

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Sistema Deep Learning para el análisis de sentimientos en opiniones de productos para la ordenación de resultados de un buscador semántico

Estudiante: Adrián Insua Yañez
Dirección: Carlos Gómez Rodríguez
Dirección: Sonia González Vázquez

A Coruña, outubro de 2019.

A mi familia y amigos por su apoyo incondicional

Agradecimientos

Debo dar las gracias al Instituto Tecnológico de Galicia por ofrecerme la oportunidad de realizar este proyecto con ellos y en especial a su gran equipo de profesionales que me ha ayudado y guiado durante todo este tiempo.

Resumen

EN esta investigación abordaremos el problema conocido como Análisis de Sentimientos, enmarcado dentro del área de estudio del Procesamiento de Lenguaje Natural, también llamado NLP por sus siglas en inglés (Natural Language Processing). La tarea consiste en identificar y extraer la polaridad de un conjunto de textos que expresan opiniones de personas con el objetivo de clasificarlos.

El desarrollo de este tipo de tareas de clasificación ha adquirido gran relevancia en los últimos tiempos dada su potencial aplicación al mundo empresarial y al crecimiento exponencial de los conjuntos de datos disponibles para realizar la investigación, gracias al uso cada vez más común de las redes sociales.

En cuanto a la importancia en el ámbito empresarial, a cualquier entidad que tenga un producto en el mercado le puede resultar interesante conocer la opinión que tienen sus clientes sobre la calidad de sus productos automáticamente, pudiendo saber tanto una opinión del público general, como realizando estudios sobre zonas geográficas determinadas. De igual forma este tipo de clasificaciones se pueden aplicar sobre otros ámbitos como por ejemplo el político, analizando el nivel de descontento de la población según la polaridad resultante de un conjunto de tweets filtrados por un hashtag (etiqueta señalada con #) determinado, sin necesidad de analizar manualmente el gran volumen de datos que puede suponer este tipo de estudios.

En este trabajo se pretende investigar distintas técnicas de aprendizaje automático aplicadas al campo NLP para obtener un análisis fiable de la información subjetiva de un conjunto de textos. En este sentido abordaremos el estudio de algoritmos de aprendizaje automático clásico (Machine Learning), que servirá para establecer una línea base sobre la que intentar mejorar los resultados, para posteriormente implementar algoritmos más modernos de aprendizaje profundo (Deep Learning), con la intención de que este tipo de sistemas sean capaces de aprender a discernir la estructura de las oraciones y gracias ello mejorar los resultados de clasificación obtenidos.

Dado que la investigación se presenta dentro de un marco profesional, se ha orientado al dominio del problema específico. En este caso se trata de un sistema que ha de clasificar un conjunto de opiniones en español sobre materiales de construcción, para posteriormente utilizar estas polaridades en un sistema de ranking que se aplicará a los resultados de un buscador semántico utilizado por usuarios expertos en el sector. Es importante que el usuario acceda rápidamente a los mejores productos de forma que mejoremos su experiencia y consigamos una mayor fidelización.

Para la validación del clasificador se ha desarrollado un sistema que además de clasificar los textos según las polaridades encontradas, devuelve un conjunto de métricas que explicaremos más adelante, y que nos permitirá comparar el funcionamiento de los distintos algoritmos.

De igual forma, aunque de manera secundaria, el trabajo abordará la implementación de la aplicación tanto en lo tocante al servidor como a la parte web. Esta aplicación tendrá una sección de comentarios en el perfil del producto para que el usuario pueda dar su opinión sobre el mismo, en esta sección se le permitirá además establecer una puntuación utilizando un sistema de “estrellas” típico que clasificará el texto en un rango de 1 a 5 siendo 1 muy negativo y 5 muy positivo.

Abstract

In this investigation we will address the problem known as Sentiment Analysis, framed within the area of study of Natural Language Processing or NLP. Our task is to identify and extract the polarity over a set of texts that express opinions from people with the objective of classifying them.

The development of this kind of classification tasks has acquired great relevance in recent times given its potential application in business and the exponential growth of available data sets for research, thanks to the increasingly common use of social networks.

As for the relevance in business area, any entity that has a product in the market may find interesting to know the opinion that their customers have about the quality of their products automatically, being able to know both an opinion of the general public, and making studies on specific geographical areas. In the same way, this type of classifications can be applied to other areas such as the political one, analyzing the level of discontent of the population according to the polarity resulting from a set of tweets filtered by a given hashtag (label marked with #), without needing a manual analysis of the large volume of data that this type of study may entail.

This work aims to investigate different machine learning techniques applied to the NLP field to obtain a reliable analysis of the subjective information of a set of texts. In this sense we will approach the study of classic machine learning algorithms, which will serve to establish a baseline on which we will try to improve the results implementing more modern deep learning algorithms by discerning the sentences structure.

Since the research is presented within a professional context, it has focused on mastering the specific problem. In this case it is a system that has to classify a set of opinions in Spanish on construction materials, to later use these polarities in a ranking system that will be applied to the results of a semantic search engine used by sector experts. It is important for the user

experience to get the best products quickly, so we can achieve greater loyalty.

For the validation of the classifier a system has been developed that in addition to classifying the texts according to the polarities found, returns a set of metrics that we will explain later, and that will allow us to compare the different algorithms.

Similarly, although in a secondary way, the work will address the implementation of the application both in terms of the server and the web part. This application will have a comments section in the product profile so that the user can give his opinion on it, in this section he will also be allowed to establish a score using a typical “ stars ” system that will classify the text in a range from 1 to 5 where 1 means very negative and 5 very positive.

Palabras clave:

- Procesamiento de lenguaje natural
- PLN
- Análisis de sentimientos
- Minería de opiniones
- Aprendizaje automático
- Aprendizaje profundo
- Análisis profundo

Keywords:

- Natural language Processing
- NLP
- Sentiment analysis
- Opinion minning
- Machine Learning
- Deep Learning
- Subjective analysis

Índice general

1	Introducción	1
1.1	Motivación	1
1.2	Objetivos	2
1.2.1	Análisis de sentimientos	2
1.2.2	Servicio web	3
2	Estructura de la memoria	5
3	Estado del arte	7
3.1	Clasificación	7
3.2	Conclusión	9
4	Fundamentos Tecnológicos	11
4.1	Análisis de sentimientos	11
4.2	Aplicación Web	12
4.2.1	Tecnología Back-end	12
4.2.2	Tecnología de datos	13
4.2.3	Front-end: Interfaz Single Page Application	13
4.2.4	Tecnologías de despliegue y construcción	15
5	Metodología de desarrollo	17
5.1	Proceso de desarrollo unificado	17
5.1.1	Características	17
5.1.2	Fases	18
6	Planificación	21
6.1	Análisis de sentimientos	21
6.1.1	Fase 1: Inicio	21
6.1.2	Fase 2: Elaboración	21

6.1.3	Fase 3: Construcción	21
6.1.4	Fase 4: Fase de transición	22
6.2	Servicio web	22
6.2.1	Fase 1: Fase de inicio	22
6.2.2	Fase 2: Fase de elaboración	22
6.2.3	Fase 3: Fase de construcción	22
6.2.4	Fase de transición	23
6.3	Diagrama de Gantt	23
6.4	Estimación de recursos	24
6.5	Estimación de costes	24
7	Casos de uso	27
7.1	Inicio de sesión	28
7.2	Visualizar comentarios	28
7.3	Añadir comentarios	28
7.4	Editar comentarios	29
7.5	Eliminar comentarios	30
7.6	Clasificar comentarios	30
8	Análisis	33
8.1	Arquitectura general	33
8.2	Arquitectura sistema de análisis de sentimientos	34
8.3	Arquitectura Web	35
8.4	Trabajo futuro	36
9	Base de datos	37
9.1	Diseño lógico: Modelo Entidad-Relación	37
9.2	Diseño físico: Modelo relacional	37
10	Diseño	39
10.1	Análisis de sentimientos	39
10.1.1	Diagrama de clases	40
10.2	Servicio web	40
10.2.1	Diagrama de clases	41
11	Implementación	43
11.1	Análisis de sentimientos	43
11.1.1	Tratamiento de los datos	43
11.1.2	Algoritmos	47

11.2	Servicio web	57
11.2.1	Visualizar comentarios	57
11.2.2	Añadir comentarios	57
11.2.3	Editar comentarios	57
11.2.4	Borrar comentarios	57
12	Resultados	59
12.1	Corpus de datos	59
12.1.1	Corpus TASS	59
12.1.2	Corpus Cine	61
12.2	Machine learning	62
12.2.1	Línea base	63
12.2.2	Experimentos de mejora Machine Learning	64
12.3	Deep Learning	67
12.3.1	Selección de parámetros	67
12.3.2	2 clases	71
12.3.3	3 clases	72
12.3.4	5 clases	74
12.3.5	Conclusiones	76
13	Solución desarrollada	77
13.1	Sección de comentarios	77
13.2	Publicación de comentarios	78
13.3	Edición y eliminación de comentarios	78
A	Material adicional	81
A.1	Análisis de resultados	81
A.1.1	Línea base - 5 clases	81
A.2	Resultados GridSearch en Machine Learning	82
A.2.1	2 clases	82
A.2.2	3 clases	83
A.2.3	5 clases	83
A.3	Trabajo futuro diseño web	84
A.3.1	Página principal	85
A.3.2	Ficha de producto	86
	Lista de acrónimos	87
	Glosario	89

Bibliografía

91

Índice de figuras

1.1	Evolución del uso de las redes sociales	1
3.1	Publicaciones sobre Análisis de sentimientos en el IEEE desde 2004 hasta la actualidad	7
3.2	Publicaciones en el IEEE desde 2010 que están relacionadas al término de búsqueda: sentiment analysis deep learning.	8
6.1	Diagrama de Gantt del proyecto (BCS).	23
7.1	Casos de uso	27
8.1	Arquitectura general del sistema	33
8.2	Arquitectura del sistema de análisis de sentimientos	34
8.3	Arquitectura del clasificador	34
8.4	Arquitectura del servicio y frontal web.	35
8.5	Ejemplo de arquitectura de un módulo REST del servicio.	35
9.1	Diagrama entidad relación para el subsistema de comentarios.	38
9.2	Modelo relacional para el subsistema de comentarios.	38
10.1	Diagrama de clases del sistema de análisis de sentimientos	40
10.2	Diagrama de clases de las entidades del modelo SQL	41
10.3	Diagrama de clases de comentario.	41
11.1	Representación de un hiper plano en una SVM	50
11.2	Representación de una red neuronal	52
11.3	Representación de una red neuronal recurrente	53
11.4	Representación de una LSTM	54
11.5	Representación de una red CNN	55
11.6	Representación de una red CNN	56

12.1	Distribución de la polarización.	60
12.2	Histograma de número de palabras por documento.	60
12.3	Distribución de las palabras polarizadas en tweets positivos	60
12.4	Distribución de las palabras polarizadas en tweets negativos	61
12.5	Distribución de las palabras polarizadas en tweets neutros	61
12.6	Histograma de número de palabras por documento.	62
12.7	Distribución de la polarización.	63
12.8	Función de activación RELU	68
12.9	Evolución entrenamiento modelo LSTM	68
12.10	Evolución entrenamiento modelo LSTM simplificado	69
12.11	Dropout con valor 0.2	70
12.12	Inicialización Glorot más Dropout	70
12.13	Inicialización Glorot, dropout y normalización de batch	71
12.14	Evolución modelo LSTM con normalización de batch.	73
12.15	Evolución modelo LSTM con normalización de batch e inicialización Glorot.	73
12.16	Evolución modelo convolucional.	74
12.17	Evolución modelo convolucional.	75
12.18	Evolución modelo LSTM con normalización de batch e inicialización Glorot.	75
12.19	Evolución modelo LSTM con normalización de batch.	76
13.1	Sección de comentarios en la ficha de articulo.	77
13.2	Publicación de comentarios.	78
13.3	Edición y eliminación de comentarios.	78
A.1	Distribución de polaridades en los conjuntos de entrenamiento y test.	81
A.2	Matriz de confusión de la clasificación en 5 clases.	82
A.3	Página principal.	85
A.4	Ficha de producto.	86

Índice de cuadros

6.1	Estimación de esfuerzo subsistema de análisis de sentimientos.	24
6.2	Estimación de esfuerzo desarrollo aplicación web.	25
12.1	Resultados en % entrenamiento para linea base.	63
12.2	Resultados en % entrenamiento para linea base de 3 clases.	64
12.3	Resultados en % entrenamiento para linea base de 5 clases.	64
12.4	Medias en % entrenamiento para gridsearch con 2 clases.	65
12.5	Medias en % entrenamiento para gridsearch con 3 clases.	66
12.6	Medias en % entrenamiento para gridsearch con 5 clases.	66
12.7	Resultados sobre el conjunto de test.	71
12.8	Resultados sobre el conjunto de test.	72
12.9	Resultados sobre el conjunto de test.	74
A.1	Resultados en % del mejor modelo obtenido en los k-folds	82
A.2	Resultados de la clasificación en el dominio de críticas de cine.	83
A.3	Resultados en % del mejor modelo obtenido en los k-folds	83
A.4	Resultados de la clasificación en el dominio de críticas de cine.	83
A.5	Resultados en % del mejor modelo obtenido en los k-folds	83
A.6	Resultados de la clasificación en el dominio de críticas de cine.	84

Introducción

1.1 Motivación

ACTUALMENTE podemos ver durante nuestro día a día como nuestra vida va cada vez más unida al uso de las redes sociales (Figura 1.1): subimos nuestras fotos, comentamos las de otros, escribimos críticas de artículos que hemos adquirido, escribimos nuestros pensamientos sobre situaciones de actualidad, etc. Aprovechando esta situación vemos que para el mundo empresarial resulta muy interesante el conocer la opinión que los usuarios tienen sobre los productos ofrecidos, o sobre la entidad misma.

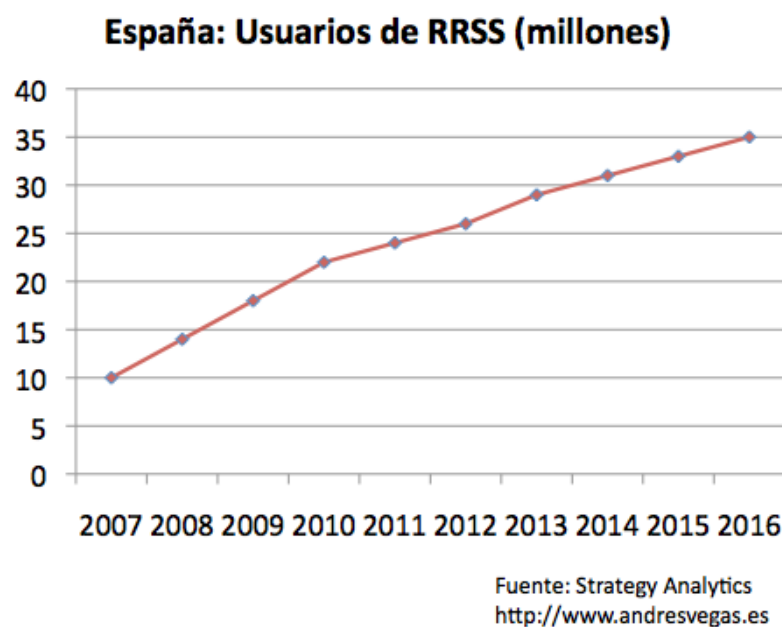


Figura 1.1: Evolución del uso de las redes sociales

Este auge de las redes sociales no solo despierta la curiosidad de los empresarios, sino que

también, y de forma muy ligada al interés económico inherente, nos permite disponer de un corpus muy extenso de textos que nos servirán como punto de partida para la construcción de un sistema clasificador.

Además de esto, cabe destacar el gran avance tecnológico realizado en los últimos años, permitiéndonos tener ordenadores cada vez más potentes, con unas capacidades de procesamiento tanto de CPU como de GPU altísimas y gracias a ello realizar entrenamientos con topologías mucho más complejas en un rango de tiempo menor. Es por esto que una de las ramas que hemos investigado para procurar una mejora del resultado final es una aproximación a las redes neuronales, las cuales se entrenan aprovechando la potencia de la GPU.

Hemos visto en este tipo de sistemas una oportunidad de negocio, ofreciendo al cliente ampliar su plataforma implementando un sistema de ranking automático de la lista de resultados mediante un pesado de la polaridad de las opiniones que los usuarios realizan sobre cada uno de los productos. De esta forma los usuarios podrán agilizar sus búsquedas, encontrando en los primeros lugares los elementos más relevantes, y el cliente podría realizar estudios sobre la aceptación de los mismos en el mercado.

1.2 Objetivos

El objetivo de esta investigación es el de analizar y comparar distintos modelos de clasificación de polaridades de textos utilizando procesamiento de lenguaje natural, para posteriormente utilizar estos modelos en la clasificación de comentarios realizados en una web de materiales de construcción.

Además se implementarán los módulos necesarios en una aplicación web para facilitar al usuario la publicación de estos comentarios.

1.2.1 Análisis de sentimientos

La tarea consiste en buscar una serie de características en el texto, en este caso comentarios, presumiblemente breves, sobre materiales de construcción, que nos ayude a realizar una clasificación de los mismos dividiéndolos en distintas clases.

Esta tarea es conocida como Análisis de Sentimiento (Das y Chen [1], Tong [2], Turney [3], Pang et al. [4]), Extracción de opiniones, Minería de Opiniones (introducido por Dave et al. [5]), Minería de sentimiento o Análisis Subjetivo.

El ámbito de clasificación de esta tarea suele ser el siguiente:

- Binario: El resultado se divide en dos clases bien diferenciadas (Opinión positiva y opinión negativa)

- Multiclase: El resultado se divide en 5 clases (Muy negativa, negativa, neutra, positiva, muy positiva)
- Tri-clase: El resultado se divide en 3 clases (Negativa, neutra y positiva)

Es de esperar que a mayor número de clases el problema se haga más complicado. Dado el dominio textual de la tarea es más difícil encontrar las diferencias entre un texto muy negativo y uno negativo que entre una opinión negativa y otra positiva.

Esta tarea suele resultar difícil incluso para humanos, ya que la valoración está fuertemente influenciada por valores subjetivos de cada observador. Por ejemplo la frase *No me ha gustado la calidad del producto* podría ser considerada negativa por un observador imparcial y muy negativa por la persona responsable de la calidad de dicho producto.

El punto principal de la tarea es la extracción de características clave del texto que ayuden a separar y clasificar los textos en clases. En este área encontramos 2 aproximaciones principales:

- Bag of words (BOW): Es el modelo más simple, se compone un vector de características en el que cada elemento se corresponde a una palabra del dominio, y su valor será o bien el número de ocurrencias en el texto o bien un valor estadístico que refleje la relación de la palabra con su contexto, como por ejemplo el valor de tf-idf.
- Word Embeddings (WE): Es un vector compuesto mediante un entrenamiento en una red neuronal que recibe como entrada un conjunto grande de textos e intenta aprender las relaciones de similitud entre las palabras que lo componen, dando como resultado para cada palabra un vector de dígitos reales de una longitud determinada. Gracias a este tipo de datos es posible buscar relaciones de similitud, o incluso operaciones de adición y sustracción entre los vectores que representan las palabras, dando como resultado otra palabra:

$$\text{Vector(Rey)} - \text{Vector(Hombre)} + \text{Vector(Mujer)} = \text{Vector(Reina)}$$

1.2.2 Servicio web

Pretendemos poner a disposición del usuario una web para realizar una búsqueda semántica de los productos, que además les ofrecerá un servicio de comentarios con las siguientes funcionalidades:

- Añadir comentarios: Los usuarios podrán añadir nuevos comentarios tanto si están registrados como si son usuarios anónimos.
- Editar comentarios: Los usuarios registrados podrán editar sus comentarios.

- Borrar comentarios: Los usuarios registrados podrán eliminar sus comentarios.

Además el objetivo final de este servicio es clasificar los comentarios de manera que este resultado se pueda utilizar en un sistema de ranking de los resultados ofrecidos por el buscador semántico.

Estructura de la memoria

LA memoria se estructurará en varios capítulos detallados a continuación:

- **Introducción:** En este capítulo explicaremos la motivación y los objetivos del estudio, además de presentar la estructura de la memoria.
- **Estado del Arte:** Se definirán términos relativos al dominio del problema y se realizará un breve análisis de las investigaciones previas sobre el análisis de sentimientos.
- **Fundamentos tecnológicos:** Capítulo en el que se describirán las tecnologías y herramientas utilizadas para la implementación del proyecto.
- **Metodología de desarrollo:** Se explicará la metodología utilizada.
- **Planificación:** Se detallará la planificación llevada a cabo para afrontar el proyecto.
- **Análisis:** Definiremos las arquitecturas seleccionadas para el proyecto, comenzando por una definición de la arquitectura general y posteriormente detallando las arquitecturas propias de cada subsección del proyecto.
- **Casos de uso:** Apartado en el que se detallarán los casos de uso a abordar en la implementación del proyecto.
- **Implementación:** En este capítulo explicaremos la puesta en marcha del proyecto, explicando los detalles de la implementación tanto para el problema de clasificación como para el desarrollo del servicio web.
- **Resultados:** Análisis de los resultados obtenidos en los distintos experimentos.
- **Conclusiones:** Se tratará de llegar a una serie de conclusiones a partir de los resultados presentados previamente.
- **Solución desarrollada:** Apartado en el que se mostrará el resultado final del desarrollo.

