Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Aplicación de Técnicas de Deep Learning para la Extracción de Términos en Dominios Específicos

Autora: Lucía Guasp Alburquerque
Tutores: Óscar Corcho y Mariano Rico

Madrid, Julio 2020

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Aplicación de Técnicas de Deep Learning para la Extracción de Términos en Dominios Específicos

Julio 2020

Autora: Lucía Guasp Alburquerque
Tutores: Óscar Corcho y Mariano Rico
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

En este Trabajo de Fin de Máster se aborda el problema de la extracción automática de términos en dominios específicos mediante algoritmos de aprendizaje automático supervisado, en particular, modelos basados en redes neuronales artificiales y aprendizaje profundo. Para resolver esta tarea, se han implementado dos modelos cuya estructura principal consiste en redes bidireccionales LSTM (*Long Short Term Memory*), puesto que permiten capturar el contexto y las dependencias de las palabras en ambas direcciones. Asimismo, se han empleado representaciones vectoriales de las palabras (*word embeddings*), en concreto, se ha usado el modelo de representación de Glove, que simboliza el significado de las palabras en base al contexto en el que se encuentran.

Los modelos se han entrenado con el dataset público INSPEC, sobre el cual también se ha realizado un proceso de validación, con la finalidad de verificar que son capaces de generalizar la extracción de términos sobre documentos distintos a aquellos empleados en el entrenamiento. Igualmente, el uso del dataset INSPEC etiquetado con una terminología ha permitido evaluar los resultados obtenidos mediante las métricas de precisión, *recall* y F1. Los rendimientos obtenidos con los modelos implementados en este trabajo han permitido verificar que se han alcanzado los resultados del estado del arte relativos a modelos basados en redes BiLSTM con representaciones vectoriales fijas, como Glove. Adicionalmente, uno de los modelos desarrollados en este trabajo ha superado el rendimiento de los modelos equivalentes de la literatura científica (basados en redes BiLSTM y representaciones fijas), hasta donde se tiene constancia. La mejora consiste en un aumento de la métrica F1 en torno a un 5 % y del *recall* alrededor de un 10 %, principalmente gracias al empleo de una arquitectura neuronal con una mayor profundidad.

Tras el proceso de implementación y evaluación, se han usado los modelos desarrollados para la extracción de terminología de corpus no anotados. La finalidad de la extracción de estos términos consiste en su posible futura incorporación a una herramienta de búsqueda llamada *KeyQ*, desarrollada en el *AI.nnovation Space* de la Universidad Politécnica de Madrid. Esta herramienta utiliza estos términos para hacer una búsqueda por palabra clave más eficaz que las búsquedas tradicionales. En concreto, se han empleado los modelos para la extracción de términos de un corpus formado por documentos técnicos de Airbus, así como para la extracción de la terminología de un corpus relativo a la COVID-19, compuesto por artículos científicos sobre esta enfermedad.

Abstract

This Final Master Project addresses the problem of automatic terms extraction in specific domains by supervised machine learning algorithms, in particular, models based on artificial neural networks and Deep Learning. To solve this task, two models have been implemented whose main structure consists of bidirectional LSTM networks (*Long Short Term Memory*), since they allow to capture the context and dependencies of words in both directions. Likewise, vector representations of words have been used (*word embeddings*), specifically, the representation model of Glove, which expresses the meaning of the words based on the context in which they are found.

The models have been trained with the public INSPEC dataset, on which the validation process has also been carried out, in order to verify that they are capable of generalizing the extraction on terms on different documents than those used in the training. Likewise, the use of the INSPEC dataset labeled with a terminology has allowed to evaluate the results obtained by means of the precision, recall, and F1 metrics. The performance obtained with the models implemented in this work has allowed to verify that the results of the state of the art regarding models based on BiLSTM networks with fixed vector representations, such as Glove, have been achieved. In addition, one of the models developed in this work has outperformed equivalent models in the scientific literature (based on BiLSTM networks and fixed word embeddings) as far as we know. The improvement consists in an increase of the F1 metric by about 5 % and of the *recall* by about 10 %, mainly thanks to the use of a neural architecture with a greater depth.

After the implementation and evaluation process, the models developed have been used for the terminology extraction of terms in unannotated corpus. The purpose of the extraction of these terms is their possible future incorporation to a search tool called *KeyQ*, developed in the *AI.nnovation Space* of the Polytechnic University of Madrid. This tool uses these terms to make a more efficient keyword search than traditional searches. In particular, the models have been used to extract terms from a corpus formed by Airbus technical documents, as well as to extract the terminology from a corpus related to COVID-19, which is comprised of scientific articles on this topic.

Tabla de contenidos

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En la actualidad, el empleo de la inteligencia artificial y, en concreto, el aprendizaje automático ha aumentado notablemente gracias a recientes avances tecnológicos. Los avances más destacables que han permitido optimizar los algoritmos de aprendizaje automático y profundo han sido, principalmente, la mejora de las prestaciones en términos computacionales de las máquinas, lo cual permite entrenar dichos algoritmos en un tiempo razonable, junto con la gran cantidad de datos disponibles, también necesarios para entrenar los modelos.

Uno de los campos más comunes en el aprendizaje automático es el procesamiento del lenguaje natural o PLN, el cual consiste en el análisis, la comprensión y la representación automática del lenguaje humano, permitiendo que las máquinas realicen un amplio rango de tareas relacionadas con el lenguaje natural en todos los niveles. Dentro de este campo, existe una gran cantidad de problemas que se pueden resolver automáticamente gracias al empleo de la inteligencia artificial, como la extracción de información, la traducción de textos entre distintos lenguajes, el resumen de documentos, la respuesta de preguntas y la clasificación y segmentación de documentos, entre otras.

La tarea desarrollada en este Trabajo de Fin de Máster se ha centrado en la resolución de uno de los problemas habituales del procesamiento del lenguaje natural mediante aprendizaje automático, en concreto, la extracción automática de terminología. Esta tarea consiste en la detección de las palabras o expresiones clave de un texto. Los términos, conocidos en inglés como *keyphrase* o *keywords*, pueden consistir en una o varias palabras, o incluso tratarse de una expresión o frase completa.

En la siguiente sección 1.1 se va a detallar el contexto y la motivación que han dado lugar al desarrollo del trabajo en torno a la extracción de terminologías en dominios específicos. A continuación, en la sección 1.2 se mencionará el planteamiento del trabajo que se va a llevar a cabo, así como la estructura de la memoria, describiendo el contenido de cada uno de los capítulos y secciones.

1.1. Motivación

Este trabajo se ha derivado del proyecto *KeyQ*, un proyecto desarrollado en el centro mixto *AI.nnovation Space* de Accenture y UPM, en el que he participado como becaria durante 6 meses. El objetivo de este proyecto consiste en crear una herramienta de búsqueda orientada a un dominio concreto, dando lugar a un sistema más especializado y preciso que un buscador genérico.

A la hora de realizar una búsqueda sobre un tema específico, el hecho de mostrar al usuario los términos relevantes o las palabras clave del tema en cuestión, permite que el usuario pueda realizar una búsqueda más precisa. De esta manera, a medida que se vayan introduciendo en la búsqueda los términos (simples o compuestos) que el usuario quiera buscar sobre un corpus[1], se mostrarán aquellas páginas relevantes del corpus que contienen dichos términos.

Por tanto, para el funcionamiento de esta herramienta se deben extraer los términos o palabras clave del dominio específico sobre el que se desarrolla la búsqueda. Tradicionalmente, la extracción de terminologías se llevaba a cabo por expertos del dominio, que seleccionaban manualmente aquellos términos que consideraban relevantes. Sin embargo, los grandes avances del procesamiento del lenguaje natural han permitido que la tarea de extracción de terminología se pueda realizar de manera automática mediante distintos algoritmos.

La importancia de incorporar una terminología relativa al corpus introducido en la herramienta de búsqueda desarrollada para el proyecto *KeyQ* ha dado lugar a la necesidad de investigar sobre los distintos métodos que existen para la extracción automática de palabras clave. En la literatura aparecen distintos enfoques para realizar esta tarea. Uno de los enfoques utilizados es el estadístico, basándose en la frecuencia de las palabras, métricas obtenidas en base a estadísticos o basándose en la posición relativa de las primeras ocurrencias. Otro enfoque importante es el lingüístico, empleando expresiones regulares, como los patrones *Part of the Speech* (POS), que determinan el análisis morfosintáctico de las palabras.

Asimismo, destaca el enfoque basado en aprendizaje automático, ya que, como se ha mencionado, se han obtenido grandes avances en el procesamiento del lenguaje mediante aprendizaje automático gracias a la potente era tecnológica actual.

Los modelos de aprendizaje automático se dividen en modelos supervisados y no supervisados. Los primeros emplean datos de los cuales se conoce la entrada y la salida que se quiere inferir, por lo que tratan de generar un modelo que establezca la correspondencia entre los datos de entrada y de salida proporcionados. Para poder llevar a cabo un modelo de extracción de términos basado en aprendizaje automático supervisado, se requieren datasets etiquetados formados tanto por los documentos en sí, como por su terminología correspondiente. Por otro lado, el aprendizaje no supervisado debe inducir los resultados disponiendo únicamente de datos de entrada, es decir, los documentos, por lo que su tarea consiste en encontrar patrones en dichos datos.

Este contexto ha dado lugar al desarrollo de este Trabajo de Fin de Máster, el cual tiene como finalidad principal estudiar y analizar los distintos modelos de aprendizaje automático supervisado para la extracción de terminología, para así implementar un

[1] Se denomina corpus a una colección o conjunto de documentos escritos.

modelo que obtenga terminologías dado un corpus de un dominio específico. Las terminologías extraídas a partir de los modelos creados en este trabajo podrán ser introducidas en la herramienta de búsqueda creada en el proyecto *KeyQ*.

Cabe destacar que, a pesar de que el proyecto se ha centrado en desarrollar el sistema para el dominio de los documentos técnicos de Airbus, por ser la finalidad del proyecto *KeyQ*, se ha implementado de manera que se pueda emplear de forma genérica para cualquier dominio del que se disponga de un corpus formado por documentos del tema específico, teniendo así una mayor versatilidad. En concreto, debido a la situación actual causada por la enfermedad COVID-19, la comunidad científica se encuentra investigando multitud de aspectos relacionados con el virus, lo cual ha dado lugar a una gran base de datos de artículos científicos que recogen información sobre la COVID-19. El hecho de disponer de una herramienta que recoja todos estos documentos y que permita realizar una búsqueda sobre todos ellos es de gran ayuda para la comunidad, al poder compartir el conocimiento de los distintos investigadores. Por ello, se ha aplicado el sistema *KeyQ* al corpus formado por los documentos sobre la COVID-19. De esta forma, tenemos un nuevo dominio al que aplicar las técnicas descritas en este trabajo para la extracción automática de su terminología.

1.2. Planteamiento del trabajo y estructura de la memoria

Tras explicar el contexto y la motivación que han dado lugar a este proyecto, en esta sección se realiza una introducción del planteamiento del trabajo que se va a desarrollar a lo largo del documento, indicando además el contenido que se va a mostrar en cada uno de los capítulos y secciones de la memoria.

El problema que aborda este trabajo consiste en desarrollar un modelo de aprendizaje automático supervisado que extraiga de manera automática la terminología (términos simples o compuestos) de distintos corpus. En concreto, se van a estudiar e implementar modelos basados en redes neuronales artificiales complejas, que pertenecen al campo del aprendizaje profundo o *Deep Learning*.

En primer lugar, el capítulo 2 expone los objetivos y la metodología. Este capítulo se divide en la sección 2.1, donde se detallan los objetivos que se pretenden alcanzar, tanto generales como específicos, y en la sección 2.2, donde se mencionará la metodología utilizada.

A continuación, se muestran en el capítulo 3 los conceptos básicos relativos a las redes de neuronas que serán necesarios en el desarrollo del trabajo. Se detallan tanto los conceptos necesarios para comprender una red neuronal (sección 3.1), como los principales tipos de redes neuronales existentes, en la sección 3.2. Además, se introducen conceptos relevantes de los modelos de aprendizaje automático (sección 3.3) y de aprendizaje profundo (sección 3.4).

En el capítulo 4 se analiza el estado del arte correspondiente al procesamiento del lenguaje natural, centrado principalmente en el aprendizaje automático supervisado con el objetivo de comprender cómo se ha afrontado este problema por los distintos autores. En concreto, en la sección 4.1 se introduce el contexto del procesamiento del lenguaje natural a lo largo de los años. En la sección 4.2 se detalla el estado del arte relativo al procesamiento del lenguaje mediante aprendizaje profundo, finalizando por las investigaciones centradas en la extracción de terminología, en la sección 4.3.

Tras haber realizado el estudio relativo al estado del arte del tema que se abarca en este proyecto, junto con los conceptos necesarios para llevarla a cabo, se menciona en el capítulo 5 la hipótesis de trabajo, detallando la definición del problema en la sección 5.1 y la propuesta de trabajo en la sección 5.2.

El capítulo 6 contiene los datos del desarrollo y la implementación llevados a cabo para alcanzar los objetivos propuestos en el trabajo. En la sección 6.1 se presentan los datos con los que se va a trabajar a lo largo del trabajo, tanto para el entrenamiento de los modelos, como los documentos sobre los que se pretende extraer la terminología, una vez definidos los modelos. La implementación de los modelos se detalla en la sección 6.2.

Posteriormente, se presentan los resultados obtenidos en el capítulo 7. En la sección 7.1 se introduce la forma de evaluar los rendimientos alcanzados. Se distinguen los resultados obtenidos sobre el dataset etiquetado, es decir, del cual se dispone de terminología previa para el entrenamiento del modelo, en la sección 7.2, de los resultados obtenidos sobre un corpus del que no se dispone de información inicial relativa a la terminología, como el de la COVID-19, en la sección 7.3.

En el capítulo 8 se exponen las conclusiones obtenidas tras la realización del trabajo (sección 8.1) y las líneas de trabajo futuro (sección 8.2).

En el último capítulo se listan todas las referencias citadas en este Trabajo de Fin de Máster.

Capítulo 2

Objetivos y metodología

Conviene destacar los objetivos que se pretenden alcanzar con este trabajo, distinguiendo el objetivo general de los específicos. Estos objetivos deben seguir las características SMART (específicos, medibles, alcanzables, relevantes y con un tiempo determinado) [1], con la finalidad de poder expresar claramente lo que se pretende conseguir, estableciendo medidas concretas para determinar si los objetivos han sido alcanzados o no. Además, se debe tener en cuenta el alcance que se puede lograr valorando limitaciones que puedan interferir en el desarrollo. Asimismo, en la última sección de este capítulo se detallará la metodología empleada para lograr los objetivos mencionados.

2.1. Objetivo general y objetivos específicos

El objetivo general de este trabajo consiste en desarrollar un modelo de extracción de terminología de manera automática que mejore la terminología empleada por la herramienta de búsqueda creada en el proyecto *KeyQ*.

La extracción de terminología consiste la selección de las palabras clave de un documento, para lo cual, con el enfoque tradicional, se necesitan terminólogos expertos que realicen esta tarea. El objetivo de la extracción automática de terminología pretende evitar tener que recurrir a estos expertos, reduciendo así el grado de dificultad, coste y tiempo del proceso. Para ello, se va a realizar la extracción mediante un modelo de aprendizaje automático supervisado, cuyas características se deben evaluar para verificar que se consigue el objetivo propuesto. Tras evaluar distintos modelos, se seleccionarán aquellos con mejor rendimiento, a partir de los cuales se pretende obtener las palabras clave de documentos que formen parte de la herramienta de búsqueda, permitiendo así que los términos sean mostrados a los usuarios para que puedan realizar una búsqueda más precisa.

Se van a analizar distintas medidas a lo largo del desarrollo del trabajo para evaluar el progreso en la consecución del objetivo. En cuanto al análisis del modelo neuronal desarrollado, se evaluará su rendimiento sobre un conjunto de datos etiquetado, del cual se dispone de una terminología con la que comparar aquella obtenida por el modelo. Además, una herramienta llamada “Terminologías interactivas”, creada para el proyecto *KeyQ* permitirá evaluar una terminología de la que no se tiene información previa en base a distintas métricas. Esta herramienta también posibilita que expertos

terminólogos interactúen con las terminologías para extraer conclusiones.

Este objetivo genérico da lugar a un conjunto de objetivos específicos que se deben analizar por separado, para poder evaluar su cumplimiento a medida que se desarrolle el trabajo.

- Identificar los modelos del estado del arte, destinados a la extracción de palabras clave o a otras tareas afines del procesamiento del lenguaje natural, que han permitido obtener buenos rendimientos.
- Explorar recursos empleados en el estado del arte que mejoran los modelos de aprendizaje automático en el procesamiento del lenguaje natural, como las representaciones vectoriales de las palabras o los modelos del lenguaje.
- Indagar los datasets existentes formados por documentos junto con su terminología asociada. Estos servirán para entrenar y evaluar los modelos desarrollados para la extracción de palabras clave de este trabajo. Se empleará aquel que tenga las características más adecuadas al problema planteado.
- Desarrollar varios modelos, basados en el estado del arte, así como modelos creados para este trabajo, para obtener distintos resultados frente a un mismo conjunto de documentos. Previamente se debe realizar un adecuado tratamiento de los datos adaptado a los modelos planteados.
- Comparar los resultados obtenidos con los distintos modelos sobre datasets etiquetados de los que se dispone, previamente, de una terminología seleccionada por expertos. Para ello, se emplearán métricas comunes en el aprendizaje automático de clasificación, como la precisión, el *recall* o la métrica F1.
- Comparar las terminologías extraídas automáticamente mediante los modelos creados sobre documentos de los que se desconoce información previa relacionada con la terminología. Para realizar esta comparativa, se utilizará la herramienta "Terminologías interactivas" creada en el proyecto *KeyQ*.
- Razonar y comprender los resultados obtenidos con las comparativas mencionadas anteriormente para extraer conclusiones a partir de ellas.

2.2. Metodología de trabajo

De cara a alcanzar los objetivos mencionados, se debe llevar a cabo un proceso organizado y estructurado. En esta sección, se introducen cada uno de los procesos o fases que componen la metodología del trabajo, la necesidad de estos procesos, así como las herramientas empleadas para llevarlos a cabo. A continuación, se van a definir las fases principales en las que se puede dividir el desarrollo de este trabajo.

2.2.1. Revisión bibliográfica

El primer procedimiento para desarrollar el trabajo consiste en el estudio del estado del arte. En concreto, se analizará el campo del procesamiento del lenguaje natural, indagando en los estudios realizados mediante modelos de aprendizaje automático o profundo. Asimismo, se revisarán los estudios sobre extracción de terminología. Se obtiene así una visión de los resultados obtenidos en el campo, además del conocimiento de los modelos que han permitido dichos rendimientos. De esta forma, se puede analizar la viabilidad del trabajo.

En esta revisión bibliográfica se ha consultado una gran variedad de fuentes formadas principalmente por artículos y libros científicos publicados en los últimos años. También se han consultado fuentes anteriores, con la finalidad de conocer los inicios y los primeros procedimientos en el tema.

Esta fase de análisis del estado del arte, pese a que ha sido más exhaustiva al iniciar el trabajo, se ha ido desarrollando a lo largo del proyecto para profundizar en distintos aspectos.

