TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Servicio Web para la Clasificación Automática de Opiniones

Sentiment Analysis App

Autor
Alberto Argente del Castillo Garrido

Directores
María Victoria Luzón García
Eugenio Martínez Cámara

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 18 de junio de 2019.

Servicio Web para la Clasificación Automática de Opiniones

Sentiment Analysis App.

Autor
Alberto Argente del Castillo Garrido

Directores
María Victoria Luzón García
Eugenio Martínez Cámara

Servicio Web para la Clasificación Automática de Opiniones: Sentiment Analysis App

Alberto Argente del Castillo Garrido

Palabras clave: REST, Flask, token, Bayes, opinión, modularidad, mongo, análisis, API, investigación

Resumen

Hoy en día la opinión es un recurso muy querido por empresas, compañías de viajes, restaurantes... Además la cantidad de información que hay no es procesable por una persona y por ello surge la necesidad de crear clasificadores que hagan esto de forma automática.

Con la opinión las empresas son capaces de ver sus errores, sus aciertos y así actuar frente a lo que desea el usuario. El procesamiento de la opinión es clave en esta tarea, ya que de la opinión las empresas pueden saber qué errores han cometido para cambiar, qué cosas hacen bien, y ya planificar cómo actuar frente a esto.

Este proyecto busca agilizar la tarea del análisis de opiniones, no sólo para aquellos que ya se dedican a este campo, sino también a aquellos que se inician en la inteligencia artificial o en el análisis de opiniones, puesto que le sirve para probar la creación de distintos modelos, o probar otros ya creados para así aprender.

Para llevar a cabo este proyecto se ha realizado una REST-API con la que el usuario puede crear y validar clasificadores de opiniones. Mediante esta aplicación el usuario cargará los datos para que la aplicación las procese y genere el modelo seleccionado por el usuario, dándole así libertad de elección. Además el usuario podrá utilizar todos aquellos modelos que ya han sido creados, o bien por él, o bien por otros usuarios, y tras esto podrá descargar los resultados.

Web Service For Classifying Opinions: Sentiment Analysis App

Alberto, Argente del Castillo Garrido (student)

Keywords: REST, Flask, token, Bayes, opinion, modularity, mongo, analysis, API, research

Abstract

We live in the era of Big Data, the era of information, which has became a very important resouce for companies. Most of the people publish their opinion about any topic, such as the places they visit, the food they eat, how was the customer support about a service... Consequently, opinions are a valuable source of knowledge to know the level of satisfaction of the people.

A web service for the classification of opinions is presented in this grade thesis. The web service will allow researchers and practitioners to generate classification models and assess their performance on different datasets. Furthermore, people who is starting in artificial intelligence or with sentiment analysis can start and prove different alternatives to resolve a problem.

The web service is built upon a REST-API. The software allows users to configure out the classification process from the preprocessing to the classification algorithm. The classification model configurated out can be trained and stored in order to use it for classifying data from a test dataset or other datasets. Also, users can choose other models created by other users and then apply the model selected to his data, and they can check the performance of the model on that data.

Yo, **Alberto Argente del Castillo Garrido**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76654048Q, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Alberto Argente del Castillo Garrido

Granada, 18 de junio de 2019.

D. **María Victoria Luzón García**, Profesor del Área de Lenguajes y Sistemas Informáticos del Departamento Lenguajes y Sistemas Informáticos de la Universidad de Granada.

D. **Eugenio Martínez Cámara**, Profesor del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Servicio Web Para la Clasificación Automática de Opiniones, Sentiment Analysis App***, ha sido realizado bajo su supervisión por **Alberto Argente del Castillo Garrido** , y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en 18 de junio de 2019.

Los directores:

María Victoria Luzón García **Eugenio Martínez Cámara**

Agradecimientos

En primer lugar quiero agradecer a mis tutores Eugenio y Vicky por la oportunidad de trabajar con ellos en este proyecto, por guiarme de la manera correcta y adecuada durante el desarrollo del mismo y también por la confianza depositada. Además quiero agradecer toda los conocimientos que me han aportado, ya que sin estos no habría sido posible. Gracias por vuestro apoyo, comprensión y enseñanza.

También quiero agradecer a mis padres, familiares, pareja y amigos por el apoyo y la motivación a lo largo de la carrera para empezarla y terminar.

Por último, pero no menos importante, agradecer a los profesores que de la carrera por la dedicación mostrada y los conocimientos que me han transmitido que me han traído hasta aquí y por todo lo aprendido durante la carrera. En especial quiero agradecer al profesor Francisco Herrera por su enseñanza y por presentarme a Eugenio y a Vicky para poder realizar este proyecto con ellos.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este proyecto de fin de grado proporcionamos un servicio web para la clasificación automática de opiniones. En la Sección 1.1 presentamos la motivación que inspira a este proyecto. En la Sección 1.2 indicamos los objetivos que se pretenden cumplir. Por último, en la Sección 1.3 se muestra la planificación del proyecto.

1.1. Motivación

Estamos en una época en la que la importancia de las redes sociales, compras online, webs de reservas, vídeos de Internet, crecen cada vez más y más, y sobre las cuales los usuarios dejan su opinión al respecto. Un ejemplo de esto es que al entrar a comprar un producto online los usuarios no sólo tienden a fijarse en las especificaciones, sino que también miran las opiniones de otros compradores previos para terminar de decidir. Pero esta información no sólo interesa a los usuarios, sino que también interesa a las compañías que venden productos, a los organizadores de un evento que utilizan un *hashtag* para obtener información de cómo lo están viviendo los usuarios, etc.

Para poder analizar toda la información hay que hacerlo a través del procesamiento de la opinión, y es que hay que saber procesarla correctamente. El procesamiento de la opinión acabará siendo determinante tanto para el entrenamiento del modelo como de una futura predicción. Con un correcto procesamiento de la opinión se estará más cerca de lograr automatizar el análisis de todas las opiniones recibidas por partes de usuarios, clientes, etc. Permitirá saber qué factores se deben cambiar y cuales pueden mantenerse como están. Este tema no solo es tratado por empresas, sino también por investigadores, y es hacia ellos a los que va dirigida la propuesta de proyecto.

El investigador centrado en análisis de opiniones suele manejar una gran

cantidad de problemas. Cuando se enfrenta a un nuevo proyecto, hay cierta tendencia a consultar análisis de problemas pasados para ayudarse del código previo. Esto lleva a hacer copias de ficheros, modificar fragmentos de código, añadir o eliminar funciones e incluso a reconfigurar algoritmos. Sería realmente cómodo para el investigador poder disponer de cierta automatización de este proceso.

Asimismo nos encontramos cada vez más usuarios no expertos que desean analizar sus datos. Por ejemplo, cierto directivo de una empresa puede desear saber cuáles son sus factores potenciales de mejora. Para ello podría analizar las críticas recibidas a través de, por ejemplo, redes sociales. Gracias a este proyecto vamos a permitir que dicho directivo, sin grandes conocimientos en el área de análisis de opiniones, pueda realizar dicha tarea consiguiendo detectar sus debilidades para mejorarlas.

Este sistema también ayudará a aquellas personas que se estén iniciando en el mundo de la ciencias de datos y que no tienen aún el criterio suficiente para diseñar un sistema de clasificación. Podrán probar con la creación de distintos modelos hasta dar con el modelo deseado, y aprender del cómo se ha llegado a ese punto.

Por ello se propone una aplicación que permita agilizar el trabajo del investigador y permita realizar un análisis básico de opiniones al usuario no experto. El usuario solo tendrá que cargar el nuevo con el que quiere trabajar, y seleccionar las funciones y algoritmos para entrenar el modelo. Una vez entrenado el usuario podrá subir nuevos datos y descargar los resultados, es decir, obtener la predicción del modelo sobre los datos subidos.

1.2. Objetivos

Los objetivos que se persiguen en el proyecto son:

1. **Estudio del estado del arte.** Se deberá realizar un estudio previo del análisis de opiniones para conocer el estado del arte.
2. **Diseño y desarrollo de plataforma de clasificación de opiniones sobre un servicio web.** Para conseguir que sea más accesible por los potenciales usuarios, la aplicación se realizará sobre un servicio web.
3. **Aplicación de principios de ingeniería del software para el análisis, diseño y desarrollo de la aplicación.** Aplicar la metodología de ingeniería del software aprendida durante el grado sobre un proyecto real.

4. **Desarrollo de los métodos adecuados para un correcto tratamiento y clasificación de opiniones.** Tras el estudio del estado del arte en el objetivo 1, se realizará el desarrollo de los principales métodos para tratar y clasificar opiniones.

5. **Desarrollo de un prototipo funcional.** Al final del proyecto se pretende tener un prototipo del servicio web que pueda ser instalado en un servidor y que además tenga la funcionalidad mínima, es decir, que se puedan procesar los datos y así aplicar un algoritmo para generar un modelo de clasificación de opiniones y que posteriormente pueda ser evaluado.

1.3. Planificación

Para la realización del trabajo se ha seguido una metodología de espiral, definida por Bohem [40].
Esta metodología insta a llevar una planificación por ciclos, cada ciclo con sus respectivas etapas. Para poder alcanzar los objetivos se decidió dividir en tareas los objetivos y resolver estas tareas mediante la metodología de espiral. Las tareas que se han realizado son las siguientes:

1. Estudio del análisis de opiniones.

2. Análisis de los requisitos del sistema.

3. Implementación del sistema.

La primera tarea de las nombradas en esta sección tomó un total de 2 ciclos, los cuales fueron necesarios para resolver las posibles dudas que surgieron durante el estudio del problema. Para la segunda tarea fueron necesarios un total de 4 ciclos, en los que se definieron todos los requisitos del servicio para sólo dejar pendiente la implementación. Para la última tarea se han realizado 5 ciclos, para así llevar una mayor supervisión del sistema y comprobar que se estaba desarrollando acorde a lo definido en la fase de análisis.

1.4. Presupuesto

En los costes del proyecto hay que sumar tanto los costes materiales como los costes humanos.

1.4.1. Costes materiales

El desarrollo del proyecto se ha realizado sobre un portátil personal que tiene un coste de 999 €[48], a lo que quedaría sumar el software utilizado durante el mismo:

- **Sistema Operativo**: Ubuntu 18.04, precio de 0€.
- **Editor de texto**: Atom, precio de 0€.
- **Desarrollo Software**: UML2 [44], y la plataforma draw.io [6], precio de 0€.
- **Framework**: Se ha desarrollado sobre el framework Flask, precio de 0€.

Por tanto el coste material del proyecto se queda sólo en el coste del ordenador usado, es decir **999€**.

1.4.2. Costes humanos

Para calcular los costes humanos, se ha utilizado el sueldo para un desarrollador Junior en España según [46], el cual es de 20.890€/año, de lo que sacamos que la hora sale a 10,88 €. Por tanto teniendo en cuenta los ciclos realizados, se obtiene lo siguiente:

Ciclo	Horas	Coste/Hora	Coste total
Primer ciclo	60h	10,88€	652,8€
Segundo ciclo	20h	10,88€	217,6€
Tercer ciclo	240h	10,88€	2611,2€
			3481,6€

Tabla 1.1: Prepuesto del proyecto referente a coste humano.

El coste humano supone un total de **3.481,6€**.

Para calcular el coste total del proyecto, habría que sumar el coste material y el coste humano. Se genera un coste total de:

$$999 + 3481,6 = 4480,6 \tag{1.1}$$

El coste final del proyecto es de **4480.6 €**.

1.5. Estructura de la memoria

Tratando de facilitar al lector su seguimiento por la presente memoria se va a exponer brevemente el contenido de los capítulos restantes.

- **Capítulo 2**: Muestra en qué consiste la tarea de Análisis de Opiniones. Se muestra la base teórica que fundamenta nuestro proyecto.

- **Capítulo 3**: Desarrolla el análisis del sistema. Cómo y por qué se han elegido la arquitectura, el *framework* utilizado, o la base de datos elegida.
- **Capítulo 4**: Muestra el diseño interno del servicio web.
- **Capítulo 5**: Describe la implementación del proyecto, desarrollando los módulos implementados, es decir, las funciones y la utilidad de cada una.
- **Capítulo 6**: Muestra las pruebas de caja negra y caja blanca realizadas para validar los requisitos del sistema.
- **Capítulo 7**: Recoge las conclusiones del proyecto y las vías futuras de mejora del servicio web planteado.

Capítulo 2

Análisis de opiniones

En este capítulo se procede a contextualizar el tipo de problema que un usuario quiere resolver al acceder al servicio proporcionado. Para ello se explican las bases de lo que conocemos como Análisis de Opiniones. En la Sección 2.1 daremos a conocer el Análisis de Opiniones. En la Sección 2.2 desarrollaremos el concepto de Opinión desde un punto de vista formal. La Sección 2.3 muestra las tareas que se pueden llevar a cabo dentro del área de Análisis de Opiniones. Finalmente la Sección 2.4 muestras los métodos de aprendizaje supervisado que se han seleccionado para realizar el Análisis de Opiniones.

2.1. Introducción

El ser humano necesita tomar decisiones a lo largo de toda su vida, ya sea para una decisión simple como elegir un dispositivo electrónico o una decisión más compleja como elegir un destino de trabajo. Para ello solemos llevar a cabo dos procesos en mayor o menor profundidad. En primer lugar, la persona en cuestión realiza un análisis de las distintas opciones que tiene: se podría preguntar por los beneficios, las deficiencias, el coste económico y las opciones de futuro de cada opción. En segundo lugar, el ser humano siente la necesidad de pedir opinión ya sea a expertos o simplemente a sus seres más cercanos. Esta petición de opinión alimenta el conocimiento de la persona decisora ya que puede conocer experiencias reales sin tener que vivirlas en sí misma. Una vez finalizadas ambas fases, intrínseca y extrínseca, la persona tiene el suficiente conocimiento como para tomar su propia decisión.

Hoy en día esta petición de opinión a terceros se realiza tanto mediante comunicación oral como a través de la Web. El ser humano tiende a solicitar y proporcionar opiniones mediante blogs, redes sociales o foros. Llegamos a un punto en el que los usuarios no solo tratan de compartir sus experiencias para que los decisores tengan más información sino que manifiestan libremente su opinión sobre temas como política, música o deporte para regodearse.

Frente a este marco expuesto nos encontramos con la tarea de Análisis de Opiniones. Para definir dicho concepto es necesario presentar previamente el concepto de Procesamiento del Lenguaje Natural (PLN). El PLN es una disciplina centrada en el diseño e implementación de aplicaciones informáticas que se comunican con personas mediante uso de lenguaje natural [43]. Dicho en otras palabras, buscamos desarrollar sistemas que presenten una interacción humano-máquina basada en el lenguaje natural.

Dentro del gran área PLN nos encontramos la tarea de Análisis de Opiniones (AO) que se centra en el tratamiento computacional de opiniones, sentimientos y expresiones subjetivas [47]. Desde un punto más amigable, podemos definir el concepto de Análisis de Opiniones como un conjunto de técnicas computacionales que se centran en extraer, clasificar, comprender y evaluar opiniones que los usuarios publican, por ejemplo, en portales web. Este concepto de Análisis de Opiniones es fruto de una traducción de los conceptos equivalentes Sentiment Analysis y Opinion Mining. Es importante destacar el echo de que el concepto "Análisis de Sentimientos" no hace justicia a Sentiment Analysis ya que el término *sentimiento* solo hace referencia al estado anímico de una persona y no a la opinión.

