneficio en la aplicación de la minería de textos y la inteligencia artificial a la web. El mismo Cooley continuó formalizando esta reflexión y se centró en la búsqueda de patrones de información en la web, comenzando así la investigación sobre las aplicaciones prácticas de la minería web. Por otra parte, podemos resaltar el trabajo de Jesús Mena en 1999 que propuso un gran número de posibles aplicaciones de la minería web. También en 1997, Menczer introdujo el concepto de rastreo web. El rastreo web es una importante aplicación de la minería web que consiste en la búsqueda de las páginas web relevantes de entre todas las existentes en Internet.

3.2. Avances en la minería de textos desde el año 2000

La minería de textos ha avanzado notablemente desde los años 90, desarrollándose nuevas técnicas a partir de las diferentes áreas de la minería de datos, la estadística y la teoría de aprendizaje estadistico, y con ayuda de los conocimientos previos de los que hemos hablado. Estos desarrollos han llevado a redefinir las tareas que conforman la extracción de información hoy en día: tokenización, análisis morfológico y sintáctico, análisis semántico y análisis del dominio.

Capítulo 4

Aplicaciones de la minería de textos

En el desarrollo de este capítulo se ha recurrido al artículo de Fernández et al. [9] y a los libros de Biemann y Mehler [5] y Aggarwal y Zhai [2].

4.1. Análisis de sentimientos

El análisis de sentimientos, cuyo objetivo es conocer la actitud de cierto grupo de personas hacia algún producto, evento, personaje, etc., ha crecido en popularidad en los últimos 15 años por varias razones: el auge de las redes sociales, los avances tecnológicos y el desarrollo de las herramientas de procesamiento del lenguaje natural, que han llevado a centrar la atención en problemas más complicados, y así más semánticos.

Uno de los problemas más prometedores en procesamiento del lenguaje natural es el análisis de críticas de productos. El objetivo es detectar de forma automática el feedback del cliente para un producto, lo que supone una gran ayuda a las empresas a la hora de desarrollar campañas de publicidad para dicho producto, así como proceder a su mejora. Algunas de las preguntas a las que se enfrenta el análisis son: ¿quién compra el producto? ¿Qué otros productos compran estas personas? ¿Qué les atrae de estos productos? ¿Con qué frecuencia los usan?

Además de el análisis del feedback, el análisis de sentimientos está presente en otros campos como son las ciencias sociales. Por ejemplo, algunos investigadores están interesados en la recabación automática de puntos de vista en cuestiones políticas a través de periódicos o redes sociales.

Para llevar a cabo un análisis de sentimientos es necesario disponer de un *corpus anotado*, esto es, un corpus en el que se han introducido cierta información extra, en este caso relativa a actitudes subjetivas no observables. Para ello, existen corpus anotados con sentimientos previamente construídos, como el *MPQA*, el más importante en inglés o el *MLSA*, en alemán. En español, disponemos del corpus *EmotiBlog Kyoto*, que posee dos partes, una compuesta por opiniones sobre teléfonos móviles extraídas de Amazon, y otra por entradas de blogs en inglés sobre el Protocolo de Kyoto. El modelo de anotación de

este corpus incluye la anotación a nivel de de término, frase y documento, distinguiendo entre objetivos y subjetivos. Cada elemento se etiqueta en términos de *polaridad*, *grado (o intensidad)* y *emoción*.

4.2. Aplicación al análisis de respuestas a preguntas abiertas en encuestas

En una encuesta, podemos encontrar preguntas de respuesta abierta o libre, y preguntas de respuesta cerrada. Son las primeras las que nos interesan en cuanto a la aplicación de las técnicas en minería de textos, pues normalmente se entrevista a un gran número de personas (de varios centerares hasta varios miles), haciendo necesaria la aplicación de estos métodos para el análisis de los resultados.

Las preguntas de respuesta libre son frecuentemente usadas en encuestas de satisfacción de clientes, estudios de preferencia de los consumidores o encuestas de opinión, entre otros. En muchos de los casos, el uso de este tipo de preguntas es necesario, pues se aporta una información específica que no podría ser captada por preguntas cerradas. Además, esta forma de preguntar condiciona menos al encuestado, evita respuestas convencionales y arroja información más fiable acerca de las opiniones reales.

Por otro lado, la codificación manual de las respuestas supondría unas limitaciones importantes, ya sea relativas al volumen de los datos recopilados, o debidas a la subjetividad del codificador, la mutilación de la forma y de la calidad de la expresión, el empobrecimiento del contenido, o la eliminación de las respuestas poco frecuentes. La aplicación de diferentes técnicas de minería de textos para extraer la información contenida en las respuestas, sin embargo, acaba con esta problemática: se opera a partir de las respuestas originales, sin efectuar reducciones a priori, conservando así la diversidad del discurso. Además, estos métodos facilitan la conexión entre respuestas abiertas y cerradas.

4.3. Análisis de textos históricos

La reciente digitalización de recursos de todo tipo ha facilitado el acceso a los registros históricos. Por otro lado, los avances en lingüística computacional han proporcionado métodos para la extracción de información de recursos históricos no estructurados o semi-estructurados, asi como para la identificación automática de elementos recurrentes en los textos. Estas circustancias han dado lugar a nuevas posibilidades de exploración de estos recursos mediante el uso de ordenadores, existiendo, por ejemplo, métodos capaces de identificar nombres de personas, lugares, o expresiones temporales.

Actualmente, este tipo de métodos son muy usuales en la investigación histórica. El análisis semi-automático de los recursos disponibles puede llevar al descubrimiento de forma casual de conexiones entre eventos que eran desconocidas e insospechadas hasta el momento. En este sentido, varios proyectos se han planteado el objetivo de modelar eventos históricos, centrándose en aspectos tales como anotaciones y proponiendo

interpretaciones alternativas para los eventos.

Algunos expertos en el campo de las humanidades reconocen que estos métodos no pueden competir con la lectura de los textos por parte de expertos en este dominio, es decir, con la manera habitual de investigación en este campo, debido a la imprecisión inherente a estos métodos computacionales. Sin embargo, es innegable la utilidad de estas técnicas en el procesamiendo de grandes cantidades de texto y la detección de patrones significativos en ellos. Por ejemplo: la búsqueda mediante palabras clave es una forma muy potente a la vez que muy simple de procesar grandes cantidades de textos; sin embargo, requiere que los investigadores sepan lo que están buscando de forma precisa.

Por un lado, los historiadores requieren resultados con gran precisión. Por otro lado, sus capacidades para procesar un gran número de textos mediante su lectura es limitada. Podemos decir, por tanto, que la opción más eficiente es la integración de la evaluación humana y el procesamiento de la información mediante técnicas de análisis de textos.

4.4. Realización de resúmenes

Una de las tareas demandadas más comunmente es la realización de resúmenes de grandes documentos de texto o de un conjunto de documentos con el objetivo de obtener una vista general de estos. Las técnicas usadas para ello son de dos tipos: resúmenes extractivos y resúmenes abstractivos. Los resúmenes extractivos están compuestos por unidades de información extraídas del texto original. Por el contrario, los resúmenes abstractivos pueden contener unidades de información sintetizada que puede no aparecer en el documento original.

Los sistemas de realización de resúmenes identifican las frases más importantes del documento o el conjunto de documentos, y posteriormente las encadenan para formar un resumen. En este proceso podemos distinguir tres pasos. En primer lugar, se crea una representación intermedia del texto original en el que se reflejen solo los aspectos claves del texto. A continuación, se puntúan según importancia las frases del documento en base a esta representación. Por último, se construye el resumen a partir de varias de estas frases.

4.5. Filtrado de emails

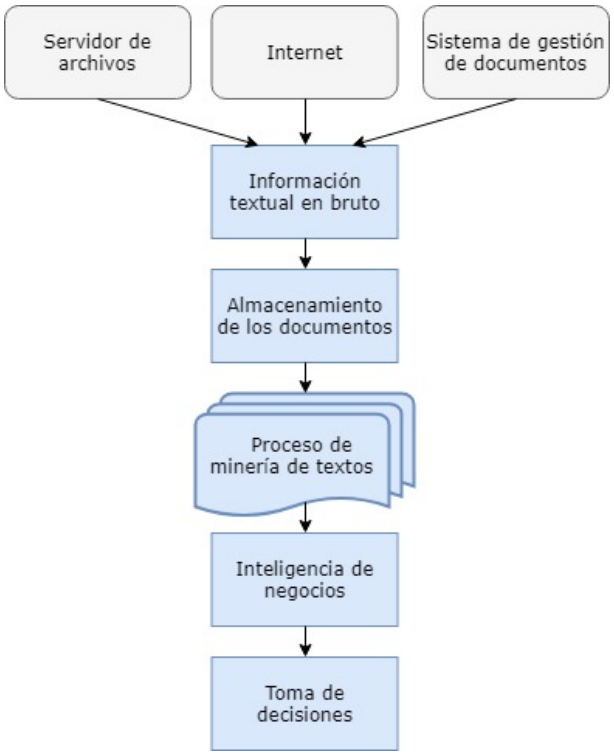
Gracias a las técnicas de la minería de textos se han desarrollado herramientras útiles y configurables para organizar en carpetas la enorme cantidad de correos electrónicos que un usuario recibe a diario: cada mensaje se tokeniza con las 15 palabras más interesantes, esto es, con menos probabilidad de ser *spam*. A partir de esta información, el clasificador decide si el correo debe ser o no clasificado como *spam* (Wiss et al., 2010).

Capítulo 5

Etapas de la minería de textos

Para el desarrollo de este capítulo se ha recurrido a los textos de Weiss et al. [17] y Billheimer et al. [8].

En la vida real, las empresas recurren a la minería de textos con el objetivo de tomar las decisiones más adecuadas. En este proceso de toma de decisiones en el ámbito de la empresa, podemos distinguir una serie de etapas y situar dentro de ellas el proceso de minería de textos que desarrollaremos a continuación:



En un proceso general de minería de textos, en el que vamos a transformar la in-

formación textual en vectores numéricos, podemos diferenciar varias etapas: recopilación de los textos, estandarización del formato, tokenización, eliminación de palabras vacías, normalización, lematización y stemming, construcción del diccionario y, por último, construcción del espacio vectorial. Una vez se han llevado a cabo estos pasos estamos en condiciones de aplicar las técnicas de minería de textos que veremos más adelante y que nos permitiran obtener la información deseada de nuestros documentos de texto. Veamos más detenidamente en qué consiste cada una de ellas.

5.1. Recopilación de los textos

Es claro que el primer paso necesario es la recopilación de los documentos de textos que vamos a usar. En muchos de los casos, los documentos relevantes van a venir dados o, más aún, determinados por el problema. Sin embargo, esto no siempre es así. Una vez los documentos que nos interesan han sido identificados y tenemos acceso a ellos, puede ser necesario limpiarlos y asegurarnos de que son de calidad, en el caso, por ejemplo, de las páginas web. Esta tarea debe desarrollarse siempre con extremo cuidado, sin comprometer la integridad del documento con nuestra intervención. En caso de que los documentos se extraigan directamente de bases de datos, podemos asumir que son de calidad y que no es necesaria su limpieza.

En algunos casos, el conjunto de documentos de interés puede ser extremadamente grande, pudiendo ser conveniente el uso de técnicas de muestreo para seleccionar un conjunto manejable de documentos relevantes. Las técnicas para seleccionar unos documentos u otros pueden variar según el interés. Por ejemplo, si estos contienen información relativa al momento de su composición, podría ser útil seleccionar los documentos más recientes.