2.2.2. Entorno de trabajo

La implementación del trabajo se ha desarrollado principalmente en Google Colab, una herramienta que permite desarrollar y ejecutar cualquier código escrito en Python mediante los servidores de Google, facilitando así la explotación de los modelos de aprendizaje automático. Además, se ha utilizado Keras, la API de alto nivel de TensorFlow que permite desarrollar modelos de aprendizaje automático de manera sencilla.

Para desarrollar los modelos de extracción de terminología, se debe emplear una base de datos compuesta por documentos de los cual se disponga de su terminología asociada, ya que, al tratarse de modelos de aprendizaje automático supervisados, deben disponer de entradas y salidas que permitan entrenar a la red neuronal. Se debe realizar un análisis de los distintos datasets públicos existentes con el objetivo de seleccionar aquel que tenga unas características adecuadas para la finalidad del trabajo.

2.2.3. Selección e implementación de los modelos desarrollados

Existe una gran cantidad de modelos de aprendizaje automático supervisado y modelos neuronales desarrollados por la comunidad científica. No obstante, cada uno de ellos tiene unas características concretas que permiten tener un rendimiento mejor para unos problemas determinados.

Tras realizar la revisión bibliográfica inicial, se deben seleccionar los modelos o las arquitecturas que mejor se adaptan al problema planteado en este trabajo. Por tanto, se hará una selección inicial de las arquitecturas en base los conocimientos adquiridos tras el estudio del estado del arte. Estas arquitecturas se implementarán y se evaluarán los resultados para poder seleccionar entonces los modelos que mejor se ajusten a la tarea planteada.

2.2.4. Evaluación y comparación de los resultados

Por último, se debe evaluar el rendimiento obtenido por cada uno de los modelos implementados. En este trabajo, se debe analizar la terminología extraída por dichos modelos de manera automática. No obstante, la terminología tiene un carácter subjetivo, puesto que depende principalmente de la interpretación de los expertos del dominio.

En este trabajo se van a evaluar dos tipos de resultados. En primer lugar, se debe comprobar cómo de correcta es la terminología extraída sobre el dataset etiquetado empleado para el entrenamiento de los modelos. En este caso, se pueden emplear métricas que tengan en cuenta la cantidad de predicciones correctas e incorrectas, como la precisión, el *recall* y la métrica F1.

Para analizar y evaluar las terminologías extraídas por modelos automáticos, se ha desarrollado en el proyecto *KeyQ* el sistema denominado "Terminologías interactivas" que permite obtener distintas métricas, como la frecuencia relativa de los términos en los documentos, o el TF-IDF. Además, permitirá que terminólogos hagan modificaciones sobre los términos, obteniendo así información adicional sobre la bondad de las terminologías. Este sistema se utilizará para analizar los resultados obtenidos por los modelos, tanto sobre el dataset etiquetado, como sobre un corpus formado por documentos de los que no se dispone de terminología etiquetada con la que comparar las predicciones, como es el corpus de la COVID-19.

Capítulo 3

Conceptos básicos sobre redes neuronales artificiales

3.1. Introducción a las redes de neuronas

A lo largo de la historia, muchos diseños se han basado en la naturaleza para desarrollar nuevas ideas. Por tanto, a la hora de construir máquinas inteligentes, es lógico emplear la inspiración sobre la arquitectura de nuestro cerebro. Esta es la idea principal que dio lugar a las redes de neuronas artificiales, las cuales fueron introducidas en la literatura en 1943 por el neurocientífico Warren McCulloch y el matemático Walter Pitts [2], presentando un modelo computacional simplificado de cómo podrían trabajar las neuronas de manera conjunta para llevar a cabo complejos problemas usando la lógica proposicional. No obstante, a pesar de su temprana introducción, no se consiguió explotar su potencial debido a limitaciones tecnológicas.

Hasta la fecha, han surgido diferentes épocas de la Inteligencia Artificial, ligadas a la aparición de nuevas técnicas que facilitaran su implementación, como sucedió con el algoritmo de *Backpropagation*, el cual se explicará más adelante, en la sección 3.3. Actualmente, nos encontramos en una de estas épocas favorable para el desarrollo e implementación de algoritmos de Inteligencia Artificial, debido a la potente era tecnológica. Principalmente, esto ha ocurrido gracias a la enorme cantidad de datos disponibles para entrenar a las redes neuronales, junto con los avances de la potencia computacional, que permite entrenar a dichas redes en una cantidad de tiempo razonable.

Los modelos basados en redes neuronales artificiales tratan de encontrar la relación entre las señales de entrada y las de salida simulando la manera en la que las neuronas de nuestro cerebro interactúan. Estos modelos son versátiles, potentes y escalables, lo que los hace idóneos para resolver complejas tareas de aprendizaje automático.

La salida de cada neurona se obtiene como la suma de las entradas ponderas con unos pesos (w), enviada a través de una función de activación, como indica la figura 3.1.

Fuente: elaboración propia.

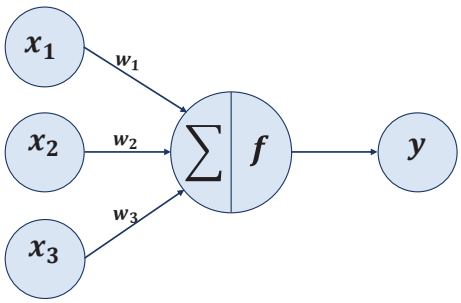


Figura 3.1: Diagrama descriptivo de una neurona artificial.

Matemáticamente, la salida se formula mediante la ecuación (3.1), donde $y(x)$ representa la salida de la neurona, $f()$ la función de activación, x_i la entrada i, y w_i el peso correspondiente.

$$y(x) = f(\sum_{i=1}^{n} w_i x_i) \tag{3.1}$$

La función de activación permite que las redes neuronales procesen la información creando modelos complejos no lineales. Su elección depende del diseño, pudiendo elegir entre diversas funciones, además de poder asignar diferentes funciones de activación a cada neuronas de la red. Las funciones de activación que se emplean habitualmente son las siguientes:

- Sigmoide o logística. Es una función diferenciable cuya salida está entre 0 y 1, por lo que se usa especialmente para predecir la probabilidad en una clasificación binaria. Se trata de una función monótona, aunque su derivada no lo es. La función softmax es una generalización de la función de activación logística, por lo que se usa para clasificación multiclase.

- Tangente hiperbólica. Otra función diferenciable, cuyo rango va de -1 a 1, por lo que también se emplea para clasificar entre dos clases. Asimismo, es una función monótona aunque su derivada no lo es. Una mejora con respecto a la función sigmoide es que sus valores están centrados en 0, lo cual mejora la eficiencia del aprendizaje de la red neuronal.

- ReLU (*Rectified Linear Units*). Es la función de activación más empleada. Su salida es nula para cualquier valor de entrada negativo, mientras que para valores positivos la salida es igual a la entrada, por lo que su rango va de 0 a infinito. En este caso, tanto la función como su derivada son monótonas. Esta función acelera la convergencia de la red, en concreto en la aplicación de *Stochastic Gradient Descent* (sección 3.3).

- Leaky ReLU. Se trata de una modificación de la anterior, para corregir el hecho de que los valores negativos se hacían inmediatamente nulos, lo cual deterioraba la capacidad del modelo para entrenar la red correctamente. Esta función incrementa el rango de la función ReLU, de menos infinito a infinito, al añadir una pequeña pendiente de 0.01 en la parte negativa de la función.

En conclusión, se tiende a utilizar la función de activación ReLU o Leaky ReLU para las capas ocultas de la red neuronal, puesto que mejora considerablemente la convergencia del modelo. El resto generan problemas de saturación (para valores absolutos altos, la pendiente de la función es nula, anulando prácticamente la propagación del gradiente e impidiendo el aprendizaje de la red), de convergencia lenta o de desvanecimiento del gradiente (el cual se explicará en el apartado 3.4.1). En la capa de salida se tiende a utilizar la función Softmax, en el caso de clasificación, o simplemente una función lineal, para problemas de regresión.

Fuente: elaboración propia.

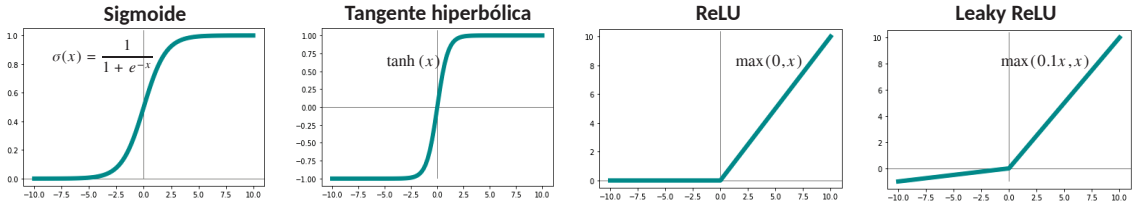


Figura 3.2: Gráficas de las funciones de activación.

Las redes neuronales, como su nombre indica, están formadas por un conjunto de neuronas formando una red, como se puede observar en la representación de la figura 3.3, donde cada una de las neuronas genera la salida mediante la ecuación (3.1). De esta forma, el conjunto de la red permite modelar modelos complejos, pudiendo variar sus características escogiendo la arquitectura de la red, el tipo de función de activación y el algoritmo de entrenamiento.

Fuente: elaboración propia.

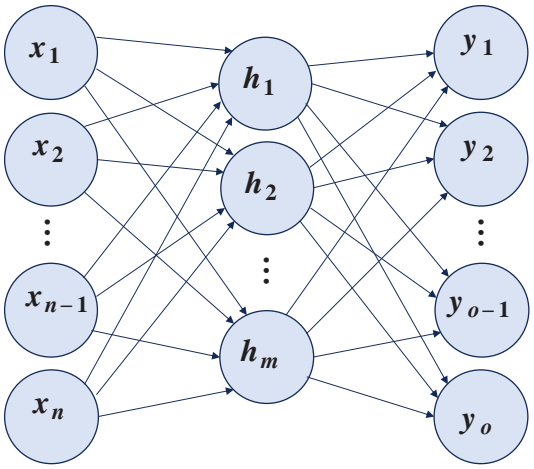


Figura 3.3: Representación de una red neuronal con una capa oculta.

La arquitectura o topología se refiere a la estructura de la red, la cual determina y condiciona el modo en el que la red aprende, así como la dificultad de los problemas que puede modelizar dicha red. La estructura depende de los siguientes aspectos:

- El número de capas de la red. Las redes neuronales forman una red mediante capas, distinguiendo entre la capa de entrada, la que recibe la información de los datos sin procesar, las capas ocultas o intermedias y la capa de salida, que genera las predicciones del modelo.
- El número de neuronas en cada capa. La capa de entrada estará formada por tantas entradas como variables de datos de entrada haya. Igualmente, el número de neuronas de la capa de salida dependerá del número de salidas. Sin embargo, no hay una cantidad determinada para las neuronas en las capas ocultas, sino que el diseño depende de muchos factores, como el número de variables de entrada, la cantidad de datos de entrenamiento o la dificultad del modelo, entre otros.
- La forma en la que se conectan las neuronas o la dirección de la información. Las redes neuronales pueden transmitir la información únicamente hacia las capas siguientes hasta alcanzar el nodo de salida, denominándose *feedforward networks*. Asimismo, se pueden diseñar redes que transmitan la información en ambos sentidos, permitiendo así aprender patrones más complicados o secuenciales. A estas redes se les conoce como redes neuronales recurrentes, las cuales se detallan en el apartado 3.2.3.

Esta flexibilidad a la hora de construir redes neuronales es también una desventaja, al haber muchos hiperparámetros que determinar. Se llaman hiperparámetros a aquellos parámetros del modelo que se deben definir a priori y que afectan al proceso de aprendizaje de los algoritmos y a su rendimiento posterior. Son hiperparámetros la tasa de aprendizaje o *learning rate*, el tamaño del lote o *batch size*, el número de épocas y la topología de la red (explicados en la sección 3.3). La elección de estos parámetros no es trivial y, a menudo, es necesario realizar procesos de optimización de los hiperparámetros para hallar los valores más idóneos. En la literatura hay diversos métodos para llevar a cabo esta optimización, destacando la optimización bayesiana, la búsqueda cartesiana, la búsqueda aleatoria, la optimización basada en gradiente o la optimización evolutiva.

3.2. Tipos principales de redes neuronales

Como se ha mencionado, la forma en la que se conectan las neuronas y la estructura de la red formada por las distintas neuronas modifican las características que definen el modelo creado con cada red neuronal.

Se denomina perceptrón, cuyo concepto fue creado en 1957 por Frank Rosenblarr [3], a la arquitectura de red neuronal más simple. Se basa en una única capa formada por una neurona artificial. A partir de esta estructura básica, surgen distintas modificaciones que permiten crear arquitecturas más completas, pudiendo así resolver una gran cantidad de tareas y problemas complejos.

3.2.1. Perceptrón multicapa

La arquitectura básica de una red neuronal compuesta por más de una capa se denomina Perceptrón multicapa (*Multilayer perceptron* en inglés, o MLP), que como su nombre indica, es un conjunto de neuronas simples situadas en distintas capas.

El número de capas ocultas de la estructura, es decir, las capas que están entre las capas de entrada y salida, determinarán el grado de complejidad de la arquitectura y, por tanto, la complejidad de las tareas que resuelve.

La estructura de un MLP presenta unas capas que están totalmente conectadas unas a otras, lo cual se denomina *fully connected layers*, ya que cada unidad en una capa está conectada a todas las unidades de la capa anterior. En este tipo de estructura, los parámetros de cada neurona son independientes del resto de unidades de la misma capa. Asimismo, se llaman *Feed Forward Neural Networks*, debido a que la información viaja en un único sentido, es decir, las salidas de las neuronas se obtienen computando únicamente la información que proviene de la capa anterior, sin ningún tipo de retroalimentación.

3.2.2. Redes de neuronas convolucionales

Se llama red neuronal convolucional (*Convolutional Neural Network* en inglés, o CNN) a cualquier red que tenga al menos una capa convolucional. La principal aplicación de estas redes es el procesado, clasificación y segmentación de imágenes, ya que permiten identificar características en la entrada, de manera similar a como lo hace el córtex visual del ojo humano, aunque también se usan para otros datos autocorrelacionados.

Las capas convolucionales se denominan así porque recogen la información a su entrada convolucionando [4], es decir, aplicando un filtro sobre su entrada, por ejemplo, una imagen. Las neuronas de estas capas no se conectan a todas los píxeles de la imagen de entrada (en el caso de la primera capa oculta), ni a todas las salidas de la capa anterior (el resto de capas ocultas convolucionales), sino que únicamente se conectan con aquellas que se encuentran dentro de un pequeño rectángulo de la imagen.

A la entrada de la capa no se le aplica un único filtro, sino varios, ya que cada uno de ellos se encarga de detectar un aspecto distinto. Tras aplicar todos los filtros, se obtiene un conjunto que se denomina *feature mapping*, ya que representa el conjunto de características de la imagen. Esta arquitectura permite a la red concentrarse en

las características de bajo nivel en la primera capa oculta (como esquinas, bordes, líneas...) y ensamblarlas en características de mayor nivel en las siguientes capas.

Estas operaciones generan una gran cantidad de parámetros, por lo que, en estas redes, se añaden unas capas denominadas *Pooling Layers*, con el objetivo de reducir la carga computacional, la memoria y el número de parámetros, muestreando sus entradas. Al igual que en las capas convolucionales, cada neurona se conecta a un número limitado de neuronas de la capa anterior. Su función consiste en agregar estas entradas mediante alguna métrica como la media o el máximo del conjunto.

Una estructura típica CNN está formada por diversas capas convolucionales, cada una de ellas seguida por una *pooling layer*. Así se consigue que la imagen se haga cada vez más pequeña, aunque más profunda, al tener cada vez más *feature maps*. Después de estas capas, se añade una o varias capas *fully connected*, que permitirán obtener la salida deseada. Se puede observar esta estructura típica en la figura 3.4.

Fuente: elaboración propia basada en [4].

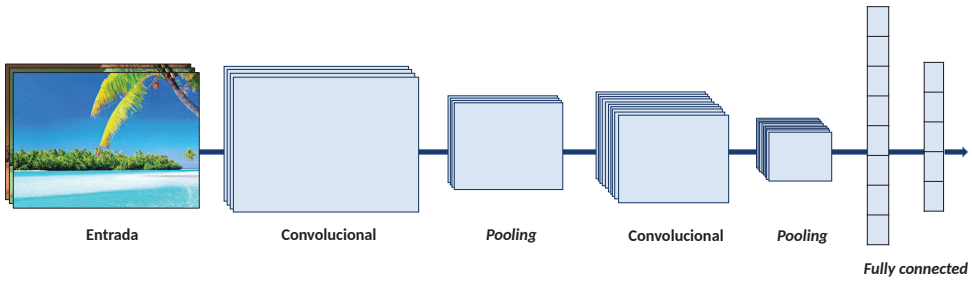


Figura 3.4: Esquema de la estructura típica de una red neuronal convolucional.

3.2.3. Redes de neuronas recurrentes

Las redes neuronales recurrentes (RNN por sus siglas en inglés) son aquellas redes en las que la información no fluye en un único sentido (de la entrada a la salida), sino que disponen de conexiones hacia atrás, permitiendo así una retroalimentación. Por tanto, en cada momento t cada neurona recurrente recibe la información de entrada $x_{(t)}$ además de una información adicional relacionada con su propia salida del momento anterior.

Las neuronas recurrentes a menudo se conocen como celdas de memoria, ya que contienen información de instantes de tiempo anteriores. La memoria viene dada por el estado de las celdas, denominado como $h_{(t)}$, el cual es una función de la entrada en ese momento y del estado del momento anterior: $h_{(t)} = f(h_{(t-1)}, x_{(t)})$.

Estas celdas pueden ser muy simples, como una neurona recurrente básica, o pueden tener arquitecturas más complejas. En una estructura básica, la salida de la celda es igual a su estado, lo cual se refleja en la ecuación (3.2), donde $\sigma(\cdot)$ es la función de activación, b el término de sesgo y w_x y w_h son los vectores de pesos para las entradas $x_{(t)}$ y para los estados del momento anterior $h_{(t-1)}$, respectivamente:

$$y_{(t)} = h_{(t)} = \sigma(x_{(t)}^T \cdot w_x + h_{(t-1)}^T \cdot w_h + b). \tag{3.2}$$

Las redes que se pueden construir siguiendo esta estructura son especialmente útiles a la hora de predecir datos relacionados con el tiempo, tales como series temporales, o

aquellos datos que dependen del orden de la secuencia de entrada, como las palabras en un texto, por lo que han obtenido resultados realmente buenos en el campo del procesamiento del lenguaje natural.

No obstante, un problema que surge es el hecho de que, cuando se tienen en cuenta largas secuencias de entrada, la memoria de las primeras va desapareciendo gradualmente conforme va atravesando múltiples celdas, ya que en cada momento de tiempo se pierde cierta información.

Para solucionar este problema, se han introducido en el estado del arte diferentes tipos de arquitecturas que han dado lugar a grandes mejoras. Una de las principales estructuras son las denominadas LSTM (*Long Short-Term Memory*) [5], que permiten obtener mejores rendimientos detectando dependencias a largo plazo.

Su arquitectura se puede observar en la figura 3.5, donde se observan dos estados diferentes: $h_{(t)}$, el cual actúa como memoria a corto plazo, y $c_{(t)}$, que se corresponde con la memoria a largo plazo. A partir de ambos estados, la celda determina qué almace-

Fuente: elaboración propia basada en [4].