Los orígenes del AO datan del siglo XX tras la publicación de trabajos expuestos en congresos científicos y revistas [41, 49, 45]. Pero no fue hasta el siglo XXI cuando se incrementó el interés por este área. La comunidad investigadora encontró en este área un gran desafío. Las personas deseaban conocer la opinión del resto de personas. Por ejemplo, el mundo empresarial anhelaba conocer la opinión de los clientes potenciales.

2.2. Opinión

Podemos inferir de la sección previa que la tarea de AO se encarga de examinar opiniones. Por lo tanto, hemos de definir dicho concepto. Para que su explicación sea más llevadera vamos a utilizar un conocido ejemplo [42].

"[1] Mi pareja y yo salimos a cenar anoche. [2] Elegimos un elegante restaurante a las afueras de la ciudad. [3] La impresión que nos dejó el personal fue buena. [4] A mi parecer la comida era de gran calidad. [5] Por el contrario a mi pareja no le gustó, según ella, el primer plato dejaba mucho que desear. [6] La decoración no nos gustó del todo a los dos ya que había muchas figuras y estaban repletas de polvo. [7] Lo mejor del restaurante, las vistas, perfectas para irse del restaurante y no volver."

Ante un comentario de este tipo nos tenemos que fijar en cuáles son aquellas oraciones que exponen una opinión. En este ejemplo vemos que serían

todas excepto la primera de ella que simplemente informa sobre el acontecimiento. De entre todas las oraciones de opinión, vemos que la segunda hace una valoración global del restaurante mientras que en el resto de oraciones se valoran componentes del mismo. Algunas de estas oraciones tienen una connotación positiva mientras que otras son negativas. También podemos extraer de este fragmento el detalle de que las opiniones no tienen porqué ser propias del usuario sino que pueden ser de terceras personas, como sería en este caso la opinión de la pareja sobre la comida y la decoración.

De este primer análisis rápido podemos ver que una opinión está constituida por un objeto de la opinión *(o)*, por la orientación de la valoración *(p)* y por el sujeto que proporciona la opinión *(h)*. También es importante tener en cuenta el momento en el que produce la opinión *(t)* ya que esta puede variar con el tiempo. Por tanto, una opinión se puede ver como una cuádrupla *(o, p, h, t)*. Por ejemplo, podríamos tener la opinión básica *(restaurante, positivo (elegante), Juan y Marta (autor y pareja), 20/06/2019 (anoche))*.

Entrando en mayor profundidad en el ejemplo nos podemos fijar en el hecho de que las opiniones proporcionas tienen distintos niveles. Se hacen valoraciones globales a nivel de restaurante y valoraciones sobre componentes del mismo, como son la comida o la decoración. Por tanto, tenemos que tener en cuenta que el objeto de la opinión se puede dividir en componentes, pudiéndose identificar distintos elementos clave. Pasaremos a llamar al objeto de opinión partitivo como entidad *(e)*. Esta entidad puede ser un producto, servicio, restaurante, hotel, persona o ente. Tras dicho inciso, la definición de formal de opinión debe de ser redefinida quedando una estructura más compleja. Una opinión se define como una quíntupla, *(e_i, a_{ij}, p_{ijkl}, h_h, t_l)* tal que:

- e_i es el nombre de la entidad
- a_{ij} es el componente o aspecto de e_i sobre el que se opina
- p_{ijkl} es la valoración de la opinión conocida como polaridad, tradicionalmente positiva o negativa
- h_h es el autor que proporciona la opinión
- t_l es el momento en el que se manifiesta la opinión

Un ejemplo de una opinión formal podría ser *(restaurante, comida, positivo (gran calidad), Juan (autor), 20/06/2019 (anoche))*.

Es importante tener en cuenta que una opinión se puede proporcionar de diversas formas. Por un lado, podemos tener opiniones directas en las que expresa de forma clara y concisa una opinión sobre una entidad o alguno de sus aspectos. Por otro lado, podemos tener opiniones indirectas en las que se expresa la evaluación de la entidad o aspecto mediante la valoración del efecto que origina la propia entidad o aspecto. Asimismo, podemos te-

ner opiniones comparativas donde se expone una relación de semejanza o disimilitud entre dos o más entidades o aspectos.

2.3. Tareas del AO

En el análisis de opiniones es importante definir las tareas que conlleva la extracción de información de un mensaje [42]. Con estas tareas se quiere representar cual sería la metodología ideal a seguir.

- **Tarea 1. Extracción y categorización de entidades**: Dado un documento D, la primera tarea consiste en la identificación de todas las entidades del documento, además trata de agrupar todas las formas de referencias una entidad con sus correspondientes.
- **Tarea 2. Extracción y categorización de aspectos**: La segunda tarea consiste en la identificación de todos los aspectos y las distintas formas de expresarlos, junto con su posterior agrupación con las entidades correspondientes.
- **Tarea 3. Extracción y categorización del autor de la opinión**: Al igual que en las tareas anteriores, en esta tarea hay que identificar los autores y las diferentes maneras de referirse al mismo.
- **Tarea 4. Extracción del momento temporal**: Para lograr completar la quíntupla definida en la sección anterior, también queda por identificar el momento en el que esta fue realizada, y esto se lleva a cabo en esta tarea.
- **Tarea 5. Clasificación de polaridad a nivel de aspecto**: Tras conocer la entidad y su aspecto ya se puede clasificar, es decir, toca indicar si la opinión es positiva, negativa o neutra.
- **Tarea 6. Generación de quíntupla de opinión**: La última tarea consiste en la recopilación de todos los elementos identificados y agruparlos para lograr formar la quíntupla definida en la sección anteior.

Las tareas a realizar dependerán del nivel de análisis que se quiera realizar. Existen tres tipos diferentes de niveles de análisis [42]:

- **Nivel de documento**: En este nivel se refiere a lo opinión total que expresa un documento de opinión. Se parte de la premisa de que en el documento de opinión únicamente se hace referencia a una entidad.
- **Nivel de oración**: En este nivel se pretende realizar el estudio de oraciones, es decir, identificar las oraciones que forman un documento y tratar de obtener la orientación de la opinión de estas.

- **Nivel de entidad y aspecto**: Este nivel es el que más pretende afinar y además corresponde totalmente con la quíntupla explicada en la sección anterior.

Tras explicar las diferentes tareas y los diferentes niveles de análisis, queda decir en cuál se centrará este sistema. Para este proyecto se ha decidido que se implementará lo necesario para cumplir la tareas 5 a nivel de documento.

2.4. Aprendizaje Supervisado

El aprendizaje supervisado es una de las tareas del aprendizaje automático. Ésta requiere de un conjunto de datos etiquetados, es decir, información que indique qué tipo de dato es en función del problema, para así poder generar un modelo que sea capaz de clasificar nuevos documentos [42].

Para generar un modelo debemos obtener la representación de los datos. Esto lo haremos a través de sus características, para que así el modelo pueda ajustarse a la distribución que siguen los datos. El algoritmo es una factor muy importante, sobre todo porque el usuario va a poder ajustarlo como él prefiera haciendo que este pueda generar un mejor modelo para los datos proporcionados. Se pueden aplicar diferentes tipos de algoritmos para evaluar un mismo conjunto de datos, por lo que el usuario podrá utilizar el que vea más conveniente.

Para poder afrontar un problema de aprendizaje supervisado en el área del Procesamiento del Lenguaje Natural, hay que ser capaz de representar los documentos proporcionados como un conjunto de términos ponderados, para así reflejar la información semántica de cada término del documento. Para lograr la obtención de una bolsa de términos que contenga toda la información semántica para cada término del documento es necesario realizar una conversión. A esta conversión se le conoce como proceso de indexación [42].

El proceso de indexación consta de cuatro etapas consecutivas: identificación de la estructura del documento y tokenización, reducción de características (opcional), normalización morfológica (opcional), y asignación de valores de importancia.

Por tanto el primer paso consiste en la obtención de los *tokens*, unidad mínima de representación del documento. Para ello se separan todas las palabras del texto en tokens. Previo a la tokenización, incluso posterior a esta, es importante que antes de que se siga adelante se decida si se querrá pasar

el texto a minúscula o si se querrá eliminar los acentos del mismo.

Tras la obtención de los *tokens*, se debe aplicar una reducción de características. Esto es porque todos los idiomas utilizan palabras que sirven para la construcción de oraciones y lograr transmitir la información con sentido, pero no siempre todas las palabras aportan realmente información al texto [42]. Dado que este tipo de información no será relevante es conveniente eliminarla. A este tipo de palabras se les conoce como palabras vacías, o en inglés, stopwords.

La siguiente etapa es la normalización morfológica. Para lograr esta normalización se pueden seguir dos métodos conocidos como lematización y stemming. En este proyecto se llevará a cabo el stemming. Se trata de un proceso heurístico que recorta la terminación de las palabras dejando en lugar de la palabra la raíz de la misma. Este método no tiene en cuenta el contexto de la palabra como sí lo haría el método de lematización [42].

La última fase previa a la utilización de un algoritmo de aprendizaje supervisado, es la de asignación de valores de importancia. Este proceso trata de calcular el valor que aporta cada token calculado previamente al texto. Para realizar este cálculo existen varios métodos, pero en este proyecto se ha decidido que se va a aplicar el método TF-IDF.

El esquema TF-IDF trata de unir dos formas de representación de información de los token en un documento como lo son el TF y el IDF. El TF (Term Frecuency) asigna un mayor valor a aquellos *tokens* que aparecen más veces a lo largo del documento, pero que discrimina a aquellos términos que aparecen muchas veces pero en pocos documentos. Por ello el idf (inverse document frecuency) asigna un mayor valor a aquellos términos que que aparecen en pocos documentos y un menor valor a los que aparecen en más. La fórmula para calcular el TF-IDF se muestra en la Ecuación 2.1:

$$TF - IDF = tf\_d * log\frac{N}{n_d} \tag{2.1}$$

2.4.1. Algoritmos

Tras haber visto el proceso de indexación, proceso previo al entrenamiento del modelo, es momento de hablar de los algoritmos presentes en el sistema de forma inicial.

Máquina de soporte de vectores

El algoritmo máquina de soporte de vectores, en inglés Support Vector Machine (SVM). Este algoritmo se fundamenta en la búsqueda de un hiper-

plano que maximice la separación entre los ejemplos pertenecientes a dos categoría diferentes [42]. SVM es uno de los algoritmos que mejores resultados ha dado para la clasificación automática de texto, además de utilizada en el análisis de opiniones.

Naïve Bayes

El algoritmo de Naïve Bayes está fundamentado en el Teorema de Bayes. Este teorema enuncia que, "dado un conjunto de sucesos mutuamente excluyentes y exhaustivos $(A_1...A_n)$ cuya probabilidad es distinta de cero, y un suceso cualquiera del que se conocen las condicionales $P(B/A_i)$, la probabilidad $P(A_i/B)$ viene dada por la siguiente expresión" [42]:

$$P(A_i/B) = \frac{P(B/A_i)P(A_i)}{P(B)} \tag{2.2}$$

Esta fórmula no está adaptada al problema de clasificación de texto, y por tanto para poder aplicar este algoritmo, antes hay que adaptarla [42]:

$$P(C = c_j/A_i = \alpha_1, ..., A_{|A|}) = \frac{P(A_1 = \alpha_1, ..., A_{|A|})/C = c_j}{P(A_1 = \alpha_1, ..., A_{|A|})} \tag{2.3}$$

Para poder aplicar este algoritmo a un problema ed clasificación, tiene que cumplirse que todas las características sean estadísticamente independientes entre sí. Esto en análisis de textos implica que las palabras son independientes del contexto, es decir, que ninguna palabra se ve afectada por sus vecinas y que la probabilidad de una palabra es independiente de su posición en el texto. Pero esto en el texto no ocurre, porque las palabras dependen las unas de las otras. A pesar de ello se ha demostrado empíricamente que el Naïve Bayes obtiene resultados aceptables.

Capítulo 3

Análisis del sistema

En este capítulo se presenta el análisis del sistema, el cual estará dividido en dos secciones. La Sección 3.1 tratará los requisitos funcionales, no funcionales y de información del sistema. La Sección 3.2 tratará los casos de uso del sistema con su correspondiente diagrama de casos de uso. En la Sección 3.3 se explicará qué tipo de arquitectura tendrá la aplicación, qué tipo de aplicación será, qué patrón arquitectónico se utilizará y el por qué.

3.1. Análisis de requisitos

Para realizar un software es importante que antes se realice un estudio de los requisitos que éste tiene, es decir, de la funcionalidad que tendrá el sistema. El objetivo del análisis de requisitos es satisfacer todas las peticiones del cliente respecto al producto.

Los requisitos son los requerimientos mínimos que el sistema debe cumplir para asegurar la resolución del problema por parte del usuario.

Dicho esto ya se puede hablar de la estructura de este capítulo, el cuál se dividirá en tres secciones. La primera sección será la de los requisitos funcionales del sistema, los que definen la funcionalidad del sistema. En la segunda sección se definirán los requisitos no funcionales del sistema, los que contienen las restricciones relacionadas con el diseño o la implementación. Y en la última sección se encuentran los requisitos de información, los cuales hacen referencia a la información que será almacenada en el sistema.

3.1.1. Requisitos funcionales

El sistema contará con los siguiente resquisitos funcionales:

RF01: Gestión de archivos: El usuario podrá cargar uno o varios archivos (csv) que contendrá el dataset y sobre el que posteriormente aplicarán distintas técnicas de preprocesamiento o algoritmos de aprendizaje.

RF02: Elección y aplicación del preprocesamiento de datos sobre el dataset: El usuario podrá aplicar distintas técnicas de preprocesamiento según crea oportuno sobre el dataset.

RF03: Visualizar los datos del dataset tras la carga de datos: El usuario podrá ver información del dataset para decidir qué funciones de preprocesamiento y qué algoritmo de aprendizaje aplicará.

RF04: Configuración del proceso de aprendizaje: El usuario podrá elegir qué algoritmo de aprendizaje que querrá aplicar sobre el dataset para así obtener los resultados.

RF05: Visualizar los resultados de los algoritmos: El usuario podrá aplicar métodos para la visualización de los resultados obtenidos.

RF06: Descargar resultados obtenidos: El usuario podrá descargarse los resultados obtenidos tras la aplicación del modelo generado en la fase de aprendizaje.

RF07: Clasificar datos con un modelo entrenado: El usuario podrá aplicar el modelo obtenido durante el aprendizaje a otros datasets que previamente haya subido.

3.1.2. Requisitos no funcionales

Los requistos no funcionales del sistema son los siguientes:

RNF01: Mantenibilidad: El sistema debe tener un mantenimiento sencillo y se debe actuar antes los errores rápido. Para poder actuar rápido frente a los errores que surjan, estos tienen que ser rápidamente localizables y así actuar sobre ellos.

RNF02: Escalabilidad: Al tratarse de una arquitectura modular podemos añadir tantos módulos como veamos necesarios sin que esto afecte al sistema en sí, teniendo funcionalidades adicionales sin cambiar la estructura del sistema.

RNF03: Usabilidad: Más propio de la interfaz web y no de la REST API, dado que el usuario no necesita conocer cómo es la API internamente

para poder utilizarla.

RNF04: Eficiencia: El sistema debe ser eficiente en todo momento para asegurar el mejor rendimiento usando los menores recursos posibles.

RNF05: Arquitectura distribuida: Al tratarse de un servicio web, la arquitectura de este debe ser distribuida. Así el usuario podrá acceder al sistema desde su máquina sin tener que preocuparse de los recursos consumidos.