Estado del arte

SON muchos los estudios que se han publicado acerca del tema de la minería de opiniones. En esta investigación nos centraremos las técnicas más comunes de aprendizaje supervisado analizando distintos métodos tanto de clasificación como de preprocesamiento del texto.

En este capítulo describiremos el estado actual de los modelos de clasificación.

3.1 Clasificación

En términos de clasificación de sentimientos son muchos los trabajos publicados en los últimos tiempos (ver fig. 3.1) debido a, como ya hemos comentado, el auge de las redes sociales y de la necesidad de las empresas por realizar un estudio de mercado y de satisfacción de sus clientes.

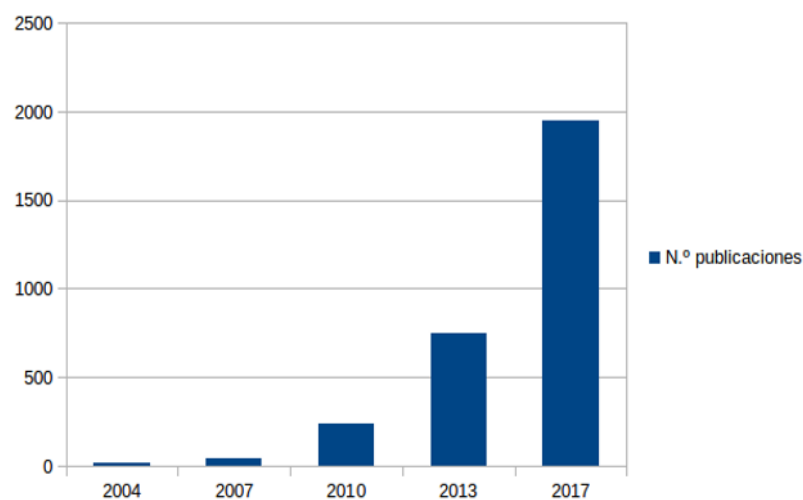


Figura 3.1: Publicaciones sobre Análisis de sentimientos en el IEEE desde 2004 hasta la actualidad

Dada la amplia extensión de métodos utilizados¹ para resolver este problema nosotros nos centraremos en las dos ramas utilizadas para el trabajo, utilizando un lexicón termino-sentimiento [6] y mediante algoritmos de aprendizaje automático, basándose la mayoría de estos estudios en los trabajos de Pang et al. [4] intentando diseñar métodos más efectivos de extracción de características para mejorar la clasificación. En este último campo podríamos realizar una subdivisión atendiendo al tipo de algoritmo utilizado:

- Machine Learning: Algoritmos tradicionales de aprendizaje supervisado, en los que se pretende obtener una matriz de características que servirán para extraer patrones que dividan los textos en distintas clases.
- Deep Learning: Se trata de algoritmos más novedosos, que están obteniendo alta relevancia en los últimos años (fig. 3.2) gracias a la mejora de la capacidad de procesamiento de las GPU actuales. Estos modelos nos permiten obtener unos resultados mejores que los modelos tradicionales y además se ejecutan de forma más rápido gracias al alto rendimiento de las GPUs.

Además mediante esta rama de clasificación el vector de características se compone de Word Embeddings (WE) los cuales son mucho más ricos en información relacionada con la palabra que las features utilizadas en algoritmos tradicionales, ya que capturan información sobre similitud y características semánticas de las palabras.

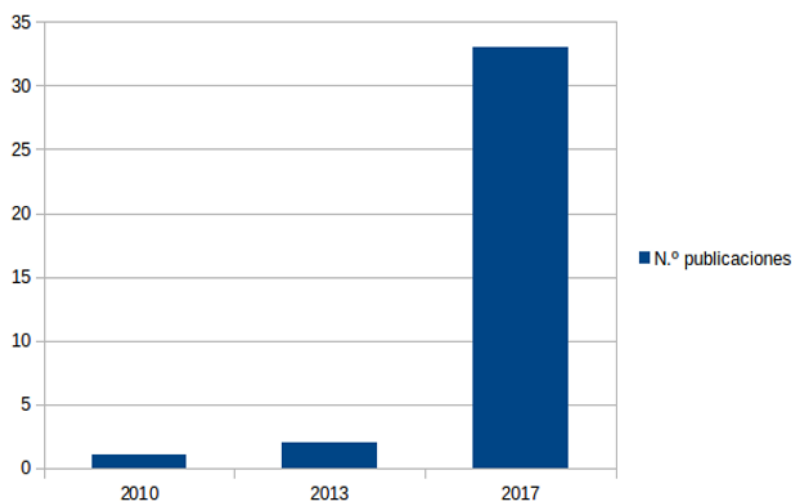


Figura 3.2: Publicaciones en el IEEE desde 2010 que están relacionadas al término de búsqueda: sentiment analysis deep learning.

Destacamos en esta sección los estudios realizados en la Universidad de Stanford [7], obteniendo una precisión del 85.4% para predicción binaria a nivel de oración.

¹<http://www.sciencedirect.com/science/article/pii/S2090447914000550>

3.2 Conclusión

Vistos los estudios anteriormente citados, vamos a realizar una implementación utilizando algoritmos típicos de Machine Learning que nos sirva como línea base, para posteriormente procurar mejorarla, tanto mejorando los procesos de tratamiento del texto como implementando modelos de Deep Learning.

Fundamentos Tecnológicos

EN este capítulo trataremos las distintas tecnologías utilizadas en el desarrollo de este proyecto.

4.1 Análisis de sentimientos

Para la preparación y estudio de los datos, así como para el desarrollo de los modelos de aprendizaje se ha elegido la versión 3.5 de Python, por su facilidad de aprendizaje y su amplia comunidad en el ámbito del aprendizaje automático.

Se trata de un lenguaje de programación multiparadigma creado por Guido van Rossum¹ y administrado por la Python Software Foundation.

Para el proyecto se han utilizado las siguientes librerías:

- NLTK: Biblioteca para trabajar en dominios de lenguaje natural.
- SciKitLearn: Ofrece herramientas eficientes para el análisis y la minería de datos así como algoritmos de Machine Learning.
- TensorFlow: Biblioteca de código abierto para la implementación de modelos de Aprendizaje Profundo. Permite el uso tanto de CPU como de GPU durante los procesos. Se ha elegido esta frente a otras como Theano o Torch por ser una biblioteca de Google que sigue mantenida en la actualidad, permite su uso en contenedores Docker de forma que la instalación para el uso de GPU se simplifica y no afecta al sistema principal de la máquina. Además disfruta de una amplia comunidad de desarrolladores lo cual facilitará la resolución de dudas durante el proceso.
- Keras: Biblioteca de alto nivel que permite agilizar la implementación de modelos de Aprendizaje Profundo. Está incluida de forma nativa en las últimas versiones de Ten-

¹<https://docs.python.org/3/faq/general.html#why-was-python-created-in-the-first-place>

sorflow pero también se puede utilizar de forma individual con otros backends como Theano.

4.2 Aplicación Web

4.2.1 Tecnología Back-end

A continuación describiremos la tecnología utilizada del lado del servidor.

Java

El lenguaje de programación Java fue creado en 1991 por Sun Microsystems (adquirida posteriormente por Oracle) y publicado en 1995 como parte fundamental de la plataforma Java, que ofrece un lenguaje de programación de propósito general, concurrente y orientado a objetos.

La tecnología Java ofrece una máquina virtual (JVM) que permite ejecutar la compilación de los bytecodes (clases de Java) en cualquier máquina, sin importar la arquitectura subyacente.

Para la ejecución como usuario final de aplicaciones Java es necesario disponer el Java Runtime Environment (JRE), mientras que para el uso como desarrollador es necesario el uso del kit JDK (java Development Kit) que incluye el JRE.

Sun define tres plataformas según el entorno de aplicación:

- Java ME: orientado a entornos de recursos limitados, como teléfonos móviles, PDAs, etc.
- Java SE: para entornos de gama media y estaciones de trabajo.
- Java EE: orientada a entornos distribuidos empresariales o de internet.

En el proyecto se utiliza la versión 8 del JDK con la plataforma Java EE.

Spring

Uno de los problemas a los que nos enfrentamos al enfrentarnos a un desarrollo software es el uso de varios frameworks (conjunto de clases que pretenden facilitarnos el trabajo), y cada uno de estos generará su propio conjunto de objetos. Esta situación puede generarnos problemas ya que los frameworks son independientes entre sí y gestionan ciclos de vida propios para los objetos.

En este sentido Spring nos ayuda cambiando las responsabilidades y encargándose él en lugar del desarrollador de generar los objetos de cada uno de los frameworks basándose en ficheros xml o anotaciones, y de integrarlos de forma correcta.

Jackson

Se trata de una biblioteca java simple pero potente pensada para serializar objetos Java a JSON y viceversa.

4.2.2 Tecnología de datos

A continuación se describen las tecnologías de gestión de datos utilizados por el sistema.

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientado a objetos y libre, publicado bajo la licencia PostgreSQL similar a la BSD o la MIT.

Hibernate

Hibernate es una herramienta de mapeado objeto-relacional (ORM) para uso sobre la plataforma Java o .Net bajo el nombre NHibernate.

Esta herramienta utiliza archivos XML o anotaciones en los beans de las entidades para facilitar el mapeado de atributos entre una base de datos relacional y el modelo de objetos de una aplicación.

4.2.3 Front-end: Interfaz Single Page Application

Una Single Page Application (SPA) es una aplicación o interfaz web de página única con el propósito de ofrecer una experiencia más fluida al usuario. Los códigos utilizados por la aplicación pueden cargarse todos de una sola vez o irse cargando de forma dinámica dependiendo de las necesidades de la aplicación.

Las herramientas modernas como AngularJS (que se explicará a continuación) permiten al desarrollador crear una SPA sin necesidad de enfrentarse al código JavaScript ni a los problemas de la tecnología.

HTML5

HTML5 es la quinta revisión importante del lenguaje HTML y especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/html), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML5 que deberá servirse con sintaxis XML (application/xhtml+xml).

La versión definitiva de la quinta revisión del estándar se publicó en octubre de 2014.

JavaScript

Es un lenguaje de programación interpretado utilizado sobre todo del lado del cliente. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

CSS3

CSS3 es la última versión disponible de CSS (Cascading Stylesheets u Hojas de estilo en cascada en Español). Cuando hablamos de CSS hablamos de un lenguaje de diseño gráfico que nos ayuda a suplir las carencias de HTML a la hora de maquetar el diseño de una página web.

AngularJS

AngularJS es un framework de javascript mantenido por Google que realiza una extensión del lenguaje HTML tradicional y nos brinda la posibilidad de diseñar páginas web dinámicas, facilitando la creación de SPA's.

Angular sigue el patrón MVVM (Model View View Model) alentando la articulación flexible entre la presentación, datos y componentes lógicos.

Se ha utilizado esta tecnología por ser la utilizada en el sistema ya existente, aunque sería recomendable hacer una migración a las nuevas versiones del framework.

AngularMaterial

Proyecto pensado para los desarrolladores de AngularJS que ofrece una serie de componentes prediseñados de interfaz siguiendo la especificación Google Material Design.

Las directivas y objetos ofrecidos por esta biblioteca están correctamente probados, y pensados para funcionar en diferentes dispositivos y a diferentes tamaños de pantalla (Responsive Web)

JSPM

JSPM es un gestor de paquetes asociado a SystemJS que funciona sobre cualquier registro como puede ser npm o GitHub y funciona sobre el sistema de carga de módulos ES6, característica que nos permita el uso de cualquier módulo javascript de forma sencilla, solo se necesita una instrucción para la instalación y una línea de código para la importación al proyecto.

Esta tecnología es la que utiliza el sistema existente, sin embargo si se migra a alguna de las últimas versiones del framework deberemos utilizar el gestor NPM.

4.2.4 Tecnologías de despliegue y construcción

Servidor HTTP Apache

Se trata de un servidor web de código abierto multiplataforma, que comenzó su desarrollo en 1995, actualmente es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (<http://httpd.apache.org/>).

La estructura del servidor es modular, es decir, además de un core o núcleo presenta diversos módulos que aportan mucha funcionalidad que podría considerarse básica para un servidor web.

Además el servidor es fácilmente extensible incluyendo nuevos módulos además de los que trae por defecto.

Apache Tomcat

Tomcat es un contenedor web con soporte de servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Apache Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002.

Maven está construido alrededor de la idea de reutilización, y más específicamente, a la reutilización de la lógica de construcción. Como los proyectos generalmente se construyen en patrones similares, una elección lógica podría ser reutilizar los procesos de construcción.

Metodología de desarrollo

EN este capítulo describiremos la metodología elegida para la planificación y seguimiento del proyecto para posteriormente explicar sus características definitorias y las fases que lo componen, esto nos servirá como introducción para la explicación de la planificación realizada explicada en el capítulo 6.

5.1 Proceso de desarrollo unificado

El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational o simplemente RUP.

5.1.1 Características

A continuación explicaremos las características que definen esta metodología.

Iterativo e incremental

El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

Cada una de estas iteraciones se divide a su vez en una serie de disciplinas:

- **Análisis de requisitos:** En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria

llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

- **Diseño:** Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software)
- **Implementación:** Es la fase en donde se implementa el código fuente.
- **Prueba:** Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente y que cumple con los requisitos, antes de ser entregado al usuario final.

Dirigido por los casos de uso

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas.

Centrado en la arquitectura

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

Enfocado a los riesgos

El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

5.1.2 Fases

El PUD se puede dividir en cuatro fases que ayudaran tanto en la elaboración del software como a la resolución de problemas.

Inicio

En la fase de inicio se define el negocio: facilidad de realizar el proyecto, se presenta un modelo, visión, metas, deseos del usuario, plazos, costos y viabilidad.

Elaboración

En esta fase se obtiene la visión refinada del proyecto a realizar, la implementación iterativa del núcleo de la aplicación, la resolución de riesgos altos, nuevos requisitos y se ajustan las estimaciones.

Construcción

Esta abarca la evolución hasta convertirse en producto listo incluyendo requisitos mínimos. Aquí se afinan los detalles menores como los diferentes tipos de casos o los riesgos menores.

Transición

En esta fase final, el programa debe estar listo para ser probado, instalado y utilizado por el cliente sin ningún problema. Una vez finalizada esta fase, se debe comenzar a pensar en futuras novedades para la misma.

Capítulo 6

Planificación

A continuación se detallan las fases seguidas en el proyecto, subdividiendo la gestión en las dos ramas principales del mismo: análisis de sentimientos y servicio web.

6.1 Análisis de sentimientos

6.1.1 Fase 1: Inicio

Se trata de una fase de documentación y aprendizaje.

En esta fase se realizará un análisis del estado del arte del problema, además se estudiará cuales son las tecnologías más comunes para abordarlo, y se tratará de familiarizarse con el dominio de la tarea.

6.1.2 Fase 2: Elaboración

Se establecen una serie de requisitos a cumplir por el clasificador, y se establecen los casos de uso. Posteriormente se realiza un primer modelado de la estructura de clases, que será posible que cambie según se vayan presentando nuevas posibilidades de experimentos a realizar.

6.1.3 Fase 3: Construcción

Se comienza la implementación del código, esta fase constará de varias iteraciones que irán sumando funcionalidades al sistema.

Iteración 1 En esta iteración se comenzará por la implementación de un algoritmo de clasificación que sirva como línea base. Además será necesario implementar el algoritmo de tratamiento del corpus de documentos, será necesario pasar de formato XML a un formato que sea válido como entrada de los clasificadores, en este caso se ha elegido un tipo DataFrame de Python, que nos facilitará el tratamiento posterior de los textos.

Iteración 2 Se procederá a implementar la clase de preprocesamiento de los textos que nos servirá para probar métodos para aumentar la eficacia del clasificador.

Iteración 3 Se desarrollarán una serie de estudios sobre los datos y las posibilidades de mejora tanto del preprocesado como de los algoritmos, partiendo de los resultados de la línea base.

Iteración 4 Se creará una clase capaz de procesar los resultados, y ofrecer recursos visuales de valor que nos ayuden a comparar los resultados obtenidos.

6.1.4 Fase 4: Fase de transición

Con el programa listo se procederá a realizar las ejecuciones de los modelos y obtener unos datos que servirán para hacer una comparativa de los mismos. Como consecuencia de estos resultados se decidirá si es necesario corregir errores, buscar nuevos modelos o consideramos que el programa está ya listo para su entrega.

Es posible que tras finalizar esta fase veamos necesario volver a la fase anterior, ya sea por necesidad de corregir errores, o porque durante la investigación se ha descubierto alguna otra estrategia que se estime pueda funcionar mejor que la adoptada hasta ahora.

6.2 Servicio web

6.2.1 Fase 1: Fase de inicio

Dado que abordaremos la implementación de una sección de una página Web perteneciente a un proyecto ya existente, en esta primera fase trataremos de comprender el funcionamiento y la estructuración del código ya existente, posteriormente se hará un análisis de los requisitos del módulo.

6.2.2 Fase 2: Fase de elaboración

Tras haber comprendido el sistema existente y haber refinado los requisitos, procederemos a la extracción de los casos de uso, del modelo de base de datos y del modelado de clases.

6.2.3 Fase 3: Fase de construcción

Con toda la documentación establecida se procederá a la implementación de las funcionalidades del módulo siguiendo un ciclo de iteraciones.

Iteración 1 Se implementará en el servidor las operaciones CRUD¹ de la entidad Comentario, así como los endpoints que permitirán al cliente acceder a estos servicios.

Iteración 2 Se creará la vista principal del módulo de comentarios que permitirá al usuario acceder a una lista de los mismos desde el perfil de cada producto.

6.2.4 Fase de transición

Se realizarán las pruebas finales para verificar que el módulo funciona tal y como se establece en los requisitos y que todos los casos de uso están implementados de forma correcta.

6.3 Diagrama de Gantt

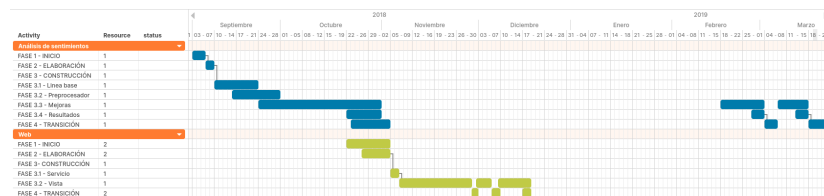


Figura 6.1: Diagrama de Gantt del proyecto (BCS).

El diagrama de Gantt realizado para el proyecto (fig. 6.1) representa los tiempos de desarrollo en un BCS (Best Case Scenario).

En esta casuística el proyecto podría durar hasta 4 meses de desarrollo como mínimo, aunque en el diagrama se incluye la posibilidad de encontrar algún fallo o método de clasificación más eficaz meses más tarde.

Como vemos las tareas no tienen por qué ser secuenciales, dado que si suponemos que la carga prevista de la tarea va a ser baja podemos paralelizarla con otra, y en muchos casos la propia naturaleza de la tarea nos obligará a ejecutarla de forma paralela, ya que, por ejemplo la construcción del preprocesador de textos se verá influida por los resultados obtenidos en la clasificación y viceversa.

En cuanto a la parte final del desarrollo web vemos que se intercalan la fase 4 y la fase 3.2, esto se debe a que esperamos encontrar fallos en las fases de prueba que necesitarán ser corregidos y probados de nuevo.

En el desarrollo real de este proyecto se ha entregado el PVM (Producto Viable Mínimo) dentro del tiempo especificado en la estimación. Sin embargo se han realizado varios ciclos de desarrollo posteriores para mejorar la implementación de ciertos algoritmos, mediante prueba

¹CRUD: Create Read Update Delete

de nuevas tecnologías, corrección de errores o afinación de hiperparámetros. Estos esfuerzos sumarán un total de 192 horas imputables a las tareas 3.3 y 3.4 del análisis de sentimientos.

6.4 Estimación de recursos

Para la realización de este proyecto se necesitarán al menos 2 personas con 2 roles diferenciados, un analista que se encargue de la recogida de requisitos y diseño inicial del sistema y un programador que lo implemente.

En caso de ser necesario podríamos recurrir a un tercer recurso para la realización de la investigación, que bien podría ser un ingeniero de datos o un especialista en el campo del aprendizaje automático.

6.5 Estimación de costes

Para realizar la estimación de costes vamos a tener en cuenta la participación de dos recursos, un analista en las fases iniciales del desarrollo web y un programador junior a lo largo de todo el proyecto. Los cálculos se realizarán sobre la estimación inicial, sin tener en cuenta el tiempo de desarrollo fuera del escenario BCS.

Los recursos tendrían una remuneración de:

- Analista: 17€/hora
- Programador: 12€/hora

A continuación se muestran las tablas con la estimación de esfuerzo de los recursos durante los tres meses y medio estimados como mejor escenario posible, separadas por subsistemas.

Tarea	Precio (€/hora)	Esfuerzo aprox. (horas)	Total (€)
FASE 1	12	24	288
FASE 2	12	16	192
FASE 3	12	384	3072
		424 h	3552 €

Cuadro 6.1: Estimación de esfuerzo subsistema de análisis de sentimientos.