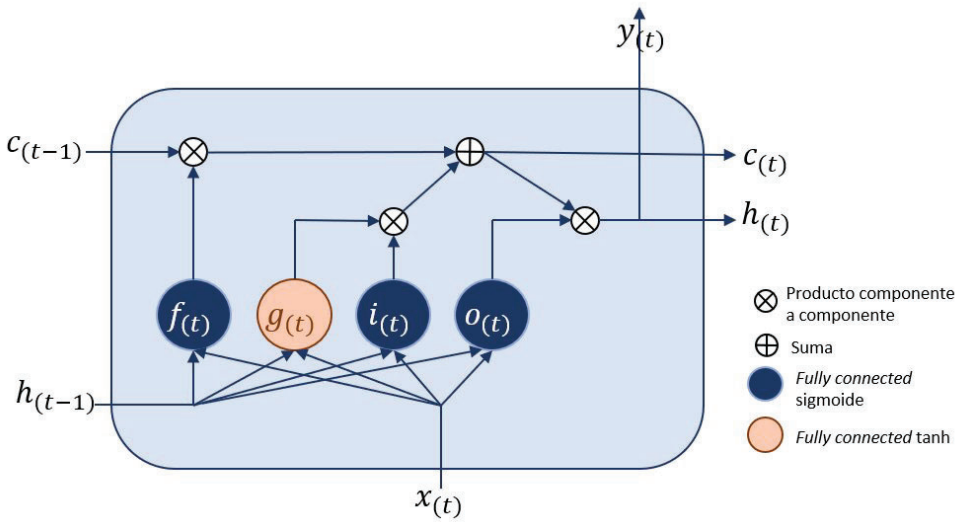


Figura 3.5: Representación de una celda LSTM.

nar y qué desechar en la memoria a largo plazo y qué sacar de ella, de la siguiente manera: En primer lugar, cuatro capas del tipo *fully connected* son alimentadas por el vector de entrada $x_{(t)}$ y el estado anterior a corto plazo $h_{(t-1)}$. La capa principal es la que obtiene $g_{(t)}$, que analiza la entrada y el estado anterior y almacena parcialmente su salida en el estado de memoria a largo plazo. Las otras tres capas, al tener como activación la función logística, permiten controlar las puertas que regulan qué información es necesaria.

- La capa $f_{(t)}$ (*forget gate*) controla qué parte del estado de memoria a largo plazo debería ser desechado.
- La capa $i_{(t)}$ (*input gate*) determina qué parte de $g_{(t)}$ se debe añadir a la memoria a largo plazo.
- La capa $o_{(t)}$ (*output gate*) establece qué parte de la memoria a largo plazo se

debería tener en cuenta para la salida $y_{(t)}$ y el estado de memoria a corto plazo $h_{(t)}$.

Este comportamiento se recoge en la ecuación (3.3), donde W_{xj} son las matrices de los pesos de cada una de las cuatro capas (subíndice j) de la conexión con la capa del vector de entrada $x_{(t)}$, y W_{hj} son las matrices de los pesos de las capas con el estado anterior a corto plazo $h_{(t-1)}$. El operador $\otimes$ indica el producto Hadamard entre dos matrices con las mismas dimensiones, en cuya matriz resultado el elemento i, j es el producto de los elementos i, j de las matrices de entrada.

$$\begin{aligned} i_{(t)} &= \sigma(W_{xi}^T \cdot x_{(t)} + W_{hi}^T \cdot h_{(t-1)} + b_i) \\ f_{(t)} &= \sigma(W_{xf}^T \cdot x_{(t)} + W_{hf}^T \cdot h_{(t-1)} + b_f) \\ o_{(t)} &= \sigma(W_{xo}^T \cdot x_{(t)} + W_{ho}^T \cdot h_{(t-1)} + b_o) \\ g_{(t)} &= tanh(W_{xg}^T \cdot x_{(t)} + W_{hg}^T \cdot h_{(t-1)} + b_g) \\ c_{(t)} &= f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \\ y_{(t)} &= h_{(t)} = o_{(t)} \otimes tanh(c_{(t)}) \end{aligned} \tag{3.3}$$

Las redes LSTM comunes aplican las operaciones indicadas en la ecuación (3.3) únicamente hacia adelante, por lo que una unidad LSTM solo tiene en cuenta las unidades anteriores a esta.

En general, no solo interesan las entradas anteriores, sino que en una secuencia conviene entender la relación con los elementos anteriores y posteriores. Para esto se emplean redes LSTM bidireccionales o BiLSTM, en las que se aplican las ecuaciones descritas en (3.3) en ambas direcciones obteniendo así dos vectores de estado $\overleftarrow{h}(t)$ y $\overrightarrow{h}(t)$, donde $\overleftarrow{h}(t)$ obtiene una representación de su entrada incorporando información de las unidades anteriores a ella y $\overrightarrow{h}(t)$ de las posteriores a ella. Concatenando estos dos estados se obtiene un nuevo estado con información de todo el contexto $\overleftrightarrow{h}(t)$, es decir, teniendo en cuenta tanto las unidades anteriores a una entrada, como las posteriores.

Estas arquitecturas se emplean habitualmente en aplicaciones como las series temporales o el procesamiento de lenguaje natural, puesto que las unidades que tratan mantienen una relación con su contexto. En el procesamiento del lenguaje estas unidades son las palabras, las cuales tienen un sentido global junto con todas las palabras que forman parte de una misma frase o varias frases. Por ello, este tipo de redes son las que se van a emplear en el desarrollo de este trabajo, en el capítulo 6.

3.3. Aprendizaje automático en las redes neuronales

Las redes de neuronas permiten aplicar la metodología de aprendizaje automático, también conocido como *machine learning*. El aprendizaje automático comprende aquellos algoritmos que permiten que los modelos aprendan automáticamente dados unos datos concretos. Se puede distinguir entre aprendizaje supervisado y aprendizaje no supervisado. En el primer caso, se emplean datos de los cuales se conoce la entrada y la salida que se quiere inferir, por lo que tratan de generar un modelo que establezca la correspondencia entre los datos de entrada y de salida proporcionados. Por otro lado, el aprendizaje no supervisado debe inducir los resultados disponiendo únicamente de datos de entrada, por lo que su tarea consiste en encontrar patrones en dichos datos.

En el caso de las redes neuronales artificiales, se usan algoritmos de aprendizaje automático para entrenar a la red, lo cual implica encontrar los valores óptimos de los pesos (w) y de los sesgos (b) de todas las neuronas que mejor describan los datos de entrada en función de la salida esperada. El algoritmo básico de aprendizaje supervisado que permite que la red realice el modelo adecuado para los datos específicos se denomina *backpropagation*.

El algoritmo compara la salida de la red con el valor real, es decir, calcula el error mediante la función de coste definida $C(x, b)$, la cual permite cuantificar cómo de buena es la aproximación generada por el modelo. Se suele utilizar el error cuadrático medio (*Mean Squared Error* o MSE), aunque existen muchas más formas de medir el error y que se pueden emplear como función de coste para el entrenamiento de la red. En el caso de que el problema a resolver sea de clasificación, existen funciones de coste que permiten clasificar cada salida una clase, en concreto, aquella que tenga una mayor probabilidad. Se suele emplear en estos casos las funciones *binary crossentropy*, para clasificación binaria, y *categorical crossentropy*, en clasificación multiclase. Por tanto, el procedimiento del aprendizaje de la red es equivalente a un problema de optimización donde se debe minimizar la función de coste.

Con el objetivo de determinar cómo se deben modificar los pesos, se utiliza la técnica del descenso del gradiente (*gradient descent*), al calcular el gradiente de la función del error con respecto a los pesos de la red neuronal. Es decir, el algoritmo busca aquella dirección de máxima pendiente descendiente en la función, con el objetivo de seguir dicha dirección hasta llegar al mínimo buscado. En el algoritmo se determina un hiperparámetro, conocido como tasa de aprendizaje (*learning rate*), el cual controla cómo se ajustan los pesos de la red con respecto al descenso del gradiente de la función de coste, es decir, la velocidad con la que se realiza el *gradient descent*. La regla que sigue el descenso del gradiente para actualizar los parámetros de la red viene dada por la ecuación (3.4), donde η es la tasa de aprendizaje.

$$\begin{aligned} w'_k &= w_k - \eta \frac{\partial C}{\partial w_k} \\ b'_l &= b_l - \eta \frac{\partial C}{\partial b_l} \end{aligned} \tag{3.4}$$

Un problema que surge con este algoritmo es que una actualización de los parámetros implica calcular el gradiente para cada uno de las instancias de entrenamiento, lo cual resulta muy costoso en entornos de *big data*, como con los que se trabaja en las redes neuronales. Para solucionarlo, se tiende a realizar el entrenamiento mediante una estimación del gradiente realizada sobre una muestra aleatoria de las instancias. Esta modificación del algoritmo se llama *stochastic gradient descent* (SGD), en el que se denomina como *batch* a los grupos de las instancias que se seleccionan para realizar el gradiente. En este caso, se entrena a la red de grupo en grupo, actualizando en cada uno de estos pasos los parámetros. Cuando se han empleado todas las instancias de entrenamiento, se dice que se ha terminado una época de entrenamiento (*training epoch*). Para completar el entrenamiento, se realiza una cantidad determinada de épocas.

El algoritmo *backpropagation* inicialmente recorre la red hacia adelante con los primeros valores de los parámetros aleatorios en lo que se conoce como *forward pass*. Entonces, calcula el error y lo propaga hacia atrás, es decir, desde la salida hasta la entrada, aplicando recursivamente la regla de la cadena, lo cual se denomina *backward pass*. De esta manera, propaga el gradiente modificando los pesos de la red para minimizar la función de coste.

Para crear cualquier modelo de aprendizaje automático, además del proceso de entrenamiento, se deben llevar a cabo los procesos de test y validación, con el objetivo de verificar que el modelo funcione correctamente y generalice al ser empleado con nuevos datos de entrada. Para ello, los datos etiquetados de los que se disponen se dividen en tres subconjuntos: entrenamiento, test y validación.

Como indican sus nombres, el conjunto de entrenamiento se emplea para entrenar la red con el algoritmo de *backpropagation* y el conjunto de test se usa para probar el funcionamiento del modelo sobre datos con los que no ha sido entrenado, evaluando así su funcionamiento y generalización mediante distintas métricas. El conjunto de validación permite evaluar el modelo a medida que se va entrenando, con el fin de establecer los valores de los hiperparámetros más adecuados para dicho modelo. Esta técnica permite distinguir de manera sencilla, por ejemplo, si el modelo presenta *overfitting* o *underfitting*. Generalmente, el ratio de división del dataset en los tres subconjuntos mencionados depende de diversos factores, como el tamaño del dataset, los tipos de datos o el modelo que se va a generar. No obstante, la proporción del conjunto de entrenamiento debe ser mucho mayor que los otros dos conjuntos, que suelen tener una proporción similar. Por ejemplo, se tiende a emplear un 80 % del dataset para el entrenamiento, un 10 % para el conjunto de test y otro 10 % para validación.

3.4. Aprendizaje profundo

El número de capas ocultas es uno de los hiperparámetros que se deben determinar a la hora de construir una red neuronal. Las redes neuronales profundas, también denominadas como *Deep Learnign* son básicamente redes de neuronas artificiales con varias capas ocultas, las cuales necesitan una gran cantidad de datos para ser entrenadas. Se trata de un tipo de aprendizaje automático, que a su vez pertenece al campo de la inteligencia artificial y se basa en la representación del problema mediante una jerarquía de conceptos o abstracciones, los cuales se obtienen mediante la distribución de las neuronas en distintas capas.

Aunque una red con una única capa oculta pueda resolver problemas muy complejos, se ha demostrado que las redes neuronales profundas tienen una eficiencia mucho mayor, puesto que pueden modelar funciones complejas usando exponencialmente menos neuronas que las redes con un reducido número de capas, haciéndolas mucho más rápidas de entrenar [4].

Adicionalmente, el uso de *Deep Learning* no solo permite que la red converja más rápido a una buena solución, sino que también mejora la capacidad para generalizar frente a nuevos datasets. Además, otra de las ventajas que tienen este tipo de arquitecturas es que permiten reusar partes de otras redes pre-entrenadas en el estado del arte que desarrollen una tarea similar. De esta manera, el entrenamiento de la red será más rápida y necesitará una menor cantidad de datos.

No obstante, desarrollar este tipo de modelos implica algunas complicaciones. En primer lugar, diseñar una red neuronal es una tarea complicada, debido a la gran variedad de valores de hiperparámetros que se deben seleccionar. A la hora de crear la arquitectura de la red, una buena táctica puede ser comenzar con una arquitectura de tamaño reducido (pocas capas ocultas y con pocas neuronas) e ir aumentándola, produciendo mejores resultados, hasta que se produzca *overfitting*, concepto que se explicará a continuación. Otra forma de abarcar el problema del diseño de la red consiste en crear una red de un gran tamaño y evitar que produzca sobreajuste (también denominado *overfitting*) mediante alguna técnica de regularización (detalladas en el apartado 3.4.4).

Otros problemas de las redes profundas aparecen en el entrenamiento, como el desvanecimiento del gradiente (apartado 3.4.1), los largos tiempos de entrenamiento, o el problema del sobreajuste (apartado 3.4.3), los cuales se desarrollan en los apartados indicados.

3.4.1. Problema de desvanecimiento del gradiente

El algoritmo de *backpropagation* que se emplea para entrenar la red funciona propagando el gradiente del error de la capa de salida hacia la de entrada. En esta propagación, los gradientes se pueden ir haciendo cada vez más pequeños al haber muchas capas, dando lugar a que los pesos de las capas más bajas no se modifiquen prácticamente y, por tanto, el entrenamiento no converja a una buena solución. A este problema se le denomina desvanecimiento del gradiente, aunque también puede ocurrir lo contrario, es decir, que los gradientes se hagan cada vez más grandes, a lo que se denomina explosión del gradiente. Por tanto, en las redes neuronales profundas se deben tratar estos gradientes inestables para evitar que las capas aprendan o modifiquen sus parámetros con velocidades distintas entre ellas.

Una manera de solucionar estos problemas es la combinación del uso de la función de activación logística sigmoide, junto con la inicialización aleatoria de los parámetros, con una distribución normal de media nula y desviación estándar la unidad [6]. Esta estrategia de inicialización es conocida como *Xavier initialization*, por el nombre del autor, y permite que la varianza de las salidas de las capas sea igual a la varianza de sus entradas, dando lugar a un adecuado entrenamiento de la red y acelerando su convergencia. Asimismo, a esta estrategia de inicialización, en el caso de que se use la función de activación ReLU, se le llama *He initialization*. La inicialización de los términos de sesgo b no es tan crítica como la de los pesos, por lo que se suele inicializarlos a 0.

Asimismo, para conseguir que el problema de desvanecimiento de los gradientes no aparezca en todo el entrenamiento, se usa la técnica *Batch Normalization* [7]. Consiste en normalizar las entradas de cada capa y, a continuación, escalar el resultado, con el objetivo de la entrada de las unidades de activación en todas las capas siga una distribución normal estándar. Para realizar la normalización, calcula la media y la desviación típica de un lote (*training batch*). Al realizar el escalado, se añaden dos nuevos parámetros que son aprendidos durante el entrenamiento de la misma manera que lo hacen los pesos. Con esta técnica se consigue mejorar la estabilidad de la red neuronal e incrementar la velocidad de entrenamiento, al poder usar tasas de aprendizaje (*learning rates*) mayores. Igualmente, permite reducir el sobreajuste de manera similar a las técnicas de regularización, las cuales se desarrollarán a continuación.

Por otro lado, para evitar el problema de que los gradientes se vayan haciendo más grandes, simplemente se suelen recortar dichos gradientes para que nunca alcancen un determinado valor, técnica que se denomina *Gradient Clipping*.

3.4.2. Optimización avanzada

Tal y como se mencionó en el subapartado 3.3, el entrenamiento de una red neuronal es prácticamente un problema de optimización, para lo cual se suele utilizar *stochastic gradient descent*.

A pesar de que este algoritmo tiene un buen funcionamiento general, en determinados casos da lugar a problemas importantes, por ejemplo al tratar con mínimos locales o puntos de silla. Una variante de SGD para resolver dichas situaciones consiste en añadir *momentum*, el cual incorpora un vector de velocidad. Este vector trata de acelerar la convergencia de la red mediante el incremento de velocidad en el descenso del gradiente en las direcciones correctas. Otra modificación sobre esta técnica es el llamado *Nesterov Momentum*, mejorando las propiedades de convergencia.

Por otra parte, el método conocido como AdaGrad (*Adaptive Gradient Algorithm*) modifica la tasa de aprendizaje de manera independiente para cada parámetro proporcionalmente a su historial de actualización [8]. El objetivo de esta idea es que a lo largo del entrenamiento, es decir, a medida que estemos más cerca del mínimo, los pasos que se dan en la actualización de los parámetros serán más pequeños para obtener una mayor precisión. No obstante, esta técnica da lugar a una reducción de la velocidad que puede ser excesiva.

El método de optimización RMSProp (*Root Mean Square Propagation*) soluciona este problema al evitar que el entrenamiento se pare en algún momento debido a la reducción excesiva de la velocidad de aprendizaje. Asimismo, el optimizador Adam (*Adaptative Moment Estimation*) combina las técnicas de RMSProp y Momentum, lo cual permite trabajar de manera eficiente con datasets ruidosos o dispersos. En general, se tiene a utilizar el método Adam, ya que tiende a obtener buenos resultados [9].

Otro aspecto clave en la optimización del entrenamiento de la red es la selección de la tasa de aprendizaje. La dificultad de su elección es debida a que se debe tener en cuenta que consiga converger a la solución óptima y el ritmo con el que lo hace. Una técnica para establecer esta tasa de forma más eficiente consiste en usar *learning rate decay*, que consiste en iniciar el entrenamiento con una tasa mayor, consiguiendo que la función de coste se reduzca rápido, y poco a poco se va decrementando la tasa, para tener una mayor precisión en la búsqueda de la solución [8].

3.4.3. Overfitting y underfitting

Se considera que un modelo de *machine learning* o *deep learning* tiene un buen rendimiento cuando generaliza y predice modelos en tests independientes al entrenamiento. En estos modelos, se debe prestar especial atención a los errores de predicción, en concreto, se debe controlar la varianza y el sesgo (también conocido como *bias*).

La varianza aparece por la incapacidad de aprender todos los parámetros del modelo de manera robusta estadísticamente, especialmente cuando los datos son limitados o el número de parámetros es elevado. Por otro lado, el sesgo es el error causado por simplificaciones en el modelo [8].

En el diseño de los modelos de *machine learning* o *deep learning*, hay dos problemas habituales que se deben tener en cuenta, el *overfitting* y el *underfitting*, causados debido a grandes valores de varianza o sesgo.

- *Overfitting* o sobreajuste: ocurre cuando la varianza es muy elevada, por lo que el algoritmo es muy sensible a los datos de entrenamiento. Da lugar a un modelo que se adapta tanto a los datos de entrenamiento, que no generaliza bien y, por tanto, no realiza predicciones correctas.
- *Underfitting*: se trata del caso contrario, en el cual el modelo no consigue adaptarse a los datos, ni generar un patrón, realizando así predicciones erróneas. Ocurre cuando tiene un alto sesgo, ya que esto causa que el algoritmo pierda relación importante entre las variables de entrada y las de salida.