3.1.3. Requisitos de información

Los requisitos de información muestran la información que es esencial almacenar en el sistema:

RI01: Almacenamiento de archivos cargdos: Se almacenarán los archivos cargados por el usuario, en los RF01 o RF07, para poder entrenar posteriormente un modelo, o para obtener la predicción de un modelo ya entrenado sobre los mismos.

RI02 Almacenamiento del modelo: Se almacenarán los modelos entrenados por los usuarios en el RF05, además de la información relevante del mismo.

RI03 Resultados de predicción: Para satisfacer el RF07 es necesario que se almacenen los resultados tras aplicar los datos cargados por el usuario y ser procesador por el modelo.

3.2. Casos de uso

En esta sección se detallarán los casos de uso del sistema, los cuales coinciden con los requisitos funcionales previamente definidos.

CU1: Gestión de archivos

Título	Gestión de archivos
Descripción	El usuario puede cargar uno o varios archivos en el formato aceptado
Actor	El usuario
Precondición	
Secuencia	1.1 El usuario selecciona los documentos que quiere cargar 1.2 El sistema comprueba que el formato de los archivos es correcto 1.2.1 Si los archivos seleccionados no corresponden con el formato, el usuario tendrá que seleccionar otros archivos. 1.3 El sistema carga los archivos.
Post-Condición	Los archivos quedan cargados correctamente
Comentarios	El formato de los archivos debe ser el aceptado por la aplicación
Requisitos Func.	**RF01**

Tabla 3.1: Caso de Uso 1: Gestión de Archivos.

CU2: Elección del algoritmo de clasificación

Título	Elección del algoritmo de clasificación
Descripción	El usuario podrá configurar los parámetros para el entrenamiendo del modelo.
Actor	El usuario
Precondición	
Secuencia	2.1 El usuario indica los parámetros para la ejecución de los algoritmos de clasificación. 2.1.1 Si el usuario no ha seleccionado ninguno, se usarán los marcados por defecto. 2.2 El sistema guarda la configuración del usuario.
Post-Condición	La configuración se aplicará sobre los datos cargados
Comentarios	El usuario podrá elegir el algoritmo de clasificación que vea más conveniente para los datos cargados.
Requisitos Func.	**RF02, RF04**

Tabla 3.2: Caso de Uso 2: Elección del algoritmo de clasificación

CU3: Elección de las funciones de preprocesamiento

Título	Elección de las funciones de preprocesamiento
Descripción	El usuario elegirá la configuración para el entrenamiento del modelo.
Actor	El usuario
Precondición	
Secuencia	3.1 El usuario elige los algoritmos de preprocesamiento para aplicar al conjunto de datos 3.2 El sistema aplica la configuración.
Post-Condición	La configuración se aplicará sobre los datos cargados
Comentarios	Quedan definidos los parámetros de configuración del entrenamiento del modelo
Requisitos Func.	**RF03**

Tabla 3.3: Caso de Uso 3: Elección de las funciones de preprocesamiento.

CU4: Visualizar datos

Título	Visualizar datos/resultados
Descripción	El usuario podrá visualizar los datos/resultados.
Actor	El usuario
Precondición	
Secuencia	4.1 El usuario pide visualizar los datos/resultados. 4.2 El sistema muestra los datos/resultados. 4.2.1 Si se encuentra en la fase de entrenamiento, el sistema mostrará una nube de palabras al usuario de los datos cargados. 4.2.2 Si se encuentra en la fase de producción, el sistema mostrará las que el usuario le indique de entre las gráficas y métricas disponibles en el momento.
Post-Condición	Se visualizan los datos/resultados
Comentarios	El usuario podrá visualizar los datos/resultados con las opciones soportadas por el sistema.
Requisitos Func.	**RF05**

Tabla 3.4: Caso de Uso 4: Visualizar datos/resultados.

CU5: Descargar resultados obtenidos

Título	Descargar resultados obtenidos
Descripción	El usuario podrá descargar los resultados generados.
Actor	El usuario
Precondición	Haya finalizado la ejecución de los algoritmos de clasificación.
Secuencia	5.1 El usuario solicita descargar los resultados. 5.2 El sistema manda los resultados al usuario.
Post-Condición	El usuario tiene los resultados en su computadora.
Comentarios	El usuario podrá descargar los datos tras el entrenamiento del modelo.
Requisitos Func.	**RF07**

Tabla 3.5: Caso de Uso 5: Descargar resultados obtenidos.

Título	Clasificar datos con un modelo entrenado
Descripción	El usuario podrá probar el modelo creado sobre nuevos archivos.
Actor	El usuario
Precondición	El modelo debe estar generado y el usuario debe haber subido otros archivos sobre los que aplicar el modelo.
Secuencia	6.1 El usuario proporciona nuevos archivos para poder probar el modelo obtenido. 6.2 El sistema comprueba si el formato es el correcto. 6.2.1 Si los archivos cargados no son aceptados por el sistema, tendrá que cargar nuevos archivos que sí lo estén. 6.3 Si el formato es el correcto, el sistema aplica el preprocesamiento sobre los datos subidos y se lo proporciona como entrada al modelo
Post-Condición	Se visualizan los datos/resultados.
Comentarios	El usuario podrá visualizar los datos/resultados con las opciones soportadas por el sistema.
Requisitos	**RF07**

Tabla 3.6: Caso de Uso 6: Exportar resultado del proceso de clasificación.

Por último se muetras el diagrama de casos de uso del sistema.

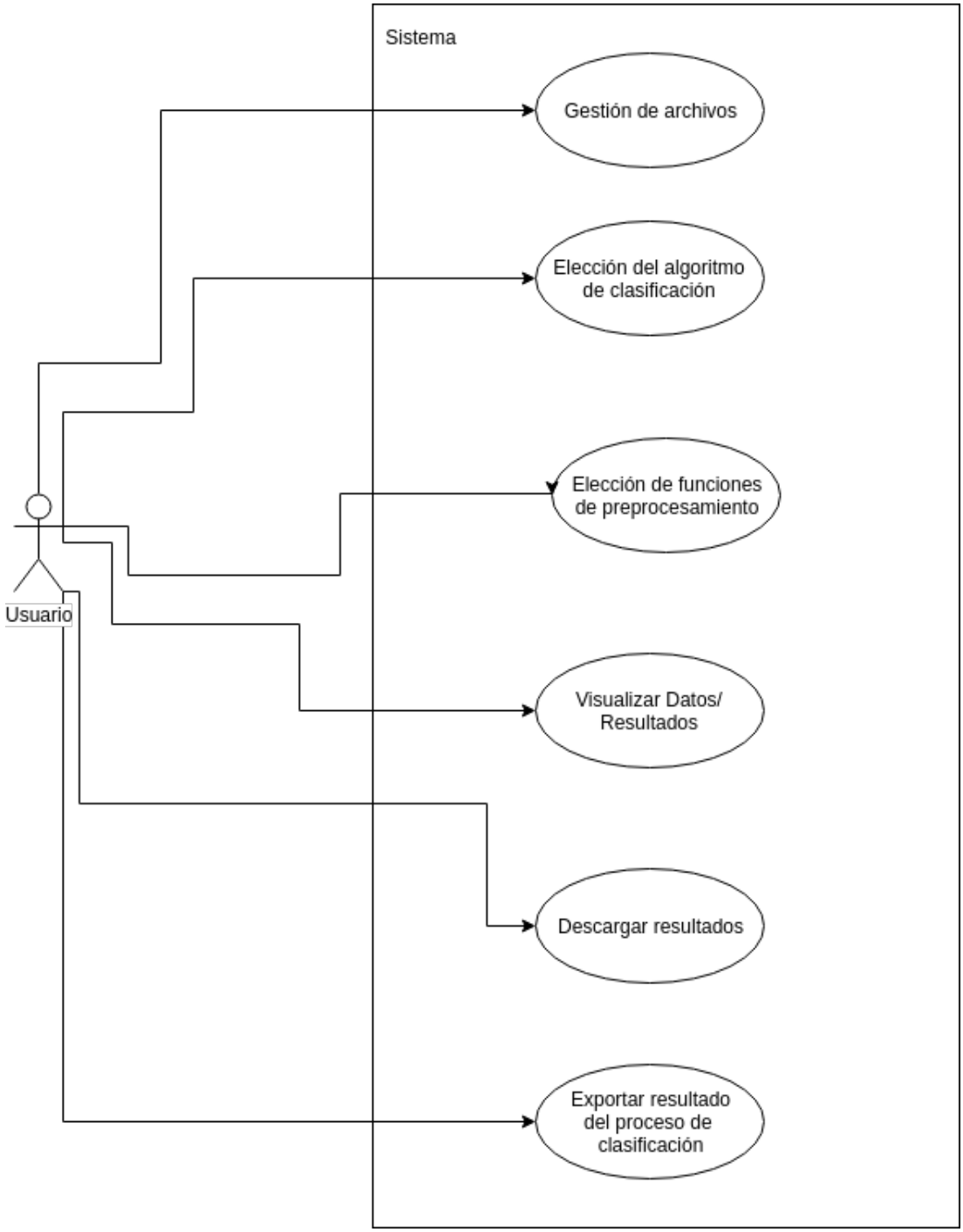

Figura 3.1: Diagrama de casos de uso del sistema

3.3. Análisis de la arquitectura

En esta sección se tratarán tres subsecciones. La subsección 3.3.1 tratará la elección de la arquitectura que tendrá la aplicación. En la subsección 3.3.2 se escogerá el *framework* con el que se desarrollará la aplicación. La subsección 3.3.3 tratará el patrón arquitectónico elegido, y la subsección 3.3.4 tratará qué base de datos se ha usado y por qué.

3.3.1. Elección de la arquitectura

La arquitectura que se busca realizar es una arquitectura distribuida. Algunas de estar arquitecturas son las que ofrecen SOAP, REST o WebSocket, y estas se analizarán a continuación para así poder elegir la más conveniente para la aplicación.

REST vs SOAP

SOAP (Simple Object Access Protocol)[30] [28], no es en sí una arquitectura, sino más bien un protocolo basado en XML el cual es usado por muchos servicios para la comunicaciónde la aplicación. Además SOAP se asemeja a una aplicación de escritorio pero estando en un servidor. SOAP es un protocolo robusto ya que permite agregar metadatos, espacios de nombre para así evitar ambigüedad.

REST (Representational State Transfer)[30] [28], es una arquitectura *stateless* más flexible que SOAP que hace que el cliente no tenga que tener un conocimiento previo porque el usuario simplemente entrará y la usará, mientras que con SOAP el cliente sí tiene que tener un conocimiento previo. Además permite enviar cualquier tipo de dato, ya sea XML, JSON, documentos, etc. por lo que podemos conseguir que el envío de datos sea más ligero y por tanto más rápido. También, si se hace bien, con REST se puede obtener una mayor modularidad de forma que se puedan realizar cambiaos en la aplicación de una forma sencilla sin que afecte al resto.

REST es independiente de HTTP, al contrario que SOAP, lo que permite que pueda usarse el protocolo que se desee, siempre que esté estandarizado por el esquema URI.Debido a la flexibilidad y modularidad que ofrece REST vs SOAP [27], se usará REST para la realización de la aplicación, ya que permitirá que se cree un modelo escalable para poder añadir módulos fácilmente sin que provoque errores en otros módulos.

	SOAP	REST
Diseño	Protocolo Estandard con reglas predefinidas.	Arquitectura Stateless con recomendaciones.
Estado de la conexión	Stateless por defecto aunque puede hacerse stateful.	Stateless por naturaleza.
Protocolos de transporte	HTTP, UDP y otros	Sólo HTTP
Formato de comunicación	Sólo XML	JSON, XML, HTML,...
Ventajas	Estandarización, robustez, seguridad, extensibilidad.	Alto rendimiento, escalabilidad, flexibilidad.
Desventajas	Más complejo, y menos flexible.	Menos seguridad

Tabla 3.7: Tabla comparativa SOAP y REST.

REST vs WebSocket

Dado que se eligió REST respecto a SOAP, queda comparar ambas opciones para decidir cuál de las dos es la mejor opción [35].

WebSocket es un protocolo de comunicación a través de una conexión TCP[26], lo que asegura una conexión punto a punto, es decir, una comunicación doble. Además, Websocket no compromete la seguridad del sistema, ya que todos los **handshackes**[1] pueden hacerse a través de las propias herramientas que vienen con el navegador.

Websocket representa un estándar entre el cliente y el servidor, mientras que REST permite que estos sean implementados de forma independiente. Dado que Websocket se representa con una conexión TCP, se obtiene una conexión más ligera, ya que el envío de las cabeceras sólo se realizan en el *handshake*. Por tanto si la interacción web es menos frecuente, es más conveniente usar REST, aún así el socket se puede cerrar tras un tiempo de inactividad[29].

Websocket es una opción muy buena si se quiere crear una plataforma en tiempo real, como por ejemplo un chat, lo que hace que también sea muy

[1] establecimiento de conexión

escalable por naturaleza y que pueda ser usado en una aplicación de escritorio, móvil o web, mientras que por su parte REST tiene sentido solamente en una aplicación Web.

REST es hasta ahora la forma más estandarizada de estructurar las APIs web [27]. La mayor parte de las aplicaciones web tienden a tener un enfoque RESTful, ya que las operaciones más frecuentes, como son, crear, leer, actualizar o eliminar se ejecutan con éxito mediante HTTP.

Por tanto REST sigue siendo la opción que más se ajusta a los objetivos propuestos, por lo que entre Websocket y REST se ha elegido también REST, puesto que REST se va ajusta más a los propósitos de la aplicación que se plantea realizar.

	WebSocket	REST
Diseño	Basado en un socket.	Basado en recursos.
Estado de la conexión	Stateful únicamente. La comunicación es bidireccional.	Stateless por naturaleza. Comunicación unidireccional.
Protocolos de transporte	HTTP en la primera conexión, TCP	Solo HTTP
Recomendación de uso	Aplicación en tiempo real.	Muchas peticiones de usuario.
Ventajas	Seguridad. Menos coste en la comunicación. Mejor rendimiento con cargas altas.	Alto rendimiento, escalabilidad y flexibilidad.
Desventajas	Se usa para casos concretos, como apps de tiempo real	Menos seguridad. Mayor carga a lo largo del tiempo.

Tabla 3.8: Tabla comparativa REST vs WebSocket.

Tras haber elegido REST frente SOAP y WebSocket, y se realizará en Python, quedará por elegir qué framework se utilizará para el desarrollo de la aplicación puesto que un framework u otro puede limitar también el tipo de arquitectura que tendrá internamente y esto puede afectar a que la aplicación no llegue a tener un tipo de arquitectura REST

De los frameworks de Python más famosos encontramos Flask y Django, ambos entre los más utilizados por los desarrolladores web de Python. Para decidir qué framework se va a utilizar se va a proceder a realizar un análisis como los anteriores.

Aunque hablar del framework utilizado sea algo pertinente al capítulo de la implementación, es importante que esto se realice durante el análisis porque esta elección puede afectar al diseño final del sistema.

3.3.2. Framework: Flask vs Django

Django [23] es un framework muy desarrollado, gracias al número de personas que lo han utilizado y lo han seguido desarrollando y por tanto tiene muchas más funcionalidades ya incluidas[23], lo cual nos permitiría un desarrollo más rápido de la aplicación. Esto es, Django incluye un panel de control, incluye un ORM (Object Relational Mapper), y la estructura para que realizar tu aplicación sea más sencilla.