Tarea	Precio (€/hora)	Esfuerzo aprox. (horas)	Total (€)
FASE 1	17	80	1360
FASE 2	–	56	–
FASE 3	12	256	3072
		336 h	4432 €

Cuadro 6.2: Estimación de esfuerzo desarrollo aplicación web.

Como vemos en la tabla 6.2 en las primeras fases solo se ha tenido en cuenta el trabajo realizado por el analista, ya que el programador tendrá que paralelizar esta etapa con la fase final del otro subsistema. Además dado que el analista paralelizará su trabajo en estas primeras fases la fase 2 de 56 horas no supondrá un costo a mayores.

Teniendo estos datos en cuenta podemos estimar un coste de 7.984 € para el proyecto. Si además sumamos el costo de las 192 horas asignadas al programador para un posible caso de arreglo o mejora futura obtenemos un total de 10.288€.

Casos de uso

DURANTE este capítulo se presentarán los diversos casos de uso que han de ser implementados en la aplicación.

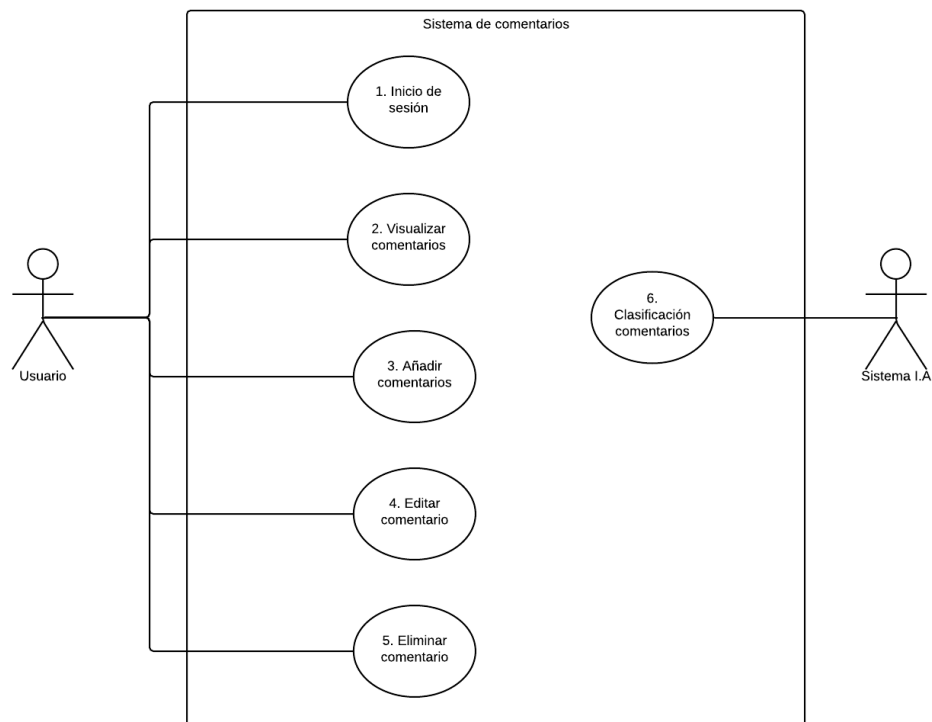


Figura 7.1: Casos de uso

7.1 Inicio de sesión

Descripción El usuario iniciará sesión en la aplicación introduciendo sus credenciales en un formulario.

Precondición N/A

Secuencia La secuencia de acciones será la siguiente:

- El usuario pulsa el botón de inicio de sesión.
- El usuario introducirá el usuario y contraseña en el formulario.
- Si las credenciales son correctas se notifica al usuario de que su sesión se ha iniciado y se cambia el botón de inicio de sesión por el botón de gestión de perfil.
- Si las credenciales no son correctas se notifica al usuario y se vuelve a pedir que las introduzca de nuevo.

Postcondición La sesión del usuario quedará iniciada en el sistema.

7.2 Visualizar comentarios

Descripción El usuario visualizará los comentarios de un artículo cuando abra la ficha del mismo haciendo click sobre el mismo en la pantalla principal.

Precondición N/A

Secuencia El caso de uso comenzará cuando el usuario haga click sobre un artículo determinado en la rejilla de resultados de búsqueda.

7.3 Añadir comentarios

Descripción El usuario podrá añadir comentarios relacionados con alguno de los artículos del sistema.

Precondición El usuario debe estar registrado.

Secuencia La secuencia de acciones será la siguiente:

- 1 El usuario hace click sobre alguno de los artículos de la rejilla de resultados de la página principal.
- 2 El usuario introduce el comentario que desee en el cuadro de texto.
- 3 El usuario puede publicar o cancelar el comentario.
 - 3.1 Si el usuario introduce una puntuación para el artículo esta se le asigna al comentario.
 - 3.2 Si el usuario no introduce una puntuación el artículo se guardará sin puntuación asociada, y se le asignará mediante un modelo de clasificación inteligente.
 - 3.3 Si el usuario cancela el envío del comentario se terminará el caso de uso.

Postcondición El comentario quedará dado de alta en el sistema.

7.4 Editar comentarios

Descripción El usuario podrá editar los comentarios realizados con anterioridad.

Precondición

- El usuario debe estar registrado.
- El comentario a editar debe pertenecer al usuario.

Secuencia La secuencia de acciones será la siguiente:

- 1 El usuario hace click sobre alguno de los artículos de la rejilla de resultados de la página principal.
- 2 El usuario elige la opción de edición sobre uno de sus comentarios.
- 3 El usuario puede aplicar o cancelar la edición.
 - 3.1 Si el usuario introduce una puntuación para el artículo esta se le asigna al comentario.
 - 3.2 Si el usuario no introduce una puntuación el artículo se guardará sin puntuación asociada, y se le asignará mediante un modelo de clasificación inteligente.
 - 3.3 Si el usuario cancela el envío del comentario se terminará el caso de uso.

Postcondición El comentario quedará modificado en el sistema.

7.5 Eliminar comentarios

Descripción El usuario podrá eliminar los comentarios realizados con anterioridad.

Precondición

- El usuario debe estar registrado.
- El comentario a editar debe pertenecer al usuario.

Secuencia La secuencia de acciones será la siguiente:

- 1 El usuario hace click sobre alguno de los artículos de la rejilla de resultados de la página principal.
- 2 El usuario elige la opción de eliminar sobre uno de sus comentarios.
- 3 El sistema pedirá confirmación antes de realizar el borrado.
 - 3.1 Si el usuario acepta la eliminación se borrará el comentario del sistema.
 - 3.2 Si el usuario no confirma la eliminación se termina el caso de uso.

Postcondición El comentario quedará borrado del sistema.

7.6 Clasificar comentarios

Descripción El sistema puntuará los nuevos comentarios según la opinión reflejada en los mismos.

Precondición

- Hay nuevos comentarios en el sistema o hay un nuevo modelo de clasificación.

Secuencia La secuencia de acciones será la siguiente:

- 1 Si hay un nuevo modelo de clasificación:
 - 1.1 Se obtienen todos los comentarios.
 - 1.2 Se clasifican los comentarios obtenidos.

- 1.3 Se actualiza la clasificación de los comentarios en el sistema.
- 2 Si hay nuevos comentarios:
 - 2.1 Se obtienen los nuevos comentarios.
 - 2.2 Se clasifican los comentarios obtenidos.
 - 2.3 Se actualiza la clasificación de los comentarios en el sistema.

Postcondición Todos los comentarios tienen una clasificación asociada.

8.1 Arquitectura general

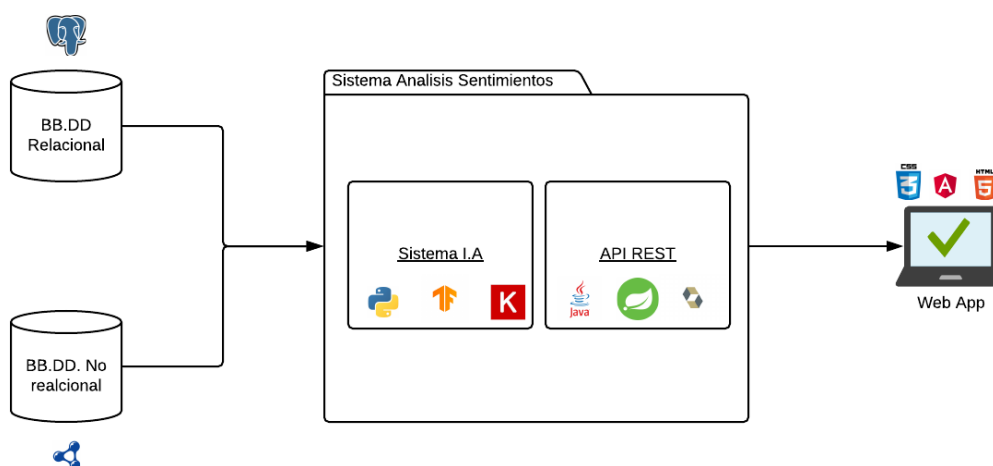


Figura 8.1: Arquitectura general del sistema

Como vemos en la figura anterior el sistema se compone de un componente principal que incluye el sistema de inteligencia artificial (clasificación de comentarios) y un sistema que publica una API rest para poder comunicarse con los usuarios finales a través de una aplicación web.

Este sistema se vale de dos bases de datos:

- BB.DD relacional: Se almacenan datos de artículos, usuarios y comentarios de forma que se puedan relacionar entre sí.
- BB.DD no relacional: Se trata de una base de datos documental con datos sobre las características de los artículos como el color o el material. Se almacena en forma de

tripletas RDF y se utiliza en el proceso de búsqueda, el cual queda fuera del alcance de esta memoria.

8.2 Arquitectura sistema de análisis de sentimientos

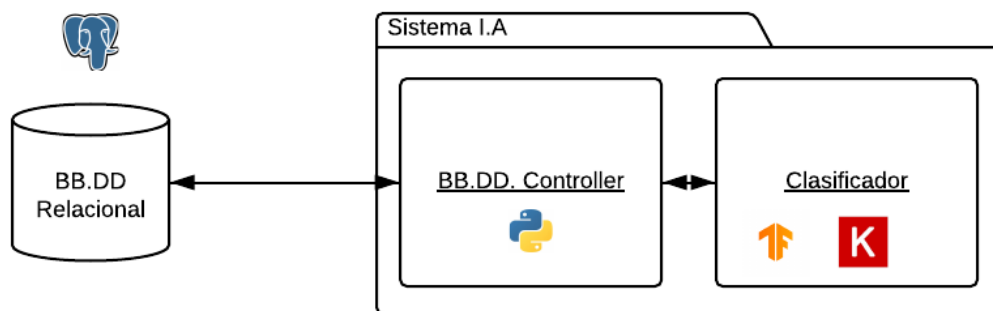


Figura 8.2: Arquitectura del sistema de análisis de sentimientos

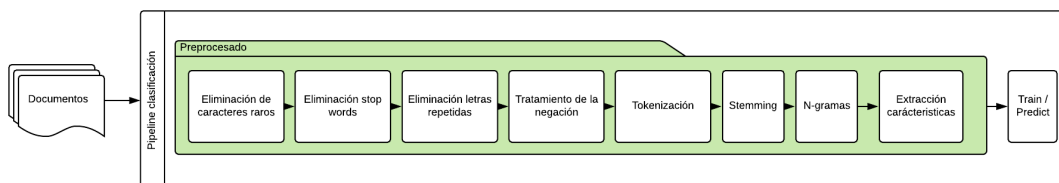


Figura 8.3: Arquitectura del clasificador

El sistema de clasificación de comentarios (8.2) se compone de dos módulos:

- Un módulo para gestionar la obtención y actualización de comentarios con la base de datos.
- Un módulo para realizar la clasificación (8.3): Compuesto por un pipeline con dos pasos principales, preprocesado y clasificación o entrenamiento. Los pasos del preprocesado no serán obligatorios y deberán activarse o desactivarse según lo que resulte más conveniente para obtener unos mejores resultados.

8.3 Arquitectura Web

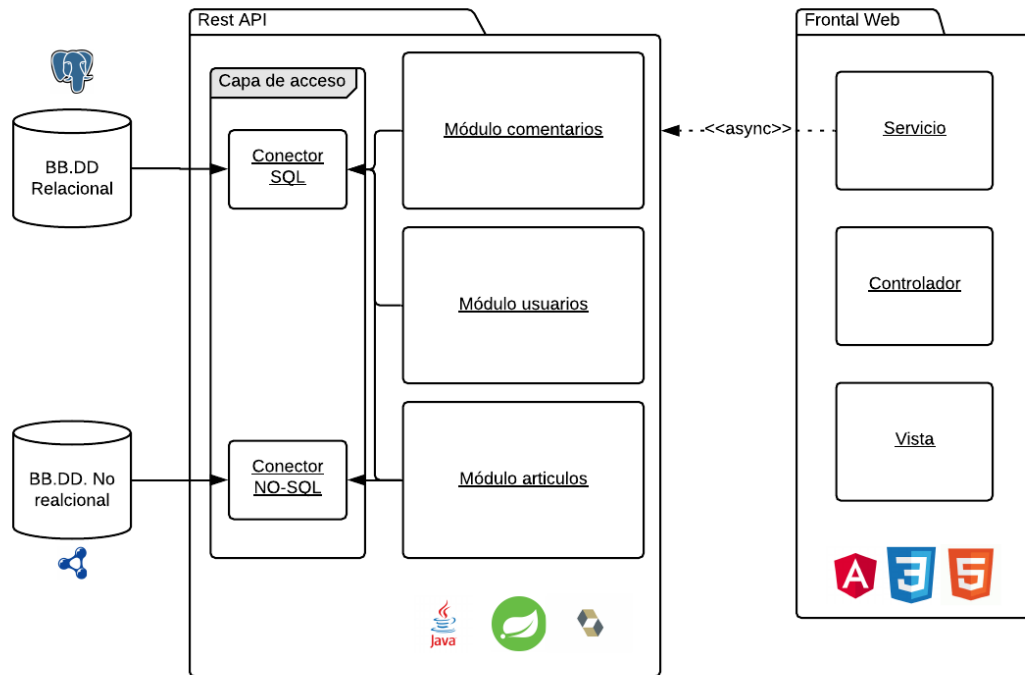


Figura 8.4: Arquitectura del servicio y frontal web.

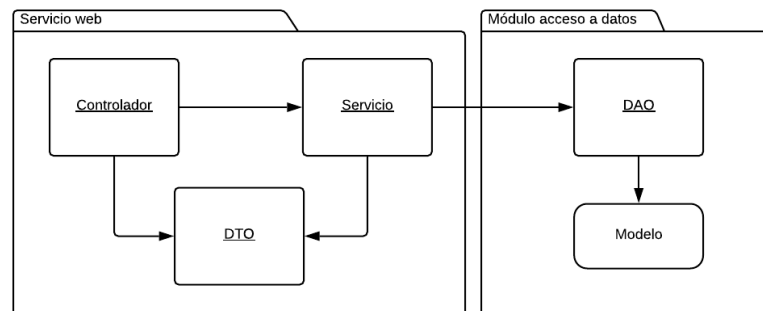


Figura 8.5: Ejemplo de arquitectura de un módulo REST del servicio.

La aplicación REST (8.4) se compone de tres módulos diferenciados para gestionar las distintas entidades del sistema: usuarios, artículos y comentarios.

Cada módulo sigue un patrón de acceso a datos DAO para abstraerse de la tecnología de base datos utilizada y ser más resistente a cambios, y un patrón DTO para la transmisión

de datos, esto nos ayuda a reducir la cantidad de llamadas realizadas al sistema y permite al servicio web abstraerse de la implementación del back-end.

El frontal web se estructura con un patrón MVC (Modelo-Vista-Controlador) en el que el servicio realiza las llamadas asíncronas al API REST y devuelve los datos al controlador, que se encargará de la comunicación entre las vistas y los servicios.

8.4 Trabajo futuro

Esta arquitectura se podría simplificar utilizando la nueva librería de javascript TensorFlowJS. Con esto podríamos eliminar el sistema de clasificación de comentarios, clasificándolos en caliente en el momento de su publicación desde la aplicación WEB. Para utilizar esta librería solo necesitamos tener un modelo guardado en formato .h5, y dado que la clasificación es un proceso ligero no se espera que suponga una carga importante para el usuario final.

Capítulo 9

Base de datos

EN este capítulo trataremos el diseño de las tablas de base de datos en las que se persistirán los comentarios realizados por los usuarios, así mismo se representarán y describirán todas las entidades relacionadas con los mismos.

9.1 Diseño lógico: Modelo Entidad-Relación

A continuación, se muestra el modelo ER del sistema de comentarios. En este diagrama (fig. 9.1) se representan las relaciones y atributos de las 3 entidades utilizadas en subsistema de comentarios: Usuario, Comentario y Artículo.

Siguiendo los requisitos mínimos expuestos por el cliente, un usuario podrá publicar de 0 a N comentarios acerca de un producto, y cada artículo podrá tener 0 o N comentarios.

9.2 Diseño físico: Modelo relacional

La figura 9.2 representa el modelo relacional del subsistema de comentarios.

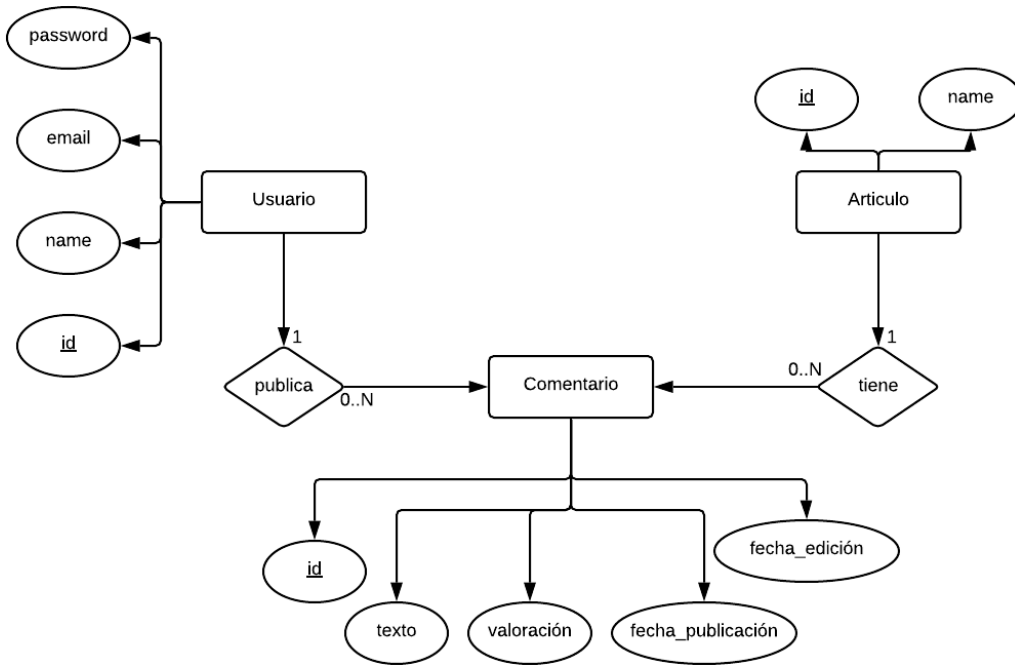


Figura 9.1: Diagrama entidad relación para el subsistema de comentarios.

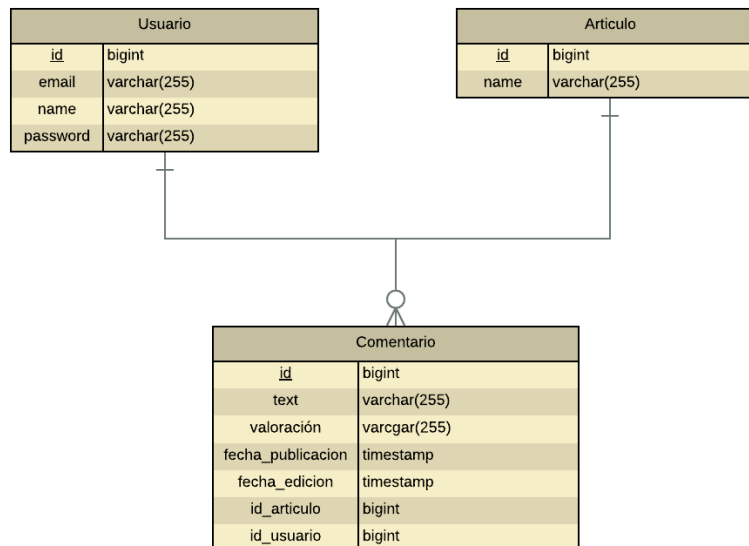


Figura 9.2: Modelo relacional para el subsistema de comentarios.

EN este capítulo describiremos la estructura de clases de los dos sistemas realizados en este trabajo.

10.1 Análisis de sentimientos

El sistema de análisis de sentimientos estará compuesto por una superclase Classifier que contendrá métodos y parámetros comunes a las clases Machine Learning y Deep Learning. Estas dos subclases implementarán de forma individual el comportamiento de los métodos de entrenamiento y predicción. Para realizar el preprocesado de los documentos se creará la clase VectorizerHelper que extenderá de las clases de la biblioteca sklearn BaseEstimator y TransformerMixin, esto nos permitirá utilizar estas clases como parte del pipe de la búsqueda de parámetros.