Por tanto, al entrenar un modelo se pretende encontrar un balance entre la varianza y el sesgo, para lo cual se deben establecer los valores de los hiperparámetros de una manera óptima.

3.4.4. Técnicas de regularización

Tal y como se ha comentado anteriormente, una ventaja y a la vez un inconveniente de las redes neuronales es su flexibilidad, debido a la gran cantidad de parámetros que se deben establecer, lo cual puede desembocar en un modelo sobreajustado al conjunto de entrenamiento y que, por tanto, no generalice bien ante nuevos datos. A continuación, se van a mencionar las técnicas más habituales para evitar el *overfitting* en *Deep Learning* [4].

- *Early Stopping*: Esta técnica consiste en parar el entrenamiento en cuanto el error de validación alcance un mínimo, ya que es en dicho punto cuando el modelo empieza a sobreajustarse a los datos de entrenamiento.
- L_1 *and* L_2 *Regularization*: Otra manera de reducir los grados de libertad del modelo en el entrenamiento es limitar los pesos de las conexiones de las neuronas. Para ello, se añade un término de regularización a la función de coste. Este término de penalización se puede obtener añadiendo el cuadrado de los parámetros en la función de coste, favoreciendo así que los pesos tengan valores absolutos pequeños ($\alpha \sum_{i=1}^{n} \theta_i^2$, dando lugar a L_2 Regularization) o añadiendo el valor absoluto de los pesos ($\alpha \sum_{i=1}^{n} |\theta_i|$, obteniendo L_1 Regularization), lo cual hace que los parámetros sean dispersos. El hiperparámetro α permite controlar la regularización del modelo. En el caso de redes neuronales, la técnica que da mejores resultados es L_2.
- *Dropout*: En cada paso del entrenamiento, cada neurona (excepto las de salida) pueden ser obviadas con una probabilidad p, es decir, no se tendrán en cuenta en dicho paso. Esta probabilidad es un hiperparámetro llamado *dropout rate*, el cual se suele establecer entre un 25 % y un 50 %. Se trata de la técnica más habitual, ya que mejora notablemente el modelo al forzar a aprender nuevas relaciones entre neuronas, impidiendo así memorizar resultados o lo que es lo mismo, sobreajustarse a los datos. El único inconveniente es la reducción en la velocidad de convergencia. Es importante destacar que el *dropout* solo se realiza en el entrenamiento.
- *Max-Norm Regularization*: Esta técnica limita los pesos de las conexiones a tener una norma vectorial menor a un determinado valor, tal que $||w||_2 \leq r$, donde r es un hiperparámetro a determinar y $|| \cdot ||_2$ es la norma L_2.

Capítulo 4

Estado del arte

En este capítulo se expone el estado del arte que permite obtener una base de conocimiento sólida para desarrollar este Trabajo de Fin de Máster.

En primer lugar, en la sección 4.1 se introduce la historia del procesamiento del lenguaje natural, al ser el campo dentro del cual se encuentra el problema de la extracción de términos. Se introducen los conceptos y los modelos implementados a lo largo de los años en la literatura científica, hasta alcanzar los avances llevados a cabo en los últimos años. Se hace hincapié sobre estos últimos avances en la sección 4.2, donde se analiza los modelos y algoritmos creados mediante técnicas de aprendizaje automático y profundo en el campo del procesamiento del lenguaje natural, ya que muchos de estos modelos serán empleados o mencionados a lo largo del documento.

Por último, en la sección 4.3 se detalla el estado del arte relativo a la tarea de la extracción automática de términos, profundizando así en el tema objeto de este trabajo.

4.1. Historia del procesamiento del lenguaje natural

El Procesamiento del Lenguaje Natural (PLN por sus siglas en castellano, NLP en inglés), también conocido como computación lingüística, es un área que implica distintos campos como la informática, la ingeniería, la lingüística, la psicología o la ciencia cognitiva. En términos generales, el objetivo de este campo consiste en el análisis, la comprensión y la representación automática del lenguaje humano, permitiendo que las máquinas realicen un amplio rango de tareas relacionadas con el lenguaje natural en todos los niveles.

Las áreas principales que abarca este campo son el modelado del lenguaje y el análisis morfológico, sintáctico y semántico. En cuanto a las áreas de aplicación, se incluyen temas como la extracción de información, la traducción entre distintos lenguajes, el resumen de textos, la respuesta de preguntas y la clasificación y segmentación de documentos, entre otras.

La computación lingüística tiene sus orígenes prácticamente desde el momento en el que apareció el ordenador. En concreto, en 1936 surge el concepto de autómata en un estudio [10], que dio lugar a la definición de los autómatas finitos y las expresiones regulares. Asimismo, en esa misma época Shannon aplicó las cadenas de Markov a los autómatas para el lenguaje en el artículo *A mathematical theory of communication*

[11]. En esa misma línea, Chomsky consideró las máquinas de estados finitos como una manera de definir el lenguaje [12]. Todos estos conceptos dieron lugar a la teoría del lenguaje formal, los cuales definían el lenguaje como una secuencia de símbolos. Por otro lado, durante esta época también se emplearon algoritmos probabilísticos para el procesamiento del lenguaje.

Entre las décadas de 1950 y 1960, los avances del PLN se desarrollan en un paradigma simbólico, el cual se basa en la inteligencia artificial, dando lugar a los primeros sistemas de comprensión del lenguaje natural [13]. Siguiendo esta tendencia surgieron sistemas básicos creados para desarrollarse en un dominio concreto mediante búsquedas de patrones.

En las siguientes décadas de 1970 y 1980, aparece una gran cantidad de investigaciones en este campo. Se presentan distintos enfoques, como el estocástico, donde se desarrollaron algoritmos de reconocimiento de voz gracias a modelos ocultos de Markov y el teorema de Shannon. Otro enfoque se basa en la lógica mediante sistemas basados en reglas. Por otro lado, la comprensión del lenguaje natural se centra en la semántica usando una representación del conocimiento en forma de redes semánticas [14].

Durante la siguiente década, los modelos de estados finitos resurgieron en el campo de la morfología y la sintaxis. Asimismo, se emplearon los modelos probabilísticos en el procesamiento del lenguaje, específicamente en el etiquetado morfosintáctico (*POS tagging*) y en el estudio de la semántica. En los años 90, los algoritmos de análisis, de etiquetado morfosintáctico, resolución de referencias y procesamiento del discurso continuaron basándose en el empleo de las probabilidades.

Por último, el auge del aprendizaje automático en torno al año 2000 permitió resolver problemas más complicados en el campo del procesamiento del lenguaje natural. En concreto, inicialmente se emplearon para este campo las máquinas de vectores de soporte (SVM), la regresión logística y los modelos bayesianos [15].

A partir de 2010, los grandes avances en la tecnología dieron lugar a la aplicación de las redes neuronales y el *Deep Learning* para tareas del procesamiento del lenguaje natural. Mediante la aplicación de estas técnicas se han conseguido algunos de los mejores resultados del estado del arte, por lo que han sido escogidos para el desarrollo de este trabajo. Debido a ello, en el apartado siguiente se detallan las técnicas que han permitido obtener los grandes avances mencionados.

4.2. *Deep Learning* en el procesamiento de lenguaje natural

Las redes neuronales han permitido obtener grandes mejoras en el procesamiento del lenguaje natural en los últimos años. A pesar de que el aprendizaje automático avanzó enormemente en torno al año 2000 en el desarrollo de distintos campos, el procesamiento del lenguaje natural ha sido uno de los que más ha tardado en obtener buenos resultados.

Inicialmente, para resolver tareas de PLN se empleaban redes neuronales convolucionales, pero no conseguían relacionar el contexto de manera adecuada, es decir, no captaban el significado de una palabra en una frase, ya que las denominadas *feed forward networks* solo tienen acceso a la entrada actual, sin contar con ningún tipo de memoria. Para conseguir representar el contexto y mantener información relativa a entradas anteriores, se aplican redes neuronales recurrentes (explicadas en el apartado 3.2.3), las cuales permiten que las neuronas se retroalimenten, aceptando así entradas secuenciales. Es por ello por lo que los resultados mejoran al emplear redes con memoria a largo plazo, como las LSTM, detalladas también en el apartado 3.2.3.

Al emplear la arquitectura LSTM, la red neuronal retiene la información que considera relevante, tanto a largo como a corto plazo, descartando la información irrelevante. Asimismo, en los problemas relacionados con el procesamiento del lenguaje, también se deben considerar las dependencias con las palabras que suceden a la actual, por lo que es beneficioso tener en cuenta las secuencias o frases en ambas direcciones. Esto se puede llevar a cabo mediante dos capas de LSTM, cada una observando las dependencias en una dirección, y combinar sus salidas, dando lugar a una estructura denominada redes LSTM bidireccionales o BiLSTM. Un inconveniente de estas redes es que son muy sensibles a los valores de los hiperparámetros cuando tratan con textos largos.

Una de las tareas más importante del PLN es el modelado del lenguaje natural. Se trata de crear un modelo que permita que, dadas unas frases o palabras previas, prediga las siguientes palabras. La potencia de los denominados Modelos de Lenguaje (LM por sus siglas en inglés), explicados en detalle en el apartado 4.2.2, radica en el hecho de que pueden capturar implícitamente las relaciones sintácticas y semánticas entre las palabras. Por ello, no solo se usan en la tarea de predicción de palabras, sino que se ha demostrado que se pueden emplear como modelos pre-entrenados. Esto quiere decir que los pesos obtenidos en dicho modelo se pueden usar como inicialización de las redes LSTM, cuyo tarea a resolver es diferente a la del modelo del lenguaje, obteniendo de esta forma una mejora en el entrenamiento y en la generalización del modelo final [16]. También se ha demostrado que el empleo de datos sin etiquetar y, por tanto, en las fases de pre-entreno, consiguen una mayor generalización en el posterior entrenamiento del modelo supervisado [17]. Este hecho adquiere gran relevancia cuando hay limitaciones en la obtención de datos.

Por otro lado, un mecanismo que ha tenido un gran impacto en las redes neuronales para el procesamiento del lenguaje ha sido el denominado *attention*, introducido por Bahdanau et al. [18]. Esta técnica consiste en dar unos determinados valores o pesos a aquellas palabras que se deben tener en cuenta con mayor atención dentro de una frase. De esta manera, la red aprende a prestar atención a determinadas palabras en función del estado oculto de la RNN y de sus pesos. En el estado del arte han surgido modificaciones a este mecanismo, donde destaca el denominado *self attention* [19], el

cual implica prestar atención a determinadas palabras dentro de la misma frase.

Otro aspecto que se debe mencionar en el procesamiento del lenguaje mediante técnicas de *Deep Learning* es la manera en la que se representa el lenguaje para ser tratado por la red. Tradicionalmente, el texto se ha representado computacionalmente mediante representaciones discretas, lo que también se conoce como *one hot encoding*. Se trata de vectores cuya longitud es igual a la cantidad de palabras del vocabulario con el que se vaya a trabajar. Para representar una palabra determinada, se indica con un 1 aquel elemento del vector que se corresponde con la palabra deseada en el vocabulario, siendo el resto de posiciones 0. Esta técnica usa el concepto de *bag of words*, donde se asigna el número de veces que aparece cada palabra en el vector del vocabulario. Se trata de una representación vectorial que no tiene en cuenta el orden de las palabras ni su contexto, por lo que pierde una gran cantidad de información sobre el texto tratado.

Con el objetivo de obtener una representación de las palabras que indiquen su información semántica y la relación entre otras palabras, surgen los conocidos *word embeddings* o representaciones vectoriales de las palabras, los cuales se detallan en el apartado siguiente.

4.2.1. *Word embeddings*

Los modelos denominados *word embeddings* representan las palabras mediante vectores dimensionales, también llamados *word vectors*, donde la dirección del vector de cada palabra guarda relación con el resto. De esta manera, se pueden identificar relaciones entre palabras, sinónimos, antónimos e incluso realizar operaciones con palabras. Un ejemplo clásico es el siguiente: partiendo de la representación de la palabra "reina", se resta la representación de "mujer" y se añade el vector de la palabra "hombre", en cuyo caso el resultado sería el vector que representa la palabra "rey", como se puede ver en la siguiente representación:

$$\alpha(\text{“}queen\text{”}) - \alpha(\text{“}woman\text{”}) + \alpha(\text{“}man\text{”}) = \alpha(\text{“}king\text{”}), \tag{4.1}$$

donde α representa la codificación vectorial de la palabra introducida como argumento. Similarmente, la figura 4.1 permite observar las representaciones de los *word embeddings*, así como la aplicación de operaciones matemáticas sobre ellas, en un espacio vectorial de dos dimensiones. Esta representación es una simplificación, ya que en la realidad, los *word embeddings* tienen muchas más dimensiones.

Fuente: elaboración propia basada en [20].

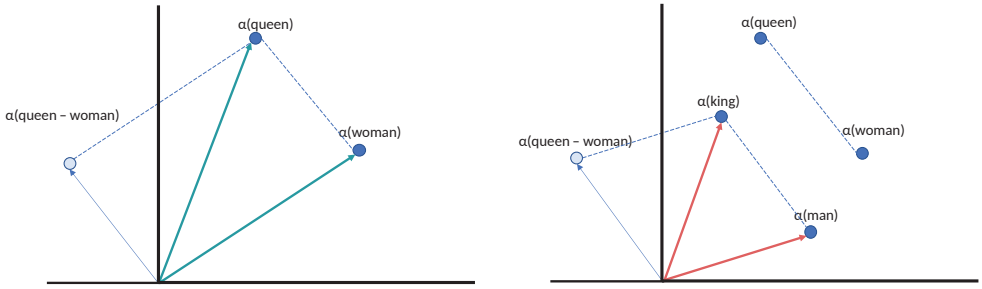


Figura 4.1: Representación de *word embeddings*.

En el año 2013 se consiguieron grandes avances en el campo del procesamiento de lenguaje natural mediante estas representaciones, gracias al desarrollo de los modelos *Word2Vec* y *Glove*, por lo que conviene detallar su metodología. La idea principal de ambos modelos consiste en que el significado de una palabra está determinado por las palabras que aparecen frecuentemente junto a ella, por lo que se extrae el significado de las palabras mediante el contexto en el que se encuentran.

En la práctica, se tiende a emplear vectores de palabras ya entrenados y disponibles en internet, conocidos como *pre-trained vectors*, que se emplean directamente en nuestro propio modelo, evitando así la necesidad de entrenar dichos vectores, para lo cual se necesitaría un corpus con unas dimensiones muy grandes. La manera de introducir estos vectores pre-entrenados consiste en añadir en el modelo de red neuronal una capa inicial, denominada *embedding layer*, la cual se encarga de mapear las palabras introducidas en la red con los vectores pre-entrenados.

4.2.1.1. Word2vec

Word2vec [21] es un modelo creado por el equipo de Google que devuelve *word embeddings* para representar las palabras de una manera adecuada y eficiente. El algoritmo se emplea sobre un corpus grande, sobre el cual se obtiene el vocabulario de palabras con su representación en forma de vector.

Word2vec entrena las representaciones de las palabras teniendo en cuenta otras palabras próximas en el corpus de entrada, para lo que recorre cada palabra del documento observando su contexto. Este enfoque se denomina *window-based* puesto que el contexto se representa como una ventana que va recorriendo el texto del corpus. Empleando similitudes de los vectores calculados hasta el momento, obtiene la probabilidad del contexto dada la palabra bajo estudio, a partir de lo cual se ajustan los vectores para maximizar dicha probabilidad. Se obtiene así, para todas las palabras, una verosimilitud en base a las probabilidades de que una palabra esté en el contexto de otra. Al maximizar esta verosimilitud se consigue que, a partir de una palabra, se puedan predecir las palabras del contexto.

Para ello, el modelo se puede realizar con dos arquitecturas distintas, *Continuous Bag of Words* (CBOW) o *Skip-Gram* [21]. La metodología de estas arquitecturas convierten la tarea no supervisada de representación vectorial de las palabras en un modelo de entrenamiento supervisado. En el sistema CBOW, la palabra actual se predice mediante su contexto, es decir, dadas las palabras anteriores y posteriores. Por otro lado, *Skip-Gram* predice la frase (el contexto) a partir de la palabra actual o central del contexto. Esta última arquitectura produce resultados más precisos sobre palabras menos frecuentes, siempre que se disponga de grandes datasets para el entrenamiento. Por otro lado, CBOW es más rápido y proporciona una frecuencia mejor para las palabras más comunes.

El modelo que permite generar la representación vectorial de las palabras consiste en una red neuronal superficial formada por una única capa oculta. Con respecto a la función de activación de la capa de salida, además de emplear la clásica función Softmax, se suele utilizar *Hierarchical softmax*, reduciendo así la complejidad de la capa. Asimismo, es muy habitual usar *negative sampling* en vez del descenso del gradiente o SGD. Se trata de un método para muestrear los datos de entrenamiento, el cual consigue mejorar la velocidad de entrenamiento eligiendo el contexto aleatoriamente.

4.2.1.2. Glove

Glove, cuyo nombre proviene de *Global Vectors*, es el modelo de representación vectorial propuesto por el grupo de PLN de Standford [20]. La aportación de este modelo frente a Word2vec reside en el hecho de que Glove no depende únicamente en estadísticas locales, es decir, la información de las palabras del contexto, sino que incorpora estadísticas globales, mediante la co-ocurrencia de palabras en el corpus, para obtener los vectores. Se trata, por tanto, de un enfoque híbrido entre los métodos basados en una ventana del contexto (*window-based*), como Word2Vec, y los métodos basados en la factorización matricial.

Esta técnica se obtiene al calcular una matriz, llamada matriz de co-ocurrencia, con unas dimensiones de $V \times V$, donde V es la cantidad de palabras del vocabulario, con el objetivo de registrar la cantidad de veces en las que cada palabra se ha relacionado con el resto. De esta manera, se obtiene la probabilidad de que cada palabra se encuentre en el contexto del resto.

El modelo de Glove no utiliza la matriz para generar los *embedings*, sino que utiliza un modelo de aprendizaje automático, el cual se optimiza utilizando el descenso del gradiente sobre una función de coste. No obstante, a diferencia del modelo de Word2Vec, en vez de predecir el contexto alrededor de cada palabra mediante redes neuronales, Glove trata de minimizar una novedosa función de coste que se obtiene mediante la diferencia del producto de los *word embeddings* y el logaritmo de la probabilidad de co-ocurrencia. Se trata, por tanto, de un modelo logarítmico-bilineal.

4.2.2. Modelos de lenguaje y transferencia de aprendizaje

Un modelo de lenguaje es una de las aplicaciones del Procesamiento del Lenguaje Natural más comunes. Su objetivo consiste en estimar la distribución de probabilidad de unidades lingüísticas, como palabras o frases. De esta manera, se consigue construir un modelo que, dada una entrada formada por varias de estas unidades, predice la siguiente. Por ejemplo, predecir la siguiente palabra o frase al introducir un texto donde cada entrada es una palabra o una frase, respectivamente. Para ello, el modelo no solo debe expresar la sintaxis correctamente, sino también la semántica de la frase.

Se ha comprobado que los modelos de lenguaje permiten desarrollar muchas facetas del lenguaje, como dependencias a largo plazo y relaciones jerárquicas [22]. Se trata de modelos entrenados de manera no supervisada, donde las muestras de entrenamiento consisten en multitud de textos. El modelo aprende a determinar la probabilidad de cada una de las palabras, dada la secuencia anterior de palabras, de manera que es capaz de generalizar para palabras o frases diferentes a las del entrenamiento. De esta manera, se obtiene un modelo que predice dichas unidades lingüísticas.

