UNIVERSIDAD AUTÓNOMA DE MADRID

Escuela Politécnica Superior

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

MACHINE LEARNING EN BASES DE DATOS DE LENGUAJE NATURAL

Álvaro García Gutiérrez

Tutor: Manuel Sánchez-Montañés

Junio 2016

Machine Learning en Bases de Datos de Lenguaje Natural

Autor: Álvaro García Gutiérrez
Tutor: Manuel Sánchez-Montañés

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio 2016

Resumen

Resumen El extenso crecimiento que ha experimentado la red durante los últimos años lleva consigo la necesidad de organizar eficientemente la nueva información que se almacena cada día. Por ello cada vez son más importantes las tareas de gestión de documentos en función del contenido. La *clasificación de textos* es una de estas tareas, la cual consiste en organizar, en diferentes clases, textos escritos en lenguaje natural.

Entre las diferentes maneras de abordar este problema, aquí nos limitaremos a tratar con técnicas de aprendizaje automático, es decir, procesos que construyen automáticamente clasificadores. Estos clasificadores aprenden las características de cada clase gracias a un entrenamiento previo sobre un conjunto de textos preclasificados. También daremos cabida en nuestro estudio a las diferentes técnicas de preprocesamiento de los textos, las cuales deben ser realizadas con cuidado con el fin de resolver eficazmente cualquier problema de clasificación propuesto.

A lo largo de este trabajo se compararán algunas de estas técnicas de clasificación y preprocesamiento de textos, centrando la mayor parte de nuestros esfuerzos en un problema específico de clasificación conocido como *análisis de sentimientos*. Este problema consiste clasificar textos según las distintas emociones que expresan, algo que cada vez tiene mayor interés debido al gran desarrollo que están experimentando las redes sociales de hoy en día.

Para ello, previamente se estudiarán las metodologías existentes de clasificación con el fin de comprender las limitaciones y ventajas de cada uno. Después, se escogerán algunas de estas metodologías y se realizarán pruebas sobre diferentes bases de datos tratando de comparar las diferentes metodologías e intentando finalmente construir el clasificador que obtenga mejores resultados. Este proceso se llevará a cabo analizando tres fases diferentes: la representación previa de los documentos, la extracción y selección de características, la construcción de un clasificador y la evaluación de dicho clasificador.

Palabras clave clasificación de textos, análisis de sentimientos, procesamiento de lenguaje natural, gestión de documentos, aprendizaje automático, clasificadores, preprocesamiento de textos

Abstract

Abstract The extensive growth that the Internet has gone through over the last few years carries with it, the need to efficiently organize the new information that is stored each day. Due to this, the job of document management, based on content, has become much more important. The *text classification* is one of these such jobs, which consists of organizing, in different categories, texts written in their natural language.

While there are various ways to approach this issue, here, we will focus on machine learning techniques. In other words, the processes that automatically build and manage classifiers. These classifiers learn the characteristics of each category due to the prior training sessions done with a pre-classified group of texts. We will also look very carefully at the different techniques used in the processing of the texts in order to efficiently resolve any classification issues that might arise.

Throughout this study, many of these classification and text-processing techniques will be compared. However, the majority of our focus will be on one specific classification problem known as *sentiment analysis*. This problem revolves around the idea of classifying texts based on the different emotions they express, something, which has recently gained interest due to the growth of social media.

To do this, we must first study the existing methods of classification to understand the limits and advantages of each one. Next, a few of these methods will be chosen to be used in trials with different databases. The trials and methods will be compared to then create a classification system that obtains the best results. This process will be conducted by analyzing four different phases: the prior representation of the documents, the extraction and selection of characteristics, the creation of a classifier and the evaluation of said classifier.

Keywords text classification, sentiment analysis, natural language processing, document management, machine learning, classifiers, text-processing

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

GLOSARIO

ASR *Automatic Speech Recognition*. 6, 8

BoW *Bag of Words*. IV, VI, 10, 16, 19, 29, 30, 33, 35, 36

CRP *Chinese Restaurant Process*. 18, 34, 35

DM *Dialogue Management*. 6

HRL *High Resource Language*. 6, 8

IMDB *Internet Movie Database*. 27, 28

LRL *Low Resource Language*. 6, 8

LSA *Latent Semantic Analysis*. VI, 16, 33, 34, 35

NLP *Natural Language Processing*. 1, 5, 18

PCA *Principle Components Analysis*. VII, 17

POS Tagging *Part of Speech Tagging*. 15

SVM *Support Vector Machines*. VII, 11, 21

TTS *Text to Speech*. 7, 8

1 Introducción y motivación

Durante este capítulo vamos a detallar algunas de las necesidades existentes en el campo del aprendizaje automático y la clasificación de textos, los cuales han motivado la realización de este Trabajo de Fin de Grado. También comentaremos los objetivos a cumplir así como la composición o estructura de este documento.

1.1 Motivación

En los últimos años se han producido enormes avance en el campo del análisis automático de documentos en lenguaje natural. La investigación en el procesamiento de lenguaje natural ha evolucionado desde épocas donde analizar una oración podía tardar hasta siete minutos hasta la era de empresas como Google o Facebook (donde se pueden procesar millones de páginas web en menos de un segundo). Estas empresas utilizan técnicas específicas desarrolladas en el área de la Inteligencia Artificial para extraer conocimiento de alto nivel a partir de este tipo de datos y realizar predicciones sobre el comportamiento de los usuarios.

Desde el nacimiento de la conocida *Social Web*, donde empresas como *MySpace*, *LinkedIn* y *Facebook* empezaron a formarse, no han parado de ofrecerse al mundo servicios de creación de contendio compartido que permiten que millones de personas se conecten diariamente a la red para difundir sus ideas y opiniones y crear contenido digital en general. Sin embargo esta gran cantidad de información, al ser producida por y para el consumo humano, está la mayoría desestructurada y no puede ser procesada fácilmente. El análisis automático de textos implica una profunda comprensión del lenguaje natural por parte de los ordenadores, una realidad de la cual aún estamos bastante lejos.

Hasta ahora, se han implementado buenos algoritmos para la recuperación de información *online*. Estos algoritmos funcionan bien a la hora de recuperar textos, comprobar la ortografía, contar las palabras, etcétera. Sin embargo, cuando se trata de interpretar las frases y extraer información significativa, sus capacidades son muy limitadas.

A día de hoy, la mayoría de las técnicas de procesamiento de lenguaje natural están basadas en la representación sintáctica de los textos, un metodo que se basa principalmente en las frecuencias de ocurrencia de las palabras. Estos algoritmos son algo limitados por el hecho de que pueden simplemente procesar la información que pueden ver. Sin embargo, los humanos no tenemos esa limitación ya que cada palabra que leemos activa un sin fin de conceptos relacionados semánticamente y aplicamos sin ninguna dificultad todas las tareas que NLP querría desempeñar satisfactoriamente (véase la desambigüación lingüistica, relación con el contexto, etcétera).

En este trabajo se buscarán las técnicas de procesamiento de lenguaje natural más recientes que tratan de superar estas limitaciones sin dejar de revisar y poner a prueba las técnicas tradicionales con el fin de establecer las mejores condiciones en las que desarrollar métodos de clasificación de textos.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado consiste en encontrar los mejores métodos y técnicas existentes en clasificación automática de textos con el fin de solucionar problemas de procesamiento de lenguaje natural. Para poder llevarlo a cabo será necesario estudiar los distintos campos de estudio del procesamiento de lenguaje natural, entendiendo el objetivo y funcionamiento de cada uno de ellos. Trataremos de encontrar sus principales aplicaciones a día de hoy así como las técnicas que se utilizan y sus posibles limitaciones.

Tras una primera revisión del campo, profundizaremos en la investigación sobre clasificación automática de textos. Comenzaremos estudiando las distintas necesidades que podemos encontrar hoy en día donde aplicar esta metodología (análisis de sentimientos, reconocimiento de autor, detección de spam, etcétera) y buscaremos técnicas específicas que optimicen la resolución de cada una de ellas. Continuaremos explicando las diferentes fases que componen un proceso de clasificación, buscando las técnicas más interesantes y utilizadas en cada fase.

El siguiente paso será entender con detalle, gracias a la literatura estudiada, el funcionamiento de los distintos métodos de preprocesamiento, representación de textos y aprendizaje automático que utilizaremos para clasificar y, una vez asimilados, será el momento de ponerlos en práctica. Utilizaremos bases de datos que nos permitan poner a prueba varios problemas y comprobar la eficiencia de cada técnica en diferentes contextos.

El último paso consisitirá en realizar una comparación exhaustiva de los resultados obtenidos en la fase de clasificación. Debido a la cantidad de parámetros y elementos que pueden modificarse en un proceso de clasificación de textos, será importante definir bien las condiciones sobre las que se realizarán las pruebas, intentando encontrar las mejores posibles en cada caso.

1.3 Estructura del documento

En el segundo capítulo de este Trabajo de Fin de Grado se realizará el **Estado del Arte**. Las primeras secciones del capítulo consistirán en revisar los

diferentes ámbitos en los que se hace uso del procesamiento de lenguaje natural. Y en la última sección profundizaremos en clasificación automática de textos. Examinaremos los problemas más importantes en este campo comentando las posibles soluciones planteadas hasta el momento.

Una vez estudiados los distintos problemas, soluciones y limitaciones de la clasificación automática de textos, es el momento de explicar en profundidad todos y cada uno de los métodos que se van a utilizar en nuestro trabajo. Esta profundización se realizará en el tercer capítulo llamado **Metodología**.

Todas estas técnicas explicadas se pondrán en práctica y evaluarán en el cuarto capítulo llamado **Resultados**. Aquí se realizará una evaluación gradual de los métodos y se mostrará el rendimiento de cada uno de ellos sobre diferentes bases de datos.

Finalmente, el último capítulo será dedicado a las **Conclusiones** del estudio. Determinaremos cuáles han sido los mejores y peores métodos en la práctica y compararemos los resultados obtenidos con nuestras expectativas iniciales. Además mostraremos las futuras líneas a seguir en este trabajo de investigación.

2 ESTADO DEL ARTE

El **Procesamiento de Lenguaje Natural** (o NLP) es un campo de la inteligencia artificial que hace uso de diferentes algoritmos y análisis estadísticos para aprender, entender y producir contenido en lenguaje humano. Su propósito es ayudar a la interacción entre humanos y ordenadores. [2] [13]

En las últimas décadas, han aparecido multitud de productos que incorporan algoritmos de procesamiento de lenguaje natural (*Cortana* de *Microsoft* [19], *Siri* de *Apple* [3], *Google Traductor* [12] y un largo etcétera). La investigación y el desarrollo que se lleva a cabo hoy en día en esta tecnología es notable. Esto implica un impactante crecimiento en la materia que ha sido posible gracias a ciertos cambios fundamentales en el ámbito informático, véase la gran cantidad de información en formato lingüistico alojada en la red, el importante aumento de la capacidad computacional, el desarrollo de eficientes y nuevos métodos de aprendizaje automático y un mayor entendimiento de la estructura del lenguaje humano y su aplicación en contextos sociales. [13]

Materias como la traducción automática, los agentes de conversación o el profundo análisis de contenido lingüistico surgen con el objetivo de mejorar la comunicación entre humanos y máquinas. Durante esta sección explicaremos los distintos ámbitos que han desarrollado el procesamiento de lenguaje natural. Analizaremos técnicas punteras que hacen uso de grandes cantidades de datos (*big data*) y los tratan con algoritmos de aprendizaje automático y análisis estadístico.