10.1.1 Diagrama de clases

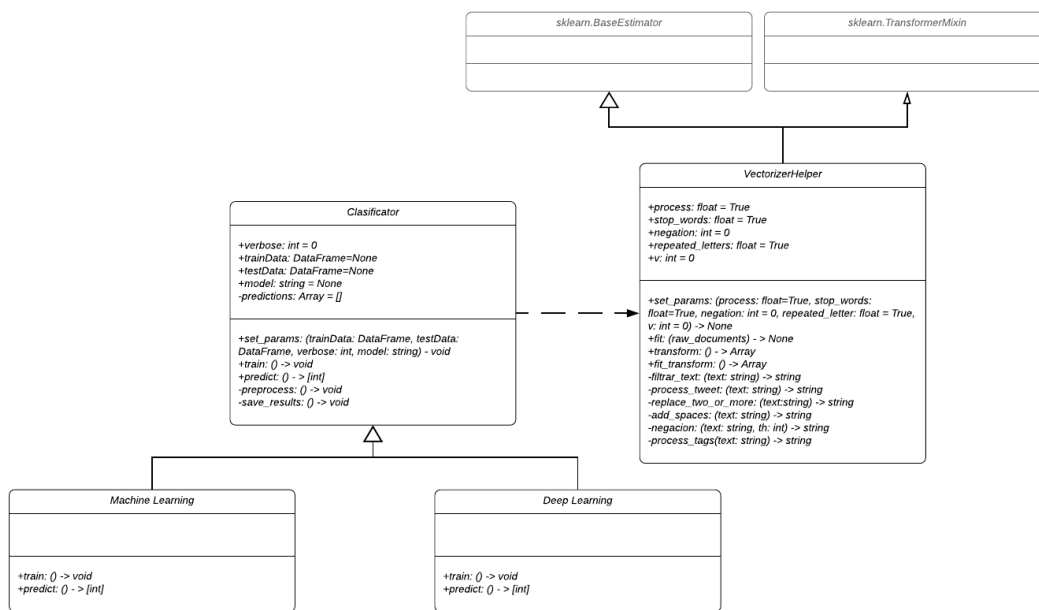


Figura 10.1: Diagrama de clases del sistema de análisis de sentimientos

10.2 Servicio web

En esta sección se mostrará el diagrama de clases del servicio de comentarios. En la figura 10.2 vemos la relación de las clases entidad relacionadas al comentario. En la figura 10.3 se describe la estructura del controlador y el servicio de comentarios, así como la descripción del DTO (Data Transfer Object) devuelto por el servicio.

10.2.1 Diagrama de clases

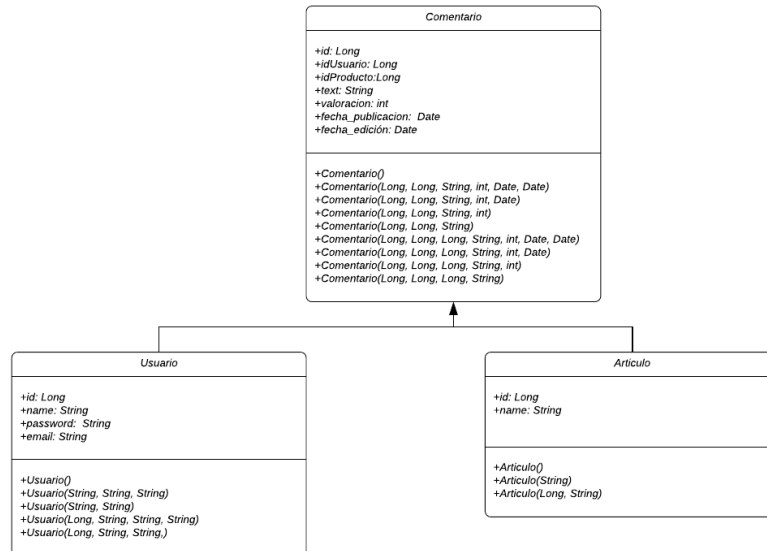


Figura 10.2: Diagrama de clases de las entidades del modelo SQL

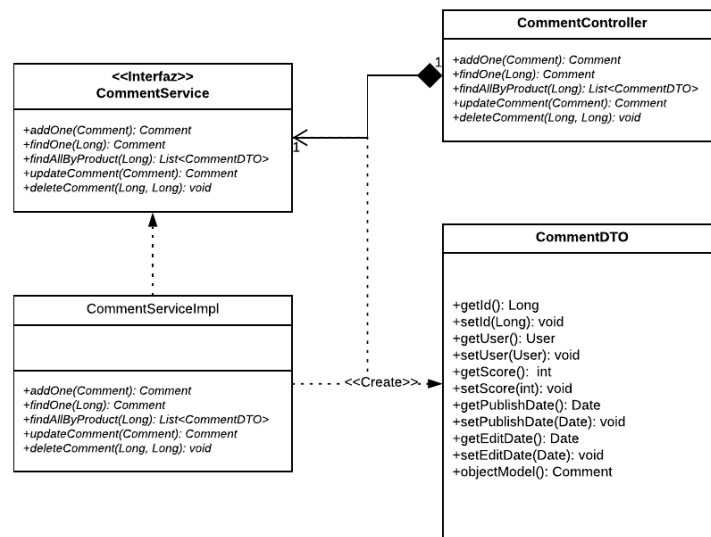


Figura 10.3: Diagrama de clases de comentario.

Implementación

11.1 Análisis de sentimientos

11.1.1 Tratamiento de los datos

Para poder realizar un trabajo de análisis de sentimientos es necesario llevar a cabo un proceso previo conocido como preprocesado con la intención de reducir la dimensionalidad del problema, y facilitar la tarea de clasificación de los algoritmos.

Los corpus de datos utilizados para las tareas de clasificación (ver sección 12.1) presentan mucho ruido, además de palabras que no tienen un impacto real en la polarización del texto. Si tenemos en cuenta que los algoritmos de clasificación utilizan palabras como característica base para la clasificación, podemos ver que mantener estos elementos en los textos aumentará la dimensionalidad del problema, por lo tanto planteemos una serie de pasos que ayudarán a limpiar los textos como la eliminación de caracteres raros, tratamiento de emoticonos, eliminación de "stopwords", tratamiento de la negación, etc. Pasos de los cuales hablaremos en profundidad en este apartado.

El tratamiento de los datos se divide en dos grandes pasos principales: transformación y filtrado, mientras el proceso de transformación hace referencia a la adaptación del contenido de los textos, el filtrado se centra más en la reducción de la dimensionalidad de los mismos ya sea mediante métodos estadísticos (TF-IDF), mediante la identificación de la categoría morfológica de la palabra y eliminación de las menos significantes o la eliminación de "stopwords".

Tokenización

La tokenización es el paso base del preprocesado de textos, este paso consiste en la división del texto en sus átomos más básicos (palabras), que se conocen como tokens. Por ejemplo el texto *Es la mejor película de mi vida* daría como resultado:

- **tokens:** [Es, la, mejor, película, de, mi, vida]

Estos tokens son utilizados como entrada del resto de procesos de los que hablamos en esta sección.

Eliminación caracteres raros

Dado que los textos proceden de entornos online es común encontrar en ellos elementos como URL, e-mails, abreviaturas, hashtags, nominaciones a usuarios (@usuario) o caracteres raros como signos de puntuación, corchetes, exclamaciones, etc. Estos elementos no aportan información importante a la hora de realizar una clasificación de los textos por lo que los eliminaremos de los mismos o los substituiremos por elementos que sí puedan aportar, como en el caso de algunas abreviaturas. Para entender el por qué de la substitución de abreviaturas hay que tener en cuenta que en el proceso de filtrado se escogen las palabras con más presencia en los documentos, por lo que si en algunos documentos aparece la palabra "gracias" que puede aportar un sentimiento positivo al mismo y en otros aparece la palabra "thx" de la abreviatura de la palabra inglesa "thanks", podemos encontrarnos un caso de disolución de la relevancia de la misma y que termine por no utilizarse como elemento del clasificador.

Eliminación letras repetidas

Se ha comprobado que es muy frecuente en los documentos utilizados en nuestro sistema el uso abusivo de caracteres repetidos como forma de enfatizar un sentimiento o representar una forma de exageración.

que maaaaaaaalo!!!!

Aunque esto resulte fácilmente interpretable para el lector humano, el sistema de aprendizaje automático no obtendrá ninguna conclusión de la aparición de estos elementos a menos de que un experto prepare los datos previamente creando algún tipo de constructo que el sistema pueda interpretar, o bien eliminándolos y reemplazándolos por una aparición única del elemento.

que malo! o que muy malo

Tratamiento emoticonos

Es común para el usuario de la red utilizar emoticonos o abreviaturas para expresar sus emociones en determinados momentos o contextos en los que se realice el comentario.

Para que el sistema interprete correctamente estos elementos es necesario crear un diccionario que los traduzca a palabras más comunes como bueno, neutro y malo, de forma que aporten un peso real en el documento.

Tratamiento negación

Los algoritmos de clasificación tradicionales no son capaces de manejar la negación por sí mismos, por eso en este ámbito se han presentado varios estudios que abordan distintas formas de preparar o interpretar los datos con la intención de solventar este problema. Sin embargo la gran mayoría de los estudios encontrados se centran en el tratamiento de los textos en inglés, dominio en el cual podemos encontrar muchos más recursos que para nuestro idioma, teniendo clasificadores POS muy avanzados, así como diccionarios de sinónimos y antónimos.

En este estudio nos basaremos en una aproximación simple utilizada en estudios como [8] [9] en la cual se niegan las palabras posteriores a la detección de la negación dentro de una cierta ventana.

Trabajo futuro Se podría utilizar una aproximación consistente en un sistema de aprendizaje profundo en la que se detecte lo que se conoce como alcance de la negación, Según la gramática, el alcance de la negación corresponde a la totalidad de palabras afectadas por la misma. Basándonos en el trabajo realizado por [10] para anotar un corpus en el que se define un alcance y un tipo de negación, el objetivo sería realizar una clasificación de los textos y un posterior tratamiento de los mismos, transformándolos según las necesidades que sugiera el resultado obtenido.

Eliminación stopwords

Es un paso común en los procesos de tratamiento de lenguaje natural. La intención de este paso es reducir la dimensionalidad del problema eliminando palabras muy comunes que no aportan valor a la polaridad del documento.

Para ello se define un diccionario de palabras comunes, compuesto en nuestro caso por preposiciones, conjunciones, pronombres, y ciertas conjunciones de verbos comunes. Tras eliminar estos elementos de los documentos obtendremos resultados como:

- Frase en el documento
Estáis ante una de las mejores películas
- Frase sin stopwords
mejores películas

Extracción n-gramas

En muchas ocasiones las palabras pueden ganar o perder fuerza enmarcadas en un contexto, es decir no es lo mismo decir que algo es "bueno" a decir que es "muy bueno", y dado que en el proceso de filtrado de las palabras el modificador "muy" puede llegar a no utilizarse

en el clasificador por ser demasiado común, hemos hecho uso de elementos conocidos como n-gramas.

Los n-gramas son constructos de palabras adyacentes dentro de una ventana determinada, por ejemplo la frase "la lluvia moja el suelo" se compone de:

- 1-gramas: [la, lluvia, moja, el, suelo]
- 2-gramas: [la lluvia, lluvia moja, moja el, el suelo]
- 3-gramas: [la lluvia moja, moja el suelo]
- 4-gramas: [la lluvia moja el, lluvia moja el suelo]
- 5-gramas: [la lluvia moja el suelo]

Stemming

El stemming pretende encontrar la raíz de la palabra, pero en este caso se utiliza un proceso heurístico mediante el cual se busca y eliminan sufijos y prefijos de forma que se puedan encontrar más palabras relacionadas bajo una misma raíz. Por ejemplo las palabras "bibliotecas" y "bibliotecario" comparten la raíz "bibliotec".

Para este proceso hemos utilizado el algoritmo SnowBall Stemmer implementado en la biblioteca NLTK.

Selección de características

Para la selección de características se utilizan dos aproximaciones en los algoritmos de Machine Learning:

- CountVectorizer: Construye un diccionario en el que cada índice es una palabra del conjunto de documentos y devuelve una matriz $M \times N$, siendo M el número de documentos y N el número de palabras del diccionario y cada posición representa el número de ocurrencias de la palabra n en el documento m .
- Tf-idf Vectorizer: En este caso en lugar de contar el número de apariciones del término se calcula el tf-idf (Term Frequency - Inverse Document Frequency) frecuencia de término – frecuencia inversa de documento, medida que expresa cuan relevante es una palabra para un documento dentro de una colección.

En los algoritmos de Deep Learning se utiliza un diccionario de Word Embeddings y se cruza con los documentos del corpus, seleccionando las coincidencias. En un segundo paso se buscan palabras similares para las palabras no encontradas utilizando la distancia de

Levenshtein. Si alguna palabra no se encuentra no será utilizada como característica en el entrenamiento.

Selección de modelos

Debido a la dependencia de los modelos del conjunto de datos de entrenamiento se ha utilizado un sistema de K-folds [11], mediante el cual realizaremos 10 divisiones distintas de los datos, compuestas cada uno de ellas de un subconjunto de entrenamiento y un conjunto de test. Esto nos permite entrenar los modelos en 10 conjuntos de datos distintos y obtener las métricas medias de los mismos para poder comparar los modelos y elegir el mejor.

Las métricas utilizadas son las siguientes:

- Accuracy (acc): Porcentaje de clasificaciones correctas (positivas y negativas) respecto al número total de observaciones o casos examinados.
- Precision (pr): Representa el porcentaje de clasificaciones positivas que han sido correctamente realizadas.
- Mean square error (mse): Error cuadrático medio de las clasificaciones, nos da una medida de la diferencia entre lo que se ha estimado y los valores reales.
- Recall: denominado también sensibilidad o exhaustividad es la tasa de positivos verdaderos, es decir, el número de observaciones clasificadas correctamente respecto al total de observaciones.

$$Recall = \frac{TP}{TP + FN} \quad (11.1)$$

- F-score (f1): denominada también F-score o medida-F. Es la media armónica de las medidas Precision y Recall y se emplea con el objetivo de obtener un valor único que resuma la evaluación ya que pondera la precisión y la sensibilidad.

$$F_1 = 2 * \frac{Precision * recall}{Precision + recall} \quad (11.2)$$

11.1.2 Algoritmos

A continuación pasamos a describir los algoritmos de los clasificadores utilizados en la investigación, que se dividen en dos grandes ramas:

- Machine learning: Algoritmos de aprendizaje automático como: Multinomial Naive Bayes, Logistic Regression, SVM, Random Forest.
- Deep Learning: Una subrama del Machine learning que utiliza redes neuronales para el aprendizaje como: LSTM, CNN y sus derivados.

Multinomial Naive Bayes

Como explican Jurafsky y Martin [12] en su estudio, este clasificador se conoce por este nombre por ser un clasificador Bayesiano que parte de una fuerte suposición previa (naive) de que los elementos (palabras) no están correlacionadas.

Este clasificador parte de un B.O.W y estudia las probabilidades de un documento de pertenecer a una clase dada la frecuencia de aparición de las palabras que lo componen a esa cierta clase, mediante una inferencia bayesiana, es decir un documento $d \in D$ pertenecerá a una clase $c \in C$ siendo C y D nuestros conjuntos de clases y documentos respectivamente, si:

$$c = \operatorname{argmax} P(x|d); x \in C, d \in D \quad (11.3)$$

Partiendo de la teoría probabilística planteada por Thomas Bayes (1702-1761) que relaciona la probabilidad de A dado B con la probabilidad de B dado A, podemos obtener a partir de $P(x|y)$ otras tres probabilidades con características que nos facilitan la resolución del problema:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \quad (11.4)$$

Si sustituimos 11.4 en 11.3 obtenemos:

$$c = \operatorname{argmax} \frac{P(d|x)P(x)}{P(d)}; x \in C, d \in D \quad (11.5)$$

Esta ecuación puede simplificarse eliminando $P(d)$ ya que vamos a utilizar 11.5 para computar todas las clases y $P(d)$ mantiene el mismo valor para todas ellas. Obteniendo:

$$c = \operatorname{argmax} P(d|x)P(x); x \in C, d \in D \quad (11.6)$$

Entendiendo un documento como un conjunto de características f podemos escribir $P(d|x)$ como $P(f_0, f_1, f_2, ..f_n|x)$. Sin embargo esta ecuación es demasiado costosa en términos de cálculo computacional, por lo que necesitamos introducir la suposición que hemos mencionado al inicio de la sección, mediante la cual suponemos independencia total entre las probabilidades de las características que forman el documento dada una clase x , de tal manera que 11.6 se simplificaría mediante el siguiente producto:

$$c = \operatorname{argmax} P(x) \prod_{f \in F} P(f|x) \quad (11.7)$$

Simplificando 11.7 determina la posibilidad de pertenencia de un documento a una clase, como la clase que obtiene más alta probabilidad del producto de la probabilidad de la clase

$P(x)$ por la probabilidad de cada una de las palabras del documento de pertenecer a dicha clase.

Logistic Regression

Este segundo algoritmo de clasificación es también conocido como algoritmo de **máxima entropía** o **MaxEnt**. Este algoritmo pertenece a la familia de los algoritmos exponenciales, y funciona de forma similar al algoritmo Naive Bayes, extrayendo una serie de características y pesos, y combinándolas linealmente.

La principal diferencia sobre el algoritmo Bayesiano es que se trata de un clasificador discriminativo, mientras el primero es generativo. Entendiendo generativo como el proceso de inducción de la probabilidad $P(y|x)$ ofreciéndole al clasificador la probabilidad $P(x|y)$, es decir sabiendo la clase y tratamos de predecir que características esperamos encontrar en el documento x . En el caso del modelo discriminativo lo que se pretende es computar directamente la probabilidad $P(y|x)$ discriminando entre los distintos posibles valores de la clase y .

El algoritmo consigue computar $P(y|x)$ extrayendo una serie de características del documento y multiplicando cada una por un peso de forma lineal como se muestra en 11.8

$$P(y|x) = \sum_{i=1}^{\infty} w_i f_i \quad (11.8)$$

Sin embargo esta ecuación produce resultados entre $-\infty$ y ∞ por lo que no son valores válidos, ya que deben estar entre 0 y 1. Esto se soluciona aplicando una función exponencial al producto wf de forma que todos los resultados sean positivos, y posteriormente aplicando un denominador que transforme los valores en resultados *legales*, es decir, entre 0 y 1. Dado que las caracterizadas f no son solo una propiedad de la observación (documento) x si no una propiedad tanto de la observación como de la clase candidata c . por la tanto la notación $f_i(x)$ pasa a ser $f_i(c, x)$, dando como resultado 11.9.

$$p(y = c|x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(c, x)\right) \quad (11.9)$$

Si consideramos N el número de características del documento y sustituimos Z obtenemos el valor final de la función de regresión logística 11.9 para computar la probabilidad de y de pertenecer a la clase c dado x

$$p(c|x) = \frac{\exp(\sum_{i=1}^N w_i f_i(c, x))}{\sum_{c' \in C} \exp(\sum_{i=1}^N w_i f_i(c', x))} \quad (11.10)$$

SVM

Las máquinas de soporte vectorial propuestas por [13] son un algoritmo 'simple' de clasificación, que se basan en intentar buscar el hiper plano que separa los datos en dos clases de la forma más óptima posible. La idea de optimizar el hiper plano se basa en conseguir tantos puntos como sea posible de la clase A a un lado del hiper plano como puntos de la clase B al otro lado, mientras que también se procura maximizar la distancia de cada punto al hiperplano (ver 11.1).

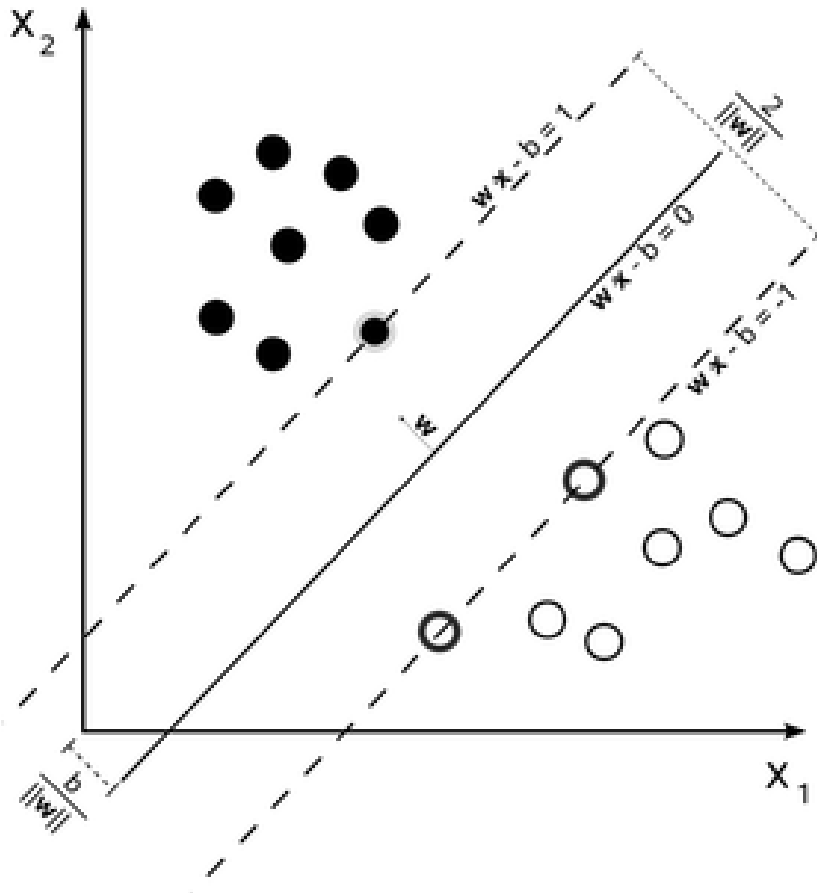


Figura 11.1: Representación de un hiper plano en una SVM

El hiper plano se define mediante:

$$\langle \vec{w} \vec{x} \rangle + b = \sum_i y_i \alpha_i \langle \vec{x}_i \vec{x} \rangle + b = 0 \quad (11.11)$$

donde \vec{x} es un vector de entrada n-dimensional e y_i es el valor de salida y $\vec{w} = (w_1, w_2, \dots, w_n)$ es el vector de pesos, o vector normal. Este último vector de pesos se construye mediante un conjunto de entrenamiento.