Inicialmente, no se había demostrado que estos modelos permitieran realizar tareas mas allá de las cuales se le había entrenado, es decir, la predicción de palabras o frases. No obstante, uno de los últimos avances del PLN ha permitido emplear estos modelos de lenguaje para optimizar diversas tareas. Se trata de la transferencia de aprendizaje (*transfer learning*), una técnica en la que, en vez de entrenar un modelo para una aplicación objetivo de PLN desde cero, se reusa un modelo pre-entrenado, en concreto, un modelo de lenguaje.

Esta transferencia de aprendizaje es similar al empleo de *word embeddings* para representar las palabras introduciendo una primera capa de neuronas denominada *embedding layer*, en el sentido de que ambas transferencias de lenguaje implican una inicialización optimizada de algunos pesos iniciales de la red neuronal. En un modelo inicializado únicamente con *word embeddings*, el aprendizaje se transfiere solo a la primera capa del modelo, teniendo que entrenar al resto de la red desde cero, lo cual requiere enormes datasets y un alto coste computacional para conseguir buenos resultados.

Sin embargo, el empleo del modelado del lenguaje durante el pre-entrenamiento ha permitido obtener los últimos avances en el procesamiento del lenguaje, al conseguir representar no solo características de uso de las palabras, como la semántica, sino también la forma de empleo de las palabras en el contexto lingüístico. En el estado del arte se ha demostrado empíricamente que los modelos de lenguaje se pueden usar como modelos pre-entrenados para resolver tareas más complejas como clasificación de textos, traducción automática o inferencia del lenguaje natural, entre otras. Esta capacidad de realizar tareas para las que no se le ha entrenado se llama *Zero-shot learning* [23].

Por tanto, se tiene a denominar a aquellos modelos que únicamente representan las palabras en su forma vectorial teniendo en cuenta la semántica como *fixed word embeddings*. Por otro lado, los modelos pre-entrenados que emplean modelo de lenguaje y obtienen relaciones más complejas de las palabras se conocen como *contextual word embeddings*, debido a que tienen en cuenta el contexto de las palabras. Un ejemplo de la mejora de los *contextual word embeddings* respecto a los anteriores es la distinción de palabras polisémicas. Estos modelos complejos permiten distinguir a qué significado se refiere una palabra según el contexto en el que se encuentre.

El modelo de lenguaje es la base para entrenar a otro modelo supervisado que realice una tarea más específica, como la traducción automática, por ejemplo. Este concepto se denomina *fine tuning*, el cual surgió en el campo de visión por computador, con *ImageNet*. Los investigadores de este campo comprobaron que los pesos aprendidos en los modelos entrenados con *ImageNet* podían ser utilizados para inicializar otros modelos similares, mejorando significativamente el rendimiento, respecto a realizarlo sin dicha inicialización. Esta técnica se puede emplear en el PLN, donde, al incluir más datos no etiquetados en el entrenamiento de los modelos de lenguaje, se consigue que el modelo supervisado requiera una cantidad menor de datos, ahorrando así en coste computacional, además de evitar la dificultad de recopilar grandes datasets etiquetados. Los últimos avances en el estado del arte del Procesamiento del Lenguaje Natural han ocurrido en torno a diversos modelos de lenguaje que permiten realizar la transferencia de lenguaje optimizando modelos de *Deep Learning* para tareas específicas. A continuación, se resumen algunos de estos modelos.

- *Semi-supervised Sequence Learning*: fue de los primeros modelos de lenguaje que permitieron llevar a cabo el *fine tuning* en tareas del procesamiento del lenguaje. El equipo de Google desarrolló este modelo para mejorar el aprendizaje de redes neuronales recurrentes [16] mediante dos perspectivas diferentes. La primera consiste en un modelo de lenguaje convencional, cuyo objetivo es predecir predecir la palabra siguiente a una frase, mientras que la segunda aproximación emplea una estructura denominada *sequence autoencoder*, la cual transforma una secuencia dada en un vector, a partir de cual se vuelve a reconstruir la

secuencia de entrada.

Utiliza estos algoritmos como modelo de aprendizaje no supervisado cuyos pesos pre-entrenados sirven para la inicialización de un modelo supervisado, por lo que se trata de una perspectiva de aprendizaje semi-supervisado.

Demuestra que al pre-entrenar redes LSTM con *sequence autoencoders*, se mejora el entrenamiento y la generalización de los modelos, en concreto, se realizaron los experimentos para clasificación de textos y análisis de sentimiento.

- *Transformer*: un equipo de investigación de Google desarrolló esta estructura de red neuronal simple, basada únicamente en el mecanismo denominado como *attention*, sin tener que emplear ni redes recurrentes ni convolucionales [19]. Estos mecanismos permiten modelar las dependencias entre las palabras independientemente de su distancia en las secuencias de entrada o salida.

  El modelo *Transformer* tiene una estructura que permite hallar las dependencias globales entre la entrada y la salida gracias a la combinación de la técnica de *self-attention* con una arquitectura tipo *encoder-decoder*. En esta arquitectura, el codificador (*encoder*) convierte las secuencias de palabras de entrada en un vector numérico, a partir del cual el decodificador (*decoder*) trata de generar de nuevo la secuencia de palabras. La diferencia con los clásicos *encoder-decoder* consiste en que en *Transformer*, tanto en el codificador como en el decodificador se emplean capas que incluyen el mecanismo *self-attention*. En la práctica, este modelo permitió obtener los mejores resultados hasta la fecha en tareas de traducción automática.

- ELMo (*Embeddings from Language Models*): este modelo pre-entrenado surge a partir de los modelos de representación vectorial de las palabras, añadiendo la capacidad de representar estos vectores teniendo en cuenta, no solo la palabra en sí, sino el contexto en el que se encuentra, denominando este concepto como *contextualized word-embeddings*. La representación vectorial de las palabras es generada a partir de una combinación lineal del estado interno de un modelo de lenguaje profundo, formado por LSTM bidireccionales, el cual es pre-entrenado mediante un corpus formado por múltiples textos [24].

  En la práctica, se demostró que al añadir las representaciones ELMo en modelo existentes para resolver seis tareas diferentes de procesamiento del lenguaje natural.

- ULM-FiT (*Universal Language Model Fine-tuning*): introduce un modelo de lenguaje y un proceso para llevar a cabo el "fine-tuning" de manera más eficiente para que el modelo de lenguaje realice varias tareas.

  La estructura del modelo pre-entrenado es similar a la que se emplea en el campo de visión por computador con el conocido ImageNet: utilizan la misma estructura con tres capas LSTM y los mismos hiperparámetros [25].

  Los experimentos sobre este método demostraron mejores resultados en seis tareas diferentes de clasificación de textos, reduciendo el error en torno al 21 %.

- GPT (*Generative Pre-training*): este modelo fue creado por Open AI con el objetivo de conseguir una representación universal que se pueda transferir a tareas específicas con los menores cambios posibles [26].

La estructura del modelo emplea una parte del modelo de *Transformer*, en concreto, un decodificador *Transformer* multicapa, lo cual le permite tratar las dependencias a largo plazo mejor que empleando redes recurrentes.

Durante la transferencia del aprendizaje, emplea adaptaciones específicas en la entrada para cada tarea objetivo, permitiendo así llevar a cabo el *fine-tuning* de manera eficiente con mínimos cambios en la estructura del modelo pre-entrenado.

GPT fue evaluado en cuatro tipos de tareas específicas (inferencia del lenguaje, respuesta de preguntas, similitud semántica y clasificación de textos), obteniendo mejores resultados que las arquitecturas diseñadas específicamente para dichas aplicaciones, en 9 de las 12 tareas estudiadas.

- BERT (*Bidirectional Encoder Representations from Transformers*): este modelo de representación de lenguaje creado por Google AI Language fue diseñado para pre-entrenar representaciones bidireccionales de texto no etiquetado teniendo en cuenta el contexto en ambas direcciones [27].

  Los modelos de lenguaje desarrollados hasta el momento tenían la limitación de ser unidireccionales, lo cual empeora las capacidades para llevar a cabo tareas de PLN, en las que es crucial incorporar el contexto en ambas direcciones. Para ello, BERT emplea un modelo de lenguaje enmascarado. Este tipo de modelos realiza un enmascarado en determinados tokens[1] de entrada de manera aleatoria, con el objetivo de predecir la identificación del vocabulario original de la palabra enmascarada basándose únicamente en su contexto. La estructura creada para este fin es un codificador Transformer multicapa bidireccional.

  Una de las grandes ventajas es que el modelo pre-entrenado BERT puede ser afinado (o lo que se conoce como *fine-tuned*) con una única capa de salida adicional. Asimismo, demostró avances en el estado del arte de once tareas de procesamiento del lenguaje natural.

  A partir de BERT, han surgido otros modelos basados en él con el objetivo de tratar dominios específicos. Concretamente, el modelo de SciBERT [28] emplea el modelo pre-entrenado de BERT y afina los *embeddings* empleando publicaciones científicas que incluyen un 18 % de artículos del dominio *computer science* y un 82 % del dominio biomédico. Especificando más el dominio, se creó BioBERT [29], un modelo de representación del lenguaje biomédico.

- RoBERTa: el equipo de Facebook AI replicó el modelo de BERT con algunas variaciones con el objetivo de mejorar su rendimiento. Los autores consideraban que BERT no explotaba al máximo su entrenamiento [30], por lo que emplearon un nuevo dataset más grande (CC-News), además de un número mayor de iteraciones. Asimismo, eliminaron el objetivo del entrenamiento del modelo original, referido a la predicción de secuencias posteriores. De esta manera, RoBERTa obtuvo resultados del estado del arte en diferentes datasets.

- GPT-2: el equipo de Open AI desarrolló una segunda versión de su modelo GPT. Este modelo ampliado cuenta con 1500 millones de parámetros que son entrenados mediante un dataset de 40 gigas de páginas web llamado WebText [31].

[1] Los tokens son las unidades de lenguaje mínimas por las que está formado un corpus.

Emplea una arquitectura basada en Transformer de manera similar al modelo anterior. No obstante, añade algunas modificaciones en este nuevo modelo, como una capa adicional de normalización tras el último bloque de *self-attention*. Además, incrementa el tamaño del vocabulario y la cantidad de tokens que se tienen en cuenta para el contexto.

Este modelo adquiere una gran capacidad en la transferencia del aprendizaje desde cero, mejorando su eficiencia y desempeño al realizar múltiples tareas. Los experimentos demostraron conseguir resultados del estado del arte en cuanto a modelado de lenguaje, en 7 de los 8 datasets empleados.

- XLNet [32]: se trata de un modelo autorregrsivo pre-entrenado que permite representar el contexto de manera bidireccional maximizando la probabilidad esperada en todas las permutaciones del orden de factorización. Este modelo se basa en BERT, tratando de superar sus limitaciones mediante la formulación autorregresiva. Además, el modelo emplea ideas de Transformer-XL [33]. Con su implementación, la arquitectura de XLNet consigue mejorar los resultados obtenidos por BERT en distintas tareas como inferencia del lenguaje natural, análisis de sentimiento o respuesta automática de preguntas.
- ERNIE (*Enhanced Representation through Knowledge Integration*): este modelo inspirado en la técnica de enmascarado de BERT, está diseñado para entrenar y aprender una representación del lenguaje mejorada al incluir en enmascarado a nivel de frase o de entidad [34]. De esta manera, no se enmascara por palabras de entrada, sino que puede tener en cuenta unidades compuestas por varias palabras, que tengan un significado en su conjunto.

  De esta forma, el conocimiento que aportan las frases y entidades es aprendido implícitamente durante el entrenamiento. Se consigue así que el modelo aprenda las dependencias semánticas a largo plazo, la relación entre entidades y sus propiedades, obteniendo una robusta representación de las palabras. Asimismo, esta técnica da lugar a un modelo con mejor generalización y adaptabilidad.

  La arquitectura empleada para desarrollar ERNIE es un codificador compuesto por una red multicapa tipo Transformer. Los resultados experimentales mejoraron los resultados obtenidos con modelos anteriores en distintas tareas de procesamiento del lenguaje natural chino.

4.3. Extracción de terminología

La extracción de términos es una técnica de análisis de textos que consiste en la extracción automática de las palabras o expresiones clave de un texto. Los términos, conocidos en inglés como *keyphrase* o *keywords*, pueden consistir en uno o varios tokens, o incluso tratarse de una expresión o frase completa.

La extracción de terminología es una de las tareas del procesamiento del lenguaje natural que se han tratado de resolver automáticamente mediante distintas técnicas o procedimientos, debido a su gran utilidad a la hora de resumir el contenido de un texto [35], analizarlo, indexarlo [36], reconocer sus temas principales o categorizarlos [37]. El hecho de que se tratar de una extracción implica que los términos seleccionado deben estar presentes en el documento.

Esta tarea está relacionada con otra tarea de PLN, en concreto, con el reconocimiento y la clasificación de entidades nombradas. No obstante, la extracción de términos implica más dificultades que la identificación de, por ejemplo, nombres de persona, puesto que la terminología varía significativamente de un dominio a otro, tanto en estructura morfosintáctica, como en significado.

A pesar de la utilidad de la terminología de un documento, únicamente una pequeña minoría de los documentos disponibles en la web tienen disponibles su términos asociados. Por tanto, disponer de una herramienta que permita extraer dicha terminología automáticamente es de gran utilidad.

A lo largo de la literatura y desde hace más de 20 años [38] [36] se ha tratado de llevar a cabo la tarea de extracción de terminología mediante diferentes técnicas, desde enfoques estadísticos simples que detectan términos contando la frecuencia de las palabras, hasta técnicas avanzadas que permiten crear modelos complejos [39].

Un enfoque muy empleado y sencillo es el estadístico [40], basándose en la frecuencia de las palabras, métricas como la puntuación tf-idf [41] o la posición relativa de las primeras ocurrencias. Por otro lado, se han usado enfoques lingüísticos, determinando la clasificación morfosintáctica de las palabras mediante etiquetado POS, mediante análisis de grafos o análisis sintáctico [42].

El enfoque lingüístico tiene en cuenta información lingüística, morfológica o sintáctica, sobre los textos y las palabras que estos contienen [43]. A menudo se emplean expresiones regulares, marcadores del discurso o información semántica para determinar los términos que se deberían extraer dentro de cada texto.

Otra perspectiva empleada para la extracción de palabras clave se basa en grafos [44], al representar las palabras como los vértices de un grafo que representa el documento. Las uniones de estos vértices pueden indicar distintas métricas sobre la relación de las palabras que permiten medir la importancia cada vértice o palabra.

En los últimos años se han empleado modelos más complejos para este fin, basados en diferentes algoritmos de aprendizaje automático con el objetivo de entrenar modelos para la selección de las palabras más relevantes de cada texto. En los siguientes apartados se detalla en dos de los enfoques más empleados para la extracción de términos: el enfoque basado en expresiones regulares y el enfoque del aprendizaje automático. Se hace hincapié en esta última perspectiva ya que se trata del enfoque empleado en este trabajo para la extracción automática de términos.

4.3.1. Extracción de términos a través de expresiones regulares

El enfoque de la extracción de terminología basado en expresiones regulares o patrones es uno de los más empleados en la literatura. Una de las ventajas de estos modelos consiste en que no se requiere de una terminología asociada al dataset para entrenar los algoritmos, como ocurre en el caso del aprendizaje automático supervisado.

Los modelos basado en expresiones regulares utilizan patrones subyacentes a la estructura sintáctica de un texto con el objetivo de seleccionar aquellos términos que simbolizan la información de un texto, teniendo en cuenta características morfosintácticas, principalmente.

Para llevar a cabo estos modelos, se debe realizar inicialmente el etiquetado gramatical de todas las palabras, también conocido como desambiguación léxica o etiquetado POS (*Part Of Speech Tagging*). Las distintas clases en las que se puede etiquetar las palabras cambian dependiendo del sistema utilizado.

Dentro de los modelos basados en expresiones regulares, se pueden distinguir distintos sistemas de etiquetado, como por ejemplo, el sistema EAGLE [45], el cual aporta bastante información mediante el etiquetado, incluso información morfológica de las palabras. Otro sistema ampliamente utilizado es el Universal POS tag (UPOS) [46], el cual incluye información relativa a la morfología y la sintaxis. Se trata de un etiquetado más simple, pero que se emplea de manera estándar en distintos idiomas.

Los sistemas de etiquetado deben resolver distintas dificultades, como la distinción de la clase de una palabra en función de su contexto, puesto que las palabras pueden pertenecer a clases distintas según su significado en la oración. A menudo, esta desambiguación de las palabras se soluciona al tener disponibles documentos cuyas palabras han sido clasificadas manualmente por expertos, o mediante reglas basadas en la categoría de las palabras vecinas.

Los modelos de extracción de terminología basados en expresiones regulares realizan la tarea de selección de las palabras clave del corpus, tras el etiquetado de todas las palabras, mediante una máquina de estados. Esta contiene unas reglas que, junto con el uso de una expresión regular, permite identificar las secuencias de etiquetas de los documentos que satisfacen dicha expresión regular, consiguiendo así seleccionar los términos del corpus.

Este tipo de modelos se ha empleado en la literatura principalmente por lingüistas, los cuales conocen las estructuras morfológicas y sintácticas que identifican los términos, en cada uno de los idiomas.

Uno de los sistemas creados siguiendo esta tendencia [47] se representa en la siguiente ecuación (4.2), donde A es un adjetivo, pero no un determinante, N es un sustantivo y P es una preposición.

$$((A|N)^{+}|((A|N)^{*}(NP)^{?})(A|N)^{*})N \tag{4.2}$$

Una de las características de este sistema consiste en que no se requiere de expertos para extraer la terminología dado un corpus. El modelo ha sido una gran aportación al campo del procesamiento del lenguaje natural, por lo cual, se mejoró el sistema dando lugar a una expresión reducida, la cual se indica en la ecuación (4.3), donde D es un determinante, y el resto de representaciones se mantienen de la ecuación (4.2).

$$(A|N)^*N(P + D^*(A|N)^*N)^* \tag{4.3}$$

Uno de los inconvenientes de este sistema consiste en que se basa en patrones desarrollados en base a la estructura morfosintáctica de los términos, lo cual varía en función de la lengua. Por ello, el modelo descrito [47] obtiene buenos rendimientos sobre corpus en inglés. Dada esta limitación, surgieron nuevos modelos con expresiones regulares basadas en distintos idiomas, como, por ejemplo, el castellano [48]. En este artículo se adapta la expresión regular a este idioma, definida mediante la ecuación (4.4).

$$N(A|N)^*(PD^*N(A|N)^*)^* \tag{4.4}$$

Este modelo ha permitido obtener mejores resultados en la extracción de terminología de documentos escritos en castellano, en comparación con el empleo de una expresión regular genérica en inglés (descrita por la ecuación 4.3) sobre corpus escritos en castellano. Además, en este artículo se emplearon algunas de las expresiones regulares más frecuentes seleccionadas por terminólogos expertos, lo cual permitió comprobar que estos patrones consiguen obtener el 91 % de los términos totales.

Estos modelos para la extracción automática de terminología desarrollados con el enfoque de las expresiones regulares son de gran utilidad en dominios concretos o en aquellos dominios sobre los que no se disponga una terminología asociada para poder generar un modelos de aprendizaje supervisado.