5.2. Conversión del formato

Una vez tenemos a nuestra disposición los documentos de texto que vamos a considerar, el siguiente pasó será la conversión de todos ellos a un formato común y que nos sea de fácil manejo para manipularlo en las siguientes etapas. Además, puede ser oportuna la diferenciación de un solo documento en más de uno, o la consideración de varios textos como uno solo. Conseguimos así nuestro *corpus*, es decir la colección de documentos de texto que vamos a manejar.

5.3. Tokenización

La tokenización es el proceso de dividir un documento de texto (o una colección de ellos) en la lista de palabras que lo conforman, mediante la identificación de, por ejemplo, los espacios en blanco o los signos de puntuación. Este paso es fundamental en el análisis posterior dado que vamos a usar modelos de espacio vectorial, en los que las palabras son los elementos básicos.

Este proceso puede resultar trivial para una persona que esté familiarizada con el idioma del texto. Por el contrario, esta tarea puede no ser tan sencilla para un ordenador, debido a que existen caracteres que son algunas veces delimitadores de palabras y otras veces no, e incluso que pueden ser considerados palabras o no, dependiendo de la situación. Por ejemplo, una coma entre dos números no debe ser considerada un delimitador, sino parte del número, pero en cualquier otra situación si lo será. Algo similar ocurre con los puntos, que pueden formar parte, por ejemplo, de una abreviación, o actuar como un signo de puntuación y, por tanto, ser un delimitador.

Resulta obvio que la tokenización es un procedimiento dependiente del idioma en el que estén escritos los textos, de forma que, aunque los principios generales son los mismos para cualquier idioma, algunos detalles variarán. Además, es conveniente adaptar el proceso al texto que estemos manejando para así reducir un posible trabajo posterior.

5.4. Eliminación de palabras vacías

Esta etapa consiste en la eliminación de las palabras que no tienen interés para la minería de textos, como son artículos, pronombres, preposiciones, conjunciones, e incluso palabras que se usan muy frecuentemente y no ayudan a la diferenciación de un documento a otro. Esta eliminación no supone una pérdida de información debido a que no contribuyen significativamente al tema del texto. Podría considerarse también oportuna la eliminación de otro tipo de palabras como adjetivos comunes, verbos comunes, o palabras que aparecen en un solo documento, conocidas como *ermitaños*. Existen listas de palabras vacías que podemos usar en esta etapa, aunque podemos optar también por crearla nosotros mismos para adecuarla más al contexto en el que nos movemos. Con este procedimiento conseguimos reducir notablemente la dimensión del espacio vectorial que construiremos más adelante.

5.5. Normalización

La normalización consiste en asignar una representación única a cada palabra, con independencia de las diferentes secuencias de caracteres empleadas en el texto para referirse a ella. Por ejemplo, los términos *Instituto Nacional de Estadística*, *INE* e *I.N.E.* se refieren a lo mismo, y deberían ser considerados como tal.

5.6. Lematización y stemming

Una vez se ha finalizado la tokenización, el siguiente paso es la conversión de cada una de las palabras a una forma estándar. En estas etapas, se presupone que el significado de las palabras no está influído por la forma gramatical que presente. El stemming es el proceso de eliminar heurísticamente partes de una palabra para reducirla a una forma común. Por su parte, la lematización se refiere a un proceso más complejo que emplea un análisis sintáctico, además análisis morfológico de las estructuras. Por tanto, uno de

los efectos de estas etapas es reducir el número de palabras que aparecerán en nuestro diccionario, identificando, por ejemplo, palabras como *diré* o *dijéramos* (ambas formas del verbo *decir*), aumentando a la vez la frecuencia con la que aparece cada una.

En cuanto al stemming, podemos optar por dos procedimientos diferentes: stemming flexional y stemming a la raíz. El stemming flexional es aquel en el que la normalización de las palabras consiste en la regularización de sus variaciones gramaticales tales como singular y plural o presente y pasado. Usando terminología lingüística, estamos realizando un análisis morfológico. El stemming a la raíz, por su parte, es un procedimiento más agresivo que el anterior. El objetivo es reducir las palabras a su raíz, prescindiendo de sufijos y prefijos flexivos o derivativos. De esta forma, la reducción a un número menor de palabras es mucho mayor.

Tras reducir las palabras a su raíz, en el caso de haber optado por el segundo tipo de stemming, podemos completar cada raiz obtenida para conseguir una palabra real. En este caso, se optará normalmente por la palabra correspondiente más frecuente en el diccionario. En este proceso final pueden surgir algunos problemas, por ejemplo, al completar la raíz de una palabra de forma automática, la palabra resultante podría no ser de la misma familia semántica que la orgininal, pese a poseer la misma raíz léxica. Por esta razón, es conveniente la revisión de las palabras obtenidas por si fuera necesario la sustitución de alguna de ellas.

5.7. Construcción de un diccionario

El diccionario, que llamaremos V, puede ser el resultado del stemming, y estar formado por el conjunto de las palabras resultantes tras este proceso, o venir dado por el problema.

5.8. Construcción de un espacio vectorial

En este sentido podemos optar por diferentes modelos según cuáles sean nuestros objetivos. En cualquier caso, la dimensión del espacio vectorial será el tamaño de nuestro diccionario, es decir, $|V|$. Dado el bag of words de un documento, le asociamos un vector en el que cada elemento representa un término y su peso en el documento, que puede quedar representado de varias formas, dando lugar a tres modelos de espacio vectorial:

- Modelo booleano: Los elementos de los vectores son 1 y 0, en función de si el término que representa cada elemento aparece o no en el documento. El espacio vectorial, por tanto, es $\{0,1\}^{|V|}$.
- Modelo de valores reales: Los vectores están formados por números reales que representan el número de veces que aparece cada elemento en el documento. El espacio vectorial resultante es $\mathbb{R}^{|V|}$.
- Modelo TF-IDF: Los elementos de los vectores se definen de acuerdo a la frecuencia de los términos en el propio documento y en el resto de los documentos, mediante

el método TF-IDF (*Term frequency - Inverse document frequency*). Según este método, si un término aparece más veces en un documento, tiene más peso. A su vez, si aparece en más documentos, tiene menos peso. De esta forma, sea t un término de nuestro diccionario, d el documento, $tf_{t,d}$ la frecuencia de t en el documento d y df_f el número de documentos en que aparece t, el peso de este término en el vector será:

$$tf - idf_{t,d} = tf_{t,d} \quad idf_t$$
$$idf_t = log\frac{N}{df_t}$$

De nuevo, el espacio vectorial es $\mathbb{R}^{|V|}$.

La definición de un espacio vectorial nos permite considerar distancias entre vectores. Cada documento va a quedar representado como un vector de espacio. Podemos recurrir a diferentes definiciones de distancia, como son las *distancias basadas en p-normas*, que estamos acostumbrados a usar, o normas más específicas como, por ejemplo, la *distancia similaridad coseno*, que define la distancia entre dos vectores como:

$$sim(d_i, d_j) = \frac{d_i d_j}{|d_i||d_j|}$$

El modelo de espacio vectorial, así como la consideración de una distancia dentro de este espacio, posibilita el uso de técnicas como las que veremos a continuación.

Capítulo 6

Técnicas en minería de textos

6.1. Análisis de la semántica latente

Esta sección se ha desarrollado usando el texto de Jorge-Botana [11].

El LSA (*Latent Semantic Analysis*) es una herramienta matemática que analiza relaciones semánticas entre diferentes unidades lingüísticas de forma completamente automatizada (Landauer y Dumais, 1997). Fue originariamente presentado en 1990 como un método de recuperación de la información, para solventar las grandes limitaciones que presentaban los motores de búsqueda en bases de datos, aunque posteriormente se ha considerado también un modelo de adquisición y representación del conocimiento. De esta forma, ha sido empleado en multitud de ámbitos: corrección de textos, medida de cohesión y coherencia de los textos, así como en la simulación de la adquisición del lenguaje del ser humano.

El funcionamiento de esta técnica es el siguiente. Para empezar, el LSA procesa los documentos que estamos manejando, que pueden ser de grandes dimensiones, conteniendo gran cantidad de párrafos y, en definitiva, de información, dando lugar al *corpus lingüístico*. A continuación, este corpus se representa mediante una matriz cuyas filas y columnas contienen, respectivamente, los diferentes términos que aparecen en el corpus y los documentos que estamos considerando. En esta matriz queda reflejado el número de veces que aparece cada término en cada documento.

Puesto que podemos asumir que las palabras excesivamente frecuentes no sirven para discriminar la información de cada documento, el LSA efectúa una ponderación en la que resta importancia a las palabras notablemente más frecuentes y aumenta la de palabras moderadamente infrecuentes. Seguidamente, aplica a la matriz obtenida el *algoritmo de descomposición en valores singulares (SVD)* con el objetivo de reducir la dimensión de la matriz con la que estamos trabajando a un número más manejable (aproximadamente 300), y sin perder información relevante. El objetivo de aplicar el SVD es ponderar cada término en función de su capacidad para representar un documento. Además, mediante este algoritmo, se habrá creado el espacio vectorial con el que vamos a trabajar en lo que sigue y con todas las ventajas que supone trabajar con un espacio vectorial (por ejemplo, la comparación de los vectores usando distancias).

Mientras que previamente a aplicar el algoritmo los vectores eran vectores huecos, los nuevos vectores no lo serán en general. Esto hace posible la detección de relaciones significativas entre pares de documentos, incluso aunque dichos documentos no tuvieran términos en común. La idea es que los términos que poseen un significado similar estarán orientados en aproximadamente la misma dirección en el espacio latente.

Por otro lado, el LSA presenta la posibilidad de introducir en el espacio nuevos vectores que representen textos que no aparecen en el corpus lingüístico que el LSA había analizado, y que llamamos *pseudodocumentos* (Landauer, Foltz y Laham, 1998). La incorporación de los pseudodocumentos no requiere recalcular el espacio vectorial, lo que supone una gran ventaja. Estos pseudodocumentos se usarán posteriormente para llevar categorizar los términos y los textos.

6.1.1. El algoritmo SVD

En este apartado vamos a estudiar con más detenimiento en qué consiste el algoritmo de descomposición en valores singulares. El SVD es una forma específica de análisis factorial. En la matriz de partida, términos y documentos son mutuamente dependientes entre ellos, mientras que, tras aplicar el algoritmo, estas relaciones aparecen desglosadas. El SDV descompone la matriz original en el producto de tres nuevas matrices T, D y S, que contienen los autovectores y los autovalores. Los autovalores, a su vez, contienen la información de la variabilidad, en cuanto a términos y documentos, explicada por cada dimensión.

La matriz T contiene la información de los factores que se han determinado mediante el análisis efectuado. Es decir, aporta la información relativa a los términos a partir de un número menor de dimensiones representada por los factores. Análogamente, la matriz D contiene los factores que representan la información relativa a los documentos, con una dimensión menor.

Por último, la matriz S es una matriz diagonal que contiene los autovalores ordenados de mayor a menor. A mayor sea un autovalor, mayor será la aporrtación para agrupar.

$$X = TSD'$$

En este proceso se requiere cierta pérdida de información pero de forma irrelevante. De hecho, la pérdida de información es, en realidad, una reducción del ruido que provocaban los factores que no eran del todo relevantes en las relaciones entre términos y documentos, y que han sido eliminados. De esta forma, al obtener la nueva matriz X, la información que esta nos proporciona será más clara que en la matriz original.