En los inicios del procesamiento del lenguaje natural, el objetivo era introducir el vocabulario y las reglas del lenguaje humano en los ordenadores, pero esta difícil tarea quedó descartada debido a la ambigüedad del lenguaje en función del contexto. Más tarde se observó que se obtenían buenos resultados al utilizar técnicas simples sobre cantidades grandes de datos. Estas técnicas hacen uso de simples conjuntos de palabras o secuencias *part of speech* (es decir, palabras que incluyen su función gramatical, ya sea nombre, verbo, adjetivo, etc). Actualmente, muchos clasificadores utilizan técnicas que requieren simplemente conjuntos de palabras contenidas en los textos (técnica conocida como *bag of words*), sin tener en cuenta la estructura o el significado de las frases. Conseguir mejoras

sobre esto puede llegar a ser difícil, pero existen herramientas que permiten mejorar este procedimiento identificando nombres propios (empresas, personas, etc), aplicando el ya mencionado *part of speech tagging* o eliminando palabras redundantes, entre otros.

La mayor limitación que sufre el procesamiento de lenguaje natural hoy en día es que la mayoría de documentos y herramientas que se utilizan sólo existen en lenguajes de altos recursos (*high-resource languages* o HRLs), véase castellano, inglés, francés, alemán, etcétera. En el futuro se planea desarrollar recursos para los lenguajes de bajos recursos (*low-resource languages* o LRLs). [13]

2.1 Traducción Automática

Hace tiempo que la traducción entre lenguajes ha dejado de ser una labor única de humanos. A día de hoy, la importancia de este proceso en el ámbito de la inteligencia artificial es creciente y por ello la *traducción automática* [1] continúa siendo objeto de estudio y desarrollo. Traducir correctamente un texto requiere poseer excelentes habilidades de análisis y creación de frases en lenguaje humano, además de poseer un razonamiento similar al humano para superar las ambigüedades de los distintos lenguajes.

Los primeros avances en este campo se debieron al análisis de texto paralelo (comparación de frases escritas en distintos idiomas que significan lo mismo). Tras recoger estadísticas de estas traducciones se crearon modelos probabilísticos de traducción automática. Finalmente, el impulso de internet (enormes cantidades de texto paralelo en la red) y el desarrollo de la informática (mayores velocidades de procesamiento) aparecieron sistemas de traducción automática basados en frases, es decir, sistemas que entendían que grupos de palabras podrían tener significado propio. Últimamente se investigando el uso de modelos basados en *deep learning* [6] en la traducción automática. Este prometedor método lo estudiaremos más adelante, la idea principal consiste en entrenar un modelo con representaciones aleatorias del conjunto de datos cada vez más restrictivas, de esta manera el clasificador podrá aprender representaciones intermedias útiles de cara al objetivo final.

2.2 Sistemas de Diálogo Hablado

Hoy en día estamos más que acostumbrados a utilizar este tipo de sistemas tratando de buscar información o gestionar aplicaciones: *Siri* de *Apple* [3], *Cortana* de *Microsoft* [19] y *Google Now* [11]. Pero también se construyen sistemas para otros múltiples usos: ayudar en terapia a personas discapacitadas [8], existen avatares para enseñar técnicas de negociación o de entrevistas, o incluso para ayudar a tomar decisiones médicas. El procedimiento de los sistemas de diálogo hablado se divide en tres fases:

- ***Automatic Speech Recognition*** *(ASR)*: fase de identificación de lo que dice la persona.

- ***Dialogue Management*** *(DM)*: fase que trata de determinar lo que la persona quiere y realizar la acción solicitada.
- ***Text to Speech*** *(TTS)*: fase encargada de devolver la información a la persona en forma de voz de lo que dice la persona.

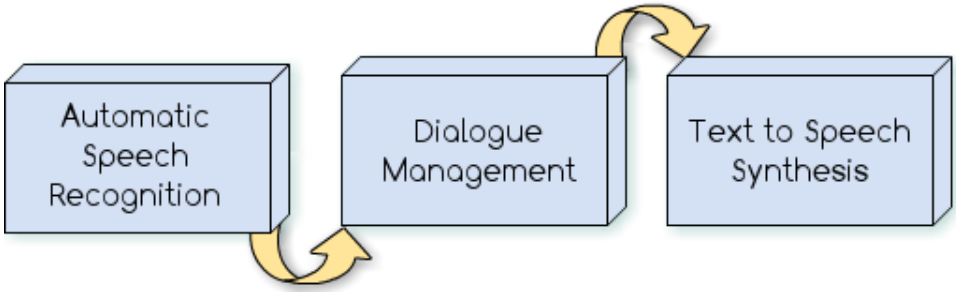

Figura 2.1: Sistemas de diálogo hablado.

Existe también una fase que se inicia al producirse un error de reconocimiento de voz, donde el sistema trata de deducir las palabras que no se han reconocido y trata de determinar lo que quería decir el usuario. Ésto puede ocurrir internamente o dialogando con la persona. En el caso de los sistemas de traducción voz a voz (*speech to speech translation systems*) también requieren tener una fase de traducción automática que les permita identificar problemas de traducción y aclararlos con los usuarios.

El progreso de esta tecnología se debe al desarrollo de técnicas de *deep learning* que asignan señales acústicas con secuencias de palabras y sus sonidos en lenguaje humano. Aunque actualmente sólo funcionan correctamente en ámbitos cerrados donde el vocabulario es predecible. Todavía quedan muchos retos por cumplir en este ámbito, como aumentar la precisión en ámbitos menos restrictivos, o conseguir que los sistemas adquieran comportamientos humanos (entender las pausas y coordinarse, usar expresiones similares al hablante en busca de empatía, etc).

2.3 Lectura Automática

Hablamos de lectura automática para referirnos a programas que pueden interpretar textos humanos. De esta manera pueden resumir información y sacar conclusiones de ellos. El objetivo de esto radica en el crecimiento que experimenta la web hoy en día, donde cada vez hay más y más información, de forma que muchas veces es inviable mantenerse al día con la literatura en ciertos campos [2].

Los mejores resultados en este campo se obtienen de técnicas de aprendizaje automático. Se construyen clasificadores que proporcionan relaciones entre palabras basándose en características extraídas antes del entrenamiento (secuencias de palabras, estructura gramatical, etc). En los últimos años, los científicos han tratado de construir bases de datos estructuradas con información de textos científicos con el fin de facilitar su trabajo. Sin embargo, construir

sistemas de inteligencia artificial que hagan esto automáticamente sigue siendo hoy en día objeto importante de investigación.

Actualmente, se sigue intentando mejorar la capacidad de los ordenadores para construir bases de datos a partir de información en formato de texto, ya que, a pesar de todas las técnicas de extracción existentes, cualquier base de conocimiento sólo será parcial e incompleta. Algunos trabajos recientes buscan complementarlos de manera probabilística, tratando de imitar el sentido común.

2.4 Minería de Medios Sociales

En los últimos años, el crecimiento que han experimentado redes sociales como *Twitter*, *Facebook*, *YouTube*, *Instagram* y otros muchos más, ha sido abrumador. La cantidad de información almacenada en dichas redes es enorme [24]. Podemos obtener y analizar datos demográficos, usos del lenguaje, temas de moda, opiniones sobre determinados productos, pensamientos políticos, interacción entre grupos de personas, etc. Ésto ha llamado la atención de gran cantidad de negocios, políticos, periodistas y anunciantes que buscan adquirir información relevante en su campo así como usar dicha información para su beneficio. Por ejemplo, encontrar opiniones acerca de un producto permitiría a una empresa mejorar sus características o incluso optimizar los anuncios del mismo.

Los datos de las redes sociales existen en todo tipo de idiomas (HRLs o LRLs) y pueden ser útiles de cara a optimizar modelos de lenguaje ASR y TTS sin la necesidad de crear corpus específico [2]. También pueden ser útiles en la producción de sistemas de diálogo en zonas de lenguajes con pocos recursos. Una aplicación específica de la minería de datos en LRLs es la extracción de datos de Twitter o blogs para obtener información valiosa con el fin de identificar los problemas más graves, donde se producen y quién los sufre.

A día de hoy, uno de los grandes problemas de la minería de datos de las redes sociales es la extensa preocupación por la privacidad. Existe un constante tira y afloja entre los propietarios de los datos y los investigadores que quieren extraerlos. Otro problema principal radicaría en la validez de los datos. No hay prcedimientos claros para validar la información encontrada referente a datos demográficos de las personas, opiniones de hoteles, restaurantes y productos, etc.

2.5 Information Retrieval

La recuperación de información (o *information retrieval*) [28] es un área del campo de procesamiento de lenguaje natural que se hace cargo de la representación, almacenamiento, organización y recuperación de todo tipo de elementos informativos (documentos, páginas web, imágenes, música, vídeo, etcétera). Los motores de búsqueda son grandes herramientas que proporcionan gran accesibilidad a información específica, lo que ha potenciado el crecimiento y la usabilidad de internet.

Uno de los problemas con los que debe lidiar todo sistema de recuperación de información es el uso de descripciones imprecisas e incompletas, tanto de los usuarios como de los documentos solicitados. Muchas veces es difícil encontrar una buena representación para las consultas realizadas o documentos buscados. Los usuarios suelen utilizar mensajes cortos y descripciones ambiguas en busca de que el sistema complete la información necesitada, por lo que normalmente el proceso de búsqueda debe ser concebido como ”prueba y error". El sistema analizará la consulta enviada y tratará de proporcionar las mejores respuestas posibles estimando la importancia de la misma. La gran dificultad de esta tarea se encuentra en tres características principales de todo lenguaje: la polisemia, la sinonimia y los errores de ortografía. Cuando una palabra o acrónimo tiene más de un significado, y además éstos no tienen relación entre ellos, es complicado escoger el correcto (o más aún, el que el usuario quiere encontrar). Lo mismo ocurre cuando un evento o concepto determinado puede ser representado con múltiples palabras.

La probabilidad de que dos personas utilicen el mismo término para describir el mismo objeto es menor de un 20 % (tal y como se demuestra en [10]). Teniendo en cuenta estos datos y los problemas anteriores, ¿cómo un motor de búsqueda puede operar en un tiempo razonable con tanta varianza? Estos sistemas asumen que palabras similares tienen significados similares, por lo que al realizar una consulta se comparan las palabras de la misma con las del documento y se establece un ranking según dicha comparación.

2.6 Clasificación automática de textos

Hasta ahora hemos hablado sobre los diferentes ámbitos en desarrollo del procesamiento de lenguaje natural. En esta sección, así como durante el resto del trabajo, centraremos nuestros esfuerzos en desarrollar teoría y práctica sobre clasificación de textos. Comentaremos los problemas y técnicas existentes en profundidad y realizaremos un análisis de aquellas que proporcionen mejores resultados en nuestro estudio.

2.6.1 Problemas de clasificación

Uno de los principales problemas de clasificación que encontramos actualmente es el conocido **análisis de sentimientos** [27]. Esta tarea consiste en la extracción de **opiniones** de un texto, determinando la positividad o negatividad que el autor ha expresado.

¿Por qué es importante este problema hoy en día? Toda empresa o negocio tiene en cuenta siempre el pensamiento de sus consumidores y/o empleados de cara a la realización de sus productos, anuncios y administración. Gracias al potente desarrollo que ha sufrido *Internet* en los últimos años, existen a libre disposición cientos de opiniones en la red (webs y aplicaciones de valoración de productos, redes sociales, blogs, etcétera) y todos estos negocios quieren sacar provecho de ésta información. [22]

La versión más simple del análisis de sentimientos es una clasificación binaria, representando la clase positiva como 1 y negativa como 0. Veamos, por ejemplo,

las siguientes frases extraídas de nuestra primera base de datos de sentimientos [26]:

- Texto positivo: *Ï thought Motorola made reliable products!. Battery for Motorola Razr. Great for the jawbone.¨*
- Texto negativo: *Ïf money is no constraint then just go for it. Try texting fast with it... What a waste of money and time!¨*

Este campo posee grandes limitaciones a la hora de clasificar, desde la ironía al hablar, las oraciones condicionales y otro tipo de expresiones que condicionarán nuestro algoritmo. Parece que en este problema la representación y el preprocesamiento de los textos será crucial con el fin de obtener una buena clasificación.