4.3.2. Extracción de terminología basada en aprendizaje automático

Dentro del enfoque de la extracción de términos mediante aprendizaje automático, se distinguen los dos tipos de modelos en función del modo en el que se entrenan los modelos para adquirir las características necesarias que les permiten llevar a cabo las tareas para las que se crean: aprendizaje supervisado y aprendizaje no supervisado.

Los modelos de aprendizaje automático supervisado, como se ha mencionado en la sección 3.3, requieren datasets etiquetados formados por los documentos junto con una terminología extraída manualmente. Para resolver el problema de la extracción automática de palabras clave, se han empleado multitud de algoritmos diferentes en el estado del arte. El proceso de la extracción a menudo se realiza mediante dos etapas [49]. La primera etapa consiste en escoger las frases o conjuntos de palabras que pueden ser candidatos de términos. La mayoría de los métodos operan a nivel de frase y emplean los patrones morfosintácticos (POS) para generar los candidatos [50]. En una segunda fase se seleccionan aquellos candidatos que serán considerados finalmente como términos.

Estos modelos de aprendizaje supervisado emplean palabras clave conocidas para identificarlas en los textos mediante una tarea de clasificación binaria, donde los candidatos a ser términos son etiquetados como positivos o negativos [38] [51] [52].

Otro enfoque de abarcar la extracción de terminología con modelos supervisados es mediante una tarea de etiquetado de secuencia [50] [53]. En estos modelos, cada uno de los tokens de los textos se clasifica como perteneciente a un término o no perteneciente. En algunos casos [54] se emplea una clasificación multiclase, estableciendo así una mayor distinción en la categoría asociada a cada token de los documentos. Se suele trabajar con tres clases diferentes, con el objetivo de distinguir si una palabra inicia un término (clase 2), pertenece a él sin ser la primera palabra del término (clase 3), o no forma parte de ninguna palabra clave (clase 1).

En el estado del arte se han empleado muchos de los algoritmos de aprendizaje automático existentes para resolver el problema de selección de términos. Por ejemplo, en 2002, Turney [52] empleó árboles de decisión, en concreto el algoritmo C4.5, y un algoritmo genético creado específicamente para este fin, al que denominó GenEx, para la extracción de terminología en diferentes corpus compuestos por artículos científicos, mensajes de correo electrónico y páginas web. Unos años después, Anette Hulth realizó diversos estudios tratando de mejorar las técnicas de extracción de términos mediante modelos de Boosting [55] y Bagging [42]. Estos métodos son tipos de algoritmos de ensamble o combinación de clasificadores, cuyo objetivo consiste en reducir, mediante la combinación de estimadores, la varianza de los modelos de aprendizaje automático.

Otro modelo relevantes de aprendizaje supervisado que se ha empleado en la literatura para la extracción de términos son las máquinas de vectores de soporte (SVM). Estos modelos de predicción no se basan en un modelo probabilístico, sino que emplean un enfoque basado en modelado geométrico, resolviendo las tareas como problemas de optimización. El uso de estos algoritmos en la extracción de palabras clave permitió obtener mejorar los resultados del estado del arte en 2012 usando SVM estructurales [56].

Asimismo, el modelo de Naive Bayes, el cual emplea los datos de entrenamiento para obtener la probabilidad de cada evento en función de su vector de características,

estimando así la clase más probable a la que pertenece cada dato, se ha utilizado en problemas de extracción de términos. Un algoritmo denominado como Kea [51] identifica candidatos a ser términos mediante métodos léxicos para, posteriormente, emplear un modelo de Naive Bayes para predecir aquellos candidatos que serán finalmente seleccionados. Los resultados obtenidos mediante un algoritmos de Naive Bayes o el modelo de Kea obtienen resultados relativamente buenos de *recall* (en torno al 30-40 %) en el problema de extracción de términos [50].

Por otro lado, también existen en la literatura científica numerosos artículos que estudian la extracción de términos mediante aprendizaje automático no supervisado, cuya ventaja principal consiste en el hecho de que no necesitan un dataset etiquetado del dominio sobre el que se pretende trabajar, lo cual supone a menudo una barrera a la hora de realizar modelos de aprendizaje supervisado.

Se han desarrollado multitud de algoritmos distintos con estas características, como es el caso del algoritmo basado en grafos conocido como PageRank [57], cuyo objetivo consiste en establecer un ranking sobre una lista de posibles candidatos de términos. A partir de este método, han surgido numerosos algoritmos de aprendizaje no supervisado que trataban de mejorar la técnica de extracción, como TextRank [58], LexRank [59], TopicRank [60], SGRank [61] o SingleRank [62].

En el estudio realizado por Judea, Schütze y Brügmann [63], realizan la adquisición de terminología de manera no supervisada, donde la selección de las características que determinan la condición de término se realiza sobre un conjunto de candidatos generados automáticamente.

Los modelos de aprendizaje no supervisado se han introducido el uso de la representación de los *word embeddings* junto con el enfoque de ranking basados en grafos para identificar la importancia de las palabras [64].

Tras un análisis del estado del arte se ha podido observar que los algoritmos de aprendizaje automático supervisado tienen mejor rendimiento que los no supervisados cuando tratan datos relativos a un dominio específico [54], ya que los modelos son entrenados para dicho campo. Los métodos de aprendizaje no supervisado tienen la ventaja de no requerir datasets etiquetados, con su terminología asociada, por lo que pueden ser empleados en cualquier dominio, pero obteniendo peores resultados respecto a los modelos entrenados en dicho dominio. Por tanto, el empleo de un tipo u otro de modelo de aprendizaje automático dependerá de las condiciones en las que se desarrollar el problema a resolver.

Entre los algoritmos de aprendizaje automático supervisado, cabe destacar la importancia de las redes neuronales en la extracción de terminologías. Dentro de este campo, se va a llevar a cabo el desarrollo de este trabajo, por lo que conviene profundizar en los modelos creados en el estado del arte.

Los grandes avances de los últimos años en el campo del *deep learning* y su implicación en tareas de procesamiento del lenguaje, ha llevado a la generación de multitud de modelos de aprendizaje supervisado basados en redes neuronales para la extracción de terminología.

Estos modelos generalmente se desarrollan como una tarea de etiquetado de secuencia, por lo que a cada uno de los tokens de entrada se le asigna una de las posibles clases, determinando así qué palabras o conjunto de palabras representan los térmi-

nos.

Para desarrollar dicha tarea, se han empleado en la literatura distintas estructuras neuronales, variando su tamaño, la topología de las redes y su configuración (véase la sección 3.1). En cuanto al tipo de capa de red neuronal, al igual que en la mayoría de tareas del procesamiento del lenguaje natural, se tiende a emplear las redes recurrentes (sección 3.2.3), y más concretamente, las redes LSTM bidireccionales [65], al ser las que relacionan mejor el contexto de las secuencias de entrada, tratándose en este caso de secuencias de palabras.

Esta estructura también se ha empleado en la literatura junto con campos aleatorios condicionales, conocidos como *Conditional Random Fields* o CRF [53] [54]. Se trata de un modelo probabilístico discriminativo, basado en grafos no dirigidos. Se ha demostrado que al incluir estos modelos probabilísticos se mejora el rendimiento en las tareas de etiquetado de secuencias [66]. Por lo general, ha permitido mejorar los resultados en distintas tareas de este tipo, como la extracción de términos, el reconocimiento de entidades (NER), el análisis del sentimiento o el etiquetado POS, puesto que el uso de CRF permite tratar las características de los documentos de manera eficiente [67]. En este último artículo, los experimentos demuestran que el empleo de un modelo basado en CRF supera el rendimiento de los resultados obtenidos en tareas de extracción de términos por otros modelos de aprendizaje automático supervisado, como SVM o regresión lineal múltiple.

Con esta combinación se obtienen las ventajas de ambas estructuras, ya que con el empleo de CRF se captura las dependencias de los términos etiquetados a través de una matriz formada por las probabilidades de transición de una término etiquetado a su vecino. Por otro lado, las redes LSTM capturan las relaciones semánticas presentes en el texto a través de dependencias a largo plazo en ambas direcciones, si se emplean redes LSTM bidireccionales o BiLSTM.

En este contexto, el empleo de los últimos progresos en el procesamiento del lenguaje natural es necesario para avanzar y mejorar los resultados obtenidos en el estado del arte de la extracción de términos. Específicamente, se han introducido los *word embeddings* en los modelos para la extracción de términos, como Glove [53] o Word2Vec [54]. Asimismo, los modelos de lenguaje descritos en el apartado 4.2.2 también han sido empleados en el estado del arte para representar las palabras y resolver tareas de extracción términos. Estas representaciones de las palabras basadas en el contexto (como BERT, OpenAI GPT o ELMo) permiten obtener mejores resultados comparados que las representaciones fijas (Glove y Word2Vec) [54].

El análisis del estado del arte permite concluir que la tarea de extracción de términos ha sido ampliamente estudiada en las investigaciones científicas y se ha empleado en diferentes entornos. Por ejemplo, se ha empleado para obtener términos en artículos científicos [68], artículos de periódico [37], páginas web [69], tweets [70], e incluso en patentes [63].

No obstante, a pesar de la importancia de la extracción de terminología y sus numerosos estudios a lo largo de los años, los resultados obtenidos en el estado del arte de esta tarea son bastante peores comparándolos con otras tareas de PLN, por lo que es un área abierta con un gran potencial de investigación [71]. Las métricas de precisión, *recall* y F1 del estado del arte actual relativo a la extracción de términos tienen valores dentro del rango entre el 30 y el 60 %. En el cuadro 4.1 se pueden observar

los valores obtenidos de una de estas métricas por distintos estudios del estado del arte, mencionados en este apartado. En concreto, en el cuadro 4.1 se ha seleccionado aquellos modelos aplicados sobre el mismo dataset, tratándose del denominado INSPEC, además de seleccionar los artículos que emplean estructuras neuronales similares a las que se realizan en este trabajo, con el objetivo de que la comparación sea precisa. Asimismo, como se explicará y justificará en el apartado 6.1.1, este será el corpus empleado para entrenar los modelos neuronales creados en este trabajo.

Estructura básica de los modelos	Métrica F1
Glove - BiLSTM [65]	42,8 %
BERT - BiLSTM [54]	50,1 %
Glove - BiLSTM - CRF [54]	45,7 %
BERT - BiLSTM - CRF [54]	59,1 %

Cuadro 4.1: Valores de la métrica F1 obtenidos en modelos del estado del arte aplicados sobre el dataset INSPEC.

El cuadro 4.1 reproduce los valores de la métrica F1 obtenidos por distintos autores del estado del arte en sus publicaciones [65][54]. Esta métrica, cuyo significado se detallará en la sección 7.1, se obtiene como la media armónica de las métricas de precisión y *recall*, representando así un balance de estas dos medidas.

En el cuadro, se han seleccionado modelos con características similares, en concreto, aquellos cuya arquitectura se basa principalmente en redes bidireccionales LSTM y que, además, emplean representaciones vectoriales de las palabras para mejorar su rendimiento. En concreto, se muestran dos tipos de arquitecturas: las centradas en redes BiLSTM y las que añaden el uso de una última capa de tipo CRF, comentadas previamente en este apartado. Por otro lado, respecto a los *word embeddings* seleccionados, se puede observar el uso de Glove (explicado en el apartado 4.2.1.2) y el de la representación vectorial de las palabras obtenida por el modelo pre-entrenado de BERT, definido en el apartado 4.2.2.

Se puede observar que el mejor modelo es aquel que combina el empleo de los *word embeddings* de BERT junto con una capa adicional CRF, además, se puede comprobar que la incorporación de una de estas características al modelo, mejora el rendimiento respecto al modelo sin BERT o sin CRF. Estos valores eran de esperar, ya que, como se ha desarrollado a lo largo del estado del arte, el empleo de modelos basados en CRF permiten optimizar los modelos basados en etiquetados de secuencia, como los definidos para extraer terminología. Además, las representaciones vectoriales de las palabras obtenidas por modelos de lenguaje como BERT, consiguen representar no solo la semántica general de las palabras, sino también su significado según el contexto, permitiendo así que el modelo adquiera más conocimiento sobre las entradas que se introducen.

Por último, se han incorporado en el cuadro 4.2 los resultados obtenidos en distintos artículos científicos en los que se ha estudiado el rendimiento de modelos similares, es decir, basados en redes LSTM bidireccionales destinados a la predicción de terminología, con el objetivo de obtener una visión más general de los rendimientos alcanzados por dichos modelos. Al igual que en el cuadro 4.1, los modelos seleccionados son aquellos que emplean redes BiLSTM, aunque cada uno tiene distintas características, como una estructura diferente en cuanto al número de capas o de

neuronas o el optimizador empleado, entre otras. También se ha representado la métrica F1 para comparar los resultados, puesto que se observa así el balance entre las métricas de precisión y *recall*. No obstante, cada uno de los artículos seleccionados emplea un dataset distinto para el entrenamiento, test y validación de los modelos, por lo que los resultados no pueden ser comparados al detalle. Se pretende, por tanto, dotar de mayor información respecto a los valores obtenidos en la métrica F1 del estado del arte de distintos modelos de aprendizaje automático supervisado para la extracción de palabras clave en documentos de texto.

Estructura básica de los modelos	Dataset empleado	Métrica F1
Glove - BiLSTM	kp20k	16,75 %
Glove - BiLSTM	WWW	16,96 %
Glove - BiLSTM	KDD	14,71 %
Glove - BiLSTM - CRF	kp20k	35,63 %
Glove - BiLSTM - CRF	WWW	39,43 %
Glove - BiLSTM - CRF	KDD	41,08 %

Cuadro 4.2: Valores de la métrica F1 alcanzados en distintos modelos del artículo de Al-Zaidy et Al. [53] .

En el cuadro 4.2 se han reproducido los resultados obtenidos en el artículo de Al-Zaidy et Al. [53], en el cual se ha llevado a cabo el problema de extracción de términos como una tarea de etiquetado de secuencia. Las estructuras empleadas se basan en redes BiLSTM, introduciendo también el uso de CRF. Los resultados obtenidos se muestran mediante la métrica F1, sobre cada uno de los datasets empleados. Este artículo, se centra en documentos científicos, usando tres datasets públicos para ello, abreviados mediante kp20k [72], WWW (*World Wide Web Conference*) y KDD (*ACM Conference on Knowledge Discovery*) [73].

Como se puede observar, los rendimientos obtenidos en la métrica F1 de los modelos indicados en el cuadro 4.2 son menores a las métricas representadas en el cuadro 4.1, lo cual se puede deber a distintos factores. Principalmente, las representaciones vectoriales de Glove empleadas en los modelos del cuadro 4.2 tienen un menor número de dimensiones (100) que las utilizadas en los otros artículos (300), lo cual genera que los modelos no aprendan tantas características de las palabras como al emplear las 300 dimensiones disponibles de Glove. Cabe destacar que, el empleo de diferentes datasets hace que la comparativa entre las métricas no se deba realizar de manera exacta, sino en términos generales.

Capítulo 5

Hipótesis de trabajo

5.1. Definición del problema

Una vez definidos los objetivos del presente trabajo, así como los conceptos básicos y el estado del arte, se puede profundizar en la definición del problema que se pretende resolver y en la propuesta de trabajo que se va a desarrollar posteriormente, en el capítulo 6.

Este Trabajo de Fin de Máster se deriva del trabajo realizado en el proyecto *KeyQ* desarrollado en el *AI.nnovation Space* de la Universidad Politécnica de Madrid. El objetivo de este proyecto consiste en crear una herramienta de búsqueda basada en terminologías, orientada a un dominio concreto, siendo, por tanto, más especializada y precisa que un buscador genérico. Para determinar dicho dominio, se debe disponer de un corpus formado por documentos del dominio sobre el que se quiere aplicar la búsqueda. Este sistema de búsqueda permite que los usuarios realicen consultas sobre una gran cantidad de documentos de manera eficiente y óptima, pudiendo navegar entre los distintos documentos para comparar los resultados obtenidos en la consulta.

A la hora de realizar una búsqueda sobre un tema específico, el hecho de mostrar al usuario los términos relevantes o las palabras clave del tema en cuestión, permite que el usuario haga una búsqueda más precisa. De esta manera, a medida que se vaya introduciendo en la búsqueda las palabras, se mostrarán aquellos términos relevantes del dominio que contienen dichos caracteres (pueden ser parte de una palabra, palabra completa o un conjunto de palabras). Al seleccionar un término, el usuario puede acceder a todos los documentos disponibles en los que aparezca dicho término.

En concreto, el proyecto se ha centrado en desarrollar esta herramienta aplicada al dominio específico de los documentos de Airbus, con la finalidad de ofrecer una aplicación que permita a sus operarios obtener toda la información relativa a su campo disponible en un mismo corpus, con una herramienta de búsqueda especializada y centrada en ofrecer las mejores soluciones para el dominio concreto con el que se trabaja.

En la aplicación de *KeyQ*, el hecho de incluir la terminología del dominio del corpus formado por los documentos de Airbus en el sistema de búsqueda genera un valor

añadido importante. Por tanto, la tarea de extracción de los términos pertenecientes a un campo concreto era un factor clave en el proyecto.

Cabe destacar que, a pesar de que el proyecto se centró inicialmente en desarrollar la herramienta para el dominio de Airbus (manuales técnicos), se ha ampliado a dominios como el legal o el bio-sanitario. Se ha desarrollado de manera que se puede aplicar a cualquier dominio del que se disponga de un corpus formado por documentos del tema específico.

Se ha empleado el sistema para un dominio de gran importancia dada la situación mundial actual causada por la pandemia de la enfermedad COVID-19. Durante estos últimos meses, la comunidad científica se ha centrado en investigar multitud de aspectos relacionados con el virus, lo cual ha dado lugar a una gran base de datos de artículos científicos que recogen información sobre la COVID-19. Por tanto, se ha determinado que dotar a la comunidad científica de una herramienta que recopile todos estos documentos y que permita realizar una búsqueda sobre todos ellos puede ser de gran utilidad para la comunidad, al poder compartir el conocimiento de los distintos investigadores. Para ello, se ha implementado la herramienta de *KeyQ* con el corpus formado por los documentos sobre la COVID-19, para lo cual se realiza en este trabajo la predicción de su terminología asociada.

Los resultados que se muestran en en la sección 7.3 relativos a la aplicación del modelo creado en este trabajo sobre documentos no etiquetados, de los cuales no se dispone de terminología etiquetada manualmente, se centrarán en los rendimientos obtenidos sobre el corpus de la COVID-19, al ser este un dataset público.

La creación del proyecto de *KeyQ* desarrollada en *AI.nnovation Space* consta principalmente de dos módulos diferenciados, cuya combinación en la herramienta final genera el valor añadido mencionado. Estos módulos son el sistema de búsqueda y la extracción de la terminología.

El primero ha sido creado por el equipo de *KeyQ*. La herramienta de búsqueda se trata de una aplicación que ofrece la interfaz para que los usuarios introduzcan un término, ya sea simple o compuesto, sobre el que se realiza la búsqueda en el corpus cargado. A partir de las coincidencias obtenidas, se construye el gráfico de dispersión léxica, y se cargan las páginas del corpus en las que se ha obtenido una coincidencia. Este módulo se llevó a cabo empleando el lenguaje de programación R, un lenguaje de código abierto que permite utilizar potentes instrumentos creados para este lenguaje, como Quanteda o Shiny, ambas empleadas en el proyecto. El lenguaje de R dispone de distintos entornos de desarrollo gratuitos, habiendo empleado en el proyecto el entorno de RStudio.