La clase de cualquier otro vector \vec{x}_i no perteneciente al conjunto de entrenamiento se puede calcular mediante la siguiente formula:

si $\vec{w} \vec{x}_i + b \geq 0$ entonces el elemento pertenece a la clase en la que estamos interesados, si no pertenece a la otra clase.

Como vemos este caso de SVM simple, es solo válido para problemas linealmente separables, para clasificaciones más complejas es necesario mapear los datos a un espacio de mayor dimensionalidad donde se vuelvan linealmente separables utilizando funciones conocidas como funciones de **kernel**

Para realizar una clasificación multiclase podemos proceder utilizando dos aproximaciones

- **Uno contra todos:** En esta aproximación se construye un clasificador por clase, cada uno de ellos toma una clase como positiva y el resto de clases como negativas, clasificando los elementos solo si son aceptados en dicha clase, y descartado en caso contrario.
- **Uno contra uno:** Se construyen clasificadores por cada par de clases y el elemento de estudio se clasifica en alguna de las dos clases pertenecientes al clasificador.

Random Forest

Este algoritmo desarrollado por Leo Breiman y Adele Cutler, y se trata de una combinación de árboles de decisión, tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.

La idea es promediar muchos modelos ruidosos pero aproximadamente imparciales, y por lo tanto reducir la variación.

Cada árbol es construido utilizando el siguiente algoritmo:

- Sea N el número de casos de prueba, M es el número de variables del clasificador.
- Sea m el número de variables de entrada a ser usado para determinar la decisión en un nodo dado; m debe ser mucho menor que M .
- Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.
- Para cada nodo del árbol, elegir aleatoriamente m variables en las cuales basar la decisión. Calcular la mejor partición del conjunto de entrenamiento a partir de las m variables.

Para la predicción un nuevo caso es empujado hacia abajo por el árbol. Luego se le asigna la etiqueta del nodo terminal donde finaliza. Este proceso se repite por todos los árboles del modelo, y la etiqueta final será aquella con mayor cantidad de incidencias.

LSTM

Este es el primero de los algoritmos perteneciente a la subsección de Deep Learning¹, esto quiere decir que a partir de este punto hablaremos de algoritmos que hacen uso de redes neuronales para obtener la clasificación de los documentos, comenzaremos por hacer una breve introducción a las redes neuronales para seguir describiendo el funcionamiento de las redes LSTM.

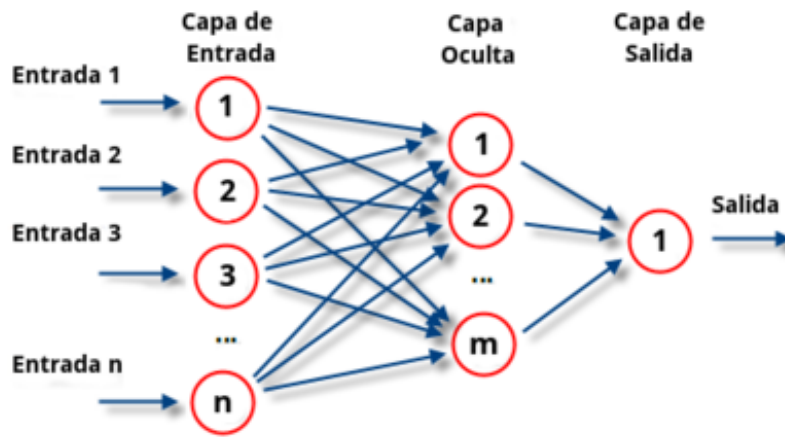


Figura 11.2: Representación de una red neuronal

Las redes neuronales son sistemas de computación basados en las redes neuronales biológicas, que pueden ser descritas matemáticamente como $f : X \rightarrow Y$, típicamente, están compuestas de una capa de entrada, una serie de capas ocultas y una capa de salida (ver 11.2), en estas capas ocultas se calculan una serie de pesos w que son propagados hacia la siguiente neurona mediante la ecuación 11.12

$$p_j(t) = \sum_i o_i(t) w_{ij} \quad (11.12)$$

Así mismo la función de una neurona $f(x)$ es definida por una composición de otras funciones $g_i(x)$ que pueden a su vez ser descompuestas en otras funciones. Típicamente esta función se representa como $f(x) = K(\sum_i w_i g_i(x))$ donde K es una función de activación predefinida como una tangente parabólica, una función sigmoideal o una función softmax por ejemplo.

Estas redes de tipo feedforward no mantienen noción de orden en el tiempo, por lo que cada input se considera solo a si mismo. Para tratar problemas del dominio del lenguaje natural necesitamos que la red tenga memoria ya que es importante el orden en el que aparecen las

¹Para los algoritmos que pasaremos a describir a partir de este punto se utilizaron Word Embeddings como características de los documentos en lugar de valores entre 0 y 1.

palabras en el documento, esto se soluciona utilizando Redes Neuronales Recurrentes en las que cada salida de una neurona es entrada de la siguiente y a su vez entrada de si misma (ver la figura 11.3). Esta propiedad permite a estas redes tener memoria, de forma que el resultado de una salida se mantiene como información en el estado oculto de la neurona (hidden state o h_t) durante todo el proceso, esto se puede describir matemáticamente como con la ecuación 11.13, en la que podemos ver que el estado oculto h_t recibe influencia del estado anterior h_{t-1} .

$$h_t = \sigma(Wx - t + Uh_{t-1}) \quad (11.13)$$

Además en el entrenamiento de estas redes se utiliza un algoritmo de propagación hacia atrás (backpropagation) mediante el cual actualizamos los pesos con los resultado obtenidos en el paso de propagación de los mismos con la intención de afinar los resultados. Es posible que durante este proceso se provoquen problemas de desvanecimiento de gradiente si los pesos son muy pequeños, o de explosión de gradiente si los pesos son muy grandes, provocando divergencia en el aprendizaje.

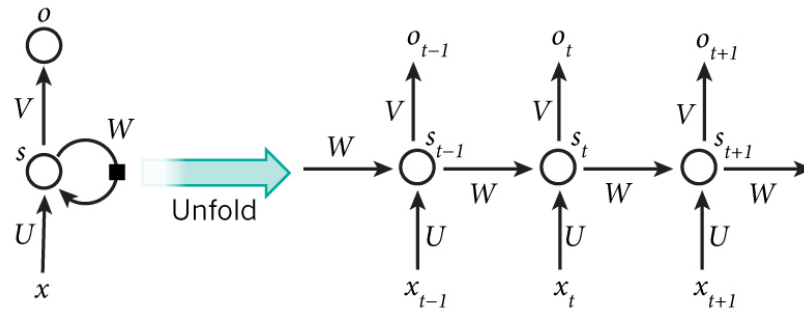


Figura 11.3: Representación de una red neuronal recurrente

Estos problemas en el algoritmo de propagación hacia atrás han sido uno de los motivos detrás de la creación de las redes LSTM o Long Short-Term Memory (ver 11.4).

Las redes LSTM son un tipo de redes neuronales recurrentes propuestas en 1997 por Sepp Hochreiter y Jürgen Schmidhuber y mejoradas en el 2000 por Felix Gers junto con los autores anteriores, y han obtenido resultados record en compresión de textos en lenguaje natural y reconocimiento de escritura manual entre otras aplicaciones. Estas redes introducen una nueva estructura llamada celda de memoria, que se compone de cuatro elementos principales:

- **input gate:** permite a las señales entrantes modificar el estado de la celda de memoria.
- **neurona con una conexión recurrente:** permite que la celda tenga memoria
- **forget gate:** modula la conexión recurrente de la celda de memoria, permitiendo que recuerde u olvide los estados previos.

- **output gate:** permite que la celda influya sobre el resto de neuronas.

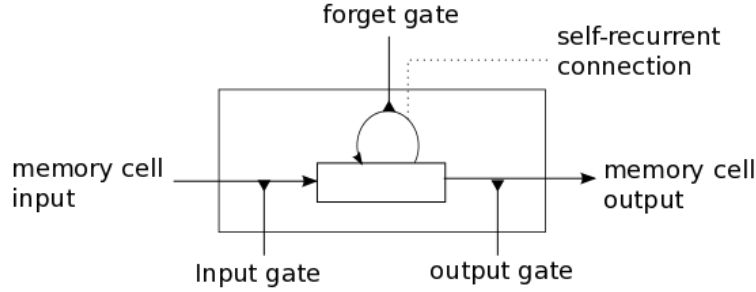


Figura 11.4: Representación de una LSTM

El estado de una celda de memoria puede ser descrito matemáticamente como:

$$h_t = o_t * \tanh(C_t) \quad (11.14)$$

donde $o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o)$, siendo:

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \quad (11.15)$$

siendo \tilde{C} el valor candidato para el estado de la celda de memoria en el instante t

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (11.16)$$

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (11.17)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (11.18)$$

LSTM Doble

En este caso nos encontramos ante el mismo algoritmo que el presentado con anterioridad, con una diferencia topológica ya que se compone de una capa de entrada, dos capas ocultas, y una capa de salida.

Las dos capas ocultas son dos LSTM, con una capa intermedia conocida como Dropout, que nos permite introducir una cierta cantidad de ruido a la red, con la intención de poder imitar el ruido existente en los problemas reales.

CNN

Este tipo de redes son ampliamente utilizadas en problemas de tratamiento de imágenes ya que se adaptan perfectamente al trabajo con píxeles, sin embargo son también muy eficaces en los problemas de lenguaje natural, además de ser más rápidas que las redes LSTM.

Como vemos en 11.5 estas redes realizan una convolución utilizando ventanas de un determinado tamaño.

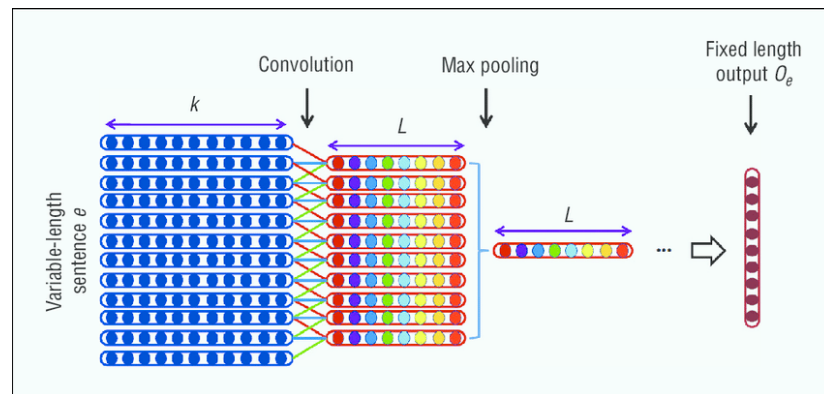


Figura 11.5: Representación de una red CNN

Tras la convolución utilizamos una capa de max-pool para convertir el resultado de la convolución en un gran vector de características, al igual que en el algoritmo anterior, se usa una capa de Dropout para introducir un cierto ruido en la clasificación.

Finalmente se computa la salida del clasificador mediante una capa softmax.

2dCNN

Mientras en el algoritmo CNN la ventana del filtro se componía de una sola dimensión, es decir un vector, en este caso la ventana es de dos dimensiones, de forma que la convolución actúa sobre una matriz (ver 11.6).

Esta versión del algoritmo nos permite tratar a la vez grupos de palabras y no solo ciertas partes del word embedding, de esta forma el algoritmo es capaz de tratar n-gramas de forma natural.

2dCNN + LSTM

Este es el primer caso en el que se experimenta con una composición de algoritmos, es decir, se usa un algoritmo híbrido, de forma que la red neuronal se compone de una capa de convolución de dos dimensiones que convierte varias palabras en un vector de características que se utilizará como entrada de las celdas de memoria de la capa LSTM que la sigue. De esta

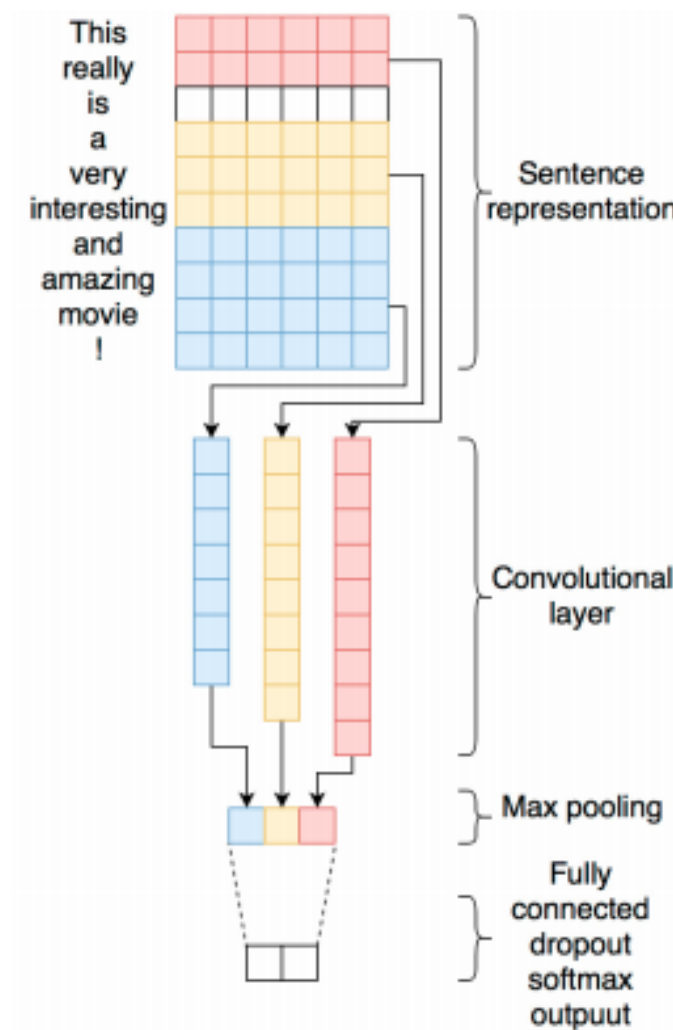


Figura 11.6: Representación de una red CNN

forma queremos combinar las bondades del tratamiento de n-gramas de las redes convolucionales con la capacidad de memoria de las redes LSTM.

Bidirectional LSTM

Este tipo de topología LSTM se compone de dos redes LSTM individuales, con una característica importante: la primera capa recibe los datos en su estado normal aprendiendo de atrás hacia adelante, y la segunda recibe los datos de forma invertida, de forma que puede aprender el efecto que tienen los datos en el “pasado”.

Los resultados de estas dos redes se combinan en una capa de fusión con una estrategia multiplicativa.

11.2 Servicio web

11.2.1 Visualizar comentarios

Se expone un endpoint, disponible mediante un método GET de HTTP que recibirá como parámetro un id de artículo.

Este endpoint buscará en la base de datos los comentarios relacionados con el artículo y lo devolverá al cliente mediante un array de objetos.

11.2.2 Añadir comentarios

Se expone un endpoint del servicio, que recibirá un objeto comentario y un id de artículo mediante un método PUT. Este método guardará el contenido del comentario en la base de datos relacionándolo con el artículo mediante una clave foránea.

11.2.3 Editar comentarios

El cliente enviará una petición POST a un endpoint del servicio, enviando el nuevo contenido del comentario y el identificador del comentario a editar.

11.2.4 Borrar comentarios

Mediante una petición DELETE que incluya un identificador de comentario, el servicio realizará un borrado del comentario en la base de datos.

Resultados

EN este capítulo analizaremos los resultados obtenidos en los modelos estudiados, realizando en todos ellos una división de los datos de entrenamiento mediante K-Folds, que nos permite comparar los modelos según precisión media.

12.1 Corpus de datos

12.1.1 Corpus TASS

Se trata de un corpus ofrecido por el TASS ¹, taller que se centra en las tareas de análisis de sentimientos y de reputación para el idioma español, organizado como evento satélite en la conferencia anual de la SEPLN (Sociedad Española para el Procesamiento del Lenguaje Natural).

Proponen dos tipos de clasificaciones: una basada en un sistema de 6 clases (Positivo +, Positivo, Neutro, Negativo, Negativo +, None) y otra en 4 clases (Negativo, Neutro, Positivo, None).

Para ello ofrecen un conjunto de 6800 tweets para entrenamiento, que vienen acompañados de una polaridad supervisada que se encontrará dentro del rango de 6 clases anteriormente citado.

En la figura 12.1 vemos una distribución con tendencia hacia los extremos, sobre todo a las polaridades positivas, mientras que la figura 12.2 nos muestra que la mayoría de los documentos contienen tan solo 10 o menos palabras, alcanzando un máximo de 28.

Hemos traducido un lexicón de palabras polarizadas del inglés para cruzarlos con los tweets de la colección y comprobar la distribución de las palabras en los mismos, obteniendo los siguientes resultados:

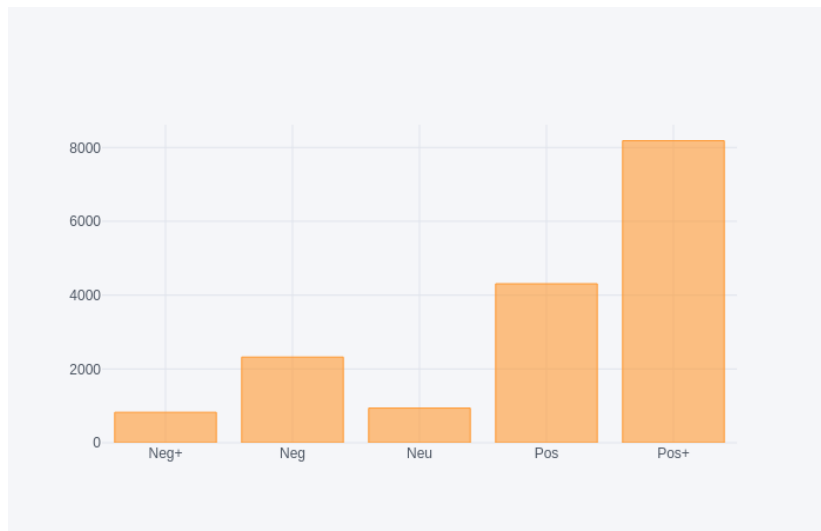


Figura 12.1: Distribución de la polarización.

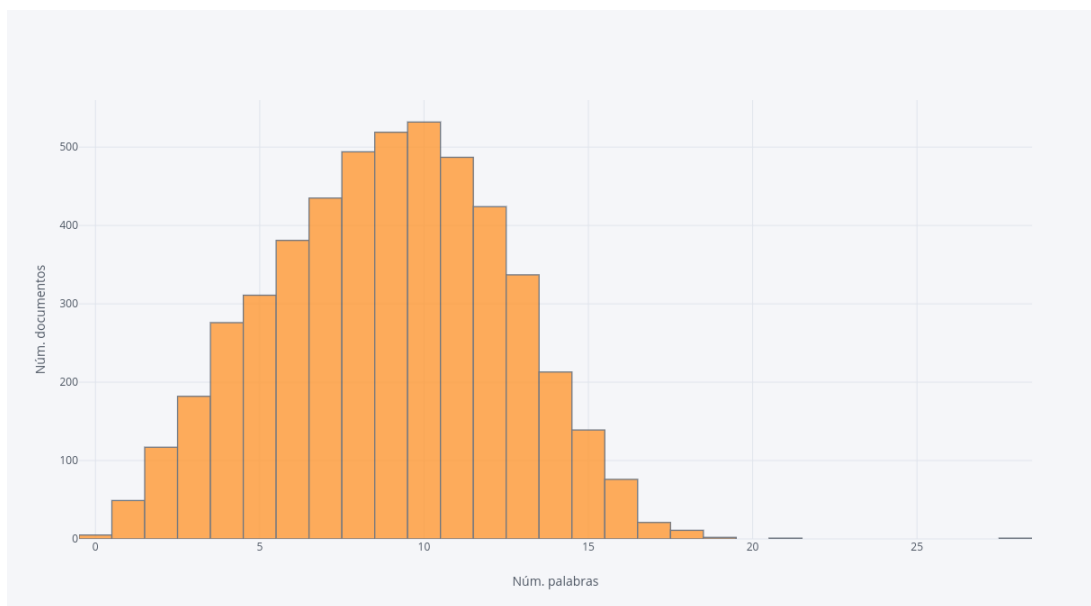


Figura 12.2: Histograma de número de palabras por documento.