Otro problema importante de clasificación binaria es la **detección de spam**, es decir, identificar si un correo electrónico o mensaje es *spam* o no. En el caso de los correos electrónicos tiene especial peso, ya que el porcentaje de *emails* que corresponden a la clase de *spam* es del 75-80 % además de que produce serios problemas económicos y legales tales como un mal uso del tráfico de red, saturación del espacio de almacenamiento, aplicación de técnicas de fraude y distribución de anuncios pornográficos, entre otros [7]. En estos casos, técnicas que detecten expresiones como ”without any cost" o ”you won" suelen tener gran peso en la labor de clasificación.

Algo similar ocurre en el caso de mensajes de spam en web [4], es decir spam alojado en forma de comentario en foros, *reviews* y demás. En estos casos identificar *URLs* y expresiones similares a las anteriores son técnicas efectivas.

Ahora pasamos a un problema no binario conocido como **identificación de autor**, donde el objetivo es determinar quién es el autor de un texto [15]. Hoy en día conocer la edad, género y lenguaje nativo del autor proporciona información valiosa para las ciencias sociales, humanidades, medicina forense y, por supuesto, el procesamiento del lenguaje natural.

En definitiva, existen multitud de problemas que resolver en clasificación de textos. En este trabajo nos centraremos exclusivamente en problemas binarios y trataremos de encontrar las mejores vías de aprendizaje automático para solucionarlos.

2.6.2 Representación de los textos

En clasificación de textos tratamos de obtener un programa que, a partir de un conjunto de datos ya clasificados, adquiera una buena capacidad de generalización que le permita clasificar correctamente nuevos textos desconocidos. Para obtener un modelo preciso es necesario realizar un buen preprocesamiento de datos que permita al clasificador entrenarse con palabras y expresiones puramente significativas. En caso contrario, una mala representación de los textos limitará la capacidad de generalización de nuestro clasificador y podría incurrir errores. Esta fase tiene un peso muy grande dentro de todo el proceso como comprobaremos en el siguiente capítulo.

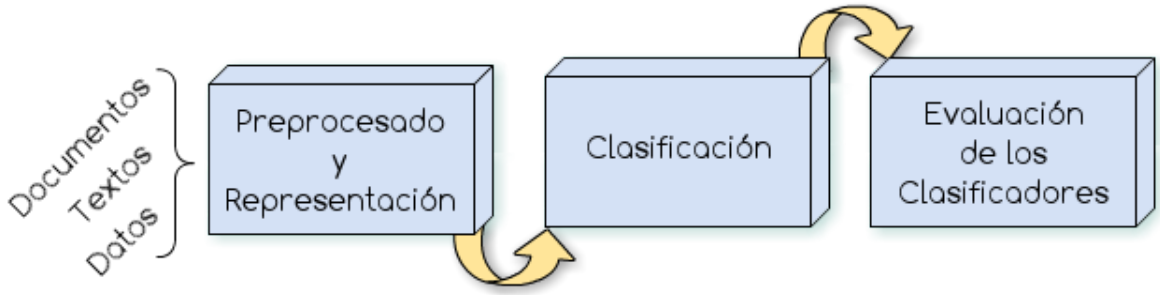

Figura 2.2: Clasificación de textos.

Hasta el día de hoy son varios los tipos de representación existentes. Uno de los más conocidos y utilizados, debido a la facilidad de implementación y su simplicidad, es el método de ***Bag of Words*** (BoW). Con esta representación obtenemos un vector cuyas componentes representan el número de veces que aparece cada palabra del *corpus*. A partir de BoW aparecen representaciones como los N***-gramas***, cuyo funcionamiento es similar pero esta vez escogiendo N palabras por cada componente.

Por otro lado nos encontramos ***tf-idf*** (*Term Frequency Inverse Document Frequency*) [17]. Esta representación determina la importancia de una palabra en función del número de apariciones de la misma respecto del *corpus* completo.

Actualmente se están desarrollando métodos basados en *deep learning* para obtener representaciones más eficientes, como por ejemplo ***Word2Vec*** y ***Doc2Vec***. El método de *Word2vec* se encarga de representar cada palabra con un vector teniendo en cuenta el resto de palabras del *corpus*, es decir, palabras similares mantendrán posiciones similares. En *Doc2vec* se va un paso más allá representando todo un texto haciendo uso de los vectores de *Word2vec*.

2.6.3 Clasificadores de texto

Tal y como ocurre con las representaciones, también existen muchos clasificadores de textos diferentes. *Naïve Bayes* es úno de los más conocidos, el cual se caracteriza por asumir la independencia de cada uno de los atributos utilizados para representar el documento. A pesar de que la eficiencia de este método es algo inferior a otros clasificadores en determinadas circunstancias [5], se ha demostrado la validez de este método en numerosos problemas reales [29] por lo que será uno de los clasificadores que escogeremos en nuestro estudio.

Otro método de clasificación muy utilizado es *Support Vector Machines* (SVM), cuyo objetivo consiste en trasladar el problema a un espacio vectorial para luego tratar de encontrar un hiperplano que separe los vectores de una clase del resto. Este clasificador proporciona mejores resultados en grandes bases de datos [15] aunque los tiempos de clasificación aumentan drásticamente.

Existen multitud de clasificadores importantes tales como *Decision Tree*, *Random Forest*, *Máxima Entropía*, etcétera. A lo largo de nuestro estudio nos centraremos en estudiar y comparar los más conocidos y utilizados.

3 METODOLOGÍA

Comenzaremos esta sección definiendo algunos conceptos básicos de aprendizaje automático y de procesamiento de lenguaje natural para posteriormente poder presentar diferentes técnicas de clasificación de textos. Trataremos de mostrar técnicas tradicionales así como las más utilizadas actualmente.

3.1 Conceptos básicos

Corpus: definimos *corpus* (o *dataset*) como el conjunto de datos que utilizamos para entrenar nuestro sistema de predicción (*training*) así como el conjunto de datos que utilizaremos para comprobar el buen funcionamiento del mismo (*test*). Éste conjunto posee *documentos* y sus *clases*.

Documento: definimos *documento* o instancia como cada una de las unidades de texto dentro de nuestra base de datos. Particularmente, en el problema de análisis de sentimientos, un documento es un texto que expresa una opinión o emoción.

Característica: definimos *característica* como cada uno de los atributos que describen propiedades de los documentos del *corpus*. Una selección de estas características son las que utilizarán nuestros clasificadores para determinar la clase de un documento.

Modelo: una vez nuestro sistema ha sido entrenado, se crea un modelo que servirá para realizar las predicciones a partir de datos nuevos. En el *training* se habrán detectado los patrones necesarios para ello.

3.2 Preprocesamiento

En la clasificación de textos, buscamos optimizar los resultados de nuestro clasificador. Por ello, dependiendo del idioma y el contexto en el que estén escritos, puede ser necesaria una fase inicial de preprocesamiento. Es decir, transformar el corpus de entrada de manera que simplifiquemos la estructura del mismo manteniendo su esencia.

Esta tarea es una fase muy importante ya que no solo nos facilitará el posterior análisis de los textos sino que también modificará los patrones descubiertos por nuestro clasificador.

Podemos encontrar diferentes técnicas de preprocesado que explicaremos a continuación.

Tokenización: esta fase consiste en separar nuestro documento de entrada en palabras o frases a partir de determinados caracteres especiales (también llamados *tokens*). Los *tokens* nos indican el fin de una palabra o sentencia y el comienzo de la siguiente.

En nuestro caso ésta es nuestra primera acción. Tokenizamos la base de datos de sentimientos por opiniones. Cada comentario de usuario es tomado como si fuera una única sentencia con un sentimiento determinado.

Normalización: en ocasiones podemos encontrar gran variedad de términos que reflejan una misma idea o concepto. La normalización es una tarea que consiste en unificar el sistema de representación de dichos términos.

Un ejemplo de ésta tarea sería sustituir ”coche", ”carro", ”auto" por sólamente una de ellas y así unificaremos su uso.

Lematización: esta tarea también permite unificar términos que transmiten una misma idea, pero esta vez consiste en reemplazar cada una por su *lema*. Por definición, un lema representa todas las palabras derivadas de ella, pero no necesariamente tiene que estar formada por la misma raíz (por ejemplo, el lema de la palabra ”mejor" es ”bueno" y el lema del verbo ”come" es ”comer").

Para encontrar el lema de una palabra tratamos de quitar todo lo añadido (conjugaciones, grado, persona, género, número, etc).

Stemming: otra forma de unificar términos consiste en obtener la raíz (o *stem*) de los mismos. Para ello eliminamos las terminaciones de cada palabra, de forma que todas quedan representadas por una pequeña parte de las mismas.

En esta tarea, a diferencia del método usado en la lematización, las representaciones pueden ser términos no existentes en el vocabulario. Pero esta tarea no representa una mejora en clasificación de textos.

Detección de negaciones: en la clasificación es importante detectar todos aquellos términos cuyo significado puede ser ambiguo. Aquí trataremos de identificar si el significado de un término es opuesto al habitual en un determinado ámbito debido a que aparece negada. Una vez identificada una palabra negada, una posible modificación consiste en añadir un prefijo identificativo (por ejemplo ”not") a todos los términos posteriores a la negación hasta el siguiente signo de puntuación.

Este procesamiento es de gran importancia en clasificación de textos, particularmente en el análisis de sentimientos ya que con esta especificación evitamos considerar un término negativo como relevante en comentarios positivos, dándole a su vez valor positivo al término negado y viceversa.

Es importante destacar que la detección de la negación es diferente en un idioma que en otro. En inglés encontramos negaciones buscando la terminación ”nt" o las palabras ”not" y ”no".

Part of Speech Tagging: esta tarea, también conocida como *POS Tagging* consiste en clasificar cada término de un texto en función de su contexto y de la función que desempeña (verbo, adjetivo, adverbio, determinante, etc).

El objetivo de esto es diferenciar palabras iguales con distinta funcionalidad (*Word Sense Disambiguation*). Por ejemplo, la palabra ”coma" puede actuar como conjugación del verbo ”comer" o como sustantivo refiriéndose al signo de puntuación.

Esta técnica tiene gran utilidad en análisis de sentimientos. Ya que un adjetivo contiene información importante en lo que a opinión se refiere, y diferenciarlo de cualquier otro tipo puede llegar a ser determinante al clasificar un texto.

Inicio/Fin de frase: cuando realizamos una extracción a modo de bigramas, no nos interesa que nuestro clasificador considere el par de palabras que contiene el final de una frase y el inicio de la siguiente. Para evitarlo podemos añadir una etiqueta al inicio (<s>) y al final (</s>) de cada sentencia, lo que nos permitiría detectar estos casos al extraer los bigramas.

Preprocesamiento Básico: además de todo lo mencionado anteriormente, existen otros cambios posibles que pueden incrementar la efectividad de nuestro clasificador. Estas tareas son más simples y podemos describirlas rápidamente:

- Eliminar términos o secuencias de palabras que no proporcionen información adicional (caracteres especiales, signos de puntuación, números, capitalizaciones, emoticonos, caracteres repetidos, *stopwords*, palabras de menos de N letras, tildes, etc). En algunos casos, los emoticonos y las capitalizaciones pueden aportar información adicional y hay que tenerlos en cuenta.
- Corregir errores ortográficos para mejorar la precisión de nuestro clasificador.