Flask [24] por su parte es un microframework, es decir, un framework mucho más ligero[34], el cual implementa la funcionalidad mínima para empezar a poder construir la aplicación, de forma que te da más libertad a la hora de elegir qué añadir a una aplicación, permitiendo así que se pueda implementar tal y como uno desee. Esto último permite que la aplicación sea tan modular como uno quiera, ya que se puede organizar la estructura como sea requerido.

Django parte con ventaja respecto a Flask por el hecho de todas las funcionalidades que vienen incluidas cuando comienzas a utilizarlo, gracias a su gran comunidad[33]. Además Django obliga a usar un tipo de base de datos, base de datos relacional, mientras que Flask sí permite que sea el desarrollador el que elija qué tipo de base de datos es la que va a mantener, relacional o no relacional

Además algo que ayuda a que Flask sea una buena opción al desarrollar una aplicación, es que al indicar que se está desarrollando la aplicación trae por defecto un *debugger* que permite detectar y localizar los errores con mayor falicidad.

Ambas opciones ofrecen escalabilidad y modularidad de cara a mantener

la aplicación y ayudan a ampliaciones futuras, sin embargo dado que Flask permite una mayor amplitud de opciones a la hora del desarrollo, se optará por Flask [32] para no limitar algunos aspectos que puedan ser más convenientes.

	Django	Flask
Arquitectura	Por defecto sigue el patrón de arquitectura MVC.	A elección por el desarrolador.
Características	Panel de control, ORM, base de datos relacional.	Puesta en producción rápida. Flexibilidad, modularidad. Minimal.
Ventajas	Mucha documentación y gran comunidad.	Minimalista, Soporta No-Sql.

Tabla 3.9: Comparativa Flask y Django.

3.3.3. Patrón arquitectónico

Tras haber elegido qué *framework* se va a utilizar, queda por elegir el patrón arquitectónico que se seguirá en la construcción de la aplicación. De entre los que hay, los que más se adaptan a la idea de la aplicación que se plantea construir son la Programación por capas [12], y el Modelo-Vista-Controlador (MVC) [25].

Ambos patrones son muy utilizados en aplicaciones web, y sus propiedades permiten el correcto desarrollo que queremos para nuestra aplicación, por tanto sólo queda ver las ventajas y desventajas que ofrece cada uno, para así poder elegir un patrón que defina la arquitectura de la aplicación.

El patrón **Programación por Capas**, está basado en una arquitectura Cliente-Servidor. En ella se pueden diferenciar tres elementos principales que son: Cliente-Servidor-Datos [1], los cuales componen, en este caso, un modelo de 3 capas. La idea detrás de esto es aislar las capas unas de otras para que en caso de tener que realizar cambios sean solo sobre un capa sin que afecten a las demás.

- **Capa de presentación (Cliente)**: Esta capa es la capa que los usuarios ven, es decir, la interfaz gráfica, o en caso de una aplicación Web es la capa de Web. Esta capa es la que se encarga de comprobar que no haya fallos de formato.

- **Capa de negocio (Servidor)**: A esta capa le llegan las peticiones del usuario y se encargar también de mandar las respuestas a estas peticiones. En esta capa se definen las distintas normas que deben cumplirse para el correcto funcionamiento de la aplicación. Además es la capa intermedia, es decir, se comunica con la cama de presentación, para recibir solicitudes y mandar resultados, y con la de datos para solicitar datos o almacenarlos.

- **Capa de datos (Datos)**: En esta capa se encuentra la base de datos, la cual está formada por uno o más gestores. En esta capa es donde se almacena la información.

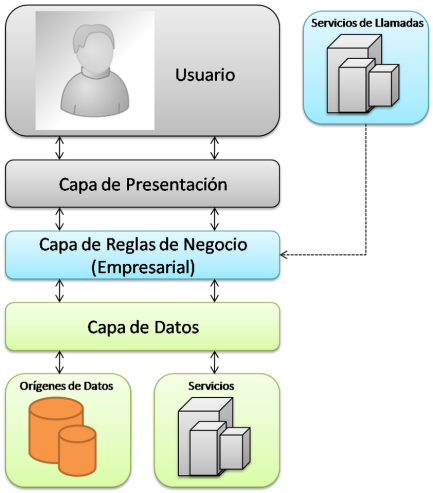

Figura 3.2: Programación de 3 capas.

Pese a que se ha hablado de un modelo cliente-servidor, el patrón permite que todas las capas puedan residir en un único ordenador, lo que permite que se pueda realizar de cara al Trabajo de Fin de Grado, pese a que lo normal sea encontrar un sistema con esta arquitectura dividido en varios ordenadores, quedando separada la capa de presentación de las capas de negocio y de datos.

Por otro lado está el patrón **Modelo-Vista-Controlador (MVC)**, el cual se basa en la reutilización de código y en la separación de conceptos, un modelo, una vista y un controlador, para así facilitar la tarea de mantenimiento del sistema o de la ampliación de la funcionalidad del mismo. Por tanto queda ver qué función hace cada uno dentro del sistema [7]:

- **Vista:** La vista es la parte que ve el usuario, es decir, la interfaz, ya sea una interfaz web, como una de escritorio. Para representar la in-

formación requiere de las normas del modelo, ya que esta sólo muestra la información tal y como el modelo se lo indique.

- **Controlador:** El controlador es el que recibe las peticiones del usuario, y le pasa el modelo esta solicitud. El controlador es un intermediario entre la vista y el controlador.
- **Modelo:** El modelo es la parte que se encarga de procesar todas las peticiones del usuario, para una vez haber cumplido todo lo solicitado por el usuario poder mostrarlo en la vista.

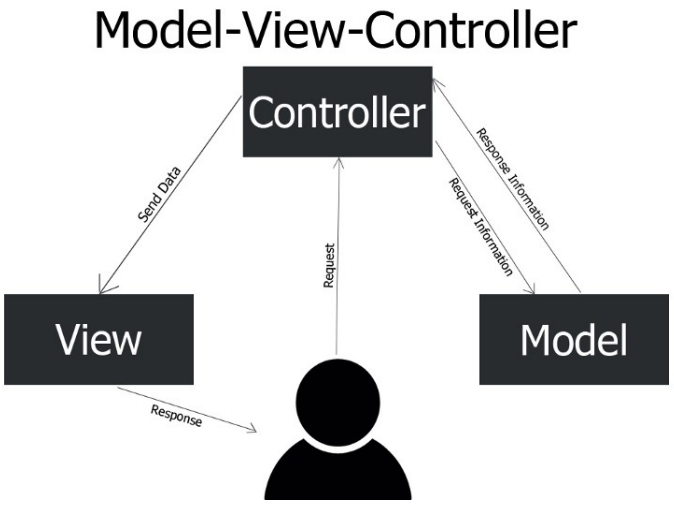

Figura 3.3: Ejemplo MVC.

Un pequeño ejemplo para entender cómo funcionan es el siguiente:

- El usuario solicita la URL de una Web.
- El Controlador recibe esta petición y encarga al modelo que prepare lo necesario para mostrarle al usuario la solicitud.
- El modelo prepara todos los datos necesarios y los organiza para manda a la vista.
- La vista muestra al usuario mediante el navegador web la url solicitada.

Se puede ver que ambos patrones son parecidos entre sí, pero MVC pretende conseguir una mayor separación entre la vista y el controlador (que corresponderían, más o menos, a las capas de presentación y de negocio respectivamente) para así mantener su implementación en paralelo, sin que se vean afectadas unas por las otras.

Una vez vistas las diferencias entre embos, queda escoger un patrón para nuestra aplicación, y dado que flask es un microframework, permite que elijamos el que más se adapte a la aplicación, cosa que por ejemplo con Django no se podría hacer ya que el framework está hecho para que se use el patrón **MVC**.

Por cómo se va a estructurar la aplicación [31], además de por cómo se ha pensado un diseño inicial de la organización de los archivos, se optará por usar el patrón arquitectónico MVC.

3.3.4. Análisis de la base de datos

Por último queda decidir qué tipo de base de datos va a ser utilizada. Dado que se está utilizando Flask se pueden utilizar tanto las bases de datos no relacionales como las relacionales, por lo que se van a exponer las diferencias entre ambas y se decidirá cual es la que más se ajusta a la aplicación que se quiere desarrollar.

Las bases de datos relacionales tienen las siguientes ventajas [39]:

- Una operación o se hace o no se hace utilizando rollback, de forma que mantenemos seguridad en si se han realizado las operaciones o no, dando así consistencia al sistema.
- Los datos deben cumplir los requisitos de integridad, tanto en tipo de dato como en compatibilidad.
- Debido al que lleva mucho tiempo en el mercado, dispone de muchas funcionalidades creadas para que pueda soportar todo tipo de datos y de documentos.

Por su parte las bases de datos no relacionales ofrecen [21] [39]:

- Un modelo de datos flexible: Estas bases de datos surgieron para poder satisfacer los requisitos de datos que predominan en las aplicaciones modernas, ya sea un documento, un gráfico, un valor clave, etc. lo que facilita el almacenamiento de datos de cualquier estructura.
- Escalabilidad y rendimiento: Este tipo de base de datos fue construida por el hecho de conseguir más escalabilidad que con una relacional, por lo que permiten que haya fragmentación o partición de esta, es decir, permite que la base de datos se amplíe en hardware, en la nube, permitiendo así un mayor rendimiendo y capacidad de almacenamiento.
- Estas bases de datos, a su vez, fueron diseñadas para estar siempre disponibles, ya que están diseñadas para ejecutarse en muchos nodos.

Sin embargo, al usar una base de datos no relacional, se han sacrificado algunas de las capacidades criticas que se suele esperar de una base de datos relacional. Pese a esto, se ha decido que una base de datos no relacional se adapta más al desarrollo de nuestra aplicación, y por tanto se usará este tipo de base de datos, en concreto se desarrollará bajo MongoDB.

SQL vs SQLIte

Pese a que se ha optado por usar una base de datos no relacional, otra opción que se planteó en el inicio junto a SQL fue el uso de SQLite[19]. Esta comparativa se realizó para explorar otras posibles soluciones al problema a parte de la escogida, para que, en cualquier caso que fuera necesario cambiar el tipo de base de datos, se supiera por cual se optaría.

SQL [2] es un lenguaje de consultas estructurado, con el cual se realizan las consultas sobre una base de datos relacional. SQL es un estándar de cómo crear un módelo relacional, cómo añadir datos ó actualizarlos. Este tipo de base de datos deben ser ejecutadas como un servicio del sistema para así estar activas, en caso contrario no funcionarán.

Por su parte SQLite[19] es una base de datos embebida basada en SQL. SQLite lee y escribe directamente en disco, ya que los datos van almacenados directamente a un archivo, es por ello que SQLite es más parecido a fopen() que, por ejemplo, a MySQL. SQLite es utilizado, sobre todo, en dispositivos móviles, ya que es server-less, y también se usa en algunos ordenadores, ya que puede llegar a ser más rápido que el propio sistema de archivos que el que trae el sistema operativo, mientras que aquellos que están basados en SQL se usan principalmente en servidores.

También hay algunas pequeñas diferencias como pueden ser que SQLite permite usar ïf not exists", que puede usarse para, por ejemplo, crear una tabla en caso de que no exista ya.

Dado que la idea es mantener la aplicación en un servidor web, lo ideal, en caso de que nos decantásemos por una base de datos relacional [2], sería usar una base de datos que use SQL, como puede ser MySQL u Oracle entre otras.

Aún así hay que recordar que en el desarrollo de la aplicación se utilizará MongoDB, ya que tras el análisis realizado se llegó a la conclusión de que era mejor opción para el sistema que otro tipo de base de datos.

Capítulo 4

Diseño

El cuarto capítulo está destinado al diseño de la aplicación que se va a desarrollar. Para ello capítulo se va a dividir en las siguientes cuatro secciones. La Sección 4.1 tendrá el diagrama de componentes del sistema y el diagrama de clases del sistema, en el que se aportará un diagrama de clases completo, el cual se dividirá en 2 diagramas más pequeños que representarán el **Modelo** y el **Controlador** del sistema. Se ha considerado hacerlo así para un mejor entendimiento del diagrama de clases y poder ver que se aplica correctamente el modelo vista controlador, además de que el diagrama del controlador es el que correspondería al de una REST-API. La Sección 4.2 será la sección del diseño de la base de datos. La Sección 4.3 corresponderá con el diseño de la interfaz web en la que se explicará el proceso por el que se ha pasado hasta llegar al estado actual de la aplicación.

4.1. Diagrama de clases

En esta sección se mostrará primero el diagrama de clases de la aplicación. Si bien, al tratarse de una RESTful-API el diagrama es distinto a un diagrama habitual [37], por tanto dado que se genera un diagrama muy amplio y que puede llegar a ser difícil de entender, se dividirá en dos diagramas, uno que mostrará la parte del controlador 4.2 del sistema, es decir, el diagrama que mostrará cómo avanza el usuario en la aplicación. El otro diagrama será el diagrama del modelo 4.3, donde sí se podrá identificar un diagrama de clases al uso.

Antes de presentar el diagrama de clases se dirán qué clases componen cada uno de los diagramas. El primer diagrama que se comentará será el diagrama de clases del modelo, el cual contiene las clases principales para el funcionamiento de la aplicación. En el modelo se encuentran el módulo de acceso y control de la base de datos, **dbmanager**, el módulo de preproce-

samiento, **Preprocesamiento**, en el que podemos encontrar las funciones para para procesar los datos que cargue el usuario. También se encuentra el módulo de clasificación, **Clasificación**, el cual contiene a la clase **Algorithm**, la clase que implemente los algoritmos que el usuario podrá elegir para generar un modelo. Otro módulo a encontrar es el módulo de lectura y escritura de datos, **Entrada**, con el cual se cargan los datos en la base de datos y se mandan al usuario. El último módulo es el de visualización y resultados, **Visualización**, con el que se obtendrán las métricas para obtener los resultados, además de mostrar algunas gráficas al usuario.

El otro diagrama que se tiene es el diagrama de clases del controlador. En el diagrama de clases del controlador es un diagrama de clases de una REST-API. Este tipo de diagramas muestra todos los caminos posibles que puede realizar el usuario dentro de la aplicación, desde el inicio, en la página principal, hasta el último paso que pueda realizar, además también hay que mostrar las conexiones con los módulos del modelo y las funciones que utilizan. En el controlador encontramos la clase **Home**, la página principal, y se conecta con la clase **Upload**, **Model** y **Models**. La clase **Upload** es la que permite al usuario que cargue los datos, y se conecta con la clase **Preprocessing**, la cual se conecta a su vez con **Classification**. Tras esto esta rama y la rama de **Models** llevan al mismo punto, la clase **Model**, desde la que se avanza hacia la clase **Testing**, la cual es la última página a la que puede acceder el usuario.

No se ha realizado diagrama de clases para la vista, ya que la vista se le muestra al usuario y en caso de haber un diagrama de clases de la vista coincidiría con el del controlador, puesto que es el controlador el que manda al usuario qué ver en cada momento, y coincide con la que se encuentre el controlador.

Una vez se ha mostrado el diagrama de clases completo (ver Figura 4.1) se procede a mostrar el diagrama de clases que corresponde al modelo (ver Figura 4.3) y el que corresponde al controlador(ver Figura 4.2) para que se puedan visualizar más fácilmente.