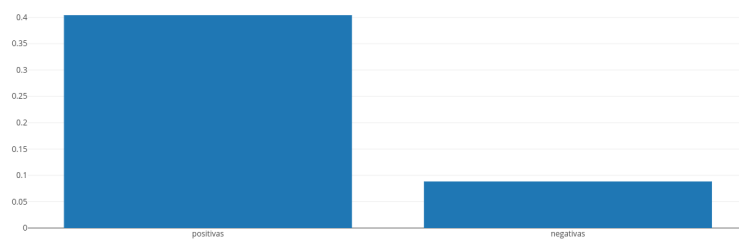


Figura 12.3: Distribución de las palabras polarizadas en tweets positivos

Como se ve en la figura 12.3 los tweets positivos presentan un cantidad notablemente mayor de palabras positivas, por lo que podemos deducir, que la clasificación de estos tweets en su clase debería ser sencilla, si somos capaces de eliminar correctamente todas las palabras que puedan aportar ruido innecesario al modelo.

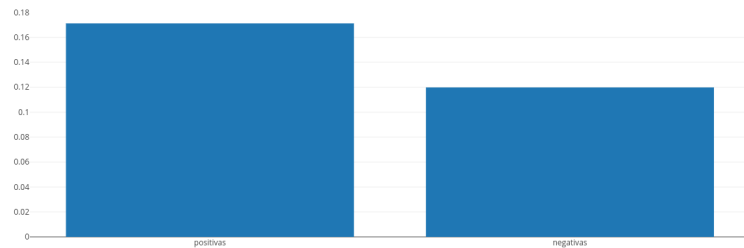


Figura 12.4: Distribución de las palabras polarizadas en tweets negativos

En el caso de los tweets negativos la gráfica 12.4 podemos observar que está más compensada, por lo que será más complicado para el modelo discernir la clase de estos tweets

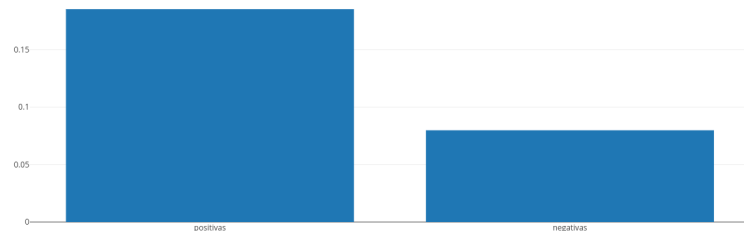


Figura 12.5: Distribución de las palabras polarizadas en tweets neutros

Para los tweets con clase neutral, se encuentran más palabras positivas que negativas, con lo que es muy probable que estos tweets acaben clasificados en una clase positiva.

Dado este análisis podemos deducir que el modelo de clasificación tendrá facilidad al clasificar los tweets positivos, pero dificultades al clasificar el resto. Esta tendencia del lenguaje a mostrar positividad ya ha sido demostrada por diversos estudios como los propuestos por Kloumann, Isabel M et al. [14] y Dodds, Peter Sheridan et al. [15].

12.1.2 Corpus Cine

Ya que en el momento actual no tenemos conocimiento de como van a ser los textos reales del dominio de la aplicación necesitamos probar los modelos contra textos mas extensos y

escritos de una forma más formal.

Se trata de un corpus descargado de la página MuchoCine.com para el trabajo, con la intención de disponer de un conjunto de documentos más extenso, tanto en número de documentos como en cantidad de palabras por documento (ver fig. 12.6). Con esto podremos comprobar el funcionamiento de los modelos en un dominio distinto al del conjunto de entrenamiento, y comprobar si son aptos para desplegar en producción.

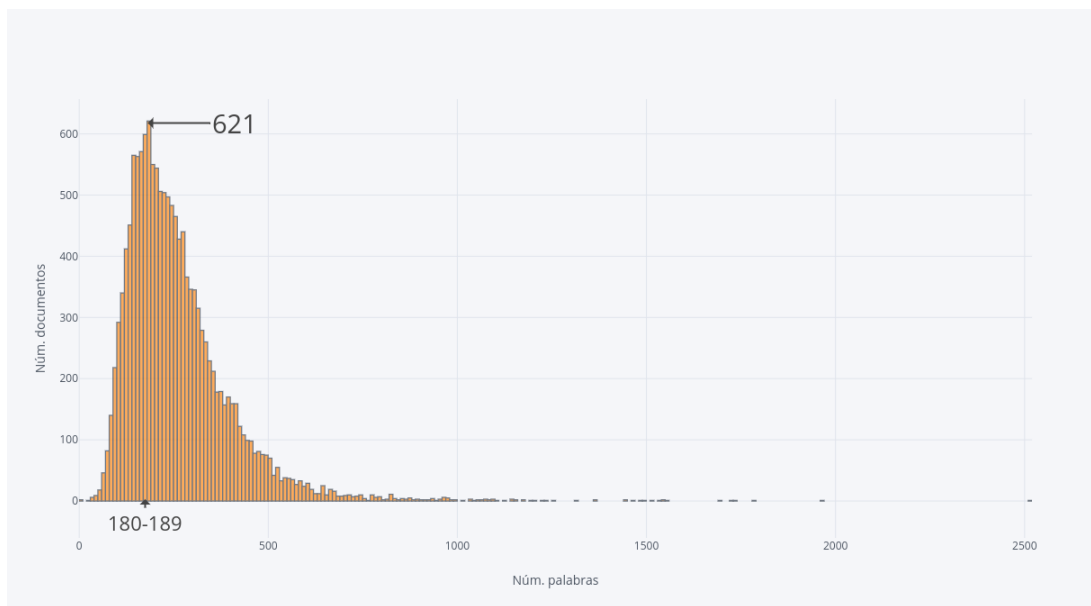


Figura 12.6: Histograma de número de palabras por documento.

Está compuesto por un total de 5000 documentos con una alta frecuencia de sentencias por documento en comparación con el corpus TASS.

En este caso los usuarios de la red tienen tendencia a realizar críticas con valoraciones medias, encontrando muy pocos casos en los extremos.

12.2 Machine learning

En las siguientes secciones comprobaremos el funcionamiento de los modelos para la clasificación de documentos del mismo dominio (TASS) y de distinto dominio (Cine).

Para establecer una línea base sobre la que probar nuevos modelos o mejoras se ha realizado un entrenamiento con los parámetros por defecto de los modelos explicados en la sección 11.1.2: MNB, LR, SVM, RF.

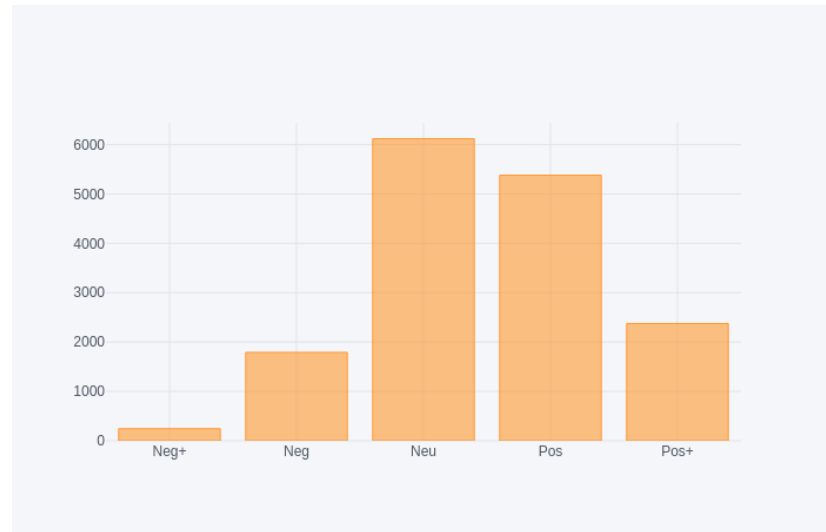


Figura 12.7: Distribución de la polarización.

12.2.1 Línea base

2 clases

	TASS				CINE			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	77.17	81.07	22.83	77.54	51.48	33.48	48.52	79.37
ls	77.29	81.02	22.71	78.01	49.08	26.58	50.92	78.07
mb	79.97	82.32	20.03	83.78	62.63	63.90	37.37	72.04
rf	73.92	78.60	26.08	74.54	51.88	48.44	48.12	63.21

Cuadro 12.1: Resultados en % entrenamiento para línea base.

El modelo con mejores resultados es MNB, con un resultado de f1 de 82.32% para el corpus TASS y un 63.90% para el corpus de críticas de cine.

3 clases

	TASS				CINE			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	72.31	72.31	93.62	72.31	31.19	31.19	150.51	31.19
ls	72.40	72.40	93.34	72.40	28.63	28.63	162.97	28.63
mb	73.19	73.19	81.34	73.19	37.35	37.35	128.21	37.35
rf	68.05	68.05	110.27	68.05	32.78	32.78	137.66	32.78

Cuadro 12.2: Resultados en % entrenamiento para línea base de 3 clases.

Al igual que en el caso de la clasificación binaria, el modelo con mejores resultados es el de Bayes con un 73.19% para el TASS y un 37.35% para las críticas de cine.

5 clases

	TASS				CINE			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	48.33	48.33	225.23	48.33	18.50	18.50	277.25	18.50
ls	48.09	48.09	240.97	48.09	16.09	16.09	307.73	16.09
mb	50.73	50.73	200.21	50.73	16.47	16.47	302.59	16.47
rf	43.53	43.53	243.98	43.53	17.92	17.92	298.46	17.92

Cuadro 12.3: Resultados en % entrenamiento para línea base de 5 clases.

En este caso las clasificaciones han bajado notablemente el resultado de la medida f1, obteniendo un 50.73% para Bayes, pero en el caso del corpus de Cine las clasificaciones se encuentran por debajo de la aleatoriedad (20% por ser 5 clases).

En el material anexo (sección [A.1.1](#)) se muestra un estudio sobre las causas de esta problemática.

12.2.2 Experimentos de mejora Machine Learning

Los experimentos han comenzado por un intento de mejora de los algoritmos de machine learning.

Como se menciona en [12.2](#) para el establecimiento de la línea base se han utilizado los parámetros por defecto de los algoritmos. Para intentar encontrar una mejor combinación se ha realizado una validación cruzada de los siguientes parámetros:

- Preprocesado de la negación (Verdadero o falso).
- Eliminación de caracteres repetido (Verdadero o falso).
- Rango de n-gramas (1 o 2).
- Tokenización o Stemming.
- Máximo número de iteraciones para SVM (1000, 2000, 1500).
- Máximo número de iteraciones para LR (100, 200, 500).
- Modificación del parámetro alpha para MB (1.0, 0.5 o 1.5).
- Número de estimadores para RF (10 o 100).

Además se han repetido las mismas utilizando preprocesado mediante CountVectorizer y mediant TFIDF.

2 clases

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	79.18	82.52	20.82	79.94	77.96	81.98	22.04	77.56
ls	78.24	81.70	21.76	79.24	79.97	82.87	20.03	81.63
mb	80.61	82.90	19.39	84.20	79.18	83.09	20.82	78.07
rf	75.14	78.85	24.86	77.30	73.89	78.03	26.11	75.75

Cuadro 12.4: Medias en % entrenamiento para gridsearch con 2 clases.

En la tabla 12.4 vemos que hemos conseguido mejorar la puntuación de todos los algoritmos. Cabe destacar el resultado del modelo de Bayes con un resultado cercano al 83% en cualquiera de los métodos de selección de características.

3 clases

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	68.88	68.88	106.81	68.88	73.68	73.68	88.24	73.68
ls	67.84	67.84	111.58	67.84	74.95	74.95	83.13	74.95
mb	70.12	70.12	101.82	70.12	74.68	74.68	84.22	74.68
rf	66.63	66.63	111.79	66.63	68.81	68.81	106.78	68.81

Cuadro 12.5: Medias en % entrenamiento para gridsearch con 3 clases.

En la tabla 12.5 vemos que el algoritmo tf-idf se comporta mejor ante la presencia de elementos con polaridad neutra. En este caso el modelo SVM es el de mayor puntuación.

5 clases

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	45.29	45.29	250.52	45.29	50.64	50.64	214.38	50.64
ls	44.68	44.68	284.19	44.68	50.70	50.70	203.83	50.70
mb	46.41	46.41	246.11	46.41	48.97	48.97	228.30	48.97
rf	39.57	39.57	278.27	39.57	45.08	45.08	241.49	45.08

Cuadro 12.6: Medias en % entrenamiento para gridsearch con 5 clases.

Como se muestra en la tabla 12.6 el algoritmo tf-idf es el que mayores puntuaciones obtiene, destacando el modelo SVM con una puntuación de 50.70%.

Conclusiones

Si comparamos los dos métodos de preprocesado de textos que hemos utilizado vemos que el Tf-idf es el que mejores resultados obtiene para todos los casos de clasificación.

En cuanto a los mejores parámetros, hemos obtenido valores dispares en cuanto al parámetro estudiado de cada algoritmo, sin embargo en todos los casos se establecido como necesario el tratamiento de la negación, el uso de Stemming y el preprocesado de letras repetidas. En cuanto al rango de n-gramas en la mayoría de los casos se ha obtenido un valor de 1.

En cuanto a los modelos vemos que Bayes es el que mejor funciona para el CountVectorizer sin embargo en el otro caso es el modelo SVM el que mayor puntuación obtiene.

Con esto podemos deducir que deberíamos escoger Tf-idf como selector de características y el modelo SVM para realizar las clasificaciones.

En el material anexo se incluyen también resultados de adaptación al dominio para los modelos optimizados por grid search. Si comparamos los resultados de los clasificadores en el conjunto de test (tablas A.1, A.3 y A.5) con los resultados en el dominio de críticas de cine (tablas A.2, A.4 y A.6) vemos que los modelos son tolerantes a la aparición de nuevos datos del mismo dominio pero su rendimiento baja notablemente en el otro caso. Esto se debe a que el algoritmo de selección de características está limitado a los elementos encontrados en el primer dominio y el resto de elementos del segundo dominio no serán tenidos en cuenta, provocando pérdida de información.

12.3 Deep Learning

En esta sección se mostrarán los resultados de los entrenamientos de modelos de Deep Learning.

Para estos casos se ha realizado una selección de parámetros manual sobre un modelo LSTM base, y posteriormente se han ejecutado el resto de algoritmos en la sección 11.1.2.

Todos los entrenamientos realizan 30 iteraciones (epochs).

12.3.1 Selección de parámetros

En esta sección mostraremos un ejemplo ilustrativo de como se ha iterado para la selección de parámetros de un modelo LSTM sobre el corpus TASS.

Este proceso se debe repetir para todas las distribuciones de clases y para todos los modelos.

Base

Este modelo se ha configurado con 64 neuronas en la capa oculta, y una activación RELU (fig. 12.8) que limita las salidas de la capa a un valor entre 0 e infinito.

La gráfica 12.9 muestra la evolución de la pérdida en los conjuntos de entrenamiento y validación durante las 30 iteraciones del entrenamiento. Se han eliminado los valores con picos muy altos para facilitar la lectura. Podemos ver que hacia el final del entrenamiento los valores de validación empiezan a divergir de los valores del conjunto de entrenamiento, esto es indicativo de que se comienza a producir un sobre entrenamiento.

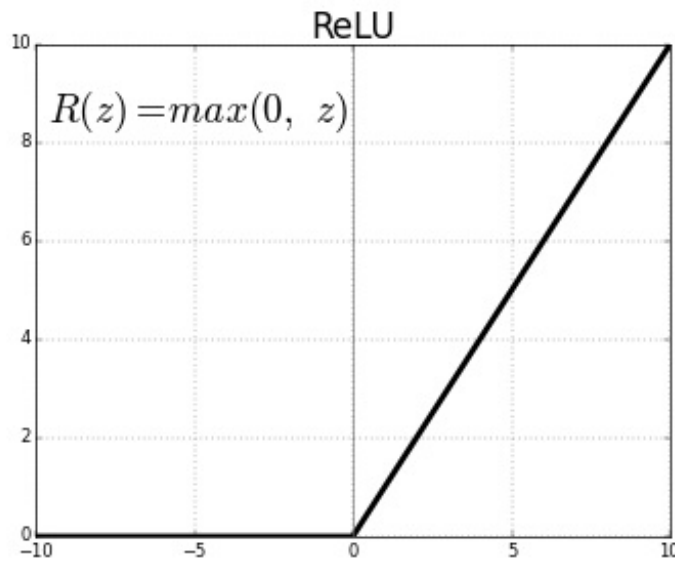


Figura 12.8: Función de activación RELU

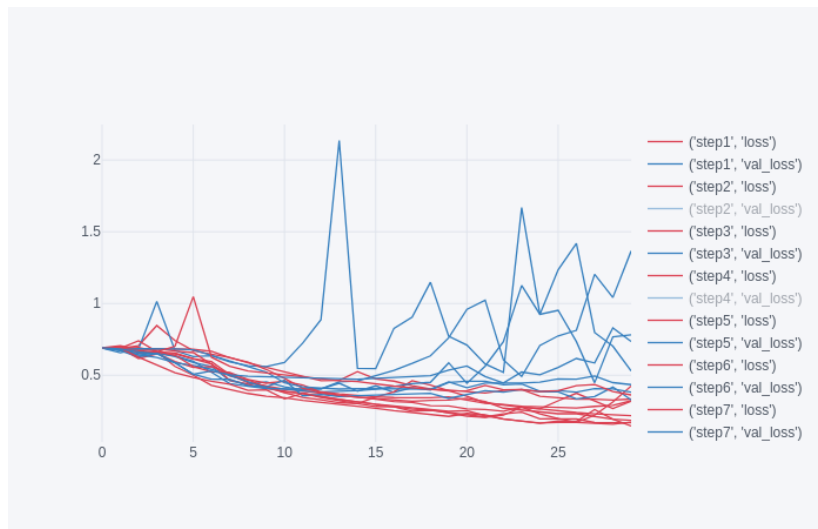


Figura 12.9: Evolución entrenamiento modelo LSTM

El sobre entrenamiento generalmente es síntoma de un modelo excesivamente complejo y por lo tanto provoca un alto nivel de varianza ajustándose en exceso a los casos de entrenamiento. Para solucionar esto se ha probado un modelo más simple de 10 neuronas.

La evolución del modelo simplificado (12.10) muestra menos rasgos de sobre entrenamiento manteniendo más tiempo la tendencia a reducir la pérdida en el conjunto de validación.

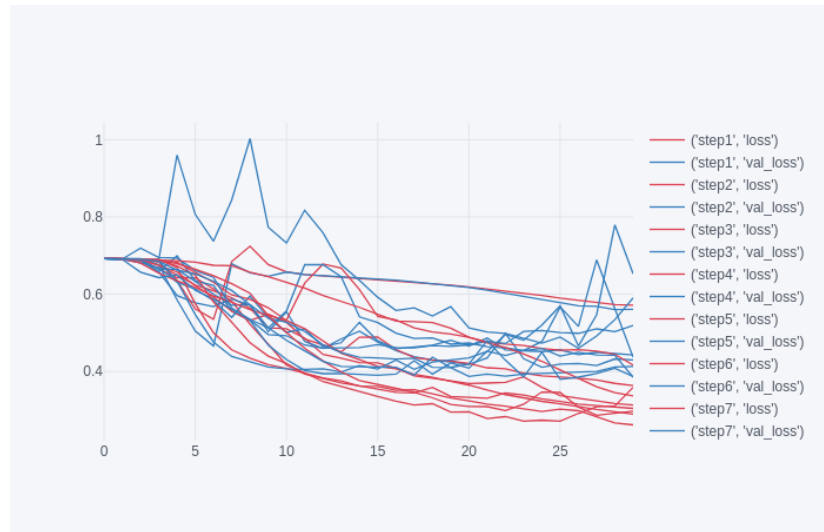


Figura 12.10: Evolución entrenamiento modelo LSTM simplificado

Dado que la tendencia de este modelo es buena se han realizado pruebas configurando otros parámetros, como:

- Dropout: Se introduce ruido sintético al modelo, apagando de forma aleatoria las neuronas durante el entrenamiento.
- Inicialización de pesos: Se cambia la inicialización de los pesos de las neuronas de la capa oculta, utilizando un tipo de inicialización conocida como Xavier initialization [16] o glorot normal, la cual asigna pesos de una distribución normal trucada centrada en el 0.
- Normalización de batch: Se normaliza la salida de la anterior capa de activación restando la media del batch y dividiendo por desviación típica. Con esto se consiguen entrenamientos más rápidos y menor varianza entre los resultados.

En la figura 12.11 vemos como han desaparecido los picos de las primeras iteraciones y parece que la evolución del entrenamiento es más adecuada, de forma que el modelo puede ser apto para seguir seleccionando mejores configuraciones.