3.3 Extracción de Características

Durante esta sección, analizaremos diferentes métodos de extracción de características. Esta tarea es de gran importancia en clasificación de textos ya que, las características (o *features*) que seleccionemos, son las que servirán posteriormente para entrenar el clasificador y determinar qué atributos debe tener en cuenta de cada texto donde desconoce la clase.

La selección de características varia en función del tipo de documentos que estemos tratando. Puede que determinados *features* no aporten información valiosa al clasificador o incluso lo confundan en algunos casos, pero puede que en otros sean de gran utilidad.

Si nuestra tarea de clasificación consistese en reconocer nombres de lugares o personas (tarea conocida como *Name Entity Recognition*), una característica

determinante que debemoríamos seleccionar es ”Empieza con mayúscula" que tomaría dos posibles valores: ”true" o ”false".

Nuestro objetivo es encontrar el mayor número de características con información relevante que sea posible. A mayor número de estos atributos, mayor precisión tendrá nuestro clasificador, y los resultados se ajustarán mejor a la realidad. Pero debemos tener cuidado ya que, según se van añadiendo variables, aumenta el coste de entrenamiento de nuestro clasificador y además el riesgo de que nuestro clasificador cometa errores de generalización es mayor. Este problema se tratará en la siguiente sección.

Ahora vamos a comentar diferentes métodos de extracción de features que hemos utilizado en nuestro estudio:

3.3.1 Bag of Words (BoW)

Bag of Words es el método más común. Se encarga de tomar como características todas las palabras que aparecen en nuestro *corpus*. Aproxima la probabilidad de ocurrencia de una palabra sin tener en cuenta las palabras que la preceden. Podemos utilizarlo de dos maneras diferentes:

- Presencia de palabras: en cada sentencia o documento, comprobamos si aparece cada palabra del corpus ignorando el número de veces que la encontramos. En este caso tomamos ”1" para las palabras que aparezcan y ”0" para las ausentes.
- Frecuencia de palabras: en cada sentencia o documento, almacenamos el número de veces que aparece cada palabra del corpus. Este caso nos ha proporcionado mejores resultados en el caso de análisis de sentimientos.

3.3.2 N-gramas

Bigramas: este método se encarga de tomar como características pares de palabras de nuestro corpus. De esta manera nuestro clasificador tiene en cuenta las palabras que preceden a cada término de nuestro corpus y no proporciona valor a cada uno por separado.

3.3.3 Latent Semantic Analysis

La idea principal de ***Latent Semantic Analysis*** (o LSA) [15] es que los datos observados, es decir, la matriz que representa palabras por documentos, están generados por unas variables ”latentes", no observables directamente, relacionadas con la semántica del problema analizado. La teoría es que el número de variables latentes es mucho menor que la dimensionalidad aparente del problema, por lo que una manera de simplificar la información del problema es expresar la información de los documentos, o de las palabras, en esas dimensiones latentes.

La técnica estándar para determinar las dimensiones latentes es el algoritmo de descomposición en valores singulares (***Singular Value Descomposition***, SVD). Este algoritmo descompone de manera exacta la matriz de observaciones en el producto de tres matrices:

$$\underset{|V|\times c}{\begin{pmatrix} \\ X \\ \\ \end{pmatrix}} = \underset{|V|\times m}{\begin{pmatrix} \\ U \\ \\ \end{pmatrix}} \underset{m\times m}{\begin{pmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_m \end{pmatrix}} \underset{m\times c}{\begin{pmatrix} \\ C \\ \\ \end{pmatrix}}$$

La primera matriz tiene dimensión $|words| \times m$ donde cada fila representa una palabra y cada columna representa una de las m dimensiones latentes. Existen tantas dimensiones latentes como rango[1] tenga la matriz original. Estos nuevos m vectores son ortogonales entre sí y además están ordenados por la cantidad de varianza de los datos.

La matriz cuadrada central es diagonal de dimensión $m \times m$ cuyos valores diagonales se conocen como valores singulares (*sigular values*) ordenados de mayor a menor. Estos autovalores coinciden además con la varianza aportada por cada dimensión. Por tanto, la magnitud del valor de la diagonal i representa la importancia de la variable latente i. De esa forma, si queremos reducir la dimensionalidad del problema y reducir el número de variables latentes utilizaremos solamente aquellas con magnitud mayor. Por último la matriz restante de dimensión $m \times c$ donde se representan los documentos en forma de columna y las dimensiones en forma de filas ortogonales entre sí.

Por tanto, una codificación natural de cada palabra consistirá en utilizar la fila correspondiente de la matriz U. Dependiendo de si queremos mayor o menor dimensionalidad, nos quedaremos con más o menos columnas (cada columna está asociada a una variable latente diferente, y estas están ordenadas de mayor a menor importancia).

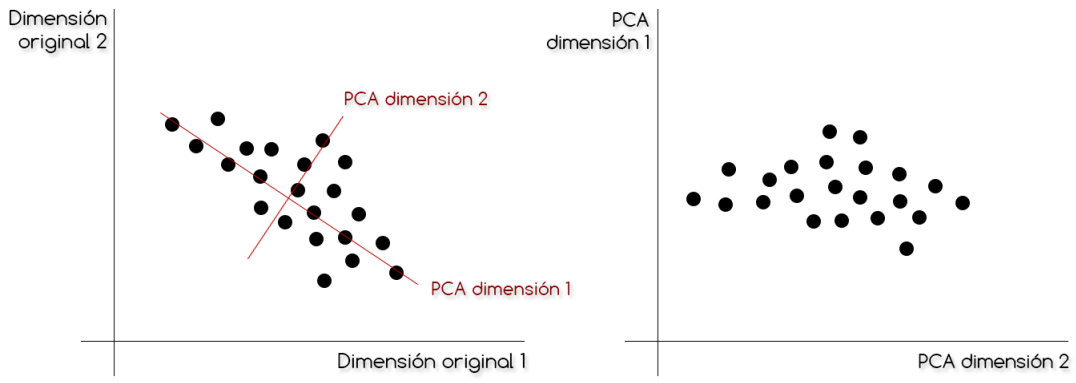

Figura 3.1: Esquema PCA

Otro algoritmo muy relacionado es PCA (*Principal Component Analysis*). Este algoritmo intenta descubrir las direcciones del espacio con mayor varianza, es decir, aquellas que capturan más información de los patrones (sin tener en cuenta

[1]El rango determina el número de filas linealmente independientes de la matriz

la clase, como hace también LSA). Para ello se calcula la matriz de correlación de los atributos, y se obtienen sus autovectores, ordenados de mayor a menor autovalor. El primer autovector nos dará la dirección de máxima varianza, el segundo la siguiente de mayor varianza, etc. Finalmente, los datos originales se proyectan sobre el conjunto de autovectores de mayor autovalor, obteniendo de esa forma una manera de representar la información con menos dimensiones.

3.3.4 Word2vec

Con el avance de la tecnología, la aparición del ***deep learning*** y la disponibilidad de grandes bases de datos, surgieron nuevos métodos de clasificación muy prometedores que hacen uso del ya mencionado *deep learning*.

Una de las técnicas de representación de textos que está generando más interés en los problemas de clasificación es ***word2vec***. Este método se inspira en los modelos tradicionales, partiendo de un gran conjunto de datos y tratando de extraer vectores de longitud fija para representar cada palabra del corpus.

Los métodos de representación tradicionales suelen ignorar la información del contexto o el orden de las palabras en los textos. Sin embargo, esta nueva representación (conocida en NLP como *word embedding*), hace desaparecer estas limitaciones, ya que los vectores generados por la red neuronal tienen un buen sentido semántico y sintáctico. Semántico en el sentido de que la palabra ”iOS" estará cerca de ”Android". Y sintáctico ya que palabras relacionadas como ”rey" y ”reina" estarán a una distancia similar que ”hombre" y ”mujer", dando a entender que ”rey es a reina como hombre a mujer".

3.3.5 Doc2vec

El método de *Word2vec* no permite combinarse fácilmente con métodos de clasificación tradicionales ya que para ello normalmente necesitamos un vector que represente un texto completo. La estrategia de ***Doc2vec*** se encarga de solucionar este problema representando no solo las palabras, sino frases y textos enteros, en vectores de misma longitud. Esto nos permite utilizar los métodos de clasificación anteriores tal y como si de vectores de *features* se tratasen.

Aunque los vectores proporcionados por *Word2vec* son precisos relacionando palabras, todavía no existe una técnica óptima para combinarlos en un vector de documento de buena calidad. Por ello en este estudio probaremos tres técnicas diferentes: calcular la media de los vectores, sumarlos proporcionalmente a su *tfidf* y por último una técnica inspirada en el conocido ”Proceso estocástico del Restaurante Chino" (también conocido en inglés como *Chinese Restaurant Process* o CRP). La idea de esta última técnica es utilizar CRP para realizar un proceso de *clustering* sobre los vectores y sumar los vectores que correspondan al mismo *cluster*.

Veamos un ejemplo: supongamos que tenemos un texto que muestra una receta de cocina. Este documento tendrá palabras referentes a comida y condimentos como ”chicken", ”pepper", ”salt", ”cheese"... Por otro lado tendrá expresiones como ”use", ”my", ”buy", ”the"... Con *Word2vec* obtenemos un vector para cada palabra. Podemos obtener un vector del texto completo haciendo la media de todos ellos, podemos realizar una suma ponderada con los pesos

proporcionados por el tfidf o podemos primero agrupar las palabras en *clusters*. De esta última manera palabras como ”chicken" o ”meat" podrían pertenecer a un mismo *cluster* y así permanecer lejos de palabras menos informativas. Si seleccionamos los *clusters* más significativos y sumamos simplemente las palabras que pertenecen a ellos, obtendremos un vector final muy representativo del documento.

3.4 Selección de variables

En la sección anterior hemos analizado los distintos métodos de extracción de variables del corpus. Pero es lógico pensar que no todas las variables escogidas son importantes a la hora de clasificar, por ello en esta sección explicaremos diferentes maneras de seleccionar las características más informativas.

Las variables menos informativas se conocen como **ruido** (o *noisy features*). No eliminar estas características no solo aumenta el coste de computación, sino que incrementa el error del clasificador. Por ejemplo, usando el modelo de BoW, tomamos la palabra ”the" como una característica. Esta palabra aparecerá en la mayoría de los textos, ya sean positivos o negativos, por lo que no proporciona ninguna información útil. Si ocurriese el caso de que esa palabra aparece más veces en casos positivos que en negativos, estaría llevando a nuestro clasificador a obtener conclusiones erróneas sobre dicha palabra. Éste último hecho se conoce como **sobreajuste** (u *overfitting*).

Los algoritmos de aprendizaje automático deben ser capaces de generalizar las condiciones de los datos de entrenamiento y usar lo aprendido para predecir nuevos casos. Cuando nuestro sistema se sobreentrena, obtenemos un clasificador ajustado a las características específicas de nuestro corpus de entrenamiento, por lo que durante esa fase el resultado será muy preciso, pero utilizando datos nuevos empeorará notablemente.

3.5 Clasificación de Textos

Nuestro estudio consiste en clasificar utilizando métodos de aprendizaje supervisado, ya que trabajaremos sobre conjuntos de datos ya clasificados previamente. Por ejemplo, en la tarea de análisis de sentimientos, cada texto del corpus está clasificado correctamente como positivo o negativo y tratamos de predecir la clase de otros textos desconocidos. En esta sección explicaremos el funcionamiento de los métodos utilizados y las conclusiones que podemos obtener de cada uno.