6.2. Análisis probabilístico de la semántica latente

Para el desarrollo de esta sección se ha usado el artículo de Hofmann [10].

Mientras que el LSA se desarrolla a partir de conceptos de álgebra lineal, el punto de partida de el PLSA (*Probabilistic Latent Semantic Analysis*) es un modelo estadístico llamado el *modelo de aspecto*. Este modelo fue propuesto por Saul y Peveira en 1997.

Es un modelo de variable latente para datos co-ocurrentes que asocia una variable no observable $z_k \in \{z_1, \ldots, z_K\}$ a cada observación, siendo cada observación la ocurrencia de un término en un documento. Llamaremos $P(d_i)$ a la probabilidad de que un término aparezca en un documento d_i, $P(w_j|z_k)$ a la probabilidad de un término w_j condicionada a la variable z_k, y $P(z_k|d_i)$ a la distribución de probabilidad de un documento d_i en el espacio latente de las variables $z_1, \ldots, z_K$.

A partir de estas definiciones, podemos generar las co-ocurrencias de términos-documentos siguiendo el segundo esquema:

1. seleccionamos un documento d_i con probabilidad $P(d_i)$,
2. tomamos un z_k con probabilidad $P(z_k|d_i)$,
3. generar un término w_j con probabilidad $P(w_j|z_k)$.

Como resultado, obtenemos un par (d_i, w_j) y la variable latente z_k queda descartada. Así, el modelo de probabilidad que se obtiene queda como sigue:

$$P(d_i, w_j) = P(d_i)P(w_j|d_i), \quad p(w_j|d_i) = \sum_{k=1}^{K} P(w_j|z_k)P(z_k|d_i)$$

Es posible una parametrización equivalente para el modelo, en que la probabilidad conjunta resulta perfectamente simétrica para documentos y términos:

$$P(d_i, w_j) = \sum_{k=1}^{K} P(z_k)P(d_i|z_k)P(w_j|z_k)$$

6.2.1. Relación entre el LSA y el PLSA

Para entender mejor la relación entre ambas técnicas, podemos reescribir el modelo parametrizado de la última forma con notación matricial. De esta forma, definimos las matrices $U = (P(d_i|z_k))_{i,k}$, $V = (P(w_j|z_k))_{j,k}$ y $\sigma = diag(P(z_k))_k$. El modelo de probabilidad conjunta, que llamaremos P, quedaría, por tanto, escrito como el producto de matrices $P = U\sigma V^t$. Comparando esta descomposición con la resultante en el LSA del algoritmo de descomposición SVD, podemos hacer las siguientes reinterpretaciones:

1. La suma ponderada de los productos entre las filas de U y de V que se obtienen al calcular P, reflejan la independencia condicional en el PLSA.
2. Los K factores que se obtienen en el LSA, parecen corresponder con los z_k del PLSA.
3. Los $P(z_k)$ del PLSA sustituyen a los valores singulares del SVD en el LSA.

El PLSA ofrece ciertas ventajas frente al LSA. Por ejemplo, P es una distribución de probabilidad bien definida y los factores tienen un claro significado probabilístico, mientras que en el LSA no se define una distribución de probabilidad. Por otro lado, no existe una interpretación evidente de las direcciones del espacio latente del LSA, mientras que las direcciones del espacio latente del PLSA son interpretables como distribuciones condicionales de términos que definen un contexto temático. En términos de complejidad computacional, por el contrario, el LSA puede resultar más ventajoso.

6.3. Asignación latente de Dirichlet

En el desarrollo de esta sección usamos el artículo de Blei et al. [6].

A pesar de que el PLSA proporciona una buena base para el análisis de los textos, presenta algunos problemas: por un lado, contiene un gran número de parámetros, que además crece linealmente con el número de documentos, y por otro lado, no existe una forma de calcular la probabilidad de un documento que no estuviera en los datos estudiados. El LDA (*Latent Dirichlet Allocation*), sin embargo, reduce considerablemente el número de parámetros a considerar y permite de forma clara el cálculo de la probabilidad para documentos arbitrarios.

Una vez más, en este método, las unidades básicas de datos son las palabras, que conforman el vocabulario y que indexamos por $\{1, \ldots, V\}$. Las palabras quedan representadas mediante vectores unitarios de dimensión V con una única componente igual a 1, y las demás iguales a 0. Los documentos, por su parte, quedan representados por la secuencia de palabras que lo componen, es decir, por un vector $d = (w_1, \ldots, w_N)$, donde w_n es la n-ésima palabra del documento. Por último, el corpus se denota como $D = \{d_1, \ldots, d_M\}$. El objetivo es encontrar un modelo probabilístico que no solo asigne una probabilidad alta a los componentes del corpus, sino que también lo haga para documentos similares a los del propio corpus.

El LDA es un modelo probabilístico generativo de un corpus. La idea general es que los documentos están representados por una mezcla aleatoria de temáticas latentes, donde cada temática está caracterizada por una distribución de las palabras. Este modelo asume el siguiente proceso generativo para documento d del corpus D:

1. Elegir un $N \sim Poisson(\xi)$.
2. Elegir un $\theta \sim Dir(\alpha)$.
3. Para cada una de las N palabras w_n:
   a) Elegir una temática $z_n \sim Multinomial(\theta)$.
   b) Elegir una palabra w_n de $p(w_n|z_n, \beta)$, una probabilidad multinomial condicionada a la temática z_n.

La dimensionalidad k de la distribución de Dirichlet, y por tanto la de la variable temática z, se supone conocida y fijada. Por otro lado, las probabilidades de las palabras

están parametrizadas por una matriz β de dimensión $k \times V$, en la que $(\beta_{ij}) = p(w^j = 1|z^i = 1)$, cantidad que debe ser estimada y que consideramos ya fijada. Por último, N es independiente de las demás variables generadoras de datos θ y z, siendo, por tanto, una variable auxiliar, lo que nos permite ignorar su aleatoriedad en lo que sigue.

Una variable aleatoria k-dimensional de Dirichlet θ puede tomar valores en el $(k-1)$-simplex, esto es, $\theta_i \geq 0$ y $\sum_{i=1}^{k} \theta_i = 1$, y tiene la siguiente función de densidad en el simplex:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}$$

donde el parámetro α es un vector de dimensión k con componentes $\alpha_i > 0$, y donde $\Gamma(x)$ es la función Gamma.

Dados los parámetros α y β, la distribución conjunta de una mezcla de temáticas θ, un conjunto de N temáticas z, y un conjunto de N palabras dado d, está dada por:

$$p(\theta, z, d|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$

donde $p(z_n|\theta)$ es simplemente θ_i para el único i tal que la i-ésima componente de z_n es 1. Integrando sobre θ y sumando en z obtenemos la distribución marginal de un documento:

$$p(d|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta$$

Por último, tomando el producto de las distribuciones marginales de los documentos por separado, obtenemos la probabilidad del corpus:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_n} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \right) d\theta_d$$

Existen tres niveles en la representación del LDA: los parámetros α y β son parámetros a nivel de corpus, las θ_d son variables a nivel de documento, y las variables z_{dn} y w_{dn}, lo son a nivel de palabra.

Es importante distinguir entre el LDA y un simple modelo de clustering multinomial-Dirichlet. Un modelo de clustering clásico asociaría cada documento a una única temática, mientras que el LDA permite relacionar cada documento con varias temáticas.

6.3.1. Relación del LDA con otros métodos

Si consideramos un modelo unigrama, las palabras de cada documento se toman de forma independiente de una distribución multinomial, de forma que:

$$p(d) = \prod_{n=1}^{N} p(w_n)$$

Si queremos hacer el modelo un poco más complejo, podemos añadir una variable temática z aleatoria discreta, obteniendo un *modelo de mezcla de unigramas*. Bajo este modelo de mezcla, cada documento es generado mediante la elección de una temática z y la generación posterior de N palabras independientemente de la distribución condicional multinomial $p(w|z)$. La probabilidad del documento, por tanto, sería:

$$p(d) = \sum_{z} p(z) \prod_{n=1}^{N} p(w_n|z)$$

Cuando las estimamos a partir de un corpus, las distribuciones de las palabras pueden considerarse representaciones de temáticas, suponiendo que cada documento posee una sola temática. Sin embargo esta suposición puede resultar muy limitante a la hora de modelar de forma efectiva una colección grande de documentos.

Por el contrario, el modelo LDA permite que los documentos incluyan varias temáticas en diferentes grados, lo que consigue simplemente añadiendo un parámetro adicional, sin suponer esto un gran coste. Mientras que en el modelo de mezcla de unigramas hay $k-1$ parámetros asociados con $p(z)$, en el LDA tenemos k parámetros asociados con $p(\theta|\alpha)$.

El PLSA, por su parte, considera que un documento d y una palabras w_n son independientes condicionalmente, dada una temática no observable z:

$$p(d, w_n) = P(d) \sum_{z} p(w_n|z)\, p(z|d)$$

El PLSA intenta relajar la suposición hecha en el modelo de mezcla de unigramas de que cada documento es generado a partir de una única temática. Por el contrario, considera que un documento puede contener varias temáticas. En este sentido, $p(z|d)$ actúa como el peso de la temática z en el documento d. Sin embargo, hay que destacar que d es un índice ficticio en la lista de documentos del corpus. Así, d es una variable aleatoria multinomial, que puede tomar tantos valores como documentos haya en el corpus, y el modelo nos da $p(z|d)$ solo para aquellos documentos estudiados. Por esto, el PLSA no es un modelo generativo de documentos bien definido, ya que no existe una forma natural de asignar una probabilidad a un documento que antes no haya sido considerado.

Otra dificultad relacionada con el PLSA es que el número de parámetros a estimar crece linealmente con el número de documentos. Los parámetros para un modelo con k temáticas en este método son k distribuciiones multinomiales de tamaño V y M mezclas de las k temáticas, es decir, tenemos $kV + kM$ parámetros, y el consecuente crecimiento lineal en M. Este crecimiento lineal sugiere que el modelo es propenso a sobreajustarse, lo que supone un gran problema que podemos intentar solucionar pero no siempre será posible.

El LDA supera ambos problemas considerando los pesos de la mezcla de temáticas como una variable aleatoria oculta con k parámetros en lugar de considerarla como un conjunto de parámetros individuales que están conectados de forma explícita con el conjunto de documentos. Como hemos visto, el LDA es un modelo generativo bien

definido y fácil de generalizar para nuevos documentos. Además, los $k + kV$ parámetros del LDA no crecen linealmente con el número de documentos.

6.4. Modelos de espacio vectorial semántico

En el contenido de esta sección se ha utilizado el artículo de Liu [13].

La idea subyacente de los modelos de espacio vectorial es que dos vectores del espacio vectorial, que representarán a un documento cada uno, estarán más próximos si son semánticamente similares, mientras que si están muy alejados, serán muy diferentes en este sentido.