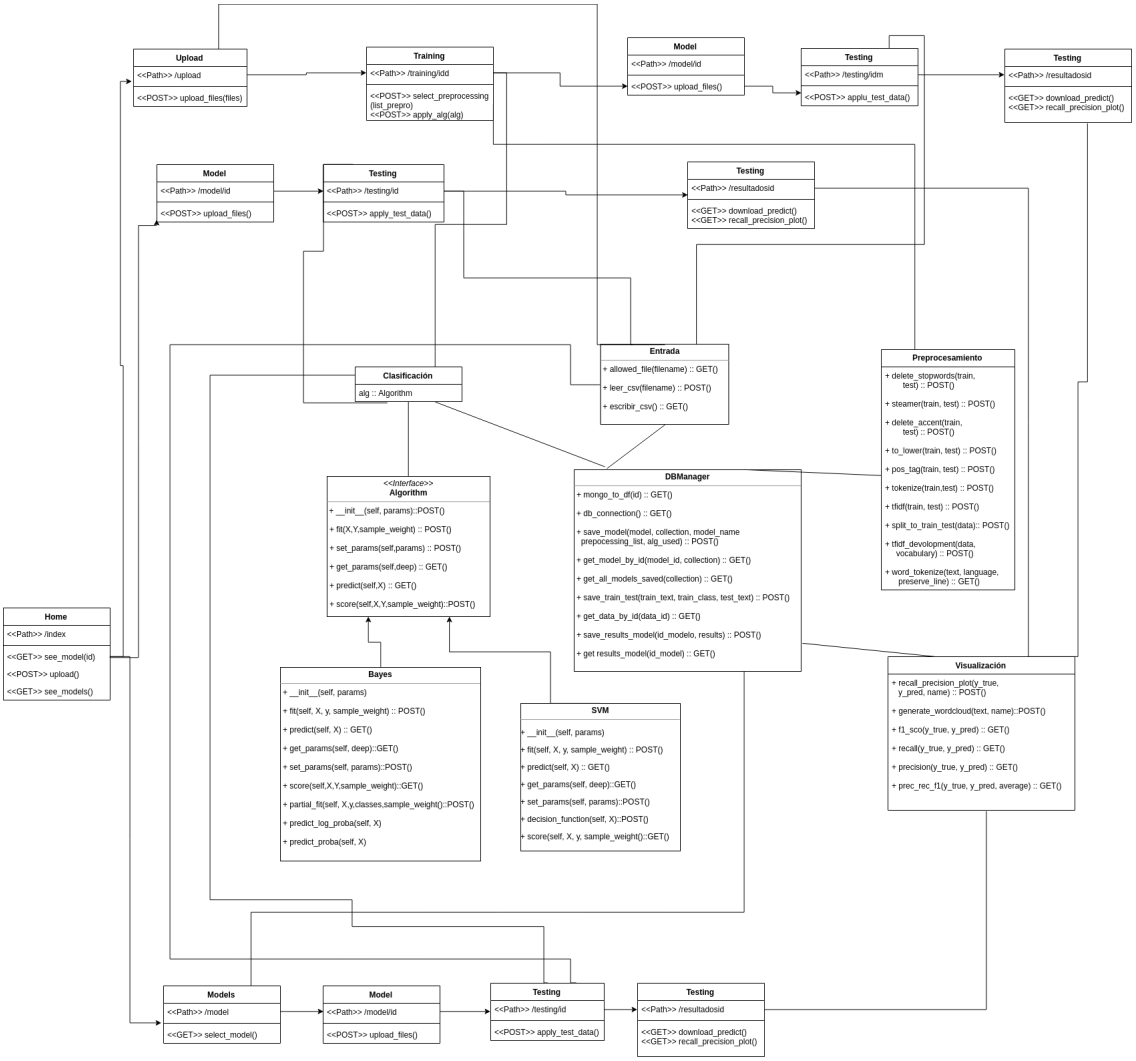

Figura 4.1: Diagrama de clases.

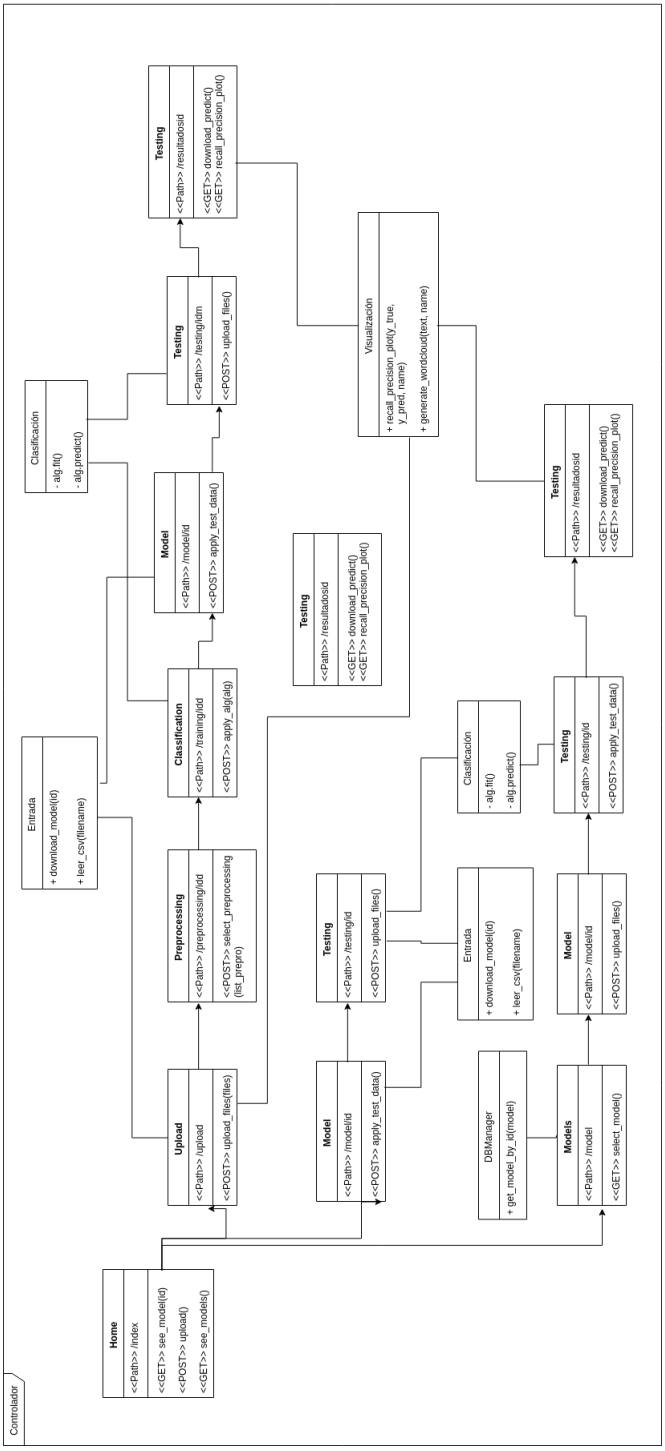

Figura 4.2: Diagrama de clases del Controlador.

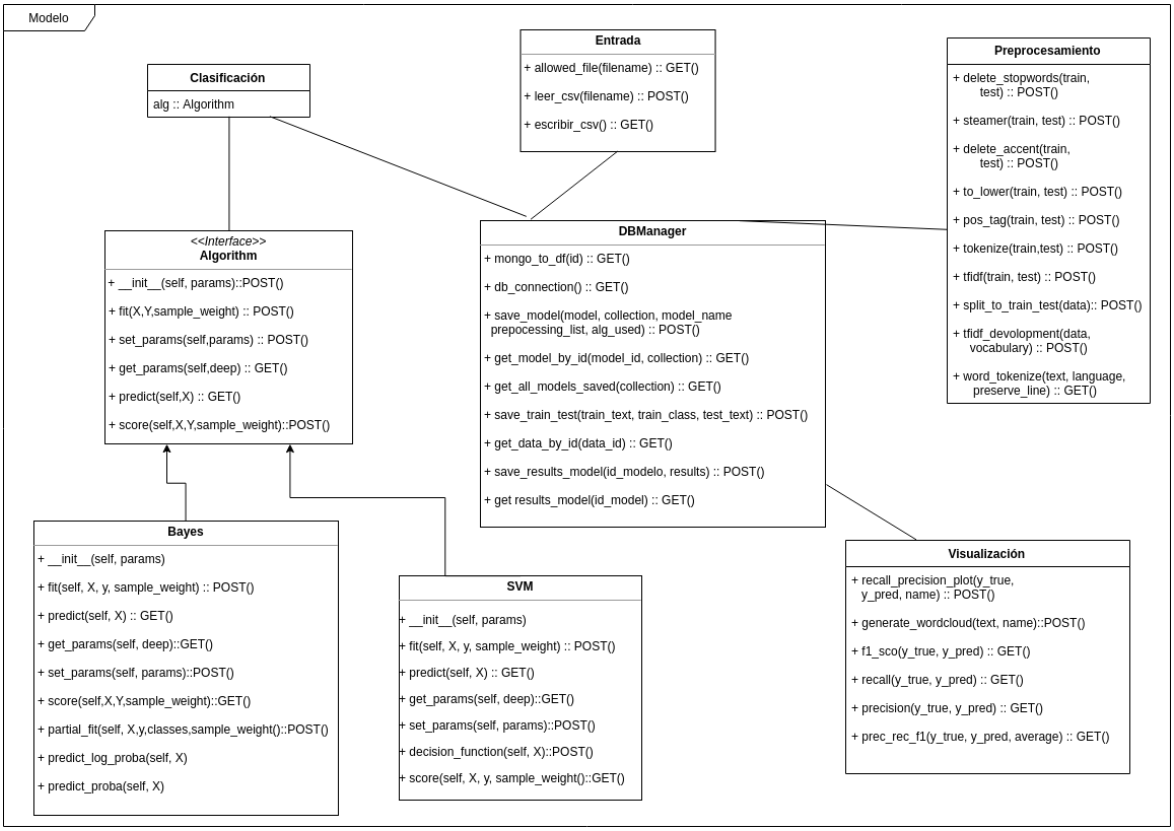

Figura 4.3: Diagrama de clases del Modelo.

4.1.1. Diagrama de componentes

Tras haber mostrado el diagrama de clases de la aplicación, también se realizó el diagrama de componentes para así poder identificar mejor la interacción con los distintos módulos del sistema.

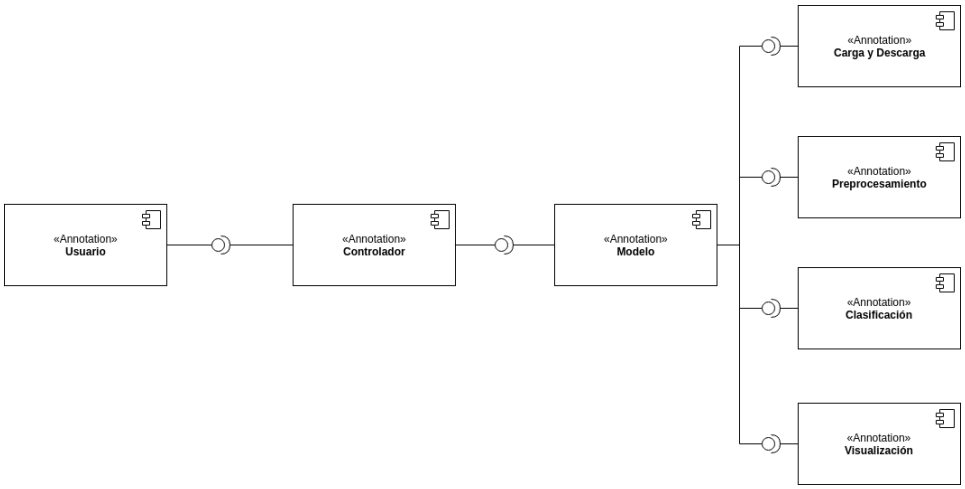

Figura 4.4: Diagrama de componentes de la aplicación.

4.2. Diagrama de base de datos

La base de datos que se ha utlizado es una base de datos no-SQL, como se concretó en el capítulo 3. Para poder realizar la representación de la base de datos hay que tener en cuenta el formato, en este caso como se está usando mongodb [39] y el formato que utiliza es JSON. Normalmente el proceso [38] consta de varios pasos, primero creando el diagrama entidad relación, después normalizando y por último de-normalizando para así obtener el resultado final. El proceso realizado ha sido el siguiente:

Una vez se ha realizado el modelo entidad relación se realiza el correspondiente paso a tablas.

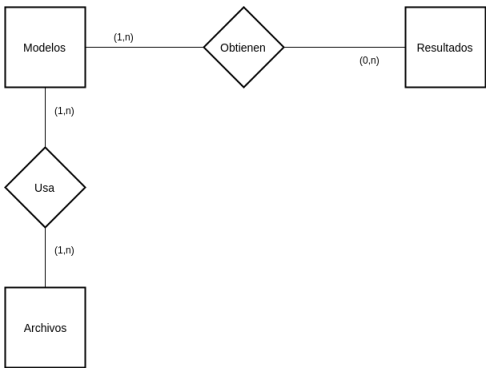

Figura 4.5: Modelo Entidad Relación de la base de datos.

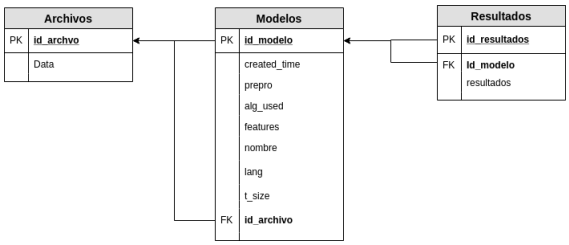

Figura 4.6: Modelo relacional de la base de datos.

Por último queda la des-normalización, para ello se toman las tablas resultantes tras el paso a tablas y se obtiene la representación NoSQL de la base de la base de datos del sistema:

```
 # Modelos                    # Resultados
{                             {
  "_id":"id_modelo",              "_id":"id_resultados",
  "created_time":"time",          "id_modelo":"id_modelo",
  "Prepro":"prepro_list",         "resultados": (
  "alg_used":"alg",                   "Recall":"recall",
  "Features":"features",              "Precicion":"precicion",
  "nombre":"nombre",                  "F1-Score":"f1score",
  "lang":"lang",                      "Predict":"pred"
  "t_size:"t_size",               )
  "id_archivos:"id_archivo
}                             }


# Archivos
{
  "_id":"id_archivo",
  "Data":data
}
```

4.3. Diseño de la aplicación Web

El diseño web se comenzó una vez se completó tras añadir toda la funcionalidad de la aplicación, para que así se supiera qué información se tenía y cual era la mejor manera de mostrarla. Antes de proceder con la programación del archivo **css** se comenzó por un primer diseño a mano (ver Figuras 4.7 y 4.8) para aproximar la idea final y cómo deberían aparecer los datos en la pantalla.

En las siguientes imágenes se muestra cual era la idea inicial, en la que se realizaron los distintos estados que alcanzaría la app según el diagrama de clases creado.

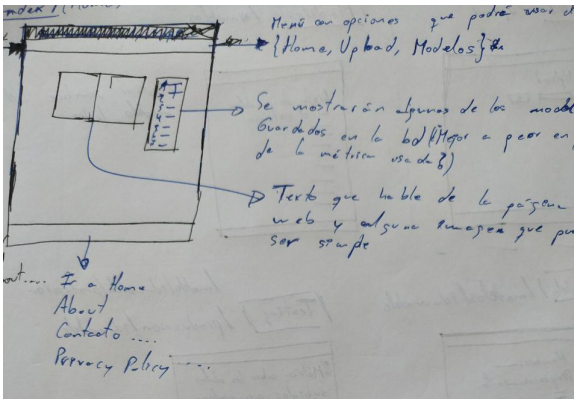

Figura 4.7: Diseño Web: Boceto inicial de la página principal.