3.5.1 Tipos de clasificadores

Dentro de los métodos de aprendizaje supervisados, vamos a realizar una división de clasificadores en función de cómo aprenden el modelo:

- **Clasificadores Generativos**: son aquellos cuya labor de predicción de la clase C para un texto T se realiza maximizando la probabilidad condicional $P(C|T)$ de manera *indirecta*, es decir, a partir de la probabilidad $P(T|C)$. Primero determinamos la clase y tratamos de encontrar las características

que esperamos encontrar en el texto, después utilizamos el **teorema de Bayes** para encontrar la probabilidad que realmente estábamos buscando.

- **Clasificadores Discriminativos** (o Condicionales): éstos son aquellos cuya labor de predicción consiste en maximizar la probabilidad condicional $P(C|T)$ de manera *directa*, es decir, analizando la estructura interna de los datos y sin tener en cuenta cómo han sido generados.

Una diferencia importante entre ambos métodos es que los modelos generativos deben tratar primero un problema intermedio antes de resolver el problema, sin embargo los modelos discriminativos ofrecen una solución directa.

3.5.2 Naïve-Bayes

Este clasificador debe su nombre a dos condiciones importantes: primero, su método de clasificación está basado en el teorema de Bayes y segundo, asume que los atributos son independientes. Pero este último hecho es difícil que ocurra en la práctica, de ahí que esta "ingenua" premisa se conozca con el nombre de *Naïve Assumption*. Aun así, como comprobaremos más adelante en nuestras pruebas, esta condición no empeora normalmente los resultados que ofrece este clasificador.

Vamos a representar el teorema de Bayes, llamando C_i a clase posible y T es un texto de nuestro corpus de entrenamiento:

$$P(C_i|T) = \frac{P(C_i)P(T|C_i)}{P(T)}$$

Nuestro clasificador tratará de escoger la clase que maximice $P(C_i|T)$, es decir, la probabilidad de que el texto T pertenezca a la clase C_i. En el cálculo se puede ignorar $P(T)$ ya que este campo es el mismo para cualquier clase.

Considerando la condición de independencia de cada una de las características dada una clase vamos a descomponer $P(T|C_i)$ considerando las características (*features*) extraídas v_k:

$$P(C_i|T) \propto P(C_i) \prod_{k=1}^{n} P(v_k|C_i)$$

La clase seleccionada por nuestro clasificador será la que maximice dicha probabilidad. Es importante tener en cuenta que las probabilidades suelen tomar valores muy pequeños por lo que, para evitar errores de precisión, conviene realizar cálculos en forma logarítmica[2] quedando la expresión de la siguiente manera:

$$\boxed{log(P(C_i|T)) \propto log(P(C_i)) + \sum_{k=1}^{n} log(P(v_k|C_i))}$$

[2]El logaritmo es una función monótona creciente no genera problemas al maximizar

En nuestro trabajo utilizaremos ***Multinomial Naïve Bayes*** ya que se recomienda su uso cuando la los vectores de características muestran frecuencia en vez de aparición, por tanto obtendremos mejores resultados con este método.

3.5.3 Máxima Entropía

El método de clasificación de **Máxima Entropía** (MaxEnt), también llamado *Multinomial Logistic Regresion* [15], es un clasificador discriminativo que se basa en extraer una lista de características de los textos del corpus y buscar la distribución de probabilidad que cumpla con todas ellas, es decir, maximiza la entropía.

Las características tomadas para este modelo son funciones binarias que se definen con $\vec{V_j}$ un vector de *features* que representa el texto j y C la clase a predecir:

$$f_i(\vec{V_j}|C) = \begin{cases} 1 & \text{si la característica } i \text{ aparece en el texto } j \text{ y pertenece a la clase } c \\ 0 & \text{en otro caso} \end{cases}$$

El modelo de regresión logística se basa en extraer estas características de los textos y combinarlas linealmente aplicando una ponderación con pesos (w_i). El problema aparece al observar que esta linealización puede tomar valores reales. Para evitarlo se toma la exponencial (estrictamente positiva) y se normaliza (restricción a valores entre 0 y 1) quedando de la siguiente forma:

$$P(C_i|T) = \frac{1}{Z} exp(\sum_i w_i f_i)$$

Generalizamos esta expresión para obtener el modelo de **Máxima Entropía**:

$$P(C_i|T) = \frac{exp(\sum_{i=0}^{N} w_{ci} f_i)}{\sum_{c' \in C} exp(\sum_{i=0}^{N} w_{c'i} f_i)}$$

3.5.4 Support Vector Machines

El modelo de clasificación conocido como *Support Vector Machines* (o SVM) es un método que trata de separar en dos grupos los vectores de características que representan los textos del corpus. En este metodo se busca la separación más grande posible entre las dos clases a través de un hiperplano[3]. Los **support vectors** son los que determinan la máxima separación entre los dos conjuntos

A diferencia de los anteriores, este clasificador no tiene un trasfondo probabilístico, pero aun así mantiene la dependencia de las características para determinar la importancia de la información del corpus.

3.5.5 Árboles de decisión

El método de clasificación por árboles de decisión (o *Decision Trees*) consiste en utilizar los valores de las características extraídas del corpus para construir

[3]Un **hiperplano** es un elemento que divide el espacio de trabajo en dos partes. En $\mathbb{R}^1$ un hiperplano será un punto, en $\mathbb{R}^2$ será una recta y en $\mathbb{R}^3$ un plano. Se generaliza para más dimensiones.

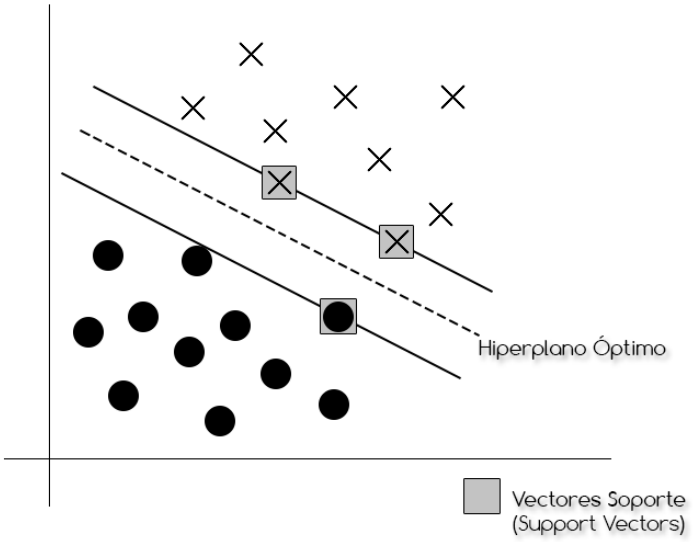

Figura 3.2: SVM.

dichos árboles. Una vez construido, clasificamos a partir de las reglas definidas en el árbol.

Veamos un ejemplo. Examinamos un problema de clasificación y finalmente tomamos el siguiente conjunto de contención de palabras para entrenar nuestro modelo: ”good", ”bad", ”not". Una vez entrenado nuestro clasificador’con los datos, podríamos obtener un arbol como la figura 3.2. De esta manera la clasificación quedaría como sigue:

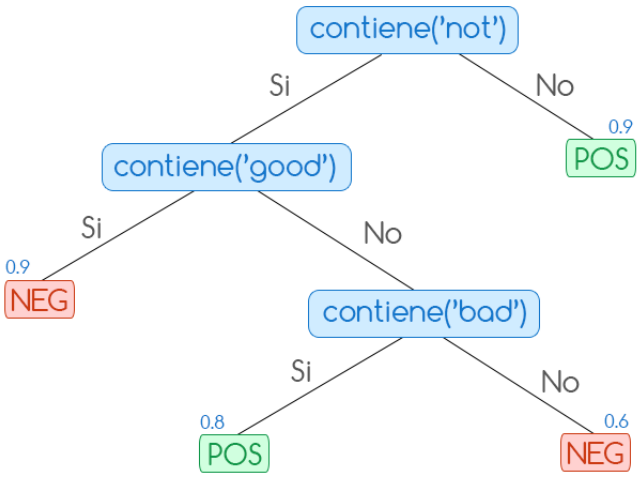

Figura 3.3: Árbol de decisión.

- Los textos que no contengan la palabra *not* serán calificados como positivos con un 90 % de probabilidad de acertar.

- Los textos que contengan la palabra *not* y además contengan la palabra *good* serán calificados como negativos con un 90 % de probabilidad de acertar.
- Los textos que contengan la palabra *not* y además contengan la palabra *bad* serán calificados como positivos con un 80 % de probabilidad de acertar.
- Los textos que sólamente contengan la palabra *not* serán calificados como negativos con un 60 % de probabilidad de acertar.

Entre todos los algoritmos que existen para construir árboles de decisión, en la práctica se han implementado los siguientes:

C4.5

Los árboles de este método se construyen tratando de encontrar, en cada nodo, la característica que maximice la ganancia de información para la clase en cuestión. Se forma un árbol del mayor tamaño posible para posteriormente acotarlo, mejorando así su capacidad de generalización sobre el corpus.

En los nodos pueden aparecer tanto valores contínuos como discretos. En el caso de valores contínuos, éstos se particionarán en un conjunto discreto de intervalos. Por otro lado, éste método transformará el árbol generado en un conjunto de reglas *if-then*, definiendo además el orden en el que deben realizarse. Tras esto, para evitar sobreajuste, se podan todas las condiciones que optimizan una regla al ser eliminadas.

Random Forest

Esta técnica se encarga de construir un conjunto de clasificadores basados en árboles de decisión. Cada uno de ellos se construye en base a un vector de números aleatorios, y todos esos vectores sean independientes entre sí. Una vez se tienen todos los clasificadores la manera de escoger el resultado final es a través de una votación no ponderada, es decir, el resultado maś repetido del conjunto de clasificación de todos los árboles. Un punto fuerte de este algoritmo es tratar de minimizar el error a través de la repetición del problema con un conjunto de árboles independientes.

Una manera sencilla para medir la efectividad de este tipo de clasificadores es utilizar la *función de margen*. Con el margen calculamos la porporción de árboles que aciertan y fallan, consiguiendo así una buena estimación de la seguridad de la predicción.

3.6 Evaluación de clasificadores

Tras construir un clasificador es de vital importancia conocer las metodologías que permiten evaluarlo adecuadamente, además es importante utilizar diferentes métricas para representar la eficacia del mismo con el fin de poder identificar la precisión de tus resultados y detectar errores.

Por ello vamos a presentar las distintas técnicas que se han utilizado para evaluar nuestros clasificadores a lo largo del estudio.

3.6.1 Métodos de evaluación

En la clasificación automática de textos existe un factor que es determinante a la hora de examinar los resultados, y esto es la división del *corpus* en datos de entrenamiento (*training set*) y datos de prueba (*test set*). La manera en que realicemos esta división y el número de veces que lo hagamos definirán la verdadera precisión de nuestro clasificador. Existen varios métodos que permiten hacer una buena división pero en este apartado vamos a explicar los método que hemos utilizado en nuestro estudio:

Hold out: esta es la manera más sencilla de evaluar un clasificador y consiste en dividir nuestro conjunto de datos en dos partes (*training* y *test*). Normalmente se utilizan porcentajes pequeños para el conjunto de prueba (20 %-30 %). Todos los resultados obtenidos con este método hacen referencia al conjunto de prueba elegido, pero ignoramos lo que ocurre con nuestro clasificador en otros casos. Por tanto, el hecho de obtener buenos resultados con este método, no quiere decir que nuestro clasificador sea bueno en cualquier ámbito.