En el modelo de espacio vectorial semántico, los documentos y las consultas se representan mediante vectores. Así, un mecanismo de búsqueda consiste en calcular la semejanza entre vectores para establecer la relevancia entre los documentos y las consultas. Veamos cómo funciona el modelo. Para cada frase de un documento d, tras realizar un análisis sintáctico, se asigna un peso, que llamaremos *peso de caso*, a cada palabra según la importancia semántica de esta en la frase para cada categoría temática que estemos teniendo en cuenta. Por otro lado, tras llevar a cabo las etapas previas tales como el stemming, eliminación de palabras vacías, etc., se asigna un valor de importancia a cada término en función de la frecuencia con la que aparezca en el documento en concreto que se esté considerando, y en el conjunto de documentos que forman el corpus. Los términos con valores de importancia mayores o iguales a un valor de referencia fijado previamente se usan como términos de indexación, mientras que los demás pueden ser o no descartados, dependiendo de nuestros intereses. A cada palabra se le asignan tantos pesos de caso como veces aparezca en el documento, por tanto, en particular, cada palabra escogida como término de indexación tendrá asignados más de un peso de caso.

Supongamos que estamos considerando m categorías temáticas. Para la i-ésima ocurrencia de un término de indexación t, su peso de caso es el vector $v_i = (w_{i1}, w_{i2}, \ldots, w_{im})$, donde w_{ij} es el peso de caso del término en su i-ésima aparición para la j-ésima categoría temática.

Supongamos ahora que la frecuencia de dicho término en el documento es k. Entonces, tendremos k pesos de caso diferentes, pudiendo definir la siguiente matriz M_t de dimensión $k * m$:

$$M_t = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{pmatrix}$$

Ahora, para el término t, definimos un nuevo vector, que llamaremos *vector de caso*, $v_t = (c_1, c_2, \ldots, c_m)$, con $c_j = (1/k) \sum_{i=1}^{k} w_{ij}$, correspondiente a la j-ésima categoría temática, para $j = 1, 2, \ldots, m$. Este vector representa la importancia del término de indexación t en cada categoría temática a nivel del documento.

El vector de caso, por tanto, da gran información en cuanto al significado de un término dentro del documento, y permite la diferenciación semántica de los términos del

documento. Sin embargo, no aporta ninguna información a cerca de la relevancia del término en la diferenciación de este documento de los demás documentos que componen el corpus. Para resolver este problema, asignamos a cada término un *peso de significado* y definimos un nuevo vector $v = (w, c_1, c_2, \ldots, c_m)$ para cada término t a partir de su vector de caso v_t y su peso de significado w. Llamaremos *vector semántico* al vector v.

El peso de significado de un término, por su parte, describe su importancia en la definición del contenido del documento que estemos considerando y su relevancia en la discriminación de este documento respecto a los demás. Su valor está determinado por su valor de importancia, es decir, por la frecuencia del término en el documento en particular, y en el corpus, en general.

Supongamos ahora que tenemos n términos de indexación en nuestra colección de textos. En un documento d, cada término de indexación presentará un vector semántico, por lo que para el documento tendremos n vectores semánticos, cada uno correspondiente a uno de los términos de indexación. Llamaremos v_i al vector semántico del i-ésimo término de indexación t_i, $v_i = (w_i, c_{i1}, c_{i2}, \ldots, c_{im})$. Podemos definir entonces la matriz M_d, que llamaremos *matriz semántica*, de dimensión $n * (m + 1)$:

$$M_d = \begin{pmatrix} w_1 & c_{11} & c_{12} & \cdots & c_{1m} \\ w_2 & c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n & c_{n1} & c_{n2} & \cdots & c_{nm} \end{pmatrix}$$

donde la i-ésima fila se corresponde con el vector semántico del i-ésimo término de indexación. La matriz M_d representa semánticamente al documento d.

Usando matrices semánticas podemos representar tanto los documentos del corpus como cualquier consulta. Así, el problema de predicción de la relevancia de los documentos en cuanto a la consulta, queda reducido a un problema de comparación de matrices, más en concreto, a un problema de cálculo de la similitud entre matrices semánticas.

6.4.1. Comparación de matrices

Para afrontar el problema de calcular similitudes entre matrices, en lugar de tratar de comparar matrices directamente, definimos la similitud entre dos matrices como el coseno del ángulo formado por determinados vectores derivados de dichas matrices.

Dada una matriz semántica M, usando la misma notación que antes, definimos el *vector semántico del texto* como
$v_M = (w_1c_{11}, w_1c_{12}, \ldots, w_1c_{1m}, w_2c_{21}, w_2c_{22}, \ldots, w_2c_{2m}, \ldots, w_nc_{n1}, w_nc_{n2}, \ldots, w_nc_{nm})$.

Cada elemento del vector semántico del texto es de la forma w_ic_{ij}, donde w_i es el peso de significado del i-ésimo término de indexación y c_{ij} es el peso de caso del término para la j-ésima categoría semántica.

Supongamos que tenemos dos matrices semánticas M_q y M_d, representando una consulta q y un documento d, respectivamente.

$$M_q = \begin{pmatrix} r_1 & q_{11} & q_{12} & \cdots & q_{1m} \\ r_2 & q_{21} & q_{22} & \cdots & q_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_n & q_{n1} & q_{n2} & \cdots & q_{nm} \end{pmatrix}, \quad M_d = \begin{pmatrix} s_1 & d_{11} & d_{12} & \cdots & d_{1m} \\ s_2 & d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_n & d_{n1} & d_{n2} & \cdots & d_{nm} \end{pmatrix}$$

Los vectores semánticos del texto de la consulta y del documento derivados de sus matrices semánticas son, respectivamente:

$$v_q = (r_1q_{11}, r_1q_{12}, \ldots, r_1q_{1m}, r_2q_{21}, r_2q_{22}, \ldots, r_2q_{2m}, \ldots, r_nq_{n1}, r_nq_{n2}, \ldots, r_nq_{nm})$$
$$v_d = (s_1d_{11}, s_1d_{12}, \ldots, s_1d_{1m}, s_2d_{21}, s_2d_{22}, \ldots, s_2d_{2m}, \ldots, s_nd_{n1}, s_nd_{n2}, \ldots, s_nd_{nm})$$

Sea θ el ángulo formado por v_q y v_d. Por otro lado, sean s y r los vectores formados por la primera columna de M_d y M_q respectivamente, y sean d_{i*} y q_{i*} los vectores de caso para el i-ésimo término de indexación en el documento d y en la consulta q respectivamente. Vamos a usar el siguiente teorema para definir la similitud entre las matrices M_d y M_q.

Teorema (Teorema de similitud coseno)**.** *Si los vectores de términos s y r y los vectores de caso d_{i*} y q_{i*} son unitarios, es decir, $||s|| = ||r|| = ||d_{i*}|| = ||q_{i*}|| = 1$, entonces los vectores v_q y v_d son también unitarios, y se tiene que $cos(\theta) = \sum_{i=1}^{n} s_i r_i (d_{i*} \cdot q_{i*})$.*

Usando la medida similitud coseno calculamos la similitud entre dos matrices semánticas. Es decir, el valor de la similitud será interpretado como una predicción de la relevancia del documento d para la consulta q.

6.5. Mapping de la semántica latente

En esta sección se ha recurrido al libro de Bellegarda [3].

6.5.1. Del LSA al LSM

El éxito del LSA en el campo de recuperación de la información ha llevado a intentar aplicar este paradigma a otras áreas de procesamiento del lenguaje natural. Podemos destacar tres características que hacen el LSA especialmente interesante:

1. El mapping de las palabras y los documentos dentro de un espacio vectorial continuo, donde posteriormente se pueden aplicar algoritmos de aprendizaje automático (*machine learning*).

2. La reducción de la dimensión, haciendo tratables problemas que eran demasiado complejos.

3. La perspectiva global del método, que complementa la aproximación local que utilizan otras técnicas más convencionales.

A través de estas características, el LSA descubre algunos aspectos útiles de la estructura semántica del documento, incluso si no son evidentes a primera vista. Resulta obvio, por otro lado, que estas características pueden ser deseables en multitud de contextos diferentes. Es por eso que estas propiedades han despertado el interés de otros usos potenciales del paradigma básico del LSA. En algunas de estas aplicaciones a otros ámbitos, el objetivo puede incluso no estar relacionado directamente con tareas lingüísticas, razón por la que se produce el cambio de terminología a *mapping de la semántica latente*, en inglés *latent semantic mapping* (LSM). Así, el LSM es una generalización del LSA, siendo el LSA específico para un contexto semántico.

6.5.2. LSM

A pesar de que el LSM es un método más general, vamos a concretar su funcionamiento para el caso que nos ocupa. Sea $\mathcal{V}$ un conjunto de n unidades individuales y sea $\mathcal{D}$ una colección de m composiciones de dichas unidades. Aunque los valores de n y m varían notablemente de un caso a otro, estos valores suelen rondar entre 1000 y 1000000. El LSM define un mapping entre los conjuntos discretos $\mathcal{V}$ y $\mathcal{D}$, y un espacio vectorial continuo $\mathcal{L}$, por el cual cada unidad w_i de $\mathcal{V}$ queda representada por un vector u_i en $\mathcal{L}$, y cada composición d_j de $\mathcal{D}$, por un vector v_j en $\mathcal{L}$. En nuestro caso, es decir, en una aproximación semántica del método, $\mathcal{V}$ es un vocabulario de n palabras y $\mathcal{D}$ un conjunto de m documentos, es decir, el corpus. Los vectores w_i representan las palabras y los vectores d_j, los documentos.

De nuevo, en este método el punto de partida es la construcción de una matriz M de co-ocurrencias entre las unidades de $\mathcal{V}$ y las composiciones de $\mathcal{D}$. Los elementos de la matriz M se definen en función de la aparición de cada unidad en cada composición, tal y como se hacía en el LSA. De nuevo, podemos representar esta aparición de varias formas. Es usual, sin embargo, definir el elemento (i, j) de la siguiente forma:

$$w_{i,j} = (1 - \varepsilon_i)\frac{\kappa_{i,j}}{\lambda_j}$$

donde $\kappa_{i,j}$ es el número de veces que w_i aparece en d_j, λ_j el número total de unidades presentes en d_j, y ε_i es la entropía normalizada de w_i en $\mathcal{D}$. De esta forma, el término $(1 - \varepsilon_i)$ refleja el hecho de que dos unidades con el mismo número de ocurrencias en d_j no aportan necesariamente la misma cantidad de información.

Veamos más detenidamente en qué consiste esta entropía normalizada de la unidad w_i. Denotemos por $\tau_i = \sum_j \kappa_{i,j}$ el número total de veces que w_i aparece en la colección $\mathcal{D}$. Entonces, la expresión de ε_i es la siguiente:

$$\varepsilon_i = -\frac{1}{log\, m}\sum_{j=1}^{m}\frac{\kappa_{i,j}}{\tau_i}\, log\, \frac{\kappa_{i,j}}{\tau_i}$$

Por definición, se tiene que $0 \leq \varepsilon_i \leq 1$, siendo igual a cero si y solo si $\kappa_{i,j} = \tau_i$, e igual a 1 si y solo si $\kappa_{i,j} = \tau_i/N$. Un valor de ε_i cercano a 1 indica que la unidad se distribuye a través de muchas composiciones, mientra que valores cercanos a 0 indican que la unidad

aparece solo en unas pocas composiciones específicas. Por tanto, el término $(1 - \varepsilon_i)$ es una medida del poder de indexación de la unidad w_i.