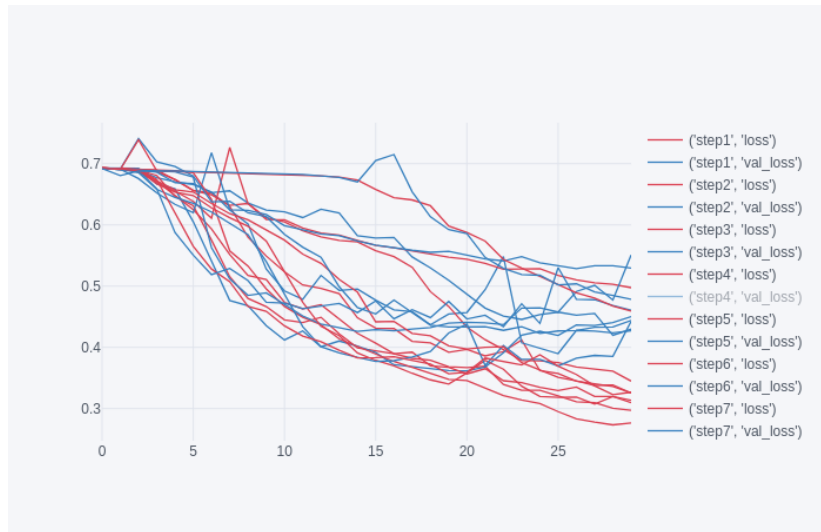


Figura 12.11: Dropout con valor 0.2

En el siguiente experimento (fig. 12.12) se aprecia mayor uniformidad en la evolución si despreciamos los picos, que pueden ser debidos a una distribución poco favorable para este modelo. Sin embargo cuando aplicamos además la normalización de batch, aunque conseguimos eliminar los picos, vemos que se produce sobre entrenamiento en las primeras etapas.

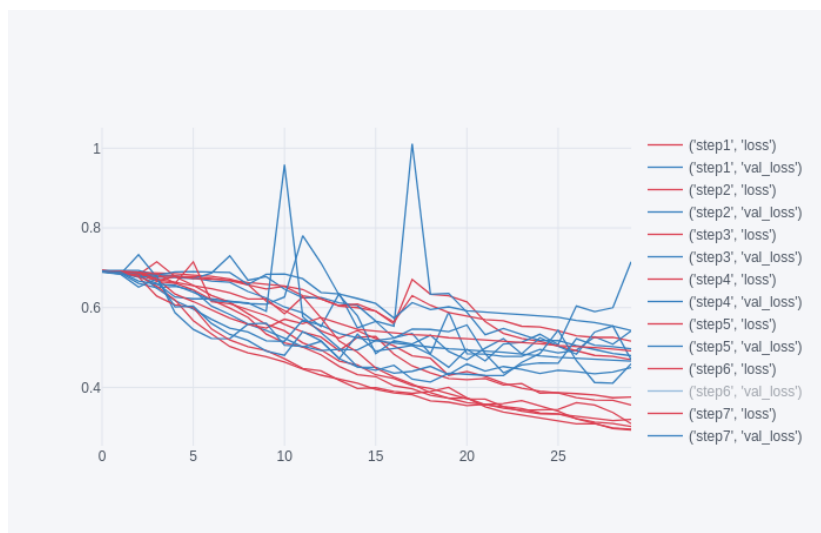


Figura 12.12: Inicialización Glorot más Dropout

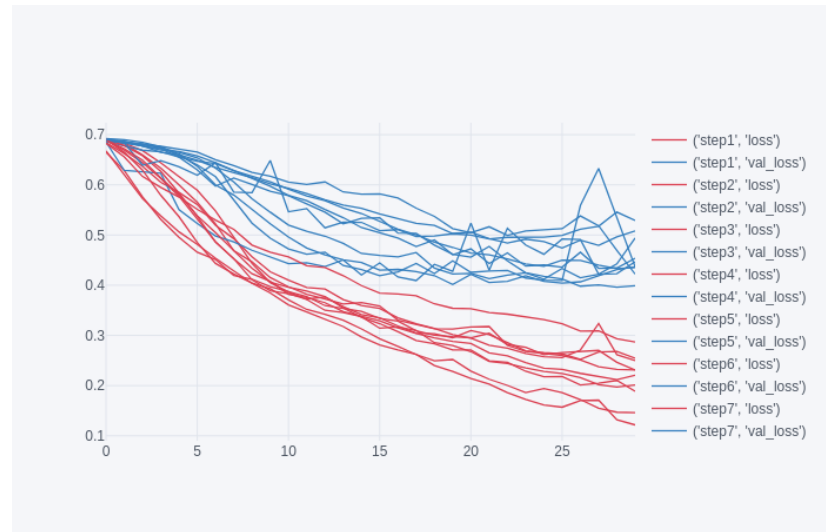


Figura 12.13: Inicialización Glorot, dropout y normalización de batch

Por lo tanto deberíamos quedarnos con el modelo con Dropout para seguir investigando ya que los otros parámetros no han dado resultados satisfactorios.

12.3.2 2 clases

	TASS				CINE			
	acc	f1	mse	recall	acc	f1	mse	recall
base	80.21	80.34	19.79	94.84	49.54	31.90	50.46	71.72
simpler	81.42	83.21	18.58	87.47	56.46	60.81	43.54	63.16
dropout	81.84	84.24	18.16	84.86	58.92	69.71	41.08	60.61
batch norm	82.27	85.12	17.73	82.95	57.80	72.05	42.20	58.26
glorot	81.56	84.97	18.44	80.59	58.26	66.35	41.74	61.95
glorot_wo_bn	83.97	85.80	16.03	88.24	56.58	60.20	43.42	63.79
double	81.70	82.73	18.30	91.42	52.38	43.70	47.62	68.55
conv	78.94	79.01	21.06	93.63	55.60	55.83	44.40	65.41
conv2d	81.42	84.93	18.58	80.22	56.14	63.19	43.86	61.18
bidirectional	81.06	82.49	18.94	88.97	52.30	46.34	47.70	65.86

Cuadro 12.7: Resultados sobre el conjunto de test.

La tabla 12.7 nos muestra el resultado de la clasificación del conjunto de test con los modelos entrenados sobre todo el conjunto de entrenamiento.

Vemos que los modelos con mejores resultados son los expuestos en la sección anterior,

debido a que la optimización de parámetros ha permitido reducir la variación entre los resultados de las distintas particiones y por lo tanto son más robustos ante distintas distribuciones.

Cabe destacar el modelo con inicialización Glorot con 85.80% en la medida f1, superando el 83.09% obtenido como máximo en los modelos de Machine Learning.

12.3.3 3 clases

	TASS				CINE			
	acc	f1	mse	recall	acc	f1	mse	recall
base	74.25	74.25	80.64	74.25	34.02	34.02	141.52	34.02
simpler	75.25	75.25	76.65	75.25	33.58	33.58	143.10	33.58
dropout	75.32	75.32	76.38	75.32	30.90	30.90	154.00	30.90
batch norm	77.18	77.18	68.93	77.18	34.24	34.24	140.64	34.24
glorot	76.58	76.58	71.32	76.58	22.78	22.78	186.48	22.78
glorot_wo_bn	74.18	74.18	80.90	74.18	35.08	35.08	137.46	35.08
double	74.12	74.12	81.17	74.12	31.48	31.48	151.96	31.48
conv	73.25	73.25	77.64	73.25	35.74	35.74	133.32	35.74
conv2d	73.52	73.52	76.78	73.52	29.40	29.40	160.06	29.40
bidirectional	75.58	75.58	71.52	75.58	34.92	34.92	137.86	34.92

Cuadro 12.8: Resultados sobre el conjunto de test.

En este experimento la mayor puntuación obtenida ha sido de 77.18% por un modelo LSTM con normalización de batch, superando el 73% obtenido con Machine Learning.

La introducción de elementos neutros en el conjunto de datos produce picos de pérdida muy altos en los modelos, los cuales son suavizados gracias a la normalización de batch.

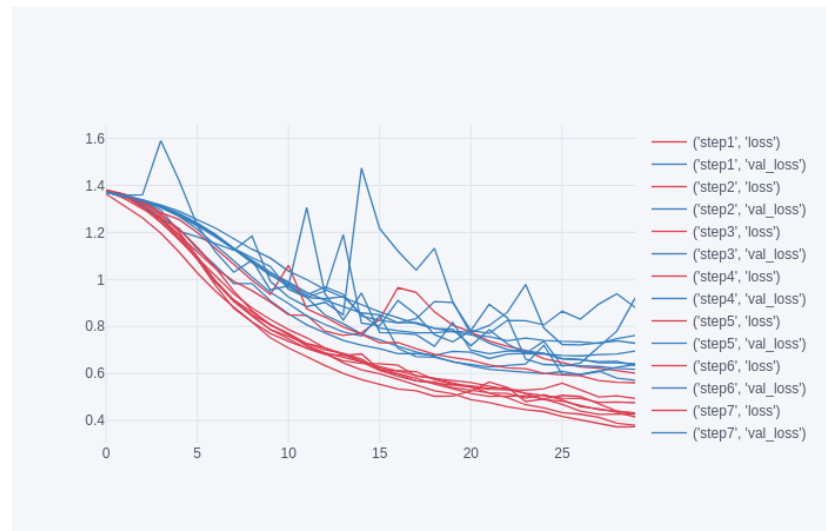


Figura 12.14: Evolución modelo LSTM con normalización de batch.

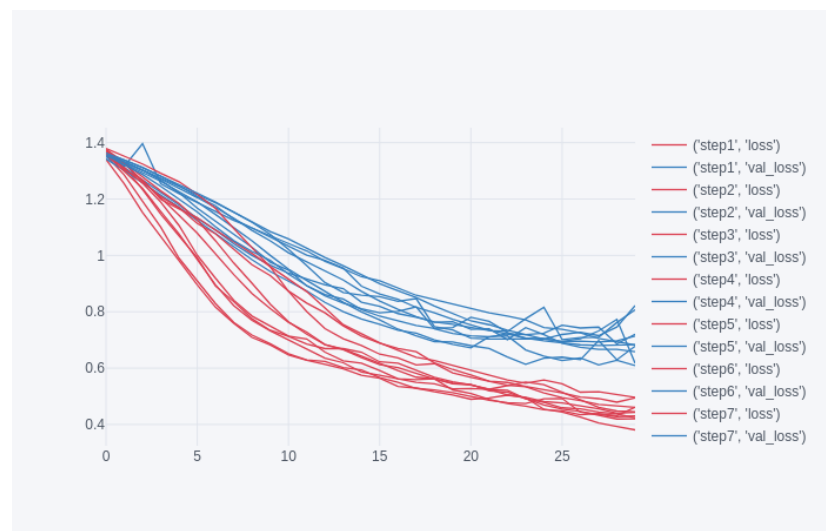


Figura 12.15: Evolución modelo LSTM con normalización de batch e inicialización Glorot.

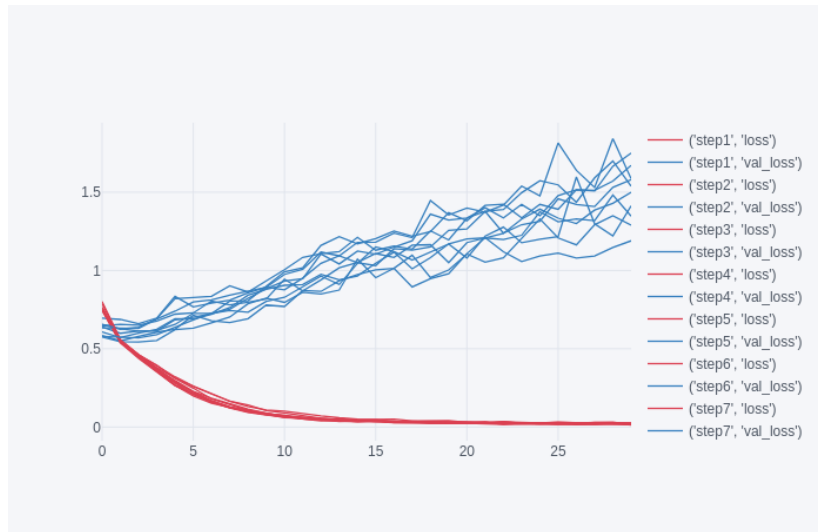


Figura 12.16: Evolución modelo convolucional.

Si vemos las gráficas de evolución de los dos mejores modelos (figuras 12.14 y 12.15) se observa una tendencia descendente que nos indica que podríamos obtener un mejor resultado afinando los parámetros o aumentando el número de iteraciones de entrenamiento. Por contra los modelos más complejos como el convolucional (fig. 12.16) presentan sobre entrenamiento en fases tempranas del entrenamiento.

Los modelos complejos se comportan mejor en la clasificación con cambio de dominio.

12.3.4 5 clases

	TASS				CINE			
	acc	f1	mse	recall	acc	f1	mse	recall
base	40.85	40.85	199.27	40.85	12.34	12.34	369.92	12.34
simpler	43.91	43.91	169.26	43.91	12.34	12.34	375.84	12.34
dropout	45.18	45.18	179.51	45.18	9.50	9.50	429.60	9.50
batch norm	43.58	43.58	176.71	43.58	8.82	8.82	455.38	8.82
glorot	45.51	45.51	201.60	45.51	11.76	11.76	382.44	11.76
glorot_wo_bn	43.05	43.05	176.31	43.05	13.24	13.24	357.64	13.24
double	41.85	41.85	142.38	41.85	21.80	21.80	219.12	21.80
conv	49.30	49.30	214.77	49.30	9.14	9.14	443.42	9.14
conv2d	43.11	43.11	182.97	43.11	11.26	11.26	400.44	11.26
bidirectional	45.51	45.51	182.37	45.51	12.00	12.00	396.64	12.00

Cuadro 12.9: Resultados sobre el conjunto de test.

A pesar de que el mejor resultado en este experimento lo obtiene una red convolucional, vemos en la fig. 12.17 que durante su entrenamiento se ha producido un sobreentrenamiento.

Por otro lado los modelos Glorot con normalización de batch y sin ella presentan una tendencia a la baja como se ve en la figuras 12.18 y 12.19 por lo que tenemos margen de mejora modificando hiperparámetros o ampliando el número de iteraciones hasta que se deje de apreciar una tendencia descendente en el entrenamiento.

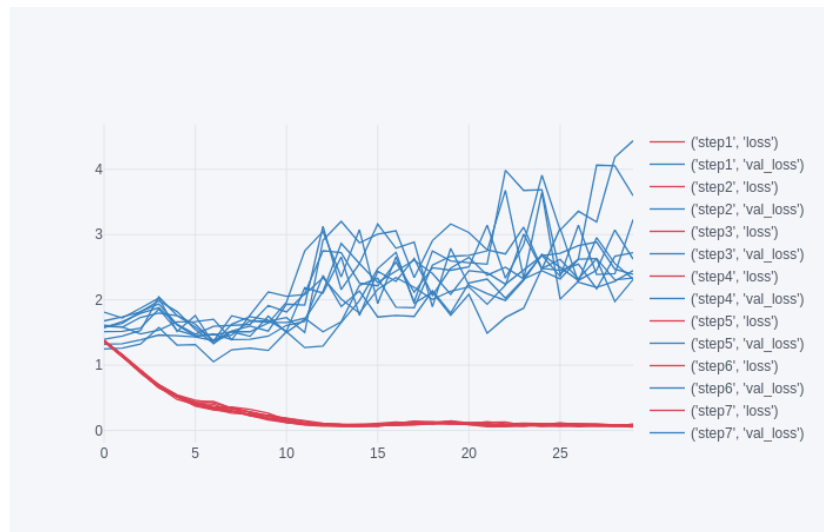


Figura 12.17: Evolución modelo convolucional.

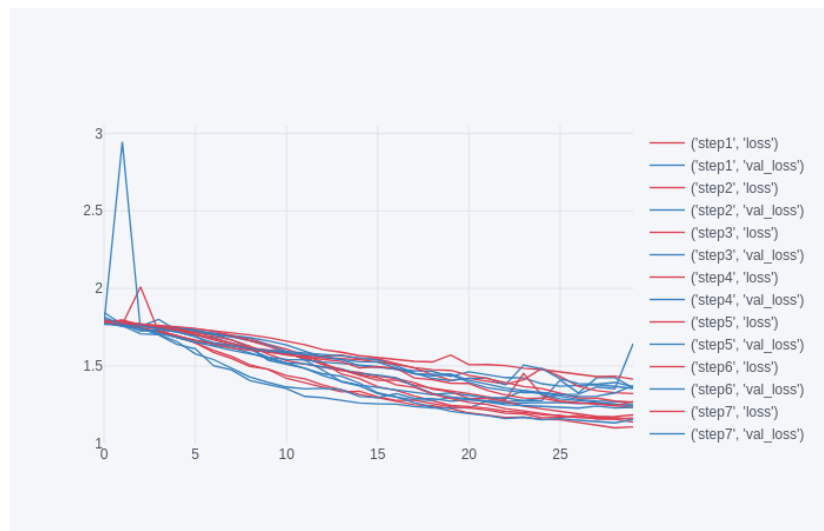


Figura 12.18: Evolución modelo LSTM con normalización de batch e inicialización Glorot.

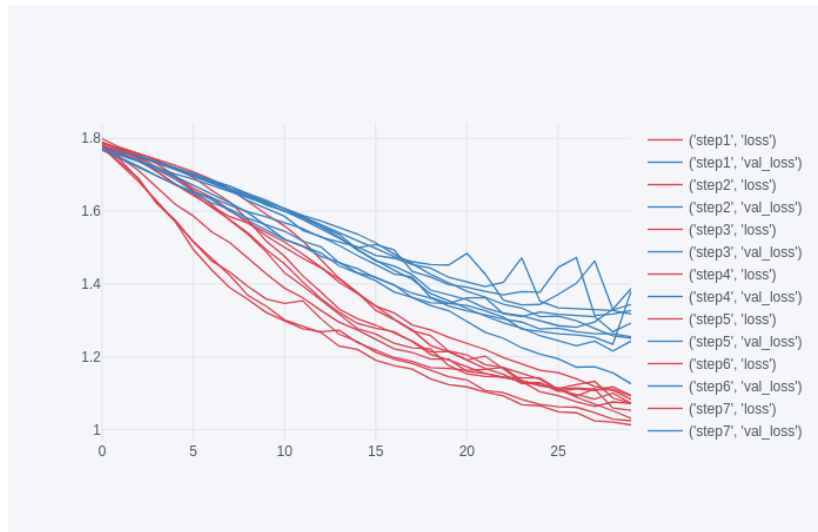


Figura 12.19: Evolución modelo LSTM con normalización de batch.

12.3.5 Conclusiones

Tras analizar los resultados de las secciones anteriores podemos concluir que la complejidad del modelo no está directamente relacionada con su precisión ya que han sido los modelos más simples los que han obtenido mejor resultado. Destacando entre estos los modelos con inicialización Glorot, normalización del batch o una combinación entre ambos.

Las topologías complejas tienen tendencia a adaptarse muy rápido a los conjuntos de entrenamiento provocando un sobre entrenamiento. Esto puede ser debido a la simplicidad del conjunto de datos utilizado.

Podemos concluir que para la puesta en producción deberíamos utilizar los modelos de Deep Learning.

Además hemos demostrado que el corpus propuesto para los entrenamientos no nos garantiza los resultados finales, dado que desconocemos como van a ser los comentarios finales publicados en la aplicación. En estos experimentos hemos demostrado que si los documentos tienen mayor dimensionalidad que el corpus TASS se puede producir pérdida de información al cambiar de dominio y por lo tanto conseguir un peor rendimiento de los modelos.

Sería recomendable comenzar con una clasificación por umbrales, de forma que solo resulten clasificados automáticamente aquellos textos que tengan una probabilidad de pertenecer a una clase mayor a un tanto por ciento establecido entre la empresa y el cliente, además de ofrecer al usuario la posibilidad de clasificar su comentario, de esta forma dispondríamos de un corpus de comentarios con un valor establecido por humanos.


Cuando la aplicación cuente con un número de comentarios suficiente sería posible utilizarlos como corpus de entrenamiento.

Solución desarrollada

EN este capítulo mostraremos una serie de capturas de la solución desarrollada, centrán- donos en la parte que se refiere a la resolución de los casos de uso expuestos en 7.

13.1 Sección de comentarios

Cantos rodados filita beige malva



NO IMAGE AVAILABLE

Nombre

Cantos rodados filita beige malva

Fabricante

Cupastone

Material

filita

Popularidad

Eco

FICHA TÉCNICA

COMENTARIOS(0)

BIM FILES

Número de serie	yy2
Tipo de producto	Tocho
Descripción	Cantos rodados de filita de color beige malva indicada para Pared exterior, Fachada, Pared interior
Sistemas de clasificación	NBS Reference: 45-25-45/380 Natural stone panels Omniclass: 23-15 15 15 11 Wall Stone Facing Uniclass 2.0: Pr_25_71_14_56 Natural stone panels Uniclass 2015: Pr_25_71_14_56 Natural stone panels Fachada

Figura 13.1: Sección de comentarios en la ficha de artículo.

13.2 Publicación de comentarios



Figura 13.2: Publicación de comentarios.

13.3 Edición y eliminación de comentarios



Figura 13.3: Edición y eliminación de comentarios.