Cross Validation: para implementar este método dividiremos nuestro conjunto de datos en k partes (cada una de éstas conocida como *fold*[4]) y repetiremos el método *Hold out* k veces. Cada vez que repitamos el proceso utilizaremos un *fold* diferente como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera todas las mediciones que realicemos sobre nuestro clasificador se harán tomando la media de todas las ejecuciones, obteniendo valores mucho más fiables sobre su eficiencia. En la figura 3.4 se muestra un ejemplo de división del conjunto de datos según este método.

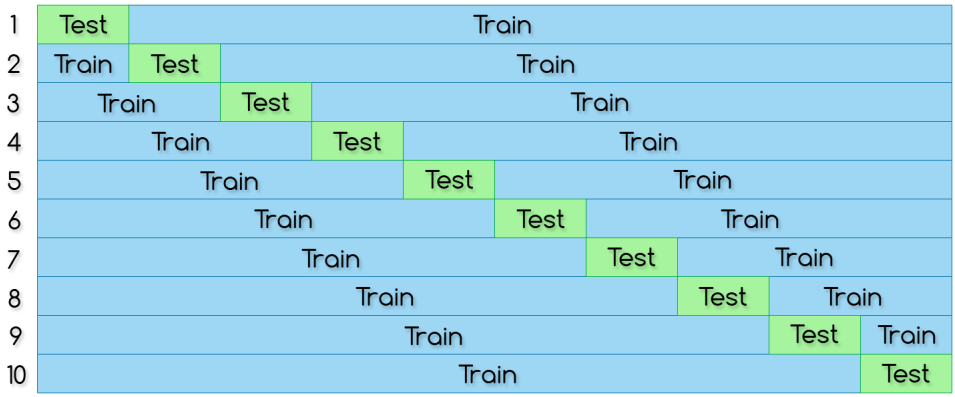

Figura 3.4: Ejemplo de división del corpus según la técnica de *cross validation* tomando $k = 10$.

3.6.2 Matriz de confusión

Esta matriz es una herramienta que nos permite visualizar rápidamente la precisión de nuestro clasificador desde múltiples perspectivas. En particular, la representación en tabla nos permite ver fácilmente si el sistema está confundiendo dos clases etiquetando de manera opuesta, y de ahí el nombre que se le atribuye.

[4]Por eso este método también se conoce como *k-fold cross validation*.

Esta matriz ordena la información de las clases determinando si el valor de predicción de nuestro clasificador ha coincidido con el valor real o, por el contrario, ha ido a parar a otra clase errónea.Las filas de nuestra matriz representan los valores que ha predicho nuestro clasificador, mientras que las columnas representan el valor real de esos datos.

Existen dos maneras de representar la matriz de confusión. Una primera forma variable, donde habrá tantas columnas y filas como clases tenga en cuenta nuestro sistema, y otra fija con dos filas y dos columnas, donde se escoge una clase en particular analizando si acierta o falla sin tener en cuenta el resto de clases.

Cada elemento de esta última representación recibe un nombre:

- Verdadero Positivo o *True Positive* (t_p): número de elementos a los que se les asignó la clase en cuestión y realmente pertenecían a ella.
- Verdadero Negativo o *True Negative* (t_n): número de elementos a los que no se les asignó la clase en cuestión y realmente no pertenecían a ella.
- Falso Positivo o *False Positive* (f_p): número de elementos a los que se les asignó la clase en cuestión pero realmente no pertenecían a ella.
- Falso Negativo o *False Negative* (f_n): número de elementos a los que no se les asignó la clase en cuestión pero ŕealmente pertenecían a ella.

	Clase	**No Clase**
Clase	t_p	f_p
No Clase	f_n	t_n

Tabla 3.1: Evaluación de Clasificadores - Matríz de Confusión.

Veamos un ejemplo de matriz de confusión para análisis de sentimientos, donde nuestras clases serán ”Texto positivo" y ”Texto negativo":

	Texto positivo	**Texto negativo**
Texto positivo	t_p	f_p
Texto negativo	f_n	t_n

Tabla 3.2: Evaluación de Clasificadores - Matríz de Confusión.

3.6.3 Métricas

Una vez introducidos los conceptos principales de la matriz de confusión, podemos dar paso a explicar diferentes métricas derivadas de ellos:

Precisión (o *Positive Predictive Value*): esta métrica representa la proporción de documentos clasificados correctamente como positivos (t_p) respecto al total clasificados como positivos (t_p + f_p).

$$\boxed{Precision_{POS} = \frac{t_p}{t_p + f_p}}$$

El *Negative Predictive Value* es un concepto similar pero con documentos clasificados correctamente como negativos:

$$Precision_{NEG} = \frac{t_n}{t_n + f_n}$$

Recall o *True Positive Rate*: esta métrica representa la proporción de documentos positivos clasificados correctamente (t_p) respecto al total de documentos positivos (t_p + f_n).

$$\boxed{Recall_{POS} = \frac{t_p}{t_p + f_n}}$$

El *True Negative Rate* es un concepto similar pero con documentos negativos clasificados correctamente (t_n) respecto al total de documentos negativos ($t_n + f_p$):

$$Recall_{NEG} = \frac{t_n}{t_n + f_p}$$

Accuracy: esta métrica representa la proporción de documentos clasificados correctamente (t_p + t_n) respecto al total (t_p + t_n + f_n + f_p).

$$\boxed{Accuracy = \frac{t_p + t_n}{t_p + t_n + f_n + f_p}}$$

Area Under the ROC Curve (o AUC): la curva ROC es una gráfica bidimensional que representa la proporción entre beneficios (*True Positive Rate* en el eje Y) y costes (*False Positive Rate* en el eje X). Muchos clasificadores se diseñan para producir una sola clase para cada instancia. Un ejemplo de ésto son los árboles de decisión.

En la figura 3.5 tenemos un ejemplo de una gráfica ROC con cinco puntos que representan diferentes clasificadores discretos. La curva queda representada en un espacio $[0, 1]$x$[0, 1]$, donde la diagonal $y = x$ representa la estrategia de clasificadores que escogen aleatoriamente sus clases. El punto $(0, 0)$ representaría un clasificador que nunca concluye valor positivo; de esta manera no comete errores por falso positivo pero obviamente tampoco hay ganancia por verdadero positivo. Una estrategia opuesta estaría representada por el punto $(1, 1)$. El punto $(0, 1)$ representaría el clasificador perfecto, donde todos los positivos son verdaderos.

En la figura 3.6 observamos una curva ROC completa generada al aplicar la técnica de *cross validation* sobre un conjunto de datos de prueba. En la gráfica se muestran 20 divisiones y la precisión obtenida en cada ejecución.

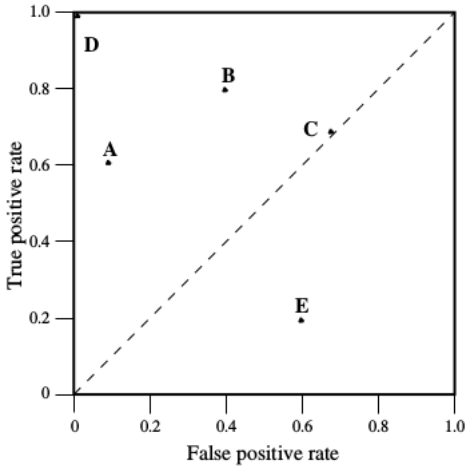

Figura 3.5: Ejemplo de gráfica ROC con cinco clasificadores discretos. Imagen obtenida de [9]

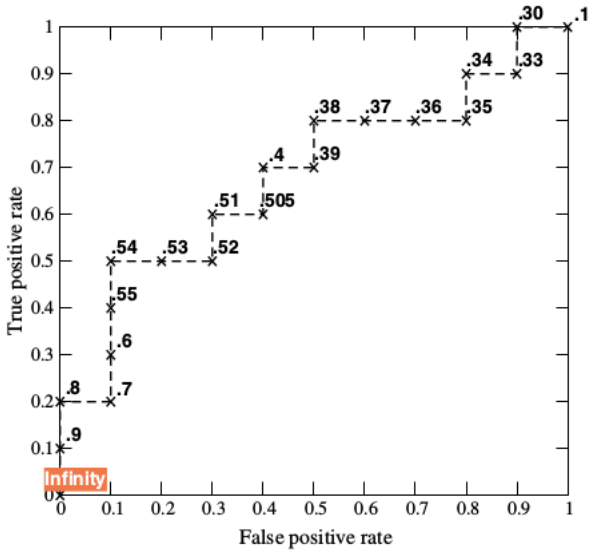

Figura 3.6: Ejemplo de una curva ROC. Imagen obtenida de [9]

3.7 Bases de Datos

En este trabajo hemos abordado dos problemas de clasificación de textos principalmente: análisis de sentimientos y detección una clase particular para la base de datos proporcionada por *Cognodata Consulting*.

Para trabajar en el análisis de sentimientos hemos utilizado dos bases de datos alojadas en la web. La primera y más pequeña se llama *Sentiment Labelled Sentences Data Set* [26]. Ésta base de datos está formada por tres documentos de diferentes contextos (*Yelp*, *Amazon* e *IMDB*). Cada documento contiene mil textos clasificados en inglés, de los cuales la mitad son positivos y la otra mitad son negativos:

Amazon: contiene comentarios y calificaciones de los productos vendidos en amazon.com en la categoría de teléfonos móviles y accesorios, y es parte del conjunto de datos recogidos por McAuley y Leskovec [18].

Yelp: contiene un conjunto de datos utilizados en el desafío Yelp, el cual contiene reseñas de diferentes restaurantes.

IMDB: contiene opiniones sobre películas del conjunto de datos IMDB para análisis de sentimientos. Este conjunto de datos es una versión reducida de nuestra siguiente base de datos [20].

Sin embargo, al observar que el número de textos no era suficiente para obtener grandes mejoras en la clasificación, pasamos a utilizar una base de datos completa de sentimientos llamada *Movie Review Data Set* [20], la cual está formada por 50.000 textos clasificados, la mitad como positivos y la otra mitad como negativos y otros 50.000 textos no clasificados.

La base de datos de incidencias, proporcionada por la empresa Cognodata Consulting, consta de 57.735 registros, cada uno descrito por un texto con la descripción aportada por el cliente u operador que la ha recogido. El problema consiste en predecir diferentes aspectos de la incidencia tales como el motivo de la incidencia (por ejemplo si es debido a un cargo indebido, o servicio no satisfactorio dado por la empresa, etc.), el canal a través del cual se ha recogido la incidencia (atención telefónica, email, etc.), o el producto sobre el que trata la incidencia.

La empresa nos da la información de los textos ya codificada. Para ello ha eliminado las stop words, y ha hecho una selección de las palabras más relevantes desde el punto de vista de negocio, quedándose con las 728 más representativas. La información de los textos que nos dan viene en formato BoW, donde hay tantas columnas como palabras en el vocabulario, y la aparición o no de las palabras está codificada como $0/1$ (1 si aparece la palabra en algún lugar del texto, 0 si no aparece).

3.8 Software utilizado

Para la realización de este estudio hemos utilizado dos entornos de programación diferentes: *Matlab* y *Anaconda*. Los programas desarrollados en *Anaconda* fueron implementados en *Python*.