El segundo módulo es el objeto de estudio y desarrollo de esta trabajo, cuyo objetivo es la extracción automática de una terminología asociada a un corpus, la cual se introduce en el sistema para optimizar las consultas. Mediante la extracción automática de las palabras clave se evita la necesidad de disponer de expertos de dominio que realicen la extracción de las palabras clave, lo cual es una tarea costosa en términos de tiempo y dinero, sobre todo tratando de grandes corpus formados por una gran multitud de documentos. Las terminologías obtenidas mediante los modelos desarrollados en este trabajo se compararán con otras metodologías basadas en enfoques diferentes, principalmente expresiones regulares, con la finalidad de determinar la terminología más adecuada para su incorporación en la herramienta de búsqueda.

En la herramienta final de *KeyQ* , la terminología asociada al corpus se carga en el sistema junto con el corpus seleccionado. Esta es empleada para realizar sugerencias durante la búsqueda a través de un elemento desplegado bajo la barra en la que se realizan las consultas.

Por tanto, el objetivo que se pretende alcanzar consiste en la extracción automática de palabras clave de documentos de texto para la optimización del proceso de búsqueda. Este problema ha sido fruto de estudio en el estado del arte, donde la terminología se emplea para indexar documentos sobre los que se realiza una búsqueda [36] [52]. Se trata de un problema abierto en el estado del arte, ya que, como cualquier problema de Procesamiento del Lenguaje Natural, a pesar de los grandes avances de los últimos años, se espera que se puedan mejorar los resultados obtenidos. En este trabajo, se ha optado por resolver esta tarea mediante modelos de aprendizaje automático supervisado, específicamente, modelos basados redes neuronales y aprendizaje profundo.

5.2. Propuesta de trabajo

Los objetivos principales de este Trabajo de Fin de Máster, como se ha mencionado y se ha detallado en la sección 2.1, consisten en analizar, estudiar e implementar distintos modelos para la extracción de terminología de documentos pertenecientes a un mismo dominio de manera automática como una tarea de aprendizaje supervisado, con la finalidad de aplicarlo a un dominio específico.

Partiendo de los últimos estudios en el estado del arte (desarrollados en el apartado 4.3.2) se pretende hacer una comparativa del rendimiento de distintos modelos de *Deep Learning* entrenados con datasets formados por documentos sobre los que existe una terminología asociada. Los distintos modelos se diferenciarán principalmente en la estructura de las capas que componen la red neuronal del modelo, variando tanto los tipos de las capas, explicadas en la sección 3.2 de este documento, como la profundidad de la red, es decir, el número de capas y el número de neuronas artificiales de cada capa, además de los optimizadores empleados para el entrenamiento del modelo, detallados en el apartado 3.4.2. Asimismo, se emplearán distintas formas de representación de las palabras, es decir, *word embeddings* (desarrollados en el apartado 4.2.1), para evaluar su rendimiento al aplicarlo en los distintos modelos.

Después de probar distintos modelos de redes neuronales, se pretende emplear el modelo o los modelos que hayan tenido mejores resultados sobre el dataset etiquetado para predecir la terminología sobre otro corpus del que se quiera extraer su terminología de manera automática. En concreto, se empleará el modelo entrenada para predecir las palabras clave del corpus formado por los documentos de Airbus, las cuales se integrarán a la herramienta *KeyQ*. Asimismo, se obtendrá la terminología extraída mediante el modelo creado sobre el corpus de la COVID-19, dada su gran relevancia en el momento actual. Los resultados obtenidos sobre un dataset no etiquetado se mostrarán sobre este último corpus, al tratarse de documentos públicos disponibles para toda la comunidad científica.

En el capítulo 6 se detalla el procedimiento llevado a cabo para conseguir los objetivos propuestos para este trabajo. Inicialmente, se describen los datos con los que se va a tratar, tanto los datasets etiquetados empleados para entrenar y evaluar los modelos (apartado 6.1.1), como los datos sobre los que se quiere extraer la terminología automáticamente con el modelo final. Se detallarán por tanto el corpus de

Airbus (apartado 6.1.2) y el de artículos relacionados con la enfermedad COVID-19 (apartado 6.1.3).

Posteriormente, en el apartado 6.2.2 se explica cada uno de los modelos creados y entrenados para conseguir obtener un buen modelo que realice la tarea deseada de manera adecuada.

Tras este proceso de entrenamiento y comparativa de distintos modelos de aprendizaje profundo basados en redes neuronales, se exponen los resultados obtenidos mediante distintas métricas en la sección 7.2. Por último, los resultados obtenidos al aplicar el modelo entrenado sobre un nuevo corpus de un dominio concreto se comentarán en la sección 7.3.

Capítulo 6

Desarrollo

En este capítulo se describe el procedimiento y la implementación llevados a cabo para alcanzar los objetivos propuestos en el trabajo.

En primer lugar, en la sección 6.1 se presentan los datos con los que se va a trabajar a lo largo del trabajo. Para ello, se presentan algunos datasets públicos existentes que permiten ser utilizados para entrenar algoritmos de aprendizaje automático supervisado, es decir, contienen una terminología seleccionada manualmente por expertos asociada a los documentos que forman el corpus. Por otro lado, también se comentan los corpus que han sido empleados para predecir una terminología, tras obtener los modelos completos. Estos corpus, por tanto, no disponen de información previa relativa a su terminología.

Tras introducir los datos que se emplean en el trabajo, en la sección 6.2 se detalla la implementación de los modelos creados para la extracción automática de palabras clave. En dicha sección se explica el preproceso que se aplica a los datos introducidos en el apartado 6.2.1, que permite que los documentos que forman los datasets puedan ser introducidos en los modelos neuronales. Por último, se desarrolla el proceso llevado a cabo para la creación de dichos modelos en el apartado 6.2.2.

6.1. Orígenes de los datos

En este trabajo se va a tratar con dos tipos de corpus, como se ha mencionado anteriormente. Inicialmente, se necesita un dataset etiquetado que permita entrenar y validar los modelos que se pretenden crear, por lo que se presentan en el apartado 6.1.1 distintos datasets públicos que se pueden utilizar para resolver la tarea de este trabajo.

Por otro lado, la finalidad de los modelos consiste en la predicción de la terminología de distintos documentos de dominio específico, de los cuales no se dispone de una terminología extraída manualmente. En los apartados 6.1.2 y 6.1.3 se muestran las características principales de los corpus empleados en el proyecto *KeyQ* y en este trabajo.

Cabe destacar que todos los documentos con los que se trata en este trabajo están redactados en inglés, tanto los datasets de entrenamiento, como los corpus sobre los que se quiere obtener su terminología asociada. Por lo tanto, los modelos implementados se podrán emplear para seleccionar términos de documentos en inglés.

6.1.1. Datasets de entrenamiento

Para desarrollar los modelos neuronales capaces de predecir terminología dado un corpus, se debe llevar a cabo el proceso de entrenamiento de la red neuronal, así como las pruebas y validaciones que verifican su correcto funcionamiento. Por ello, como en cualquier modelo de aprendizaje automático supervisado, se necesita un dataset etiquetado del cual se pueda extraer unas entradas del modelo y las correspondientes salidas que se espera. En el caso de la tarea de extracción de terminología, se requiere un dataset que disponga de la terminología asociada a los textos, la cual será la salida que debe producir el modelo. Esta terminología debe ser extraída manualmente, por lo cual adquiere un carácter subjetivo, ya que no hay un criterio estándar definido para determinar qué términos pertenecen o no a la terminología de un dominio.

Existen distintos datasets públicos en la literatura que permiten entrenar modelos de extracción de terminologías, por lo que en este apartado se van a introducir algunos de los más conocidos y empleados en el estado del arte.

En muchos de estos datasets, se distinguen dos tipos de terminología en función de quién ha llevado a cabo dicho etiquetado [42]. Se denomina terminología controlada a aquella que ha sido seleccionada por los autores de los documentos. Muchos de los datasets (corpus) anotados existentes están formados por artículos de carácter científico o por sus abstracts. En estos casos, la terminología controlada suele estar formada por aquellas palabras que se indican al principio de los artículos como palabras clave o *keywords*.

Estos términos no están presentes necesariamente en los documentos bajo estudio, sobre todo cuando los documentos únicamente están formados por los abstracts, sino que el autor, como experto del tema, determina que se trata de un término representativo del dominio del documento. Por otro lado, se conoce como terminología no controlada a la que extraen otros profesionales distintos a los autores de los documentos. Estos términos sí tienden a estar presentes en los textos asociados, aunque no es una característica necesaria.

En este estudio, puesto que se va a llevar a cabo la extracción de la terminología mediante una tarea de etiquetado de secuencias a partir de los documentos, se tendrá en cuenta únicamente la terminología denominada como no controlada, la cual está presente en los documentos.

A la hora de escoger el dataset (corpus) con el que se entrenará el modelo, se deben tener en cuenta sus características, como la cantidad de documentos para entrenamiento, test y validación; o el dominio sobre el que tratan los textos. A continuación, se mencionan algunos de los datasets más empleados en la extracción de terminología y sus características.

6.1.1.1. Dataset INSPECT

El dataset conocido como INSPEC [42] está compuesto por 2.000 *abstracts* de artículos en inglés, con sus correspondientes títulos y términos. Estos artículos pertenecen a revistas científicas de los campos de Computadores y Control, y Tecnologías de la Información, publicados entre los años 1998 y 2002. La colección de documentos está dividida en 1.000 documentos para el entrenamiento del modelo, 500 para test y 500 para validación.

Se dispone de tres tipos distintos de archivos, distinguiéndose cada uno de ellos mediante la extensión del fichero. Cada abstract (ficheros con extensión *.abstr*) tiene asociado dos conjuntos distintos de terminología extraída manualmente. El conjunto controlado (archivos *.contr*) es creado por los autores de los artículos o por profesionales, mientras que la terminología no controlada (ficheros *.uncontr*) es extraída por lectores.

Como se ha mencionado anteriormente, en la tarea de extracción de términos a partir de los documentos, se ha optado por tener en cuenta únicamente la terminología no controlada. Por ello, la entrada del modelo neuronal serán los abstracts con extensión *.abstr* y las salidas serán los términos etiquetados con formato *.uncontr*.

Los abstracts tienen una longitud mínima y máxima de 15 y 557 palabras, respectivamente. La longitud media de los documentos es de 125 palabras. Por otro lado, cada abstract tiene asociados una media de 10 términos no controlados y 5 controlados.

6.1.1.2. Dataset SemEval-2010

SemEval-2010 [68] fue creado para desarrollar distintas tareas propuestas en el Workshop sobre Evaluación Semántica de 2010. Con este fin, se recopilaron 284 artículos completos de revistas ACM, escogidos por autores y lectores para que trataran sobre temas diversos. Entre las tareas creadas, la número 5 se corresponde con la extracción automática de palabras clave de los artículos científicos.

El corpus está dividido en un conjunto de entrenamiento, otro de test y otro de validación, con 144, 100 y 40 documentos, respectivamente.

Cada uno de los artículos contiene términos designados por los autores, así como una terminología extraída por estudiantes contratados para desarrollar dicha tarea. El dataset está formado por tres conjuntos de etiquetas o salidas: uno con la terminología controlada (ficheros con extensión *.author.final*), otro con la terminología no controlada (archivos *.reader.final*) y el último con una combinación de dichos conjuntos (ficheros *.combined.final*).

6.1.1.3. Dataset SemEval-2017

A lo largo de los años, se ha realizado el Workshop sobre Evaluación Semántica ofreciendo distintas tareas en este campo, modificando los datasets anualmente. El corpus SemEval-2017 [74] está formado por 500 artículos científicos completos publicados en ScienceDirect con acceso abierto, tratando dominios como la ciencia de los computadores, ciencias materiales y física.

La parte del corpus relativa al entrenamiento consiste en 350 documentos, disponiendo de otros 100 documentos para test y 50 para la validación. En este corpus, la tarea relativa a la extracción de términos es la número 10.

A diferencia de los anteriores datasets, SemEval-2017 no contiene anotaciones de los autores, sino que únicamente dispone de un conjunto de terminología extraída por estudiantes voluntarios, es decir, por lectores de los documentos no expertos en el dominio.

6.1.1.4. Comparativa de los datasets

En el cuadro 6.1 se ha recopilado la información más relevante relativa a los datasets mencionados en los apartados anteriores, es decir, INSPEC, SemEval-2010 y SemEval-2017, con el objetivo de aportar una visión general de las características de cada uno de ellos. Se indica el número de documentos que compone cada uno de los datasets, distinguiendo entre los conjuntos de documentos seleccionados para el entrenamiento, para el test y para la validación. También se indica el número medio de términos etiquetados por cada documento, en cada uno de los conjuntos mencionados. Por último, se representa el número medio, máximo y mínimos de tokens de cada uno de los documentos.

Fuente: elaboración propia basada en [54].

Características	Datasets								
	INSPEC			SemEval-2010			SemEval-2017		
	Entr	Test	Valid	Entr	Test	Valid	Entr	Test	Valid
Nº documentos	1000	500	500	130	100	14	350	100	50
Nº medio términos	9,8	9,8	9,1	9,9	9,8	9,7	15,6	17,7	19,5
Nº medio tokens	142	135	133	185	207	201	187	220	225
Nº máx. tokens	557	384	330	432	395	416	350	389	399
Nº mín. tokens	15	23	16	55	65	111	65	102	137

Cuadro 6.1: Estadísticas generales de los datasets seleccionados.

Tras el análisis y estudio de las características de estos datasets, se decidió escoger el corpus de INSPEC. El principal factor que ha dado lugar a su selección consiste en que es el dataset que contiene una mayor cantidad de documentos, lo cual permite que el modelo de *Deep Learning* represente más características representativas de los documentos, al tener una mayor cantidad de datos con la que ajustar sus parámetros durante el proceso de entrenamiento. En cuanto al dominio de los datasets, todos los mencionados en este apartado están formados por artículos y documentos científicos redactados en inglés, por lo que se considera que pueden tener un buen rendimiento al extrapolar el modelo al dominio del corpus de la COVID-19, por ejemplo, al tratar este también sobre artículos científicos.

6.1.2. Corpus de Airbus

La finalidad de la extracción de terminología en este trabajo es su incorporación a la aplicación *KeyQ*, del proyecto de AI.nnovation Space. Por ello, uno de los corpus del que se pretende extraer los términos consiste en un conjunto de documentos de la empresa Airbus. Estos documentos son manuales en formato PDF, que la empresa ha proporcionado para la realización del proyecto. El corpus está formado por un conjunto de 198 documentos, cada uno de los cuales tiene un tamaño entre 0,006MB y 31,18MB, ocupando un total de 355,78MB. El conjunto de los documentos tiene un total de 28.201 páginas y 8.993.815 tokens.

Se trabajará con este corpus en la parte de predicción del modelo. No obstante, para poder ser tratado por el modelo, previamente se lleva a cabo una tarea de preproceso, la cual se desarrolla en el apartado 6.2.1.

6.1.3. Corpus sobre la COVID-19

Otra de las aplicaciones para las que se emplea este Trabajo de Fin de Máster consiste en la extracción de la terminología de un corpus basado en artículos científicos sobre la enfermedad COVID-19[1]. Los términos extraídos, así como su integración en la herramienta de *KeyQ* pueden ser de gran utilidad para la comunidad científica.

Este corpus consta de 657 documentos en formado PDF, cada uno de los cuales tiene un tamaño entre 0,07MB y 23,57MB, ocupando un tamaño total de 1.562,12MB. El conjunto de todos los artículos científicos tiene un total de 8.405 páginas y 7.294.168 tokens.

Al igual que con el resto de documentos que se introducen en los modelos creados, se debe realizar un preproceso de los datos previos, explicado en el apartado 6.2.1.

[1] https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

6.2. Desarrollo de los modelos de extracción de terminologías

En esta sección, tras haber introducido los datos que se emplean en el trabajo en la sección 6.1, se especifica el proceso y la implementación que se ha llevado a cabo para desarrollar modelos basados en redes neuronales que resuelvan la tarea de extracción automática de terminologías. Inicialmente se define los procesos aplicados a los datos de entrada de los modelos en el apartado 6.2.1, para continuar con la explicación del desarrollo que ha dado lugar a la creación de los modelos, en el apartado 6.2.2.

6.2.1. Preproceso de los datos

Como se ha comentado en el apartado 6.1.1.4, el dataset empleado finalmente para el entrenamiento del modelo ha sido INSPEC (explicado con mayor detalle en el apartado 6.1.1.1, debido a que contiene un mayor volumen de texto para tratar, lo cual es de gran utilidad para poder entrenar desde cero cualquier modelo de aprendizaje profundo. Además, al tratarse de *abstracts* de artículos científicos de distintos campos, permite generar un modelo aplicable a distintos campos de dominios específicos.

Antes de que dicho dataset sea usado por los modelos creados, se debe llevar a cabo un proceso de tratamiento de los datos para que el formato y las características de estos sea adecuado para los modelos de aprendizaje profundo. En concreto, en el procesamiento del lenguaje natural las entradas y salidas se corresponden con palabras y frases, por lo que se deben transformar en valores numéricos que puedan ser tratados por la red neuronal. Este preproceso se lleva a cabo en cualquier texto o documento que se quiera introducir en el modelo, por lo que se aplica tanto al dataset INSPEC como a los corpus sobre los que se pretende predecir una terminología, en concreto, el corpus de Airbus (apartado 6.1.3) y el corpus sobre la COVID-19 (apartado 6.1.3).

Inicialmente se dispone de los documentos que componen el corpus en texto plano. Con el objetivo de organizar los documentos que se van a tratar, se crean en Python estructuras de datos tipo diccionario para distinguir los documentos que pertenecen a los conjuntos de entrenamiento, test y validación. Cada uno de los diccionarios creados está formado por claves, que indican el nombre de los documentos, y por sus respectivos valores, que contienen los textos de los documentos a los que se hace referencia. Asimismo, se generan diccionarios para las terminologías asociadas, distinguiendo también entre los conjuntos de entrenamiento, test y validación.

A continuación, se lleva a cabo el proceso conocido como *tokenización*, por el cual se separan las palabras de los textos en entidades llamadas tokens, representando la unidad mínima de lenguaje natural para su procesamiento posterior. En función del modelo empleado, se han pueden emplear distintos tokenizadores. No obstante, en los modelos descritos en el apartado 6.2.2, se ha utilizado un tokenizador de la plataforma NLTK (*Natural Language Toolkit*) [75], que divide una cadena de caracteres en subcadenas empleando expresiones regulares.

A partir del texto tokenizado, se debe asignar a cada token distinto un identificador único asignado como un valor entero. Se debe crear un objeto que almacene el vocabulario presente en todos los documentos que se vayan a emplear en el modelo, que permita mapear cada uno de los identificadores únicos con el token asociado.

La secuencia de identificadores únicos será la entrada del modelo neuronal creado, descrito en el apartado 6.2.2. Asimismo, también se debe obtener una matriz que contenga la representación vectorial de cada uno de los tokens que aparezcan en el vocabulario del corpus. En esta matriz, el índice en el que se encuentra cada vector se corresponde con el identificador del token que representa dicho vector. Esta matriz también formará parte de las entradas del modelo.