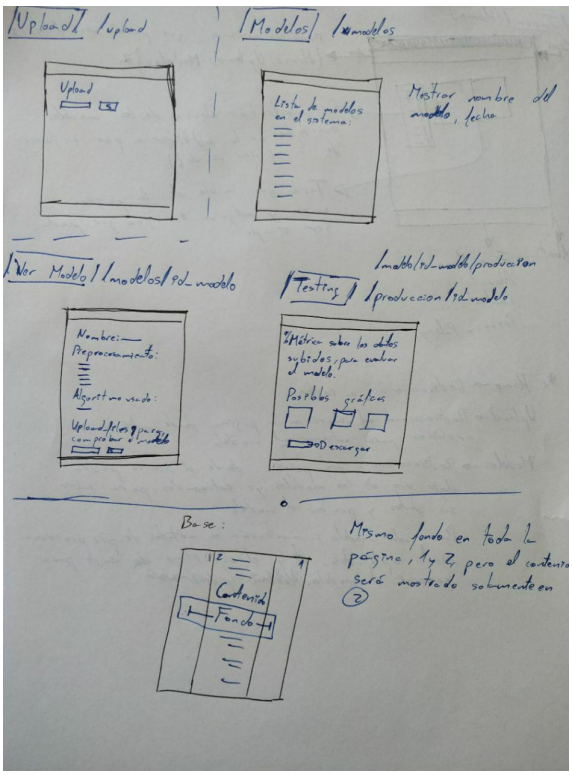

Figura 4.8: Diseño Web: Boceto inicial de todos los estados que alcanza la aplicación.

Tras realizar los diseños a mano se procedió con la creación del archivo **css** intentando aproximar y realizando algunos cambios que fuesen convenientes. Además para este proceso se tomó a una persona ajena al proyecto y que no tiene conocimiento del campo para tomar su opinión en consideración a lo largo del desarrollo, sobre el aspecto de la aplicación, ya que es importante que la aplicación llegue a ser accesible para todo el mundo.

En las primeras fases en las que se estaba probando la funcionalidad y añadiendo poco a poco el diseño planteado, se plantearon varios fondos, algunos de ellos son los siguientes:

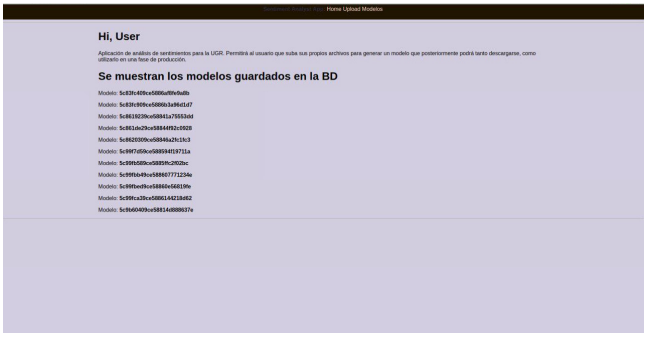

Figura 4.9: Diseño Web: Pruebas del color del fondo en el diseño de la aplicación I.

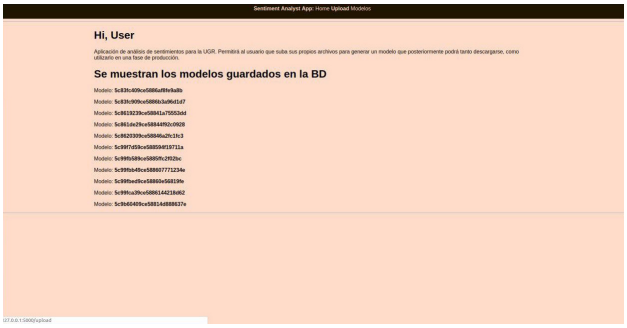

Figura 4.10: Diseño Web: Pruebas del color del fondo en el diseño de la aplicación II.

Finalmente se optó por la última opción de las dos anteriores mostradas, ver Figuras 4.9 y 4.10, de entre las posibles que había, no solo estaban estas dos, y se prosiguió con la mejora del diseño y adición de texto que ayudase al usuario. Además se consultó con esta persona el tipo de fuente y se presentaron distintas alternativas entre las que se encuentran las siguientes dos:

Tras distintas muestras de fuentes [36], ver Figuras 4.11 y 4.12, se decidió elegir dos fuentes distintas, una para la barra de navegación, y otra para el texto de la aplicación. También se decidió aumentar el tamaño de la fuente para lograr una lectura más fácil.

Al finalizar todo el proceso se llegó al siguiente diseño para cada uno de los estados de la aplicación:

Dentro del sistema se pueden observar dos fases distintas, la fase de desarrollo y la fase de producción. En la fase de desarrollo el usuario podrá entrenar un modelo para unos datos cargados, mientras que en la fase de

Figura 4.11: Diseño Web: Pruebas de fuentes I.

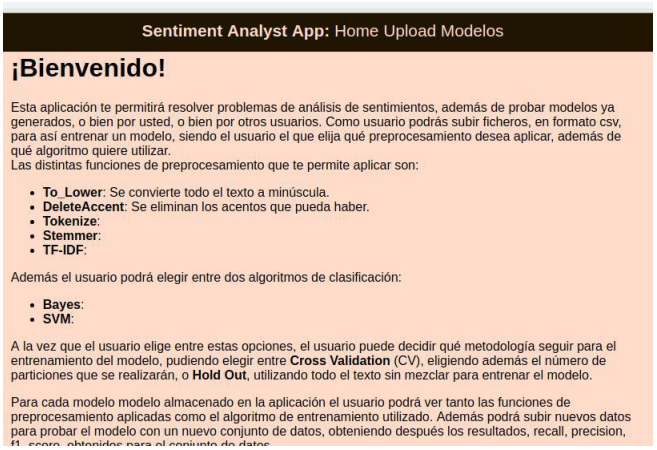

Figura 4.12: Diseño Web: Pruebas de fuentes II.

producción el usuario podrá usar cualquiera de los modelos que haya ya y proporcionar nuevos datos para obtener resultados y posteriormente descargarlos.

Por tanto la fase de producción abarca las partes de carga de archivos del modelo (ver Figura 4.15) y la del entrenamiento del modelo (ver Figura 4.16), terminando en (ver Figura 4.14). Además en esta última no sólo termina la fase de desarrollo, sino que también empieza la fase de producción, en ella el usuario cargar nuevos datos y posteriormente obtiene los resultados.

Figura 4.13: Diseño Web: Estado final del Index.

Figura 4.14: Diseño Web: Estado final de la página del Modelo.

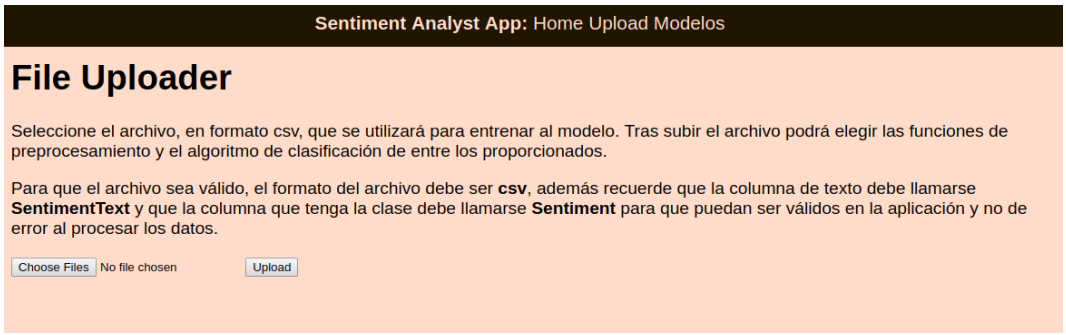

Figura 4.15: Diseño Web: Estado final de la carga de archivos.

Figura 4.16: Diseño Web: Estado final de la fase de entrenamiento.

Capítulo 5

Implementación

En este capítulo se va a explicar tanto el lenguaje de programación utilizado como la implementación del proyecto que se ha realizado, para ello se dividirá en secciones, cada una para un módulo del proyecto, ya que se planteó una implementación modular para así poder realizar futuras ampliaciones sobre la aplicación sin que suponga mucho trabajo. Para cada sección se dirá qué métodos se han implementado y la funcionalidad que tiene ese método. En la Sección 5.1 se hablará tanto del lenguaje utilizado como del framework utilizado para desarrollar el sistema. La Sección 5.2 será la correspondiente al módulo encargado de manejar la base de datos. La Sección 5.3 corresponde al módulo de preprocesamiento. La Sección 5.4 corresponde al módulo de entrenamiento. La Sección 5.5 engloba la visualización y los resultados. La Sección 5.6 es la dedicada al controlador. Por último la Sección 5.7 para tratar la implementación de la vista, es decir, los templates.

En la imagen 5.1 se muestra la estructura [31] que se ha tomado a la hora de realizar la aplicación.

5.1. Lenguaje de programación utilizado

Como se vio en el capítulo del análisis que se iba a realizar un servicio web, quedaba la elección del lenguaje de programación utilizado en la implementación. Para la realización de este sistema se decidió que se utilizaría Python[14], ya que cuenta con librerías conocidas y de calidad para la ciencia de datos, big data, e incluso para el desarrollo web. Por tanto Python se vio como el mejor lenguaje de programación posible para este proyecto desde el primer momento.

La versión Python utilizada para el desarrollo de la aplicación web es la 3.6.6, y para poder realizar un desarrollo en el que se mantuvieran las

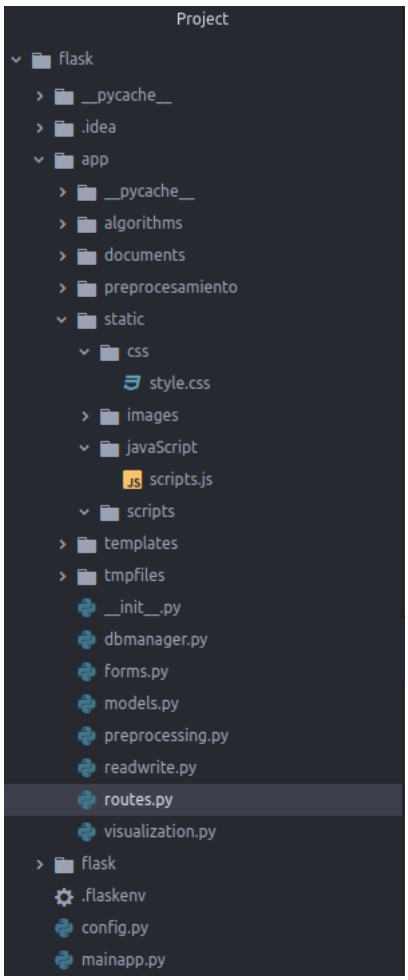

Figura 5.1: Estructura de la programación de la aplicación.

versiones de los paquetes desde el inicio y no actualizar a no ser que se quisiera se creó un entorno virtual.

5.2. Control de la base de datos

Este bloque combina las funciones que se van a encargar de acceder a la base de datos y el archivo que corresponde con estas operaciones es "db-manager.py". De esta forma nos aseguramos que ninguna otra función del sistema pueda acceder a la base de datos por motivos de seguridad. Como la implementación se está realizando en Python y como base de datos se ha optado por MongoDB [39], se usará el paquete PyMongo [4] para poder realizar todas las operaciones relacionadas con la base de datos.

db_connection: Este método sirve para establecer una conexión con la base de datos, y así no tener que repetir este código en el resto de funciones.

mongo_to_df: Devuelve un DataFrame con los datos cargados por el usuario en el formulario de subida.

save_model: Guarda el modelo entrenador en la fase de entrenamiento en la base de datos junto a sus características, para así poder usarlo en la fase de producción.

get_model_by_id: Devuelve el modelo entrenado para que pueda ser utilizado durante la fase de producción para así realizar el predict de nuevos datos.

get_pickle_model: Devuelve el modelo comprimido. Esta función se ha creado para permitir en un futuro que el usuario pueda descargar el modelo que ha entrenado y no sólo tener que usarlo en la aplicación web.

get_all_models_saved: Devuelve una lista con los ids y los nombres de los modelos almacenados en la base de datos para mostrarlos al usuario.

get_data_by_id: Devuelve el conjunto de datos que será utilizado durante el entrenamiento del modelo.

save_results_model: Guarda los resultados obtenidos por el modelo durante la fase de producción.

get_saved_results: Devuelve los resultados obtenidos por el modelo durante la fase de producción.

save_archivos: Guarda en la base de datos los archivos cargados por el usuario en el formulario de subida.

5.3. Preprocesamiento

El módulo de preprocesamiento es el que tiene las posibles funciones que se han aplicado para procesar los datos antes de ser entrenados. Para la creación de estas formulas se han usado las librearías NLTK [11] y Scikit-Learn[15]. NLTK[2] se trata de una librería que contiene alrededor de 50 corpus y recursos lingüisticos, y estos últimos permitirán la realización del procesamiento del texto comentado en el capítulo dos. Scikit-Learn[1] es la otra librería utilizada tanto en el modulo de preprocesamiento como en el

[2]Natural Language Toolkit, `https://www.nltk.org/`

[1]`https://scikit-learn.org/stable/`

módulo de entrenamiento, y tiene diferentes herramientas para el análisis de datos. Las herramientas que incluye Scikit-Learn, van desde corpus, pasando por funciones de preprocesamiento para tratar los datos, hasta diferentes algoritmos tanto para aprendizaje supervisado, como no supervisado, como semi-supervisado. Dentro de este módulo se pueden diferencias las siguientes funciones:

split_to_train_test: Método para dividir los datos cargados por el usuario en train y test y poder usarlos en el entrenamiento.

word_tokenize_t: Tokeniza las frases que recibe utilizando el Treebank-WordTokenizer [22]. Funciona para datasets en español y en inglés.

tokenize: Método para tokenizar los conjuntos de datos recibidos. Este método se ayudará del método word_tokenize_t para tokenizar cada unas de las instancias de los conjuntos de datos.

delete_stopwords: Método que elimina las stopwords de los conjuntos de datos, tanto en español como en inglés.

pos_tag: Método que obtiene la categoría gramatical de cada palabra.

stemmer: Método para calcular el stem (raíz) de cada palabra.

delete_accent: Método para eliminar todos los acentos que puedan encontrarse en los conjuntos de datos.

to_lower: Método que sirve para convertir todo el texto a minúscula.

TF-IDF: El esquema TF-IDF mezcla dos medidas en una, la medida TF y la medida IDF. La primera medida calcula las veces que aparece una palabra en todos los documentos, dando más importancia a las que más veces aparezcan, y la segunda da importancia a aquellas palabras que aparezcan mucho en pocos documentos. De esta forma se logra dar una mayor importancia a términos que son relevantes y restándosela a otros que aparezcan más. Por tanto este método que calcula el esquema TF-IDF para los datos recibidos. Entrena un modelo que servirá posteriormente para aplicarlo sobre los datos en la fase de producción.

tfidf_devolopment: Método que calcula el TF-IDF para los datos cargados durante la fase de producción. Utiliza el modelo generado durante la fase de desarrollo para mantener el mismo vocabulario, ya que sino no se podría calcular la predicción sobre el conjunto da datos nuevo.