Las librerías de *Python* utilizadas han sido las siguientes:

- ***Natural Language Toolkit*** (NLTK) [21]: esta librería especializada de *Python* permite un trabajo sencillo con datos lingüísticos. Proporciona gran cantidad de *corpus* de datos así como eficientes herramientas para tratar, procesar y clasificar textos. Dispone de un libro manual [25] donde podemos aprender todo lo necesario sobre la misma.
- ***Scikit Learn*** [23]: librería muy completa enfocada al procesamiento de lenguaje natural que, al igual que NLTK nos permite tratar texto de una forma sencilla así como clasificar con diferentes algoritmos de aprendizaje automático.

RESULTADOS

4

A lo largo de esta sección, mostraremos los resultados de las pruebas llevadas a cabo en este trabajo, aplicando toda la metodología explicada anteriormente. Vamos a realizar un análisis de los resultados de forma gradual, es decir, vamos a comenzar por estudiar el sistema más básico de representación (BoW) junto con los distintos preprocesamientos y metodos de clasificación en busca de la mejor combinación. Una vez encontremos un buen sistema de preprocesamiento de nuestros textos probaremos a cambiar el sistema de representación, los métodos de clasificación y las bases de datos.

4.1 Base de Datos: Sentiment Analysis

Las primeras implementaciones se realizaron trabajando sobre esta base de datos [26]. Empezamos con BoW junto con un preprocesamiento mínimo basado en convertir todas las palabras del texto en minúsculas (a) y poco a poco se irán variando las técnicas de preprocesado hasta completar el análisis.

Como los tres ficheros de esta base de datos pertenecen a ámbitos distintos, hemos decidido analizar cada uno por separado con el fin de que nuestro clasificador cometa los menos errores de generalización posibles. El análisis mostrado en esta sección será llevado a cabo por completo sobre el fichero de *Yelp* realizando una división de un 90 % *training* y 10 % *test*[1], además aplicamos la técnica de *cross validation* que nos permite obtener 10 pruebas y poder hacer un error que acompañamos en la tabla. Además, independientemente del sistema de representación utilizado, se realizará una selección de las 50 variables más significativas a la hora de realizar la clasificación:

[1] El motivo de esta división es que fue la que nos proporcionaba los mejores resultados dentro de este análisis

	Decision Tree	Random Forest	Máxima Entropía	SVC Kernel Linear	NB Multinomial
a	0.611 ± 0.069	0.628 ± 0.062	0.651 ± 0.089	0.648 ± 0.052	0.588 ± 0.052
b	0.618 ± 0.092	0.654 ± 0.036	0.659 ± 0.081	0.649 ± 0.041	0.611 ± 0.069
c	0.605 ± 0.045	0.669 ± 0.041	0.661 ± 0.079	0.655 ± 0.035	0.595 ± 0.075
d	0.632 ± 0.118	0.674 ± 0.046	0.672 ± 0.058	0.668 ± 0.042	0.597 ± 0.103
e	0.580 ± 0.023	0.579 ± 0.101	0.551 ± 0.069	0.603 ± 0.022	0.540 ± 0.029
f	0.618 ± 0.072	0.691 ± 0.063	0.694 ± 0.041	0.699 ± 0.051	$\mathbf{0.630 \pm 0.110}$
g	$\mathbf{0.647 \pm 0.083}$	$\mathbf{0.697 \pm 0.089}$	$\mathbf{0.699 \pm 0.076}$	$\mathbf{0.701 \pm 0.059}$	0.626 ± 0.094

Tabla 4.1: Tasa de acierto (en tanto por uno) en test de los diferentes clasificadores

Leyenda:

$a = BoW$
$b = BoW + Preprocesamientoimbolos$
$c = BoW + Preprocesamientoimbolos + Stopwords$
$d = BoW + Preprocesamientoimbolos + Stopwords + Stemming$
$e = BoW + Preprocesamientoimbolos + Stopwords + Stemming + PosTagging$
$f = BoW + Preprocesamientoimbolos + Stopwords + Negaciones$
$g = BoW + Preprocesamientoimbolos + Stopwords + Stemming + Negaciones$

Aclaración: a la hora de filtrar los símbolos eliminamos todos aquellos que no proporcionan información a excepcción de los emoticonos debido a su gran influencia en el análisis de sentimientos, por ello escogemos una palabra genérica para sustituir los que representan felicidad y otra palabra genérica para los que representan tristeza.

Podemos observar que, aunque las mejoras paso a paso no son excesivas, en la última prueba hemos alcanzado una mejora de aproximadamente un 5 % en el mejor clasificador (SVC con kernel linear). Además el resto de clasificadores también han mejorado su rendimiento notablemente, siendo *Random Forest* el que más ha incrementado su precisión (6.3 %).

Por otro lado cabe destacar que, para llevar a cabo la técnica de *POS Tagging*, hemos utilizado la extensión de NLTK llamada *Stanford POS Tagger*. Aunque la funcionalidad de esta librería es correcta, la velocidad con la que encuentra las tiquetas es extremandamente baja con esta pequeña base de datos. También es importante comentar el resultado tan negativo que produce en los resultados, donde rara vez supera el 60 % de precisión en media. Esto puede deberse a que la mayoría de palabras que se etiquetan tienen usos gramaticales diferentes. De esta manera la importancia de las palabras puede dividirse entre sus diferentes usos e incluso puede ser que muchas desaparezcan tras la selección de variables.

Una vez detectado el preprocesamiento que mejor resultados ofrece, lo mantendremos durante en el resto de pruebas. Tras utilizar el sistema más simple (BoW), es el momento de escoger pares de palabras (bigramas). Vamos a analizar y comparar los resultados con cada uno de ellos y sus combinaciones. Además compararemos también estos métodos cuando se les aplica una transformación tfidf:

	Decision Tree	Random Forest	Máxima Entropía	SVC Kernel Linear	NB Multinomial
a	**0.647 ± 0.083**	**0.697 ± 0.089**	**0.699 ± 0.076**	0.701 ± 0.059	0.626 ± 0.094
b	0.497 ± 0.013	0.497 ± 0.013	0.490 ± 0.020	0.496 ± 0.004	0.495 ± 0.015
c	0.635 ± 0.065	0.642 ± 0.098	0.683 ± 0.037	0.696 ± 0.044	0.624 ± 0.106
d	0.620 ± 0.070	0.676 ± 0.044	0.714 ± 0.046	**0.702 ± 0.078**	**0.687 ± 0.063**
e	0.499 ± 0.011	0.496 ± 0.024	0.494 ± 0.016	0.502 ± 0.018	0.495 ± 0.015
f	0.578 ± 0.072	0.652 ± 0.068	0.672 ± 0.068	0.682 ± 0.048	0.660 ± 0.050

Tabla 4.2: Tasa de acierto (en tanto por uno) en test de los diferentes clasificadores

Leyenda:

$$
\begin{aligned}
a &= BoW \\
b &= Bigramas \\
c &= BoW + Bigramas \\
d &= BoW + tf - idf \\
e &= Bigramas + tf - idf \\
f &= BoW + Bigramas + tf - idf
\end{aligned}
$$

Como se puede comprobar, los bigramas producen un efecto muy negativo en la clasificación. Esto es debido a que los bigramas no tienen apenas repeticiones entre las frases. No existen expresiones de dos palabras que coincidan con frecuencia en estos textos. Por esta razón no ha sido necesario realizar pruebas con mayores ”N-gramas".

También observamos que la aplicación de los pesos de *tfidf* antes de clasificar mejora un poco los resultados de todos los casos. Por tanto se entiende que los pesos determinados por esta técnica generan mejores vectores y por tanto es conveniente aplicarla.

Por último vamos a comparar las representaciones que basadas en la técnica de *word2vec*. Empezaremos utilizando el conjunto de vectores ya entrenados de *GloVe* [14] (*pre-trained vectors*) de dimensión 200 debido a que es la representación con la que mejores resultados hemos obtenido. En este caso, *a priori* no es posible utilizar técnicas de preprocesamiento cuya funcionalidad se base en aplicar etiquetas sobre las palabras ya que dichas palabras transformadas no tendrán un vector representativo en la base de datos de GloVe. Por ello decidimos hacer una primera prueba manteniendo simplemente un preprocesado básico, es decir, eliminando símbolos no significativos, *stopwords*, *lematización* y cambiando los emoticonos por las palabras ”happy" y ”sad" con el fin de que sean reconocidas como vectores. Pero por otro lado se nos ocurrió una segunda prueba manteniendo el preprocesado original y representando las palabras negadas con el vector opuesto al de la palabra original.

También hemos entrenado nuestro propio modelo obteniendo vectores que representen todas nuestras palabras. En este caso podremos utilizar la detección de negaciones sin problemas ya que tendrán su propio vector representativo. En la siguiente tabla comparamos los resultados de todas estas pruebas:

	Decision Tree	Random Forest	Máxima Entropía	SVC	SVC Kernel Linear
a	0.629 ± 0.091	0.788 ± 0.042	$\mathbf{0.817 \pm 0.043}$	$\mathbf{0.798 \pm 0.042}$	0.807 ± 0.053
b	$\mathbf{0.666 \pm 0.094}$	0.783 ± 0.057	0.794 ± 0.066	0.786 ± 0.064	0.789 ± 0.061
c	0.634 ± 0.046	0.786 ± 0.054	0.808 ± 0.032	0.766 ± 0.054	$\mathbf{0.822 \pm 0.038}$
d	0.640 ± 0.050	$\mathbf{0.789 \pm 0.071}$	0.805 ± 0.065	0.558 ± 0.042	0.805 ± 0.035
e	0.572 ± 0.038	0.657 ± 0.073	0.730 ± 0.040	0.722 ± 0.058	0.724 ± 0.046

Tabla 4.3: Tasa de acierto (en tanto por uno) en test de los diferentes clasificadores

Leyenda:

$$a = GloVe + mediadelosvecs$$
$$b = GloVe + mediadelosvecs + negaciones$$
$$c = GloVe + tf - idf$$
$$d = GloVe + tf - idf + negaciones$$
$$e = Doc2vec + mediadelosvecs$$

Aclaración: en Naïve Bayes no se permite la utilización de vectores de características con valores negativos por lo que se ha excluido este clasificador en el análisis y a cambio hemos incluido un clasificador SVC básico para observar las diferencias con el que utiliza Kernel Linear.

Revisando los resultados no podemos observar ninguna gran mejora en las diferentes técnicas pero podemos observar un gran cambio respecto a las anteriores pruebas, llegando incluso al 82 % de precisión. Llama especial la atención el bajo rendimiento de SVC básico cuando se está aplicando la técnica de *tfidf* junto con las negaciones y otro hecho a destacar es la diferencia de precisión de los que utilizan vectores preparados de GloVe respecto a nuestro clasificador Doc2vec. Esto tiene cierta lógica ya que los vectores que proporcionan en *GloVe* han sido generados a partir de millones de datos mientras que los nuestros simplemente han utilizado 1000 textos.

Por último hemos realizado un análisis en la plataforma de *Matlab* para probar la técnica de *LSA* junto con dos de nuestros clasificadores (SVC y Naïve Bayes). Vamos a realizar pruebas escogiendo vectores de diferentes longitudes y comprobando el efecto que tiene en el resultado una normalización de los mismos.

	SVC Kernel Linear	Naïve Bayes
a	0.583	0.506
a	0.585	0.506
b	0.599	0.595
b	0.580	0.580

Tabla 4.4: Tasa de acierto (en tanto por uno) en test de los diferentes clasificadores

Leyenda:

$$a = LSA + tf - idf$$
$$b = LSA + Normalizar + tf - idf$$
$$c = LSA + tf - idf + CRP$$
$$d = LSA + Normalizar + tf - idf + CRP$$

Como se puede comprobar, aplicando el algoritmo de LSA obtenemos resultados mucho peores de los obtenidos anteriormente. Podemos destacar la mejora producida en *Naïve Bayes* al realizar la normalización, aumenta hasta casi un 9 %, mientras que en SVC todo permanece estable y con valores muy bajos.