Trabajo futuro En el apartado de contenido extra se incluyen una serie de capturas que documentan una propuesta de rediseño y modernización de la interfaz.

Apéndices

Apéndice A

Material adicional

A.1 Análisis de resultados

A.1.1 Línea base - 5 clases

En la figura A.1 se aprecia una gran diferencia en la distribución de clases, mostrando en las polaridades del conjunto de entrenamiento una tendencia de agrupación en torno a los extremos, mientras que en el conjunto de test se agrupan en torno a la neutralidad.

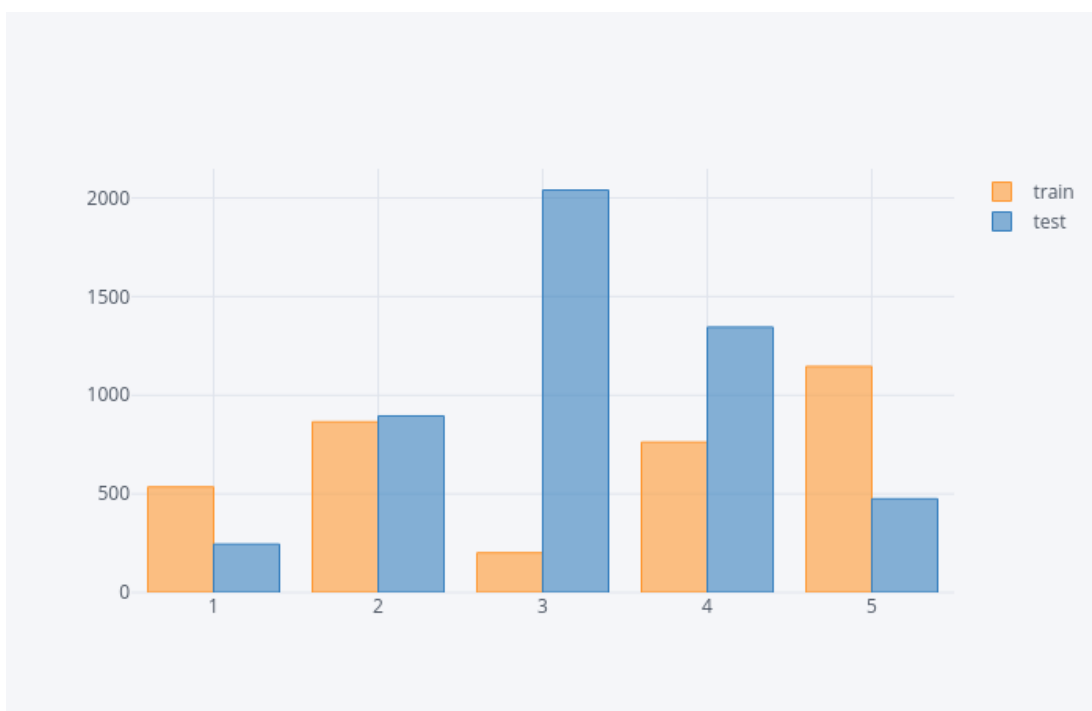


Figura A.1: Distribución de polaridades en los conjuntos de entrenamiento y test.

Este desbalanceo es una de las causas de los problemas de clasificación en distinto dominio,

ya que el modelo tiende a clasificar los documentos con las polaridades que más destacan en el conjunto de entrenamiento, en este caso las clases con etiquetas 2 y 5, como se ve en la matriz de confusión representada en la figura A.2.

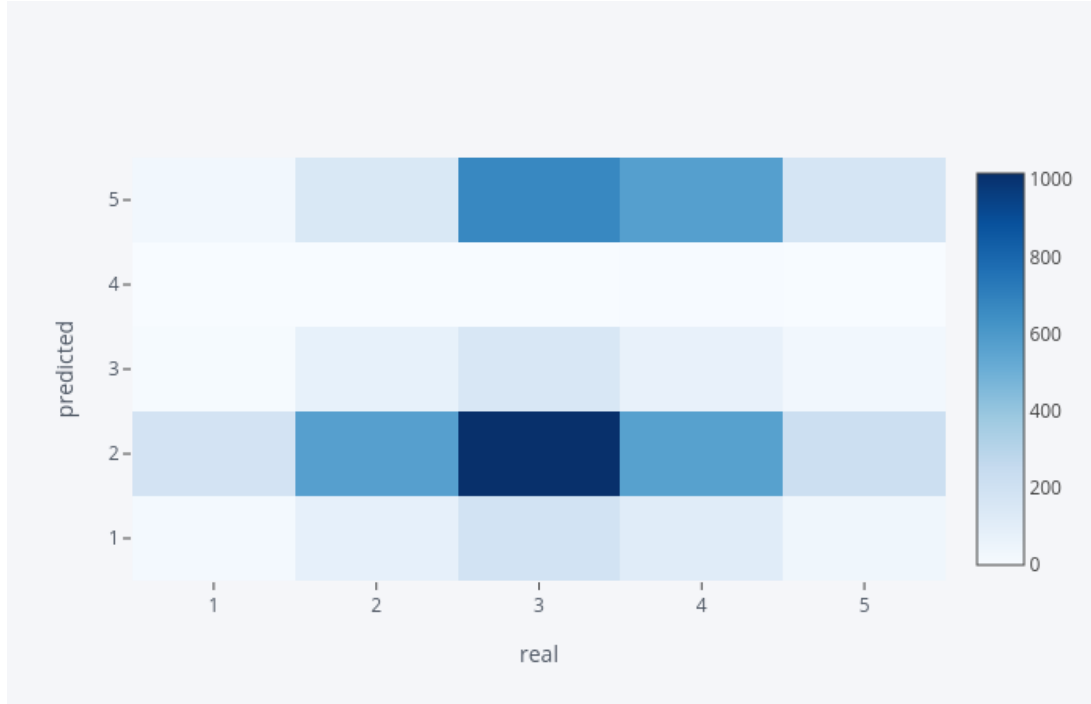


Figura A.2: Matriz de confusión de la clasificación en 5 clases.

A.2 Resultados GridSearch en Machine Learning

A.2.1 2 clases

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	76.88	80.87	23.12	77.77	79.22	82.04	20.78	82.29
ls	76.60	80.52	23.40	77.85	80.99	83.21	19.01	85.35
mb	79.93	82.39	20.07	83.90	77.38	81.89	22.62	76.46
rf	73.55	77.10	26.45	77.44	70.92	74.85	29.08	75.12

Cuadro A.1: Resultados en % del mejor modelo obtenido en los k-folds

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	63.44	69.51	36.56	63.82	65.02	72.47	34.98	66.30
ls	62.55	65.74	37.45	65.66	64.56	73.09	35.44	64.97
mb	60.34	54.67	39.66	72.72	60.88	73.29	39.12	60.41
rf	52.69	56.06	47.31	56.87	50.16	41.53	49.84	64.08

Cuadro A.2: Resultados de la clasificación en el dominio de críticas de cine.

A.2.2 3 clases

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	67.86	67.86	105.59	67.86	74.32	74.32	80.37	74.32
ls	67.80	67.80	106.45	67.80	75.25	75.25	76.65	75.25
mb	70.13	70.13	96.74	70.13	71.86	71.86	90.22	71.86
rf	64.01	64.01	116.23	64.01	69.66	69.66	98.80	69.66

Cuadro A.3: Resultados en % del mejor modelo obtenido en los k-folds

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	39.74	39.74	118.28	39.74	38.74	38.74	122.64	38.74
ls	38.36	38.36	124.34	38.36	38.86	38.86	122.16	38.86
mb	36.06	36.06	133.36	36.06	37.14	37.14	129.04	37.14
rf	34.94	34.94	119.24	34.94	34.42	34.42	139.92	34.42

Cuadro A.4: Resultados de la clasificación en el dominio de críticas de cine.

A.2.3 5 clases

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	41.05	41.05	268.06	41.05	45.58	45.58	222.82	45.58
ls	40.79	40.79	299.14	40.79	46.04	46.04	214.90	46.04
mb	42.12	42.12	257.29	42.12	42.38	42.38	285.56	42.38
rf	36.13	36.13	287.09	36.13	43.65	43.65	248.17	43.65

Cuadro A.5: Resultados en % del mejor modelo obtenido en los k-folds

	CountVectorizer				TF-IDF			
	acc	f1	mse	recall	acc	f1	mse	recall
lr	14.08	14.08	373.52	14.08	13.54	13.54	360.28	13.54
ls	12.44	12.44	384.74	12.44	12.70	12.70	376.00	12.70
mb	18.52	18.52	267.64	18.52	11.56	11.56	390.06	11.56
rf	21.38	21.38	257.68	21.38	15.26	15.26	334.78	15.26

Cuadro A.6: Resultados de la clasificación en el dominio de críticas de cine.

A.3 Trabajo futuro diseño web

Con la intención de mejorar la UX/UI de la aplicación web, se plantea un rediseño enfocado a la minimalización y simplificación de la misma.

Se plantea el diseño entorno a 3 puntos:

UX simple : Un usuario debería saber que puede hacer en la aplicación de un vistazo. Una interfaz poco intuitiva puede suponer la frustración y por lo tanto pérdida de muchos usuarios.

Adaptarnos a la imagen de empresa : Se deben seguir los estándares de diseño de la empresa cliente, véase: utilizar su logo, su paleta de colores, etc. Esto permitirá al cliente suavizar el impacto de la implantación de la herramienta en su ecosistema.

Enfocada en el producto : Enfocar la interfaz al contenido importante, el producto.

A.3.1 Página principal

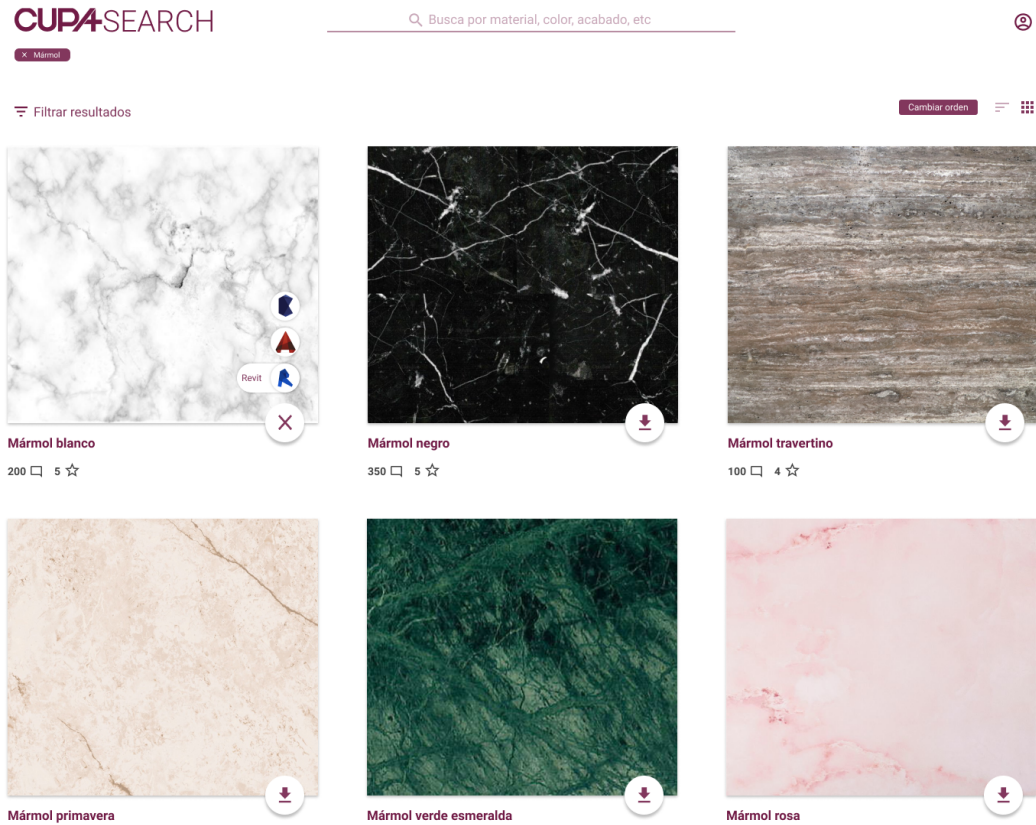


Figura A.3: Página principal.

Como vemos en la figura anterior hemos simplificado la interfaz y adoptado la paleta de colores de la empresa cliente.

A primera vista el usuario verá los artículos más destacados y toda la información relevante sobre los mimos, como número de comentarios, valoración y botón de descarga.

En este diseño asumimos el correcto funcionamiento del buscador, por lo que el usuario debería ser capaz de encontrar lo que desea entre los primeros 6 o 9 productos.

A.3.2 Ficha de producto

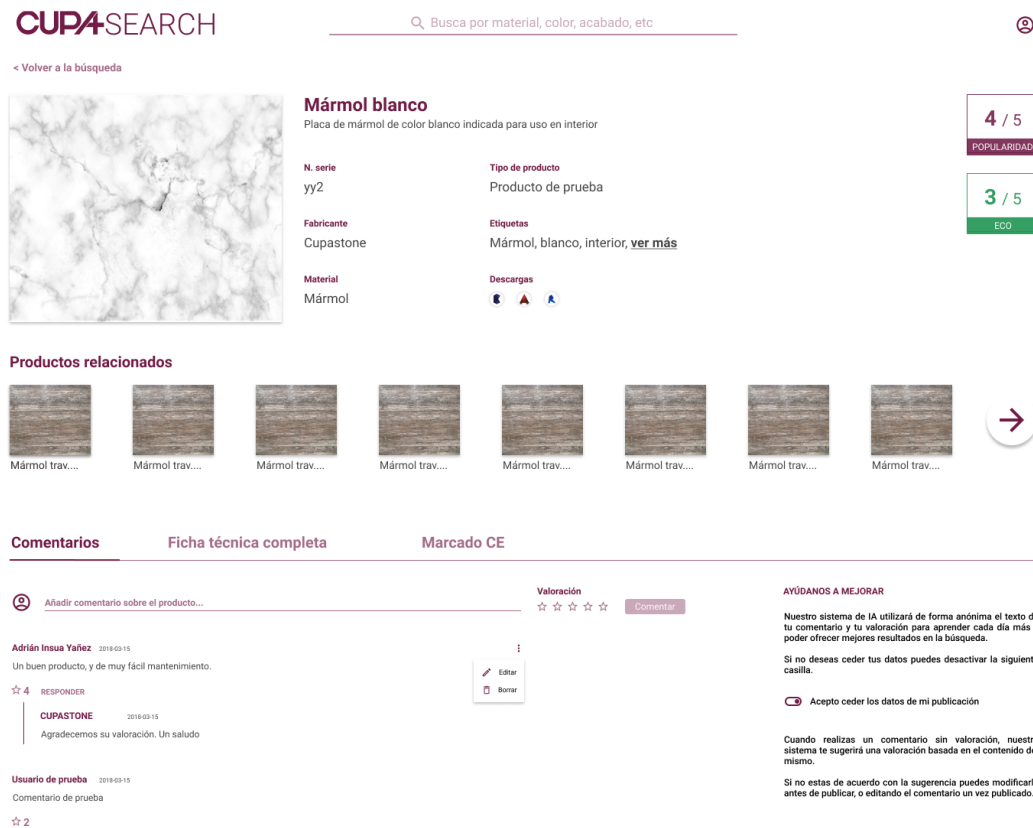


Figura A.4: Ficha de producto.

Para esta sección proponemos cambiar el modelo de ventana modal a una navegación completa, lo que nos permite introducir más información relevante sobre el artículo, sin que resulte molesto.

En esta ficha vemos además una sección de artículos relacionados, típica de las webs actuales, que se obtendrán a partir de las etiquetas del producto actual.

En la parte inferior se encuentra la sección de comentarios, y otras secciones secundarias. Hemos incluido el cuadro de inserción de comentarios y la nota sobre el uso de datos para mejorar el sistema de inteligencia artificial en la parte derecha de la sección de comentarios, evitando al usuario tener que dar pasos innecesarios.

Lista de acrónimos

MNB *Multinomial Naive Bayes*

LR *Logistic Regression*

SVM *Support Vector Machines*

RF *Random Forest.*

Glosario

Corpus Un corpus es un conjunto de documentos que se utilizarán como entrada para un sistema de análisis de sentimientos. Dichos documentos poseen una clasificación previa y generalmente realizada por humanos.

Lexicón de polaridad Vocabulario que relaciona una palabra con una polaridad determinada, basada en la teoría de las emociones básicas del ser humano propuesta por Plutchik ([17],[18],[19])

Para el trabajo se ha utilizado el diccionario término-emoción creado por Mohammad y Turney [20] a través del servicio Mechanical Turk de Amazon, sobre el que se han realizado algunas correcciones de errores derivadas de la traducción automática de los términos mediante el servicio de traducción de Google.

Feature Con este término nos referiremos a las características o propiedades de los textos que utilizaremos para predecir su polaridad mediante los distintos modelos de clasificación.

N-grama Según la definición de Wikipedia¹: *un n-grama es una subsecuencia de n elementos de una secuencia dada*. Para el estudio del lenguaje los n-gramas son composiciones generalmente formadas por fonemas, sílabas, letras o palabras. Los n-gramas se clasificarán según el valor de n:

- unigrama: n-gramas de 1 solo elemento.
- bigrama: n-grama de 2 elementos, también llamados digramas.
- trigramas: n-grama de 3 elementos.

Token Cadena de caracteres que forman un componente léxico básico del sistema para la realización de la clasificación.

Una sentencia está compuesta por una colección de tokens que corresponderán a las palabras que la forman.

¹<https://es.wikipedia.org/wiki/N-grama>

Frase: *“La clasificación de textos en español”*.

Tokens: [la, clasificación, de, textos, en, español]

Si nos ceñimos al modelo de n-gramas un token sería considerado un unigrama.

Stopwords Colección de palabras de uso común en un idioma, que dado que no aportan información relevante sobre los textos pueden ser eliminadas en un proceso previo de limpieza.

Comúnmente esta colección está formada por preposiciones, pronombres, verbos comunes, signos de puntuación, y otras palabras dependientes del dominio del problema.

Preprocesamiento Proceso de tratamiento previo de los datos que pretende mejorar la calidad de los mismos. Dado que se trata de un dominio no profesional es común encontrar faltas de ortografía cometidas por los usuarios, caracteres especiales del dominio (hash-tags o menciones de twitter, emoticonos, etc.) que no aportan información válida para la tarea, y necesitaremos deshacernos de ellos antes de utilizar los textos.

Bibliografía

- [1] S. Das and M. Chen, *Yahoo! for Amazon: Extracting market sentiment from stock message boards*, 2001, vol. 35.
- [2] R. M. Tong, *An operational system for detecting and tracking opinions in on-line discussion*, 2001, vol. 1, no. 6.
- [3] P. D. Turney, *Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews*, 2002.
- [4] B. Pang, L. Lee, and S. Vaithyanathan, *Thumbs up?: sentiment classification using machine learning techniques*, 2002.
- [5] K. Dave, S. Lawrence, and D. M. Pennock, *Mining the peanut gallery: Opinion extraction and semantic classification of product reviews*, 2003.
- [6] P. Gamallo, M. Garcia, and S. Fernández-Lanza, *TASS: A Naive-Bayes strategy for sentiment analysis on Spanish tweets*, 2013.
- [7] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, 2013.
- [8] G. Grefenstette, Y. Qu, J. G. Shanahan, and D. A. Evans, *Coupling niche browsers and affect analysis for an opinion mining application*, 2004.
- [9] M. Hu and B. Liu, *Mining and summarizing customer reviews*, 2004.
- [10] S. M. Jiménez-Zafra, M. Taulé, M. T. Martín-Valdivia, L. A. Ureña-López, and M. A. Martí, *SFU Review SP-NEG: a Spanish corpus annotated with negation for sentiment analysis. a typology of negation patterns*. Springer, 2018, vol. 52, no. 2.
- [11] M. Stone, *Cross-validatory choice and assessment of statistical predictions*. Wiley Online Library, 1974, vol. 36, no. 2.

-
- [12] J. H. Martin and D. Jurafsky, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/-Prentice Hall Upper Saddle River, 2009.
- [13] C. Cortes and V. Vapnik, *Support-vector networks*. Springer, 1995, vol. 20, no. 3.
- [14] I. M. Kloumann, C. M. Danforth, K. D. Harris, C. A. Bliss, and P. S. Dodds, *Positivity of the English language*. Public Library of Science, 2012, vol. 7, no. 1.
- [15] P. S. Dodds, E. M. Clark, S. Desu, M. R. Frank, A. J. Reagan, J. R. Williams, L. Mitchell, K. D. Harris, I. M. Kloumann, J. P. Bagrow *et al.*, *Human language reveals a universal positivity bias*. National Acad Sciences, 2015, vol. 112, no. 8.
- [16] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, 2010.
- [17] R. Plutchik, *The emotions*. University Press of America, 1991.
- [18] —, *A general psychoevolutionary theory of emotion*. Elsevier, 1980.
- [19] —, *On emotion: The chicken-and-egg problem revisited*. Springer, 1985, vol. 9, no. 2.
- [20] S. M. Mohammad and P. D. Turney, *Crowdsourcing a word–emotion association lexicon*. Wiley Online Library, 2013, vol. 29, no. 3.