5.4. Entrenamiento

De cara a conseguir una implementación que sea modular y que permita añadir diferentes algoritmos para el entrenamiento de un modelo en el futuro, se ha optado por crear una clase abstracta [5] de la cual heredarán el resto de algoritmos implementados. La implementación se ha llevado a cabo dentro la carpeta de algorithms 5.1, en el archivo .ªlgoritmos.py"

5.4.1. Class Algorithm

Esta clase abstracta sólo tiene aquellas funciones que son comunes a los algoritmos implementados,:

- **__init__**: Método que inicilizará el algoritmo con los parámetros recibidos.
- **fit**: Método que llamará el algoritmo para así generar un modelo entrenado.
- **set_params**: Método que permite cambiar parámetros del algoritmo.
- **predict**: Método que, tras haber sido entrenado el modelo, calcula la predicción que realiza el modelo sobre los datos.
- **get_params**: Método para obtener los parámetros del modelo
- **score**: Calcula la métrica .ªccuracy"del modelo sobre los datos de test.

Los algoritmos que se utilizan en la aplicación son Naive Bayes [8] [9] y SVM [17] [18]

Bayes

Esta clase utiliza el algoritmo MultinomialNB [8] de sklearn. Los parámetros que se pueden cambiar son los siguientes:

- alpha
- fit_prior
- class_prior

Además de implementar los métodos de la clase Algorithm, implementa los siguientes:

- **partial_fit**: Realiza algunos ciclos del entrenamiento del modelo. Para que se entrene el modelo mediante este método, deberá usarse múltiples veces.
- **predict_log_proba**: En vez de hacer el predict con el entrenamiendo del modelo, a este le aplica la probabilidad logarítmica.
- **predict_proba**:Realiza el predict basándose en la probabilidad.

SVM

Esta clase utiliza el algoritmo SVC [18] de sklearn. Los parámnetros que se permiten configurar son los siguientes:

- gamma
- C
- kernel
- degree
- coef0
- shrinking
- probability
- tol
- cache_size
- class_weight
- verbose
- max_iter
- decision_function_shape
- random_state

Por su parte esta clase además de implementar los métodos de la clase Algorithm, implementa el siguiente método:

- **decision_function**: Evalua la función de decisión, parámetro del algoritmo, para el conjunto de datos dado.

5.5. Visualización y resultados

Se ha decidido unir en el mismo módulo las funciones de visualización y de resultados. El archivo que contiene las funciones implementadas es "visualization.py". Las funciones que se han implementado son:

- **prec_rec_f1**: Calcula las métricas "Precision", Recallz "F1_score"[10].
- **precision**: Calcula la métrica "Precision"[10].
- **recall**: Calcula la métrica Recall"[10].
- **f1_sco**: Calcula la métrica "F1_Score"[10].
- **recall_precision_plot**: Genera y almacena un gráfico que compara las métricas "Precisionz Recall"[3].
- **generate_wordcloud**: Genera una nube de palabras a partir de los datos cargados por el usuario en la fase de desarrollo y la almacena [20].

5.6. Controlador

El módulo del controlador es el que se va a encargar de guiar al usuario a través de la web. Por tanto este modulo tendrá distintas funciones que redirigirán al usuario a una fase u otra. Las distintas funciones que contiene el controlador son:

- **index**: Muestra al usuario a la página principal de la aplicación.
- **pr_modelo**: Muestra al usuario a la página que muestra todos los modelos que hay almacenados en la base de datos. Clickando en alguno de los modelos mostrados le redirige a la página que contiene la información del modelo.
- **ver_modelo**: Muestra al usuario a la página que contiene la información relativa al modelo seleccionado. Además el usuario puede cargar un archivo para entrar en la fase de producción.
- **download_model_pred**: Manda al usuario el archivo con las predicciones del modelo sobre los datos cargados. El formato del archivo será csv.
- **testing**: Prepara los datos para que puedan se procesados por el modelo para posteriormente calcular la predicción del modelo sobre los mismos. Después redirige al usuario a la página de resultados.

- **upload**: Muestra la pagína desde la que el usuario puede seleccionar los archivos con los que se entrenará el modelo. Tras ello redirige a preprocessing.
- **preprocessing**: Muestra al usuario las distintas opciones con las que se puede hacer el entrenamiendo con los datos que han sido cargados, y tras haber sido seleccionados se entrena el modelo redirige a la página que muestra la información del modelo.

5.7. Vista

En este módulo se incluyen todos los archivos correspondientes a la vista. Esta está compuesta de las carpatas "static", la cual almacena el archivo **.css** y el archivo **.js** creados para la aplicación. Mientras que en la carpeta "templates"se encuentran todos los archivos **.html** que el controlador muestra al usuario. Estos archivos son:

- **base**: Contiene la estructura base, de forma que en los demás archivos sólo se cambie el bloque principal y no afecte al resto.
- **index**
- **pr_modelo**
- **upload**
- **testing**
- **train**: Página a la que redirecciona el controlador cuando está en "preprocessing".
- **ver_modelo**

Capítulo 6

Pruebas

Dado que no se hizo un desarrollo basado en tests, para comprobar que el sistema cumple con los requisitos nombrados en el capítulo de análisis. Para esto se utilizarán pruebas de caja negra sobre las fases principales y sobre el módulo de preprocesamiento. Este capítulo tendrá dos secciones en las que se pondrán los resultados de las pruebas. La Sección 6.1 tratará las pruebas de caja negra en los bloques principales, y después se realizarán pruebas de caja negra sobre las principales funciones del sistema 6.2, para así asegurar la mínima funcionalidad del mismo.

6.1. Pruebas caja negra para los bloques principales

Una prueba de caja negra[13] ignora cómo está programado sino los "datos de salida" para unos "datos de entrada". Dado que en el sistema se pueden diferenciar dos fases, una fase de desarrollo y una fase de producción, se van a hacer dos pruebas de caja negra, una para cada fase.

Para identificar ambas fases, se hará según el flujo de trabajo(ver Figura 6.1). La fase de desarrollo corresponde con "Upload" y "Preprocesamiento-Train", y la fase de producción está formada por "Testing" y "Resultados".

Figura 6.1: Flujo de trabajo del sistema.

Con estas fases se pretende que los requisitos del sistema expuestos en el capítulo 3 queden satisfechos, y que se pueda afirmar que se han logrado los objetivos.

Fase de desarrollo

La fase de desarrollo es la fase en la que se entrena al modelo en función de uno o varios archivos cargados por el usuario. En esta fase los datos de entrada serán los archivos en el formato aceptado por el sistema, formato **csv**, y los parámetros de entrenamiento seleccionados por el usuario. Pese a que se de la libertad al usuario para que elija qué preprocesamiento aplicar a los datos, se consideran mínimos y obligatorios la tokenización y la representación TFIDF. Por tanto aunque el usuario no los seleccione, estos serán aplicados por el sistema para que pueda ejecutarse correctamente. Como datos de salida se tiene al modelo generado.

Dado que las pruebas de caja negra se tienen que realizar por la interfaz del sistema, en este caso por la interfaz de la aplicación web, se añadirán imágenes del proceso seguido.

Primero el usuario desde el menú "Upload" deberá cargar los archivos en formato csv (ver Figura 6.2).

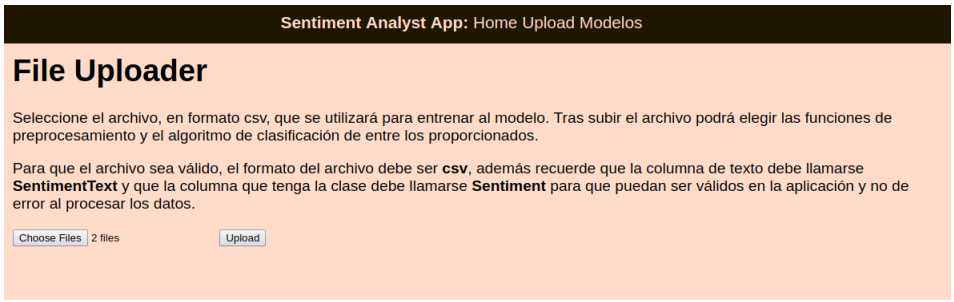

Figura 6.2: Pruebas de Caja Negra, fase de desarrollo: Carga de archivos.

Una vez se han cargado los archivos el usuario deberá elegir cómo se entrenará el modelo (ver Figura 6.3), para ello se le permite elegir las funciones de preprocesamiento, el idioma del texto que ha cargado, el tipo de partición para train-test que desea, además del algoritmo de clasificación a utilizar.

Tras terminar el proceso de entrenamiento el modelo quedará generado como se observa en 6.4

Fase de producción

La fase de producción parte de (ver Figura 6.4), donde se cargará el archivo el cual el usuario querrá realizar una predicción de los datos según el modelo entrenado. A estos datos se le aplicará el preprocesamiento mostrado en la pantalla del modelo.

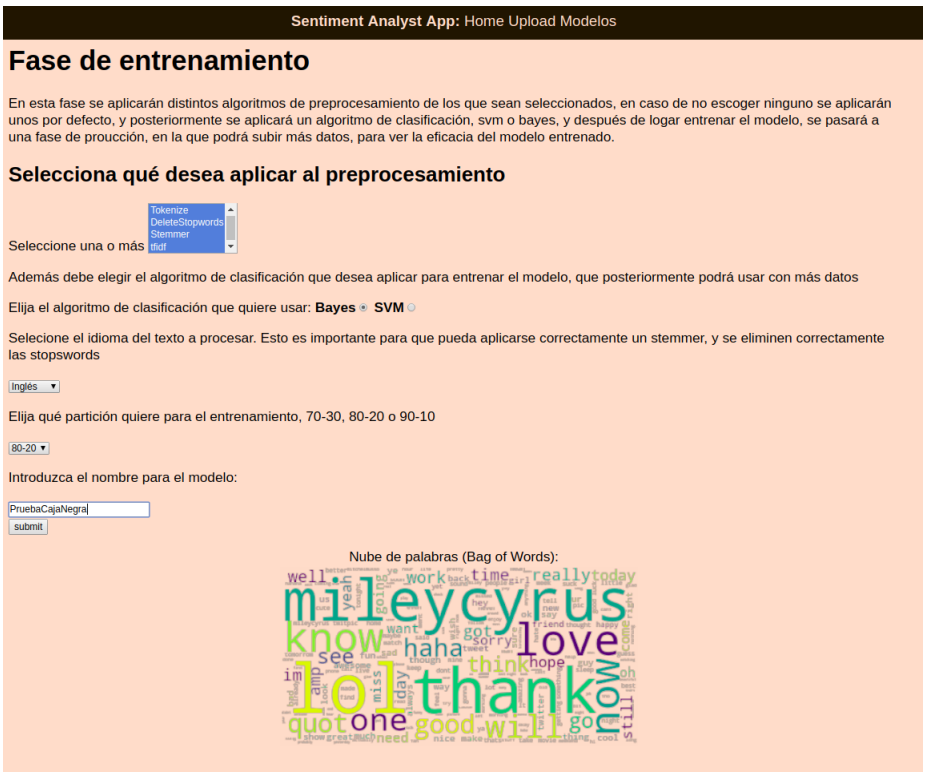

Figura 6.3: Pruebas de Caja Negra, fase de desarrollo: Elección de parámetros de entrenamiento.

Figura 6.4: Pruebas de Caja Negra, fase de desarrollo y producción: Modelo entrenado.

Tras cargar los archivos al usuario se le mostrarán los resultados obtenidos, además el usuario podrá descargar un archivo, en formato csv, con los resultados del predict del modelo (ver Figura 6.5).

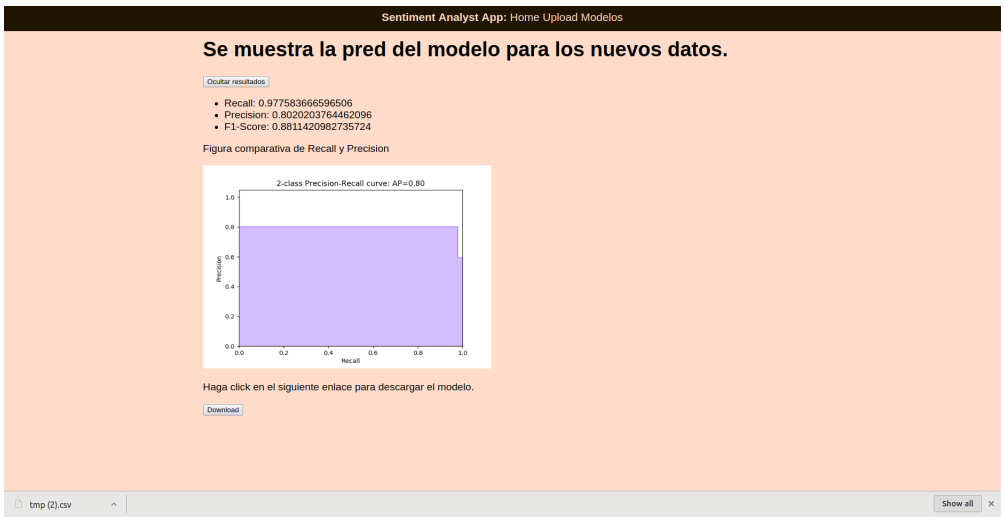

Figura 6.5: Pruebas de Caja Negra, fase de producción: Resultados del modelo y descarga.

Con estas pruebas de caja negra se ha comprobado que los requisitos descritos en el capítulo 3 han quedado satisfechos.

6.2. Pruebas de caja negra en las funciones

Estas pruebas se realizarán sobre las funciones nombradas en la implementación, en las cuales nos centramos principalmente en las que están dentro de bloque de preprocesamiento, ya que estas son importantes en el análisis de opiniones..

Pruebas de caja negra en preprocesamiento

Ahora se realizarán las pruebas de caja negra sobre las funciones del bloque de preprocesamiento. Para ello se utilizarán ejemplos que ayuden a una mejor explicación de estas pruebas.

Función **split_to_train_test**: Esta función recibe todos los datos cargados por el usuario como si fuera un DataFrame, y divide los datos en un conjunto de train y otro de test. Para ello se cargará un fichero de 5000 instancias y se aplicará una división de 80-206.6, es decir, 80 % de los datos serán para train y el 20 % restantes para test. Por tanto train debería quedar con una longitud de 4000 y test con una longitud de 1000.

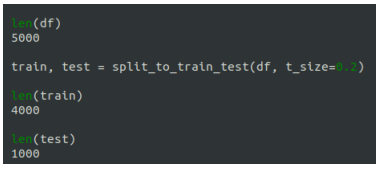

Figura 6.6: Prueba de Caja Negra: split_to_train_test 80-20.

En la imagen (ver Figura 6.6) se puede ver que la división de los datos ha sido correcta. Para comprobar que funciona para las opciones que se le ofrecen al usuario se añadirán las pruebas correspondientes, una para la división 70-30 (ver Figura 6.7)y otra para una división 90-10 (ver Figura 6.8).

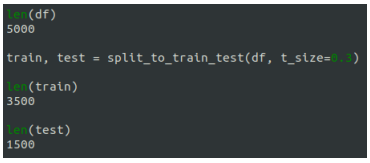

Figura 6.7: Prueba de Caja Negra: split_to_train_test 70-30.

Tras haber comprobado que funciona correctamente ahora se pueden comprobar las siguientes funciones.

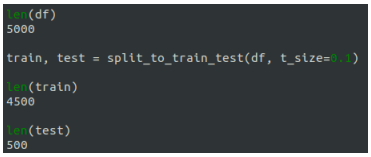

Figura 6.8: Prueba de Caja Negra: split_to_train_test 90-10.

Se hicieron 2 funciones de tokenización, realmente la que tokeniza es la función word_tokenize_t, que es la que se encarga de tokenizar, y con la función tokenize lo que se hace es dividir el texto en frases, ya que la primera de las funciones sólo tokeniza frases individuales.