4.2 Base de Datos: IMDB

Sobre esta base de datos vamos a mostrar una tabla representativa con los mejores métodos de cada una de las fases anteriores. Al ser un corpus bastante más grande es probable que obtengamos resultados diferentes:

	Decision Tree	**Random Forest**	**Máxima Entropía**	**SVC**	**SVC Kernel Linear**
a	**0.670 ± 0.040**	**0.820 ± 0.030**	0.850 ± 0.035	0.515 ± 0.060	0.835 ± 0.025
b	0.664 ± 0.041	0.814 ± 0.042	**0.856 ± 0.024**	0.662 ± 0.033	**0.871 ± 0.030**
c	0.658 ± 0.042	0.773 ± 0.037	0.820 ± 0.031	**0.727 ± 0.064**	0.818 ± 0.048
d	0.657 ± 0.078	0.781 ± 0.035	0.783 ± 0.067	0.688 ± 0.067	0.796 ± 0.054

Tabla 4.5: Tasa de acierto (en tanto por uno) en test de los diferentes clasificadores

Leyenda:

$$a = BoW$$
$$b = BoW + tf - idf$$
$$c = GloVe + mediadelosvecs$$
$$d = GloVe + mediadelosvecs + tf - idf$$

Observando los resultados obtenidos, parece que esta vez el análisis con BoW a aumentado su precisión de forma notable, hasta un 87 % en el mejor de los casos. Parece ser que en bases de datos de mayor tamaño, la capacidad de generalización de algunos clasificadores es mucho mayor y nos permite obtener resultados de gran calidad. Lo más sorprendente es que en estos casos, la simple representación con BoW, supera a las que utilizan *Word2vec* y, en la mayoría de ellos, con un porcentaje bastante amplio. Como consecuencia de estos resultados podemos determinar que el sistema de preprocesamiento escogido ayuda mucho al clasificador en su capacidad de generalización de los datos y además parece ser una fase muy importante en este proceso.

4.3 Base de Datos proporcionada por Cognodata Consulting

Para el análisis de esta base de datos hemos escogido SVC con Kernel Linear como clasificador principal ya que es el que mejores resultados nos ha proporcionado en las anteriores pruebas.

En esta sección vamos a poner a prueba las tres versiones de nuestra representación LSA, aplicando en distintos momentos variantes como la normalización de los vectores o aplicar tfidf.

Comenzamos analizando un problema binario sobre el tipo de familia de producto, en nuestro caso queremos comprobar su correspondencia con el segundo tipo, de esta manera nuestro clasficador abandonará el análisis de sentimientos para centrarse en comprobar si nuestros documentos pertenecen a esta clase o no.

	SVC Kernel Linear	**Naïve Bayes**
a	0.803	0.561
b	0.823	0.577
c	0.723	0.671
d	0.757	0.724

Tabla 4.6: Tasa de acierto (en tanto por uno) en test de los diferentes clasificadores

Leyenda:

$$a = LSA + tf - idf$$
$$b = LSA + Normalizar + tf - idf$$
$$c = LSA + tf - idf + CRP$$
$$d = LSA + Normalizar + tf - idf + CRP$$

En este nuevo problema de clasificación hemos implementado LSA, otro de los métodos que explicamos en el cápítulo de Metodología. Para llevar a cabo este método hemos establecido el número de dimensiones en 200, de forma que los vectores tendrán el mismo tamaño que los que usamos con GloVe anteriormente.

La tabla anterior nos muestra la comparación de diferentes representaciones. Como podemos observar aquellas que implementan la normalización de los vectores suponen una mejora sustancial que hace ver que esta idea se complementa en gran manera con el método de LSA. Los resultados mejoran tanto si utilizamos la técnica CRP como si no. En los casos en los que aplicamos CRP hemos establecido el número de clusters en 100 y podemos observar que la técnica de Naïve Bayes mejora notablemente al aplicar este método, mientras que SVC, el clasificador que mejores resultados nos había ofrecido hasta el momento, disminuye su rendimiento hasta casi un 10 %.

5 Conclusiones y Trabajo Futuro

En este Trabajo de Fin de Grado hemos llevado a cabo una profunda investigacion sobre aprendizaje automático y las técnicas utilizadas actualmente para clasificación automática de textos. Comenzamos presentando el concepto de **Procesamiento de Lenguaje Natural** y los distintos ámbitos donde se aplica. Este estudio inicial nos permitió comprender las bases de este sistema y las limitaciones que podríamos encontrar. Posteriormente desarrollamos también las diferentes técnicas de preprocesamiento de textos, los métodos de clasificación escogidos para el trabajo y algunos métodos de evaluación posibles para la eficiencia de los clasificadores.

Para dar sentido a nuestra investigación implementamos un conjunto de programas, sobre dos plataformas diferentes, que clasificaban y evaluaban automáticamente textos procedentes de bases de datos en dos idiomas (español e inglés). Estos programas se probaron en multitud de contextos con el fin de establecer unas buenas condiciones iniciales de estudio (vectores de características de 50 componentes, 90 % de datos para *training* y 10 % para test en *cross-validation*, etcétera) y posteriormente realizamos un análisis de eficiencia de todos los clasificadores presentados inicialmente.

Todas las pruebas realizadas muestran que los distintos métodos de preprocesamiento de texto, la selección de características, la proporción en los conjuntos de datos de *training* y *test*, el tamaño del *corpus* utilizado y la proporción de documentos de cada clase tienen un gran impacto en la eficiencia del clasificador.

Además hemos encontrado que clasificadores supervisados como ***SVC con Kernel Linear***, ***Random Forest*** y ***Máxima Entropía*** son métodos de aprendizaje automático que, por norma general, proporcionan mejores resultados que el resto en las mismas condiciones, mientras que algoritmos como *Decision Tree* y *Naïve Bayes* no proporcionan tan buenos resultados. Cabe destacar también que la precisión de SVC con Kernel Linear es mucho mayor que la de su correspondiente modelo con kernel por defecto.

También hemos mostrado que la fase de preprocesamiento de texto influye de manera considerable en los resultados del clasificador y presentamos tablas con las mejoras de precisión y rendimiento obtenidos en cada caso. En particular los mejores resultados los hemos obtenido a partir de representaciones derivadas de ***deep learning*** y ***word2vec*** llegando a alcanzar un 82 % de precisión en los mejores casos. En particular estos resultados surgieron de utilizar vectores entrenados de la base de datos de *GloVe* los cuales fueron combinados con técnicas como tfidf o la media de los mismos. Cabe destacar la gran utilizad que tiene la técnica de ”detección de negaciones" en problemas de análisis de sentimientos. Al introducir dicha técnica hemos obtenido importantes mejoras en la eficiencia de los clasificadores, ya fuese con representación básica (*BoW*) o tomando los vectores opuestos en *Word2vec*. También es importante comentar la mejora que supone para algoritmos como *Naïve Bayes* la combinación de CRP y LSA en la representación de los documentos.

En cuanto al tamaño del *corpus* es importante comentar las grandes diferencias experimentadas al comparar la representación con BoW y su correspondiente preprocesamiento en las dos bases de datos de sentimientos. Se pudo observar que al trabajar en un *corpus* mayor la precisión aumentó considerablemente, llegando a obtener la mejor precisión de todo el estudio con nuestro clasificador SVC de Kernel Linear (87 % aproximadamente).

Por último, la investigación llevada a cabo en este Trabajo de Fin de Grado así como los datos y técnicas aquí recopiladas forman una base importante para la investigación en el campo de la clasficación automática de textos, permitiendo escoger las técnicas más apropiadas en función del objetivo deseado.

En el futuro se recomienda investigar la manera de mejorar nuestros resultados en este difícil problema de clasificación. Se propone evaluar nuevos preprocesamientos de texto que en ciertos casos puedan aportar gran valor (como por ejemplo la corrección ortográfica), añadir otro tipo de características que puedan ser determinantes o incluso nuevos sistemas de representación de texto que ayuden en la generalización de la información para los clasificadores (por ejemplo, redes de palabras). También sería interesante profundizar en el ámbito del *deep learning* y las redes neuronales más allá de *Word2vec*, o incluso encontrar nuevos sistemas de formación de vectores de documentos para *Doc2vec* que reduzcan el ruido que observamos en los métodos utilizados en este trabajo.

BIBLIOGRAFÍA

[1] A. A. Abiola O.B. and O. A. A review of the various approaches for text to text machine translations. *International Journal of Computer Applications*, 120(18):pages 7–8, 2015.

[2] P. P. Alpa Reshamwala and D. Mishra. Review on natural language processing. *IRACST – Engineering Science and Technology: An International Journal*, 3:113–116, 2013.

[3] Apple. *Siri*, 2015. http://www.apple.com/es/ios/siri/.

[4] N. S. Z. H. Atefeh Heydari, Mohammad ali Tavakoli. Detection of review spam: A survey. *Elsevier Journal*, pages 3634–3642, 2015.

[5] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. *Department of Computer Science, Cornell University, Ithaca, NY*, 2006.

[6] K. C. Dzmitry Bahdanau and Y. Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*, 2015.

[7] A. B. Enrico Blanzieri. A survey of learning-based techniques of email spam filtering. *Springer Netherlands Journal*, pages 64–65, 2009.

[8] J. Fasola and M. J. Mataric. A socially assistive robot exercise coach for the elderly. *Journal of Human-Robot Interaction*, 2013.

[9] T. Fawcett. An introduction to roc analysis. *Elsevier*, pages 861–874, 2005.

[10] L. M. G. G. W. Furnas, T. K. Landauer and S. T. Dumais. The vocabulary problem in human – system communication. *Communications of the ACM*, 1987.

[11] Google. *Google Now*, 2015. https://www.google.com/intl/es/landing/now/.

[12] Google. *Google Traductor*, 2015. https://translate.google.com/.

[13] J. Hirshberg and C. D. Manning. Advances in natural language processing. *Science*, 349:261–266, 2015.

[14] Jeffrey Pennington, Richard Socher and Christopher D. Manning. *Global Vectors for Word Representation*, 2015. http://nlp.stanford.edu/projects/glove/.

[15] D. Jurafsky and J. H. Martin. Speech and language processing. 7, 16, 2015.

[16] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[17] J. S. Man Lan, Chew Lim Tan and Y. Lu. Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4), 2009.

[18] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. *In Conference on Recommender Systems*, pages 165–172, 2013.

[19] Microsoft. *Cortana*, 2015. https://www.microsoft.com/es-es/windows/Cortana.

[20] National Science Foundation. *Large Movie Review Dataset*, 2012. http://ai.stanford.edu/~amaas/data/sentiment/.

[21] NLTK Project. *Natural Language Toolkit*, 2015. http://www.nltk.org/.

[22] B. Pang and L. Lee. Opinion mining and sentiment analysis. *NOW - the essence of knowledge*, 2008.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] H. L. Pritam Gundecha. Mining social media: A brief introduction. *Informs*, 2012.

[25] E. K. Steven Bird and E. Loper. *Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit*. NLTK Project, 2015. http://www.nltk.org/book/.

[26] UCI Machine Learning Repository. *Sentiment Labelled Sentences Data Set*, 2015. https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences#.

[27] H. K. Walaa Medhat, Ahmed Hassan. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, pages 1094–1113, 2014.

[28] C. Zhai. Statistical language models for information retrieval a critical review. *NOW - the essence of knowledge*, 2008.

[29] H. Zhang. The optimality of naive bayes. *Faculty of Computer Science, University of New Brunswick*, 2004.