La matriz de co-ocurrencias M define las representaciones vectoriales para las unidades y las composiciones. Cada unidad w_i se asocia de manera única con un vector fila de dimensión m, y cada composición d_j se asocia, también de manera única con un vector columna de dimensión n. Estas representaciones vectoriales son, sin embargo, poco prácticas por tres razones: la dimensiones m y n pueden ser extremadamente grandes, los vectores w_i y d_j suelen ser vectores huecos y, por último, generan dos espacios diferentes. Para resolver estos problemas, podemos acudir al algoritmo de descomposición en valores singulares (SVD), tal y como se hacía en el LSA, y proceder de manera análoga. Así, obtenemos una descomposición como la que sigue:

$$\hat{M} \approx M = TSD'$$

Las matrices T y S son matrices ortonormales, es decir, los vectores columna de cada una de estas matrices definen una base ortonormal para el espacio vectorial de dimensión r generado por los vectores w_i y d_j respectivamente, donde r es la dimensión de S, es decir, el orden de la descomposición. A partir de esta descomposición, podemos reescribir los vectores w_i y d_j.

$$w_i = \sum_{k=1}^{r} \langle w_i, D'_k \rangle D'_k = \sum_{k=1}^{r} (u_i\, S)_k D'_k, \quad d_j = \sum_{k=1}^{r} \langle d_i, T_k \rangle T_k = \sum_{k=1}^{r} (v_j\, S)_k T_k$$

donde T_k y D_k son los vectores columna de T y S. Estas expresiones representan w_i y d_j como una cobinación lineal de conceptos abstractos y previamente ocultos, representados por los vectores singulares T_k y D_k. De esta forma, se define un mapping $(\mathcal{V}, \mathcal{D}) \longrightarrow \mathcal{L}$ en el que cada unidad $w_i \in \mathcal{V}$ corresponde de manera única con el *vector de unidad* $\hat{u}_i = u_i S$ y cada composición $d_j \in \mathcal{D}$ corresponde de manera única con el *vector de composición* $\hat{v}_j = v_j S$. Este es, por tanto, un mapping biyectivo.

La matriz $\hat{M}$ captura las asociaciones estructurales más fuertes, mientra que desprecia otro tipo de efectos que no son interesantes. Por tanto, dos unidades cuyas representaciones son "cercanas"en alguna métrica, tienden a aparecer en el mismo tipo de composiciones, ya aparezcan o no en el mismo contexto dentro de estas composiciones. Por el contrario, dos composiciones cuyas representaciones son "cercanas"tienden a poseer la misma significación, ya contengan o no las mismas unidades.

6.6. Técnicas de clasificación

Para el desarrollo de las diferentes técnicas incluídas en esta sección, se ha recurrido a los textos de Adankon y Cheriet [1], Berger et al. [4], Wu et al. [18], Quinlan et al. [15], Aggarwal y Zhai [2] y Z y Lior [19].

6.6.1. Máquinas de vectores de soporte

Las máquinas de vectores de soporte (SVM, en inglés *Support Vector Machines*) son un conjunto de algoritmos clasificadores lineales, basados en el principio de maximización marginal. Estos algoritmos llevan a cabo la clasificación de los datos mediante la construcción, en un espacio de dimensión mayor, de un hiperplano que los separa de forma óptima en dos categorías. Llamaremos *clase* a cada una de las regiones en que queda separado el espacio.

La idea de los SVM es, por tanto, encontrar el hiperplano que presente mejor capacidad de generalización. Para ello, busca el margen máximo entre las dos categorías de datos para luego determinar el hiperplano que está en el medio de este margen máximo. Así, los puntos más cercanos de cada categoría al hiperplano óptimo se encontrarán a la misma distancia de este.

Veamos en qué consiste la formulación del SVM. Sea un conjunto de datos $\{(x_1, y_1), \ldots, (x_l, y_l)\}$, con $x_i \in \mathcal{R}^d$ y $y_i \in \{-1, 1\}$. El objetivo es estimar los parámetros w y b para la función de decisión $f(x) = w \cdot x + b$ que define el hiperplano óptimo. Los puntos cercanos a la frontera de decisión definen el margen. Consideremos dos puntos x_1 y x_2 en lados opuestos del margen tales que $f(x_1) = 1$ y $f(x_2) = -1$. El margen correspondiente es igual a $(f(x_1) - f(x_2))/||w|| = 2/||w||$. De esta forma, el problema de maximizar el margen es equivalente al de minimizar $||w||/2$ o $||w||^2/2$. Por esta razón, para encontrar el hiperplano óptimo, basta resolver el siguiente problema de optimización:

$$\begin{cases} min_{w,b}\, \dfrac{1}{2} w'w \\ s.a.\, y_i(w' \cdot x_i + b) \geq 1, \forall i = 1, \ldots, l \end{cases}$$

La transformación del problema anterior en su dual nos lleva al siguiente problema, en el que no aparece el parámetro b:

$$\begin{cases} max_\alpha \displaystyle\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \\ s.a. \displaystyle\sum_{i=1}^{l} y_i \alpha_i = 0;\ \alpha \geq 0,\ \forall i = 1, \ldots, l \end{cases}$$

La solución de este último problema lleva a la elección del parámetro $w = \sum_{i=1}^{l} y_i \alpha_i x_i$ del hiperplano óptimo. Ahora, usando las restricciones del problema primal, se obtiene que $b = -1/2[max_{y=-1}(w \cdot x_i) + min_{y=1}(w \cdot x_i)]$. Es posible probar mediante las condiciones de Karush-Kuhn-Tucker que los únicos x_i correspondientes a α_i distintos de 0 son aquellos que cumplen que $y_i(w \cdot x_i + b) = 1$. Estos x_i son los llamados *vectores de soporte*.

6.6.2. Máxima entropía

El objetivo de este método es construir un modelo que represente adecuadamente el comportamiendo de un proceso aleatorio, en este caso, la asignación de ciertos elementos a diferentes categorías. Más rigurosamente, es un método para estimar la probabilidad condicional $p(y|x)$ de que, dado un contexto x, el procesolo asigne a y.

Para conseguir este objetivo, observamos el comportamiento del proceso aleatorio, recolectando una cantidad notable de muestras $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, que suponemos generadas por una persona experta. Podemos representar entonces la muestra en términos de su distribución de probabilidad empírica $\tilde{p}$, definida por:

$$\tilde{p}(x, y) \equiv \frac{1}{N} \times n(x, y)$$

donde $n(x, y)$ es el número de veces que el par (x,y) aparece en la muestra. Normalmente, un par (x, y) o bien no aparece en la muestra, o bien aparece al menos unas pocas veces.

6.6.3. Análisis discriminante lineal escalado

El LDA (*Linear Discriminant Analysis*) es un método muy conocido cuyo objetivo es encontrar los ejes mediante los cuales los datos de diferentes clases quedan bien separados mientras que los datos pertenecientes a la misma clase resultan cercanos unos de otros. Esta técnica se utiliza frecuentemente tras aplicar una técnica de reducción de la dimensionalidad, pudiendo acudir a alguno de los métodos anteriores que cumplen este objetivo. A continuación, por tanto, vamos a ver en qué consiste el LDA clásico, sin introducir ningún método previo.

Consideremos el siguiente problema de clasificación supervisado multi-clase: sea un conjunto de N datos etiquetados, esto es, asignados previamente a una clase, pertenecientes a K clases $\{C_1, C_2, \ldots, C_K\}$, cada clase de un tamaño $N_1, N_2, \ldots, N_K$ respectivamente, donde $N_1 + N_2 + \cdots + N_K = N$. Por otro lado, sea $X = \{x_1, x_2, \ldots, x_N\}$ la muestra observada e $y_i \in \{1, 2, \ldots, K\}_{i=1,\ldots,N}$ la clase a la que pertenece x_i. El objetivo del LDA es construir un clasificador basado en el conjunto de entrenamiento X para predecir la clase de un conjunto no etiquetado $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_M\}$. El objetivo dl LDA es buscar una matriz W que maximice el siguiente cociente:

$$J(W) = argmax_w \frac{det(W' S_b W)}{det(W' S_w W)}$$

donde las matrices S_b y S_c representan, respectivamente, la dispersión de la variabilidad entre clases y dentro de las clases.

$$S_b = \frac{1}{N} \sum_{i=1}^{K} N_i (m_i - m)'(m_i - m)$$

$$S_w = \frac{1}{N} \sum_{i=1}^{K} \sum_{x_j \in C_i} (x_j - m_i)'(x_j - m_i)$$

donde $m = \frac{1}{N}\sum_{i=1}^{N} x_i$ es la media de los datos de entrenamiento y m_i es la media de los datos pertenecientes a la clase C_i.

De esta forma, la similitud entre los datos S_w se minimiza y las diferencias entre las clases S_b se maximiza. La matriz W de la proyección de discriminación óptima se puede obtener calculando los autovectores de la matriz $S_w^{-1}S_b$. Dado que el rango de S_b es $K-1$, existen como máximo $K-1$ autovectores correspondientes a los autovalores distintos de 0.

Una desventaja de este método es que, al implicar la descomposición de matrices mediante autovalores, puede resultar computacionalmente costoso en cuanto a tiempo y memoria en los casos en los que el número de características o la muestra sean grandes: la complejidad computacional y la memoria requerida crece con N y d. Por esta razón, puede ser conveniente la aplicación previa al método del algoritmo SVD para solventar estos problemas, trabajando en una dimensión menor.

6.6.4. Bagging

El *bagging*, así como el *boosting*, que se tratará en el siguiente apartado, es un método para mejorar la capacidad predictiva de los sistemas de clasificación. Supongamos un conjunto de N casos, que en nuestro caso serán documentos, cada uno perteneciente a una de las K clases, y un sistema de aprendizaje que construye un clasificador a partir de un conjunto de entrenamiento de documentos. Tanto el bagging como el boosting construyen varios clasificadores a partir de los datos de entrenamiento. El número de repeticiones T se supone fijado, aunque este parámetro se puede determinar automáticamente por validación cruzada. EL clasificador desarrollado en la repetición t se denota por C^t, mientras que $C*$ representa el clasificador compuesto. Para un documento d, $C^t(d)$ y $C*(d)$ son las clases predichas para este documento por C^t y $C*$ respectivamente.

Para cada etapa $t = 1, \ldots, T$, se toma un conjunto de entrenamiento de tamaño N mediante un muestreo con reemplazamiento del conjunto de documentos original. Este conjunto de entrenamiento tiene el mismo tamaño N que el conjunto de datos original, sin embargo, algunos documentos aparecerán más de una vez, mientras que otros no aparecerán. El sistema de aprendizaje genera un clasificador C^t a partir de la muestra, y el clasificador $C*$ se crea a partir de los T clasificadores resultantes. Para clasificar un documento d, cada elección de una clase k es recordada por cada clasificador tal que $C^t(d) = k$ y $C*$ es la clase que haya sido más veces elegida.

6.6.5. Boosting

Mientras que en el bagging la clasificación se lleva a cabo mediante una sucesión de selecciones de muestras independientes de los datos originales, el boosting mantiene un peso para cada documento, de forma que a mayor peso, la influencia del documento en el clasificador es mayor. En cada etapa, el vector de los pesos se ajusta para reflejar el rendimiento del clasificador correspondiente, con el resultado de el peso de los documentos no clasificados se incrementa. El clasificador final agrega las elecciones de los diferentes clasificadores aprendidos, al igual que en el baggind, pero en este caso, además, la

elección de cada clasificador es unna función de su precisión.