Para comprobar la primera función se usará la frase "Esta frase es de prueba para tokenizar", y se debe obtener el siguiente resultado: ['Esta', 'frase', 'es', 'de', 'prueba', 'para', 'tokenizar'].

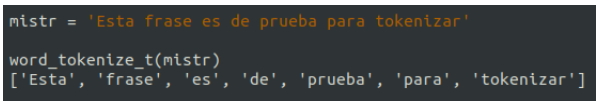

Figura 6.9: Prueba de Caja Negra: Función word_tokenize_t.

Ahora se comprobará que tokenize funciona bien, para ello se ha creado una lista con la misma frase anterior, añadida 3 veces, y por tanto se obtendrá el mismo resultado que para la función anterior, pero repetido 3 veces.

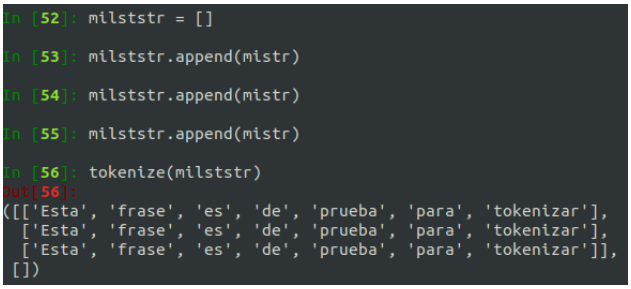

Figura 6.10: Prueba de Caja Negra: Función tokenize.

Ahora se comprobará el funcionamiento de las funciones que pasan a minúscula y eliminan acentos. Para ello se creará una frase que contenga tanto mayúsculas como acentos. La frase utilizada será: "ElimiNar TODAS LAS MAYUS y las ácétÓn qüe nòs encontrèmós por el camNO". El resultado tras aplicar las minúsculas será "eliminar todas las mayus y las ácétón qüe nòs encontrèmós por el camno", y el resultado tras eliminar los acentos

"eliminar todas las mayus y las aceton que nos encontremos por el camno". Dado que la función to_lower funciona con el tipo de dato "Series" de la librería "Pandas", se creará una variable con este tipo de dato.

```
In [98]: mistr = "ElimiNar TODAS LAS MAYUS y las ácétÓn qüe nòs encontrèmós por el camNO"

In [99]: milststr = []

In [100]: milststr.append(mistr)

In [101]: milststr.append(mistr)

In [102]: miserie = pd.Series(milststr)

In [103]: milower = to_lower(miserie)

In [104]: milower.values
Out[104]:
array(['eliminar todas las mayus y las ácétón qüe nòs encontrèmós por el camno',
       'eliminar todas las mayus y las ácétón qüe nòs encontrèmós por el camno'],
      dtype=object)

In [105]: miaccents = delete_accent(milower)

In [106]: miaccents
Out[106]:
(['eliminar todas las mayus y las aceton que nos encontremos por el camno',
  'eliminar todas las mayus y las aceton que nos encontremos por el camno'],
 [])
```

Figura 6.11: Prueba de Caja Negra: Convertir a minúsculas y eliminar acentos.

En la imagen (ver Figura 6.11) se ve que los resultados que se esperaban son los resultados obtenidos, por tanto las funciones que se están usando son correctas.

La siguiente función a comprobar es la función que elimina las palabras vacías, delete_stopwords. La lista varía según el idioma, en este caso se pondrá un ejemplo en español, se tokenizará y luego se eliminarán las stopwords. Para ello se usarán dos frases en inglés, la primera: "this is an example to prove that the delete stopwords function will work", y la segunda: "you may know now that some words will dissapear".

Para poder aplicar correctamente la eliminación de palabras vacías, previamente se deben tokenizar las frases, por tanto se usará antes la función tokenize, y después se mostrará el resultado tras eliminar las palabras vacías.

En la figura 6.13 se observa que varias palabras han sido eliminadas, y esto es porque esas palabras están categorizadas como palabras vacías. El sistema soporta tanto inglés como español, por tanto se podrán eliminar palabras vacías para ambos idiomas. La última función a comprobar de este módulo es stemmer, en esta función se usa el SnowballStemmer[16], puesto que permite seleccionar que se elija el idioma a usar. Para poner un ejemplo, se ha utilizado el ejemplo anterior, para ello se mostrará el resultado tanto

```
In [19]: mstr1
Out[19]: 'you may know now that some words will dissapear'

In [20]: mistr
Out[20]: 'this is an example to prove that the delete stopwords function will work'

In [21]: milststr
Out[21]:
['this is an example to prove that the delete stopwords function will work',
 'you may know now that some words will dissapear']

In [22]: delete_stopwords(tokenize(milststr))
Out[22]:
[['example', 'prove', 'delete', 'stopwords', 'function', 'work'],
 ['may', 'know', 'words', 'dissapear']]
```

Figura 6.12: Prueba de Caja Negra: Eliminar palabras vacías.

en frases en las que se han eliminado las palabras vacías como en las que no.

```
In [27]: stemmer(delete_stopwords(tokenize(milststr)), tokenize(milststr))
Out[27]:
([['exampl', 'prove', 'delet', 'stopword', 'function', 'work'],
  ['may', 'know', 'word', 'dissapear']],
 [['this',
   'is',
   'an',
   'exampl',
   'to',
   'prove',
   'that',
   'the',
   'delet',
   'stopword',
   'function',
   'will',
   'work'],
  ['you', 'may', 'know', 'now', 'that', 'some', 'word', 'will', 'dissapear']])

In [28]: milststr
Out[28]:
['this is an example to prove that the delete stopwords function will work',
 'you may know now that some words will dissapear']
```

Figura 6.13: Prueba de Caja Negra: Obtención de la raíz.

El módulo que contiene las clases con los algoritmos de entrenamiento se quedó validado en la sección anterior, ya que se comprueba cómo se entrena un modelo, se queda guardado. Además se pueden acceder a los resultados del modelo además de utilizar la función que calcula el resultado.

A pesar de las pruebas realizadas, sería conveniente la realización de más pruebas como pueden ser las pruebas de caja blanca, o pasar a un desarrollo basado en test, para así agilizar el desarrollo y mejorar la calidad final de la aplicación.

Capítulo 7

Conclusiones y trabajo futuro

Este proyecto se planteó para realizar la base de un sistema que sirva para aquellos que se dediquen al análisis de opiniones, o para aquellos que quieran iniciarse. No se pretendía la creación del sistema más completa, pero sí lograr que se crease un sistema funcional y que cumpliera unos requisitos para que en un futuro el sistema pueda ser ampliado en pocos pasos.

Los objetivos planteados al inicio del proyecto se han acabado cumpliendo, puesto que se ha hecho un estudio acerca del análisis de opiniones antes de realizar la aplicación. Esto era necesario puesto que para entender cómo trabaja una persona que se dedique a este campo y que así pueda desarrollarse un sistema acorde a sus espectativas. Además se ha seguido la metodología de ingeniería del software para así crear un sistema modular y de fácil mantenimiento, puesto que cuando hay un fallo se sabe dónde ir a resolver el fallo rápidamente.

Para la realización de este proyecto se ha tenido que realizar un estudio del tema de análisis de opiniones, puesto que el tema era desconocido antes de empezar, pero que ha despertado el interés al tratarse de un tema interesante y el cual desemboca en otro mayor y más interesante aún, como lo es el Procesamiento del Lenguaje Natural (PLN). Así mismo se ha trabajado por primera vez con un framework de Python, Flask, el cual también ha requerido de un estudio previo a plantar las bases del sistema para así asegurar un correcto desarrollo, y sobre el cual se han ido adquiriendo más conocimientos conforme se realizaba la aplicación y se ha intentado aprovechar ese aprendizaje para resolver posibles errores y para mejorar en todo lo posible la aplicación.

Trabajo futuro

Pese a que los objetivos principales del sistema han quedado cubiertos, el sistema cuenta con la funcionalidad justa para poder resolver un sistema de análisis de opiniones. No obstante se tienen pensadas las siguientes ampliaciones:

- Permitir al usuario cargar otro tipo de formato de archivo que no sea sólo csv.
- Añadir más funciones de preprocesamiento. No sólo optar por usar TF-IDF, sino también se puede optar por una representación vectorial de los documentos (embeddings).
- Se pretende que el usuario pueda personalizar complemtamente los parámetros de entrenamiento del modelo, no sólo con los parámetros por defecto, ya que al usuario puede interesarle probar distintos parámetros para así tratar de obtener mejores resultados.
- Ampliar el número de algoritmos disponibles. Actualmente sólo se permite usar Bayes y SVM, pero se quieren añadir otros algoritmos de Deep Learning.
- Añadir más funciones de visualización. En este campo la visualización de datos es algo muy importante, por ello se añadirán más funciones de datos para que el usuario pueda contar con más información.
- Instalar en un servidor para que pueda comenzar a usarse y recibir feedback de usuarios potenciales para mejorar la aplicación.

Bibliografía

[1] Comparación mvc y patrón por capas. https://www.academia.edu/9578780/CONCEPTO_DE_ARQUITECTURA_EN_3_CAPAS_MVC. Last access on 2-6-2019.

[2] Comparación sql-sqlite - stackoverflow. https://stackoverflow.com/questions/4539542/sqlite-vs-sql-server. Last access on 6-6-2019.

[3] Curva recall-precicion - sklearn. https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics. Last access on 4-6-2019.

[4] Documentación oficial de pymongo. https://api.mongodb.com/python/current/. Last access on 4-6-2019.

[5] Documentación oficial de python sobre las clases abstractas. https://docs.python.org/3/library/abc.html. Last access on 4-6-2019.

[6] Herramienta utilizada para la generación de diagramas. draw.io. Last access on 4-6-2019.

[7] Información del patrón mvc. https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html. Last access on 2-6-2019.

[8] Multinomialnb, una de las variantes de bayes para clasificación de texto. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB. Last acces on 4-6-2019.

[9] Métodos de naive bayes por sklearn. https://scikit-learn.org/stable/modules/naive_bayes.html. Last access on 4-6-2019.

[10] Métricas de sklearn. url. Last access on 4-6-2019.

[11] Nltk, sitio oficial. https://www.nltk.org/. Last access on 16-6-2019.

[12] Patron por capas y sus niveles. https://www.ecured.cu/Arquitectura_de_tres_niveles#Arquitectura_por_capas. Last access on 2-6-2019.

[13] Pruebas de caja negra. https://es.wikipedia.org/wiki/Caja_negra_(sistemas). Last access on 11-6-2019.

[14] Python, sitio oficial. https://www.python.org/. Last access on 16-6-2019.

[15] Scikit-learn, sitio oficial. https://scikit-learn.org/stable/. Last access on 16-6-2019.

[16] Snowabll stemmer - nltk. http://www.nltk.org/howto/stem.html. Last access on 12-6-2019.

[17] Support vector machines (svm) - sklearn. https://scikit-learn.org/stable/modules/svm.html. Last access on 4-6-2019.

[18] Svc, implementación de svm por sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Last access on 4-6-2019.

[19] Web oficial sqlite. https://www.sqlite.org/whentouse.html. Last access on 6-6-2019.

[20] Wordcloud - documentación oficial. https://amueller.github.io/word_cloud/. Last access on 4-6-2019.

[21] ¿qué son las bases de datos no-sql? https://aws.amazon.com/es/nosql/. Last access on 2-6-2019.

[22] Nltk - treebanktokenizer source code. https://www.nltk.org/_modules/nltk/tokenize/treebank.html, 2001. Last access on 2019.

[23] Web oficial de django. https://www.djangoproject.com/, 2005. Last access on 2-6-2019.

[24] Web oficial de flask. http://flask.pocoo.org/docs/1.0/foreword/#what-does-micro-mean, 2010. Last access on 2-6-2019.

[25] Patrón mvc resumido con lego. https://realpython.com/the-model-view-controller-mvc-paradigm-summarized-with-legos/, 2012. Last access on 2-6-2019.

[26] Comparación websocket y rest para api en tiempo real. https://stackoverflow.com/questions/28613399/websocket-vs-rest-api-for-real-time-data/28618369#28618369, Febrero 2015. Last access on 2-6-2019.

[27] Comparación entre rest y soap. https://stackify.com/soap-vs-rest/, Marzo 2017. Last access on 2-6-2019.

[28] Comparativa soap y rest. https://www.oscarblancarteblog.com/2017/03/06/soap-vs-rest-2/, Marzo 2017. Last acces on 2-6-2019.

[29] Diferencias en el envío de paquetes entre rest y websocket. https://stackoverflow.com/questions/45460734/websocket-vs-rest-when-sending-data-to-server, Agosto 2017. Last access on 2-6-2019.

[30] Diferencias entre soap y rest. https://stackoverflow.com/questions/19884295/soap-vs-rest-differences, Octubre 2017. Last accesed on 2-6-2019.

[31] Estructura de una aplicación flask. https://flask-restful.readthedocs.io/en/0.3.5/intermediate-usage.html, 2017. Last access on 2-6-2019.

[32] Flask vs django. por qué flask es mejor. https://www.codementor.io/garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v, Febrero 2017. Last access on 2-6-2019.

[33] Uso de flask y de django. https://stackshare.io/stackups/django-rest-framework-vs-flask, Octubre 2017. Last access on 2-6-2019.

[34] Amplia comparativa entre flask y django. https://www.educba.com/django-vs-flask/, Abril 2019. Last access on 2-6-2019.

[35] Amplia comparativa entre websocket y rest. www.educba.com/websocket-vs-rest/, Marzo 2019. Last acess on 2-6-2019.

[36] Fuentes ofertadas por google para la web. https://fonts.google.com/, Junio 2019. Last access on 3-6-2017.

[37] Guía para crear un diagrama de clases para restful api. https://www.visual-paradigm.com/support/documents/vpuserguide/276/3420/85154_modelingrest.html, Abril 2019. Last access on 3-6-2019.

[38] Guía para modelar una base de datos nosql. https://mapr.com/blog/data-modeling-guidelines-nosql-json-document-databases/, Mayo 2019. Last access on 3-6-2019.

[39] Web oficial de mongodb. https://www.mongodb.com/es, Mayo 2019. Last access on 3-6-2019.

[40] B Boehm. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes*, 11(4):14–24, August 1986.

[41] Jaime G. Carbonell, editor. *Subjetive Understanding: Computer Models of Belief Systems*. Tesis Doctoral, University of Yale, 1979.

[42] Eugenio Martínez Cámara. *Sentiment Analyst in Spanish*. Tésis doctoral, Universidad de Jaén, Octubre 2015. Last access on 4-6-2019.

[43] Robert Dale, H. L. Somers, and Hermann Moisl, editors. *Handbook of Natural Language Processing*. Marcel Dekker, Inc., New York, NY, USA, 2000.

[44] Martin Fowler. *UML Distilled*. 1997.

[45] Marti A. Hearst. Text-based intelligent systems. chapter Direction-based Text Interpretation As an Information Access Refinement, pages 257–274. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1992.

[46] Indeed. Sueldo medio de un desarrollador en españa. `https://www.indeed.es/salaries/desarrollador/a-junior-Salaries`, Mayo 2019. Last access on 2-6-2019.

[47] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.

[48] PCComponentes. Coste del portatil en la página de pccomponentes. `https://www.pccomponentes.com/lenovo-legion-y520-15ikbn-intel-core-i7-7700hq-8gb-1tb-gtx-1050-156`, Junio 2019. Last acces on 28-5-2019.

[49] Yorick Wilks and Janusz Bien. Beliefs, points of view, and multiple environments. *Cognitive Science*, 7(2):95 – 119, 1983.