Sea w_d^t el peso del documento d en el paso t donde, para cada d, $w_d^1 = 1/N$. Para cada $t = 1, \ldots, T$, se construye un clasificador C^t a partir de los documentos dados bajo la distribución w^t, esto es, como si el peso de un documento d representara su probabilidad de ocurrencia. El error e^t de esta clasificación es medido también en función de los pesos, y consiste en la suma de los pesos de los documentos que no han sido clasificados. Si e^t es mayor que 0.5, la etapas se finalizan y se cambia T por $t-1$. Por el contrario, si C^t clasifica todos los documentos, y por tanto $e^t = 0$, las etapas se finalizan y T pasa a ser t. En otro caso, el vector de peso w^{t+1} para la siguiente etapa se genera multiplicando los pesos de los documentos que C^t clasifica correctamente por el factor $\beta^t = e^t/(1-e^t)$, y posteriormente renormalizando para que $\sum_d w_d^{t+1} = 1$. El clasificador $C*$ se obtiene mediante la suma de las elecciones de los clasificadores $C^1, C^2, \ldots, C^T$, donde la elección del clasificador C^t tiene un peso de $log(1/\beta^t)$ unidades.

Si el error obtenido en las diferentes etapas, bajo la distribución uniforme de w^1, es siempre menor que 0.5, la tasa de error de $C*$ se aproxima a 0 exponencialmente con el crecimiento de T. Así, una sucesión de clasificadores "débiles"$\{C^t\}$ puede dar lugar a un clasificador "fuerte"$C*$ que es al menos tan preciso como el clasificador mejor de los clasificadores débiles construídos en las T etapas, y normalmente mucho más preciso. Esto no garantiza, sin embargo, la precisión de $C*$ en la generalización para documentos nuevos, pudiéndose mejorar esta actuación mediante mecanismos tales como el de *Vapnik* (1983).

6.6.6. Random forest

El *random forest* es un clasificador de conjuntos que consiste en la construcción de una colección de clasificadores con estructura de árbol. Es, por tanto, una variante del bagging. Para llevar a cabo este proceso, se considera una muestra de los datos originales y una muestra aleatoria de las caracteristicas, esto es, de variables de clasificación, para construir una colección de árboles de decisión, que posteriormente se promedia. El algoritmo de RF asociado a un conjunto de datos de entrenamiento, en este caso documentos, D y a un conjunto de características N_f se puede describir de la siguiente forma:

1. En primer lugar, se usa bagging para generar K subconjuntos $\{D_1, \ldots, D_K\}$ mediante un muestreo aleatorio con reemplazamiento sobre D.

2. Para cada subconjunto de datos D_i, se construye un modelo de árbol de decisión. Durante la construcción del árbol de decisión, se selecciona aleatoriamente un subespacio de dimensión $m < N_f$ de entre las características disponibles presentes en los documentos de cada nodo. A continuación, se calculan todas las posibles divisiones basadas en estas m características. Las particiones de los datos resultado de la mejor división (en el sentido del *índice de Gini*) se usan para generar nodos secundarios. Este proceso se repite hasta que lo indique un criterio de parada, como puede ser que el número de documentos de cada clase sea menor que un n_{min} fijado previamente.

3. Por último, se combinan los árboles sin podar en un conjunto random forest y se usa la elección mayoritaria a lo largo de los árboles para decidir la clasificación de los documentos.

El método de random forest, como ocurre con el bagging y el boosting, incrementa la precisión conseguida al considerar un solo árbol de decisión, ya que construye la clasificación en función de las decisiones tomadas en los árboles construídos a lo largo de las diferentes etapas.

El índice de Gini

El índice de Gini es uno de los métodos más comunes para cuantificar el nivel de discriminación de una característica. Sea $p_i(w)$, para cada una de las $i = 1, \ldots, k$ clases, la probabilidad condicionada de que un documento pertenezca a la clase i, sabiendo que contiene la palabra w. Por tanto, tenemos que:

$$\sum_{i=1}^{k} p_i(w) = 1$$

Entonces, el índice de Gini para la palabra w, denotado por $G(w)$ se define de la siguiente forma:

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

El valor del índice está siempre comprendido entre $1/k$ y 1. Valores altos del índice de Gini indican un poder discriminativo más fuerte de la palabra w. Por ejemplo, si todos los documentos que poseen una palabra w pertenecen a una misma clase, se tiene que $G(w) = 1$, mientras que si los documentos que contienen la palabra w se distribuyen a lo largo de las k clases, será $G(w) = 1/k$.

6.6.7. Redes neuronales

La unidad básica en una red neuronal se llama *neurona*. Cada unidad recibe un conjunto de inputs, denotados por el vector X_i, que en este caso se corresponde con las frecuencias de los términos en el i-ésimo documento. Cada neurona está asociada también a un conjunto de pesos A, que se usa para el cálculo de una función $f(\cdot)$ de sus inputs. Una función típicamente usada en redes neuronales es la siguiente función lineal:

$$p_i = A \cdot X_i$$

Así, para un vector X_i construído a partir de un vocabulario de d palabras, el vector de pesos A debe contener también d elementos. Consideremos ahora un problema de clasificación binaria en el que todas las etiquetas se toman de $\{+1, -1\}$. Sea y_i la etiqueta de clase de X_i, determinada por el signo de la función p_i. Así, el objetivo de

esta aproximación es determinar el conjunto de pesos A a partir de los datos de entrenamiento. La idea es la siguiente: si comenzamos considerando pesos aleatorios y los vamos actualizando gradualmente al producirse un error aplicando la función al ejemplo de entrenamiento. La magnitud de la actualización está regulada por una tasa de aprendizaje μ. Esta idea es la base del *algoritmo perceptrón*:

Algoritmo perceptrón
Inputs:
Tasa de aprendizaje: μ
Datos de entrenamiento: $(X_i, y_i)\, \forall i \in \{1, \ldots, n\}$
Inicializar los vectores de peso en A con 0 o números pequeños aleatorios
repetir
Comprobar para cada dato de entrenamiento, comprobar si el signo de $A \cdot X_i$ coincide con el de y_i
Si el signo no coincide, actualizar los pesos en A basándose en la tasa de aprendizaje μ
hasta los pesos en A convergen

Los pesos en A suelen actualizarse proporcionalmente a $\mu \cdot X_i$, con el objetivo de reducir las direcciones del error de la neurona. Existen, sin embargo, formas alternativas de llevar a cabo esta actualización.

En el caso de que nuestras clases no estén separadas unas de otras a través de un separador lineal, podemos recurrir a *capas múltiples de neuronas* para introducir estas fronteras de clasificación no lineales. La consecuencia de estas capas múltiples es la introducción de varias fronteras lineales a trozos. Sin embargo, a cambio de estos beneficios, la complejidad del método se incrementa notablemente.

6.6.8. Árbol de clasificación

Un árbol de decisión es un modelo predictivo que puede ser usado como modelo de clasificación y como modelo de regresión. En el primer caso, nos referiremos a un árbol de clasificación, y en el segundo a un árbol de regresión. En esta sección, vamos a centrarnos en los árboles de clasificación.

Los árboles de clasificación se usan para clasificar una colección de objetos, que en este caso serán documentos, en un conjunto predefinido de clases basadas en las caracteristicas de estos objetos. Es un clasificador expresado como partición recursiva del espacio, es decir, del conjunto de documentos. El árbol consiste en un conjunto de nodos que forman un *árbol enraizado*, es decir, existe un nodo *raíz* al que no llega ninguna rama, mientras que al resto de los nodos llega exactamente una rama. Un nodo con ramas salientes a otros nodos se llama *interno* y los nodos sin ramas salientes, *hojas*. En un árbol de decisión, cada nodo interno divide el espacio considerado en dos o más subespacios de acuerdo a ciertas características.

De esta forma, cada hoja se asigna a la clase que representa de manera más adecuada algún valor de sus caracteristicas. Otra opción es asignar a cada hoja un vector de probabilidad indicando la probabilidad de que cierta característica presente cierto valor.

Las ramas salientes de un nodo interno, que pueden ser dos o más de dos, corresponden a rangos de valores de cierta característica que deben ser mutuamente excluyentes y completos. Asi, los documentos son clasificados descendiendo desde la raiz a las hojas.

El árbol de clasificación es muy útil como técnica exploratoria. Sin embargo, esta técnica no puede reemplazar a otras técnicas disponibles para la clasificación o predicción de datos, como pueden ser las máquinas de vectores soporte o el método de random forest, por ejemplo.

Capítulo 7

Aplicaciones con R

Para llevar a cabo las técnicas de minería de datos, vamos a usar el software R (R Development Core Team, 2012), un software libre para la computación estadística o la realización de gráficos que proporciona una gran variedad de técnicas en este sentido. Además, R nos permite extender el programa mediante paquetes, existiendo aproximadamente 4000 paquetes disponibles en el CRAN. R es ampliamente usado a nivel tanto académico como industrial.

En esta sección vamos a aplicar a un caso real algunos de los métodos de clasificación estudiados en la sección previa y el LDA. Para ello, vamos a usar los datos procedentes de una encuesta atendiendo a las respuestas relativas a la ocupación. Estos datos han sido recogidos en forma de respuesta cerrada, categorizándose cada sujeto en una de las 17 categorías existentes o en una categoría extra en caso de que esta pregunda no procediese, y en forma de respuesta abierta, en la que cada sujeto ha especificado en qué consiste su ocupación de forma libre. La encuesta posee 5380 registros en los que, además, se han recogido de forma similar datos relativos a la actividad.

Para explicar las librerías usadas en ambas secciones vamos a usar la información procedente del propio programa R. Por otro lado para el desarrollo de los scripts, se han utilizado los textos de Collingwood et al. [7] y Silge y Robinson [16].

7.1. Técnicas de clasificación en R

7.1.1. Librerías usadas

Para aplicar las técnicas de clasificación vistas anteriormente en R, vamos a recurrir a las librerías *RTextTools* y *tm*. Veámos en qué consisten:

- RTextTools es un paquete de aprendizaje automático para la clasificación automática de textos. Este paquete permite llevar a cabo una clasificación de forma relativamente simple, a la vez que ofrece la posibilidad de experimentar de forma sencilla a través de los diferentes ajustes y combinaciones de los algoritmos disponibles. Incluye nueve algoritmos: *svm*, *slda*, *boosting*, *bagging*, *random forests*, *glmnet*, *decision trees*, *neural networks* y *maximum entropy*. RTextTools ofrece una aproximación

exhaustiva a la clasificación de textos, acudiendo a rutinas de pre-procesamiento de textos y algoritmos de aprendizaje automático, a la vez que ofreciendo nuevas funciones analíticas. A pesar de que algunos paquetes ya existentes pueden usarse para el procesamiento de textos y el procesamiento automático, ningún paquete combina estos procesos con funciones que permitan evaluar la exactitus de ambos. En conclusión, RTextTools acelera el proceso de clasificación de textos: todos los pasos, desde el proceso de instalación al entrenamiento y la clasificación, han sido optimizados para hacer el aprendizaje automático con datos textuales más accesible.

- El paquete tm constituye un marco para aplicaciones de la minería de textos dentro de R, proporcionando una infraestructura para construir un corpus, por ejemplo, leyendo datos de texto de archivos PDF y transformando el corpus en una matriz de término-documento.

7.1.2. Aplicación de las técnicas

En esta sección vamos a clasificar, de acuerdo las técnicas de máxima entropía, máquinas de vectores de soporte y análisis discriminante lineal escalado, distintos documentos de texto que, en este caso, van a ser las respuestas abiertas a la pregunta de "ocupación". El objetivo es comparar la clasificación de las respuestas a estas preguntas abiertas llevada a cabo a partir de los algoritmos citados, con la clasificación real que conocemos atendiendo a las respuestas cerradas.

En primer lugar, vamos a cargar las librerías necesarias y los datos de la encuesta, procedentes de un archivo *csv*, así como a la eliminación de los registros en los que la respuesta a las preguntas relativas a la ocupación no procediera, ya que estos casos no nos van a resultar de ninguna utilidad para nuestro propósito.

```
library(RTextTools)
library(tm)

esm2011<-read.csv2(file="ESM2011.csv")
esm2011b<-esm2011[esm2011$OCUPA!=-9,]
esm2011c<-esm2011b[esm2011b$OCUPA_O!="-9",]

dataESM <- esm2011c
rm(esm2011,esm2011b,esm2011c)
```

A continuación, vamos a establecer el tamaño del conjunto test y del conjunto de entrenamiento.

```
nESM<-dim(dataESM)[1]
nt<-250
```

Ya estamos en condiciones de crear la matriz de término-documento. Para ello, usamos la función create_matrix que nos proporciona la librería tm, en cuyos argumentos introducimos los textos que considerar más adelante para aplicar los algoritmos, el lenguaje en que se encuentran dichos textos, indicamos qué modificaciones hacer en los documentos y el índice que vamos a usar, en este caso el TfIdf.

```
matrixESM <- create_matrix(cbind(dataESM["OCUPA_O"],dataESM["LOCUPA"]),
                           language="spanish",removeNumbers=TRUE,
                           stemWords=FALSE, weighting=tm::weightTfIdf)
```

La matriz términos-documentos que obtenemos presenta las características detalladas en la tabla 7.1. Observamos que, como era esperable, la matriz es hueca y de gran dimensión.

DocumentTermMatrix (documents: 2190, terms: 1447)	
Non-/sparse entries:	15577/3153353
Sparsity:	100 %
Maximal term length:	18
Weighting:	term frequency - inverse document frequency (normalized) (tf-idf)

Cuadro 7.1: Características de la matriz término-documento

Una vez disponemos de la matriz de término-documento, procedemos a la creación de un contenedor de datos para la clasificación; esto es, se efectúa una partición de la matriz, obteniendo dos matrices huecas, de entrenamiento y test, separadas. Usamos para ello la función create_container, también de la librería tm, especificando cuáles van a ser los conjuntos test y de entrenamiento: los 250 primeros registros se utilizarán para probar el modelo, mientras que el resto de registros se usarán para entrenar el modelo.

```
containerESM <- create_container(matrixESM,dataESM$OCUPA,
                                 trainSize=(nt+1):nESM,
                                 testSize=1:nt, virgin=FALSE)
```

Ya podemos aplicar las técnicas de clasificación. Para ello, aplicamos la función train_models al contenedor de datos previamente definido y especificamos qué algoritmos queremos usar. Esta función efectúa un modelo de clasificación a partir de los datos de entrenamiento.

```
modelsESM <- train_models(containerESM,
                          algorithms=c("MAXENT","SVM","SLDA"))
```

Ahora, vamos a calcular la precisión de los distintos algoritmos usados. Primero, vamos a definir algunos conceptos:

- Precisión (P): número de verdaderos positivos (T_p) dividido por la suma del número de verdaderos positivos más el número de falsos positivos (F_p).

$$P = \frac{T_p}{T_p + F_p}$$

- Recall (R): número de verdaderos positivos (T_p) dividido por la suma del número de verdaderos positivos más el número de falsos negativos (F_n).

$$P = \frac{T_p}{T_p + F_n}$$

- F-score (F_1): media armónica de precision y recall.

$$F_1 = 2\frac{P * R}{P + R}$$

La función classify_models, de la librería RTextTools utiliza los modelos que devuelve train_models para llevar a cabo la clasificación.

```
resultsESM <- classify_models(containerESM, modelsESM)
head(resultsESM)

##   MAXENTROPY_LABEL MAXENTROPY_PROB SVM_LABEL  SVM_PROB SLDA_LABEL
## 1                6       1.0000000         6 0.7348717          6
## 2                2       1.0000000         2 0.9871216          2
## 3                5       0.5571947         6 0.5741915          5
## 4                4       1.0000000         4 0.9867922          4
## 5               16       0.9999988        16 0.8904056         16
## 6                1       1.0000000         1 0.9977155          1
##   SLDA_PROB
## 1 1.0000000
## 2 1.0000000
## 3 0.9700087
## 4 1.0000000
## 5 1.0000000
## 6 1.0000000

analyticsESM <- create_analytics(containerESM, resultsESM)
summary(analyticsESM)

## ENSEMBLE SUMMARY
##
##        n-ENSEMBLE COVERAGE n-ENSEMBLE RECALL
## n >= 1                1.00              0.94
```

```
## n >= 2                1.00              0.94
## n >= 3                0.97              0.96
##
##
## ALGORITHM PERFORMANCE
##
##        SVM_PRECISION           SVM_RECALL           SVM_FSCORE
##            0.9482353            0.9470588            0.9470588
##       SLDA_PRECISION          SLDA_RECALL          SLDA_FSCORE
##            0.9370588            0.9364706            0.9358824
## MAXENTROPY_PRECISION    MAXENTROPY_RECALL    MAXENTROPY_FSCORE
##            0.9335294            0.9329412            0.9329412
```

Atendiendo a estos resultados podemos observar que, en este caso, el algoritmo que mejor funciona es el de máquinas de vectores de soporte, aunque la ejecución de los tres algoritmos es buena.

Veamos, por último, una comparación entre la categoría asignada por los algoritmos en cada caso y la categoría real.

```
dataESM250<-dataESM[1:250,]
dataESMOCUPA<-dataESM250$OCUPA

categ_maxen<-as.numeric(as.character(resultsESM[,1]))
categ_svm<-as.numeric(as.character(resultsESM[,3]))
categ_slda<-as.numeric(as.character(resultsESM[,5]))

comparacion<-data.frame(dataESMOCUPA,categ_maxen,categ_svm,categ_slda)
head(comparacion)
```

```
##   dataESMOCUPA categ_maxen categ_svm categ_slda
## 1            6           6         6          6
## 2            2           2         2          2
## 3            6           5         6          5
## 4            4           4         4          4
## 5           16          16        16         16
## 6            1           1         1          1
```

7.2. Asignación latente de Dirichlet en R

7.2.1. Librerías usadas

- Los paquetes RTextTools y tm, dado que, en este caso, se requiere de nuevo el procesamiento de los datos y la construcción de una matriz de término-documento, que sirve de punto de partida.

- El paquete topicmodels nos posibilita el ajuste del modelo de LDA con el algoritmo VEM y con el muestreo de Gibbs, con la posibilidad, además, de realizar este ajuste mediante diferentes métodos de estimación. Este paquete se basa en el paquete tm, ya que proporciona el punto de partida para aplicar los modelos de tópicos. En topicmodels, el codigo se llama directamente a través de una interfaz en el nivel C, lo que evita la entrada y salida de archivos mejorando así notablemente el rendimiento.

- El paquete tidytext hace las tareas de minería de textos más fáciles, eficientes y consistentes. Proporciona funciones y conjuntos de datos de apoyo que permiten convertir los textos a un formato ordenado.

- El paquete ggplot2 ofrece un potente lenguaje para la creación de gráficos elegantes y complejos. Permite construir gráficos que representen datos numéricos o categóricos, univariantes o multivariantes, de forma directa, así como el uso de colores, símbolos, tamaños y transparencias.

- Al trabajar con datos debemos enfrentarnos a las tareas de plantearnos qué queremos conseguir, describir estos objetivos en forma de un programa de ordenador y ejecutar dicho programa. El paquete dplyr facilita estas tareas al restringir las opciones, proporcionar funciones "verbo"* e introducir backends, que hacen los programas más rápidos.

7.2.2. Aplicación del LDA

En primer lugar, de nuevo, cargamos las librerías y los datos, selecionamos los registros de interés y creamos la matriz término-documento de la misma forma que en la aplicación de las técnicas de clasificación.

```
library(topicmodels)
library(RTextTools)
library(tm)
library(tidytext)
library(ggplot2)
library(dplyr)

esm2011<-read.csv2(file="ESM2011.csv")
esm2011b<-esm2011[esm2011$OCUPA!=-9,]
esm2011c<-esm2011b[esm2011b$OCUPA_O!="-9",]

dataESM <- esm2011c
rm(esm2011,esm2011b,esm2011c)
```

*funciones que corresponden a las tareas de manipulación de datos más comunes

```
matrixLDA <- create_matrix(cbind(dataESM["OCUPA_O"],dataESM["LOCUPA"]),
                           language="spanish", removeNumbers=TRUE,
                           stemWords=TRUE, weighting=weightTf)
```

En este caso, dado que vamos a aplicar el LDA, en los argumentos de la función create_matrix indicamos que se realice stemming y elegimos que se construya de acuerdo a la frecuencia de los términos. Las características de la matriz obtenida se recogen en la tabla 7.2.

DocumentTermMatrix (documents: 2190, terms: 1447)	
Non-/sparse entries:	15577/3153353
Sparsity:	100 %
Maximal term length:	18
Weighting:	term frequency (tf)

Cuadro 7.2: Características de la matriz término-documento

Vamos a usar la función LDA del paquete topicmodels. Tomamos $k = 17$ para crear un modelo LDA de 17 tópicos. Tomamos $k = 17$ debido a que, en nuestra encuesta, se utilizan 17 categorías en las preguntas de respuesta cerrada relativas a la ocupación (OCUPACION Y LOCUPACION).

```
lda <- LDA(matrixLDA, k = 17, control = list(seed = 1234))
lda

## A LDA_VEM topic model with 17 topics.
```

Para interpretar el modelo que acabamos de obtener, podemos usar la función terms, que nos proporciona el término más probable por tópico.

```
terms<-terms(lda)
terms

##         Topic 1         Topic 2         Topic 3         Topic 4
## "profesionales" "instalaciones"   "transportes"  "construccion"
##         Topic 5         Topic 6         Topic 7         Topic 8
##       "maestra"      "maquinas"      "camarera"    "maquinaria"
##         Topic 9        Topic 10        Topic 11        Topic 12
##      "comercio"     "servicios"   "cientificos"       "oficina"
##        Topic 13        Topic 14        Topic 15        Topic 16
##       "empresa"          "jefe"     "ensenanza"   "funcionario"
##        Topic 17
##    "directores"
```

Veamos ahora, para cada documento, esto es, para cada respuesta abierta, el tópico asignado por el modelo (representado por su palabra más probable), así como la categoría en la que se había situado cada persona dentro de LOCUPA.

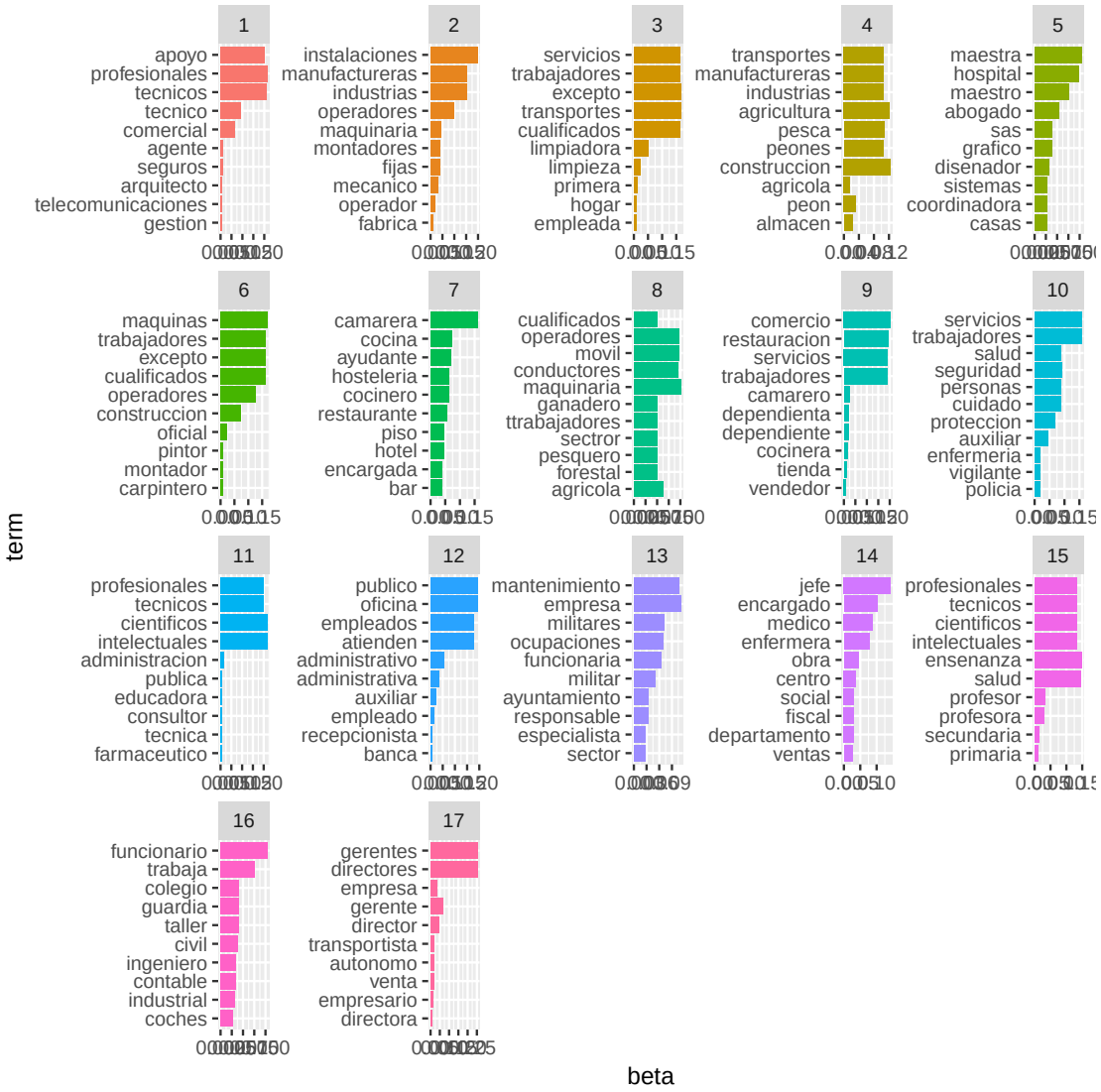
```
topics<-topics(lda)
r = cbind(as.vector(dataESM$LOCUPA), as.vector(dataESM$OCUPA_O),
          as.vector(terms[topics]))
head(r)

##      [,1]
## [1,] "F.Empleados oficina atienden publico"
## [2,] "B.Tecnicos y profesionales cientificos e intelectuales salud y ensenanza"
## [3,] "F.Empleados oficina atienden publico"
## [4,] "D.Tecnicos y profesionales de apoyo"
## [5,] "P.Peones agricultura, pesca, construccion, industrias manufactureras y transp
## [6,] "A.Directores y gerentes"
##      [,2]                              [,3]
## [1,] "Teleoperadora"                   "oficina"
## [2,] "Profesora"                       "ensenanza"
## [3,] "ADMINISTRATIVO"                  "oficina"
## [4,] "Jefe de Compras"                 "profesionales"
## [5,] "Labores del campo con la aceituna" "construccion"
## [6,] "TRANSPORTISTA"                   "directores"
```

Usando la función tidy del paquete tidytext obtenemos beta** (β) para cada tópico, y para cada término. Visualizamos gráficamente, para cada tópico, los diez términos más probables utilizando ahora funciones de las librerías dplyr y ggplot2.

```
ap_topics <- tidy(lda, matrix = "beta")
ap_topics

## # A tibble: 23,001 x 3
##    topic term               beta
##    <int> <chr>             <dbl>
##  1     1 abastecimiento 4.04e-153
##  2     2 abastecimiento 1.86e-152
##  3     3 abastecimiento 1.55e-284
##  4     4 abastecimiento 1.07e-  3
##  5     5 abastecimiento 1.80e-104
##  6     6 abastecimiento 6.17e-169
##  7     7 abastecimiento 5.11e-119
##  8     8 abastecimiento 3.12e-168
```

** probabilidad de cada término para cada tópico

```
##  9     9 abastecimiento 1.75e-169
## 10    10 abastecimiento 1.46e-168
## # ... with 22,991 more rows

ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



Para terminar, obtenemos las probabilidades gamma[***] (γ) para cada documento, y para cada tópico, usando, de nuevo, la función tidy.

```
ap_documents <- tidy(lda, matrix = "gamma")
ap_documents

## # A tibble: 37,230 x 3
##    document                                                topic   gamma
##    <chr>                                                   <int>   <dbl>
##  1 Teleoperadora F.Empleados oficina atienden publico          1 0.00525
##  2 Profesora B.Tecnicos y profesionales cientificos e intel~     1 0.00385
```

[***] probabilidad de cada tópico para cada documento

```
##  3 ADMINISTRATIVO F.Empleados oficina atienden publico           1 0.00525
##  4 Jefe de Compras D.Tecnicos y profesionales de apoyo           1 0.552
##  5 Labores del campo con la aceituna P.Peones agricultura, ~     1 0.00275
##  6 TRANSPORTISTA A.Directores y gerentes                         1 0.00826
##  7 ADMINISTRATIVA DE OBRAS. TRABAJA EN UNA CONSTRUCTORA. E.~     1 0.00340
##  8 ATIENDE EN UNA TIENDA DE ROPA AL POR MENOR G.Trabajadore~     1 0.00340
##  9 ENCARGADO DE CANTERAS EN UNA EMPRESA DE CEMENTOS D.Tecni~     1 0.404
## 10 Empleado Cajasol (equipo de rotacion, direccion, subdire~     1 0.00275
## # ... with 37,220 more rows
```

Capítulo 8

Líneas futuras

- **Mejora de los métodos de comprensión del lenguaje natural:** Actualmente, la mayoría de los enfoques se basan en la representación de la bolsa de palabras. Sin embargo, esta representación que puede resultar demasiado simple. Las técnicas de extracción de la información van un poco más allá en la comprensión del lenguaje natural con la representación semántica, pero la mayoría de los métodos de los que disponemos hoy en día se limitan al aprendizaje supervisado y, por lo general, solo funcionan cuando se dispone de una cantidad suficiente de datos de entrenamiento, lo que supone una limitación. Por tanto, es importante desarrollar métodos más robustos y efectivos en este ámbito.
- **Adaptación del dominio y aprendizaje de transferencia:** Muchas de las tareas desarrolladas en minería de textos se basan en el aprendizaje supervisado que, como ya se ha dicho, presenta limitaciones al requerir cierta cantidad de datos de entrenamiento para ser efectivo. Los métodos en adaptación del dominio y aprendizaje de transferencia pueden solventar este problema aprovechando los datos de entrenamiento disponibles en dominios o tareas relacionadas. Sin embargo, los métodos actuales en estos ámbitos presentan todavía muchas limitaciones cuando los datos en el dominio de interés no existen o son pocos.
- **Análisis contextual de los textos:** Normalmente, los textos están asociados a una información contextual como pueden ser los autores de dicho documento o información temporal. Esta información puede ser interesante en algunos casos. Por esto, seria interesante la incorporación de este tipo de información para así llevar a cabo análisis más potentes de los textos.
- **Minería de textos paralela:** En muchas aplicaciones de la minería de textos, la cantidad de datos textuales es enorme y está continuamente incrementándose, lo que hace imposible almacenar todos los datos en una sola máquina. En estos casos es por tanto necesario desarrollar algoritmos de minería de textos paralela que funcionen a la vez en un grupo de ordenadores para llevar a cabo tareas de minería ded textos en paralelo. Esta es una línea de mejora aplicable a todas las técnicas actuales.

Bibliografía

[1] M. Adankon and M. Cheriet. *Support Vector Machine*, pages 1303–1308. Springer US, Boston, MA, 2009.

[2] C.C. Aggarwal and C.X. Zhai. *Mining Text Data*. Springer New York, 2012.

[3] J.R. Bellegarda. *Latent Semantic Mapping: Principles & Applications*. Synthesis lectures on speech and audio processing. Morgan & Claypool, 2007.

[4] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[5] C. Biemann and A. Mehler. *Text Mining: From Ontology Learning to Automated Text Processing Applications*. Theory and Applications of Natural Language Processing. Springer International Publishing, 2014.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[7] Loren Collingwood, Timothy Jurka, Amber E Boydstun, Emiliano Grossman, WH van Atteveldt, et al. Rtexttools: A supervised learning package for text classification. 2013.

[8] Dean D. Billheimer, Andrew James Booker, Michelle Keim Condliff, Mark Thomas Greaves, Fredrick Baden Holt, Anne Shu-Wan Kao, Daniel John Pierce, Stephen Robert Poteet, and Yuan-Jye Wu. Method and system for text mining using multidimensional subspaces, August 26 2003. US Patent 6,611,825.

[9] Javi Fernández, José Manuel Gómez, Ester Boldrini, and Patricio Martínez-Barco. Análisis de sentimientos y minería de opiniones: el corpus emotiblog. *Procesamiento del lenguaje natural*, (47), 2011.

[10] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, Jan 2001.

[11] G. Jorge-Botana. *La técnica del ánalisis de la Semántica Latente (LSA/LSI) como modelo informático de la comprensión del texto y el discurso una aproximación*

distribuida al análisis semántico. PhD thesis, Universidad Autónoma de Madrid, 2010.

[12] María del Consuelo Justicia de la Torre. Nuevas técnicas de minería de textos: Aplicaciones. 2017.

[13] Geoffrey Z Liu. Semantic vector space model: Implementation and evaluation. *Journal of the American Society for Information Science*, 48(5):395–417, 1997.

[14] G. Miner, J. Elder, and T. Hill. *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. Academic Press, 2012.

[15] J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[16] Julia Silge and David Robinson. *Text Mining with R: A tidy approach*. .º 'Reilly Media, Inc.", 2017.

[17] S. M. Weiss, T. Zhang N. Indurkhya, and F. Damerau. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer New York, 2010.

[18] Qingyao Wu, Yunming Ye, Haijun Zhang, Michael K. Ng, and Shen-Shyang Ho. Forestexter: An efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems*, 67:105 – 116, 2014.

[19] M.O. Z and R. Lior. *Data Mining With Decision Trees: Theory And Applications (2nd Edition)*. Series In Machine Perception And Artificial Intelligence. World Scientific Publishing Company, 2014.

[20] Gustavo Adolfo Valencia Zapata. Minería de datos. `http://www.gustavovalencia.com/app/webroot/img/Documents/BI/Actividades/001/Articulo%20DM.pdf`.