Otro aspecto importante en el preproceso de los datos es la determinación del tamaño de las secuencias de entrada. Este valor se corresponderá con un parámetro del modelo y, por tanto, se podrá modificar según la arquitectura diseñada y según el corpus empleado. Pese a que las redes recurrentes permiten tratar con secuencias de longitud variable, es necesario establecer un tamaño constante ya que, en la práctica, la red no se alimenta con un único ejemplo en cada instante, sino que se utiliza un lote o *batch* completo. Cada muestra dentro del lote debe tener el mismo tamaño para que pueda ser tratado por el modelo.

En el problema planteado en este trabajo, las secuencias son conjuntos de palabras, por lo que se debe seleccionar la cantidad de palabras que se introducen en las unidades del modelo neuronal. El tamaño escogido debe ser suficiente para capturar el contexto relevante sobre el que se va a trabajar, con el objetivo de que las secuencias permitan obtener información a largo plazo, pero no debe ser mayor. Esto es debido a que secuencias cortas son más fáciles de entrenar y permiten, además, simular la técnica de *Data Augmentation*[2] en el procesamiento de imágenes, la cual permite mejorar los resultados y evitar *overfitttting*.

En los modelos descritos en el siguiente apartado se ha establecido un tamaño de secuencia que abarca prácticamente la longitud total de los abstracts que forman el conjunto de documentos de entrenamiento, permitiendo así que se capture el contexto total del texto introducido. El valor escogido ha sido 550, puesto que, como se puede comprobar en el cuadro 6.1, el número máximo de tokens de los documentos se supera superficialmente en uno de los conjuntos del dataset INSPEC. Además, este valor de longitud de secuencia no es demasiado grande para una red neuronal recurrente, por lo que permitirá un entrenamiento y una convergencia adecuados.

Por otro lado, la terminología asociada a los documentos del dataset INSPEC también necesita un preprocesado, distinto del que necesitan los documentos. En primer lugar, para asociar los textos de los documentos con su terminología, se genera una clasificación de cada uno de los tokens de los documentos, indicando si se corresponde con el inicio (1) o el cuerpo (2) de un término, o si no pertenece a ninguna palabra clave (0). Esta clasificación se expresa de forma categórica, de manera que mediante un vector de tamaño 1×3 se indica la codificación *one hot encoding* de las clases[3]. De esta forma, la salida de la red neuronal consistirá en establecer la probabilidad de que un token dado pertenezca a cada clase.

Tras este preproceso se obtienen los datasets y los corpus preparados para su incorporación en los modelos de aprendizaje profundo creados, descritos en el siguiente apartado.

[2]La metodología de *Data Augmentation* es una técnica específica al tratamiento de imágenes en *Deep Learning* que permite optimizar modelos entrenados sobre un dataset con un tamaño limitado, con el objetivo de lograr una buena generalización[76]

[3]Un token perteneciente a la clase (1) tendrá un vector salida asociado [1, 0, 0], uno de la clase (2) vendrá determinado por el vector [0, 1, 0] y uno de la última clase se representará mediante [0, 0, 1]

6.2.2. Modelos de *deep learning*

En cualquier problema de aprendizaje automático, un factor determinante para obtener los mejores resultados para un problema concreto es la elección del modelo. En lo que concierne a este trabajo, la determinación del modelo de red neuronal, los tipos de capas y su estructura son factores cruciales para obtener unos buenos resultados. Por tanto, se ha optado por reproducir, aunque con ciertas variaciones, los últimos modelos del estado del arte para la extracción de palabras clave.

Los últimos avances en el campo de *deep learning* aplicado al procesamiento de lenguaje natural, como se ha visto en la sección 4.2, se han llevado a cabo principalmente mediante el empleo de redes neuronales recurrentes, en concreto, con LSTM (véase el apartado 3.2.3), ya que tienen en cuenta el orden de la secuencia de entrada del modelo, detectando dependencias tanto a corto como a largo plazo. El uso capas LSTM bidireccionales dota al modelo con la capacidad de representar las dependencias de las secuencias de entrada en ambas direcciones, es decir, tanto hacia atrás como hacia delante. Por ello, el desarrollo de este trabajo se ha centrado en probar distintas estructuras conteniendo alguna capa BiLSTM.

Además, como todo modelo de aprendizaje automático, en el procesamiento del lenguaje natural se debe escoger la forma para representar las palabras de manera que puedan ser tratadas por el modelo computacional, tratando de conservar la mayor información posible, en cuanto a su semántica y la relación con otras palabras. Como se ha explicado en los apartados 4.2.1 y 4.2.2, existen en la literatura distintos modelos de representación de las palabras, llamados *word embeddings*. Principalmente, cabe diferenciar los modelos fijos (*fixed word embeddings*), como Word2Vec o Glove, cuya representación es invariante, sin tener en cuenta el significado de la palabra en la frase en la que se encuentra. Por otro lado, los modelos de lenguaje (apartado 4.2.2) dan lugar a vectores de palabras contextuales o *contextual word embeddings*, ya que su representación indica la semántica concreta de la palabra según el contexto en el que se encuentra. Por ello, es evidente que estos últimos modelos generan una representación más precisa, dando lugar a mejores resultados en los modelos que los incorporan, como se ha estudiado en el estado del arte [54]. En este trabajo, se han analizado distintos *word embeddings*, tanto fijos como contextuales. No obstante, los modelos finales se han desarrollado con representaciones vectoriales fijas, en concreto, se han empleado los *word embeddings* de Glove (definidos en el apartado 4.2.1.2), ya que, como se ha observado en el estado del arte (apartado 4.3.2), permiten obtener unos resultados adecuados, sin tener que utilizar grandes cantidades de memoria RAM y de tiempo, como ocurre con las representaciones vectoriales más complejas, como es el caso de BERT [77].

Como se ha visto en la sección 4.3, existen distintas formas de abordar la extracción de palabras clave de los textos de un corpus. En este trabajo, los modelos generados son supervisados, y la tarea es el etiquetado de las secuencias de entrada. Se ha utilizado un modelo de etiquetado múltiple, determinando la probabilidad de que cada uno de los tokens de entrada pertenezca a una de las tres clases establecidas: (1) palabras que no pertenecen a ningún término, (2) palabras que inician los términos y (3) palabras que pertenecen a un término, pero no en primera posición.

Al ser un modelo multiclase, cuya predicción debe corresponderse con la probabilidad de pertenencia a cada una de estas clases, se ha empleado como función de activación de la última capa la función *softmax* (explicado en la sección 3.1), y *categorical*

crossentropy (sección 3.3) como función de coste.

La implementación del trabajo se ha llevado a cabo en un cuaderno de Google Colab[4], puesto que permiten ejecutar cualquier código en Python mediante los servidores de Google, accediendo a sus GPUs y TPUs[5] de manera gratuita.

Para construir y entrenar el modelo de *deep learning* se ha empleado la API funcional Keras de TensorFlow Se trata de una API de alto nivel implementada en la última versión de TensorFlow (v. 2.0). que permite crear modelos de aprendizaje profundo de manera sencilla manteniendo todas las funcionalidades de TensorFlow. En Keras, se ensamblan capas neuronales para construir los modelos. El modelo habitual se denomina *Sequential* y consiste en una pila de capas.

En este trabajo se han elaborado distintos modelos que permiten resolver la tarea de extracción de términos de forma automática, siguiendo la línea que se ha desarrollado en el estado del arte (véase el apartado 4.3.2). Se ha comenzado por un modelo sencillo, a partir del cual se han ido añadiendo estructuras más complejas para tratar de obtener mejores resultados. Para ello, la primera tarea ha consistido en reproducir los resultados descritos en el estado del arte, y después se ha modificado la estructura del modelo para tratar de obtener mejores resultados.

En los siguientes apartados, se describe el ensamblaje en Keras de los modelos diseñados, detallando los tipos de capas necesarios, las funciones de activación de las neuronas artificiales, así como otros parámetros necesarios para obtener las estructuras deseadas.

[4]El código implementado para el desarrollo de este Trabajo de Fin de Máster se ha publicado en Github: `https://github.com/luciaguasp/DL-terminology-extraction`.

[5]Las GPUs (unidades de procesamiento gráfico) y las TPUs (unidades de procesamiento tensorial) son procesadores con mayor poder computacional que una CPU, por lo que se emplean en el entrenamiento de modelos de aprendizaje automático y profundo para optimizar este entrenamiento, dotándole de una mayor velocidad. En concreto, las TPUs fueron creadas específicamente por Google para acelerar el procesamiento de modelos de aprendizaje automático creador con TensorFlow.

6.2.2.1. Modelo 1: BiLSTM empleando Glove *embeddings*. Reproducción del modelo de Basaldella et al. [65]

El modelo de partida de este trabajo consiste en una arquitectura de *deep learning* basada en redes neuronales recurrentes de tipo LSTM bidireccionales, que no requiere características especiales para un dominio específico, sino que puede ser aplicado a una gran variedad de situaciones, dependiendo de los datos con los que se entrene al modelo.

En primer lugar, se ha reproducido un modelo existente en el estado del arte con estas características para, posteriormente, realizar modificaciones sobre dicha arquitectura. En este caso, se ha reproducido un modelo diseñado por Basaldella et al. [65] cuya estructura se basa en BiLSTM mediante los *word embeddings* de Glove.

Fuente: elaboración propia.

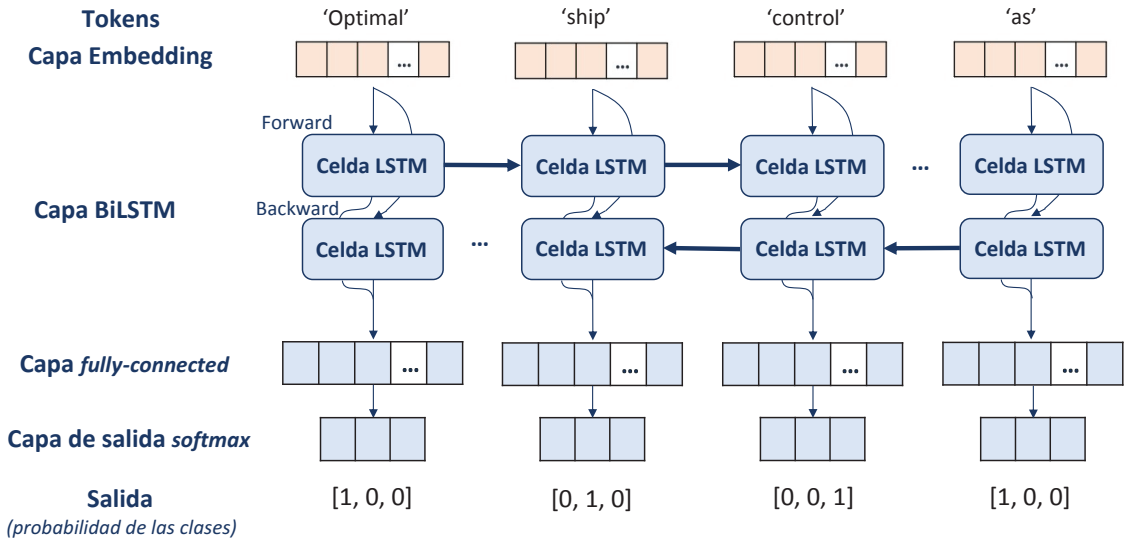


Figura 6.1: Esquema del Modelo 1.

Tras el preprocesado de los datos explicado en el apartado 6.2.1, se asocia cada identificador de los tokens a una representación vectorial o *word embedding*. En este modelo, se han empleado el modelo de representación de Glove (detallado en el apartado 4.2.1.2). La representación de 6 mil millones de tokens extraídos de documentos de Wikipedia puede descargarse desde la página de Stanford[6]. Para el modelo, se han empleado los vectores de 300 dimensiones, ya que contienen mayor información, consiguiendo así una mejor representación.

Por tanto, la primera parte del modelo consiste en una capa tipo *Embedding*, en la que se debe establecer el tamaño del vocabulario de la representación, así como la concatenación de la lista de pesos de esta capa, que se corresponde con la representación vectorial asociada a los tokens de entrada, obtenidos durante el preproceso. Se indica en la configuración de la capa que los pesos no son entrenables, es decir, no se entrenan con los datos de entrada de la red, sino que se usan las representaciones obtenidas por el modelo de Glove.

[6] https://nlp.stanford.edu/projects/glove/

A continuación, los *word embeddings* se introducen en celdas BiLSTM (sección 3.2.3), que pueden tratar con secuencias de longitud variable y extraer características útiles de las palabras en su contexto. Para reproducir los resultados del modelo escogido, se ha diseñado la estructura con una capa BiLSTM de 300 neuronas que emplean como función de activación la tangente hiperbólica y como función de activación en el paso recurrente la sigmoide (explicadas en la sección 3.1).

Las entradas y las salidas de estas capas son tridimensionales, al tratarse de las muestras de cada uno de los instantes de tiempo, con sus respectivas características. En el ámbito del procesamiento del lenguaje natural, las operaciones relativas a instantes de tiempo en las redes recurrentes se refieren a las subdivisiones de las secuencias de entrada, normalmente tokens. Es decir, cada instante de tiempo en una secuencia temporal se corresponde con cada token en una secuencia de palabras. No obstante, por defecto, las capas LSTM generan salidas bidimensionales (las muestras y sus características), por lo que se debe establecer en la configuración de la capa el argumento *return_sequences* para conseguir que la capa LSTM genere las salidas tridimensionales.

La segunda capa BiLSTM se conecta a una última capa oculta, construida con Keras como una capa *TimeDistributed Dense* formada por 150 unidades de neuronas artificiales. El término *"Dense"* indica que se trata de una capa totalmente conectada (capa *fully-connected*, explicada en el apartado de Perceptrón multicapa de la sección 3.2). Al emplear la capa *TimeDistributed*, se realiza una operación *Dense* a cada instante de tiempo del tensor tridimensional de entrada.

Para finalizar la arquitectura neuronal, se añade la capa de salida, también de tipo *TimeDistributed Dense*. Esta capa estará formada por tres neuronas con función de activación softmax (sección 3.1), con el fin de que la red neuronal prediga la probabilidad de pertenencia de cada uno de los tokens de entrada a las tres clases establecidas (inicio de término, pertenencia a un término, o no perteneciente a ninguna palabra clave).

En la figura 6.1 se puede observar una representación de la arquitectura de este modelo, distinguiendo la capa de entrada tipo *Embedding*, dos capas ocultas (BiLSTM y *fully-connected*) y la capa de salida. En el ejemplo de la figura, la entrada del modelo sería la secuencia de palabras *"Optimal ship control as..."* tokenizada. En la salida se puede observar que tanto el primer token como el último pertenecen a la primera clase (0), es decir, no forman parte de ningún término. Por otro lado, el token *"ship"* pertenece a la clase (1) y *"control"*, a la clase (2), lo cual implica que *"ship control"* es una palabra clave. Con esta metodología se entrena el modelo, de manera que, cuando se vaya a emplear este modelo para predecir la terminología de un documento, se obtenga la probabilidad de pertenecer a cada una de las clases, escogiendo por tanto la clase con mayor probabilidad.

Como se ha visto en el apartado 3.4.3, los modelos de aprendizaje profundo son tan complejos que generalmente tienden a modelizar, de forma demasiado exacta, los datos del entrenamiento, dando lugar a lo que se conoce como sobreajuste (*overfitting*), que provoca que el modelo no generalice bien. Por lo que, en el modelo descrito anteriormente, se deben añadir técnicas de regularización (apartado 3.4.4) que traten de evitar el sobreajuste a los datos de entrenamiento para generar así un modelo que pueda ser empleado para predecir la terminología de un corpus no etiquetado.

En concreto, se ha añadido la técnica de regularización L_2, que permite reducir los grados de libertad del modelo al penalizar los pesos de las conexiones de las unidades neuronales. Como se desarrolla en el apartado 3.4.4, se debe establecer el valor del hiperparámetro α, que determina el valor de la penalización aplicada a los pesos del modelo en el entrenamiento. Para este modelo se ha utilizado un valor de $\alpha = 0.01$ en la última capa oculta, lo que significa que la penalización es de un 1 % del sumatorio del valor absoluto de los pesos.

Asimismo, se ha empleado la técnica de *Dropout* con un valor del 25 %, por lo que, en cada paso del entrenamiento, cada neurona tiene un 25 % de probabilidad de ser desactivada, produciendo así que las neuronas que sí permanecen activadas en ese paso aprendan nuevas relaciones.

Una vez definida la arquitectura del modelo secuencial en Keras, se debe configurar la función de coste y el optimizador que se quiere emplear para el entrenamiento. En este modelo, se ha usado la función de coste *categorical crossentropy* (3.3), al tratarse de una tarea de clasificación multi-clase. En cuanto al optimizador, se ha empleado RMSProp (detallado en el apartado 3.4.2).

Finalmente, se realiza el proceso de entrenamiento del modelo creado, especificando el número de épocas y el tamaño del lote (*batch size*). Como se ha descrito en la sección 3.3, el tamaño del lote indica el número de muestras que se seleccionan para llevar a cabo cada paso del algoritmo *backpropagation*, mientras que el número de épocas determina cuántas veces se realiza el entrenamiento sobre todas las muestras.

El entrenamiento del modelo neuronal detallado en este apartado tiene por entradas los textos pertenecientes al dataset INSPEC y por salidas las terminologías etiquetadas de dicho dataset, tras el preproceso explicado en el apartado 6.2.1. Se ha entrenado con un tamaño de lote de 32 muestras, durante un total de 30 épocas, evaluando la precisión tanto de las entradas de entrenamiento, como las de validación, permitiendo así comprobar si el modelo funciona correctamente, además de permitir la detección de posibles problemas de *overfitting* (apartado 3.4.3).

Fuente: elaboración propia.

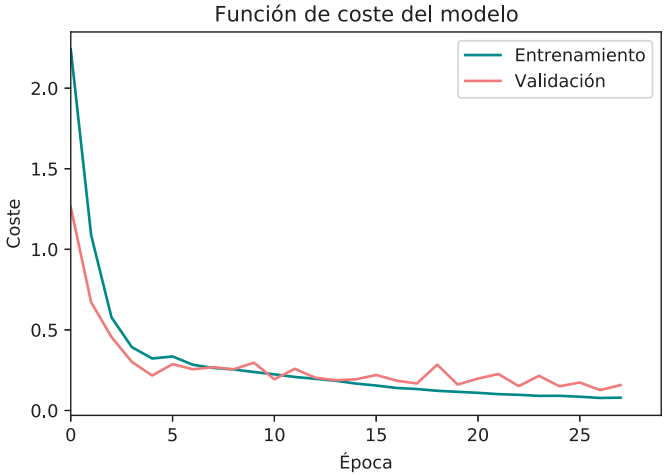


Figura 6.2: Variación de la función de coste en el entrenamiento del Modelo 1.

En la figura 6.2 se puede observar la evolución de la función de coste durante el entrenamiento, tanto del subconjunto de entrenamiento como del de validación. El valor de esta función disminuye a medida que avanzan las épocas, aunque a partir de la quinta época la disminución es muy reducida. Sin embargo, al no aumentar el valor de la función de coste no hay sobreajuste, por lo que consideramos que el entrenamiento se ha realizado con éxito.

Los resultados obtenidos con este modelo en los procesos de entrenamiento, test y validación se desarrollan en el apartado 7.2.1, donde se comprueba que se han obtenido los resultados del modelo del estado del arte reproducido.

6.2.2.2. Modelo 2: Doble capa BiLSTM usando los *word embeddings* de Glove

A continuación, se ha tratado de mejorar la arquitectura del apartado anterior, pero manteniendo sus características principales, es decir, se trata de otro modelo formado por unidades LSTM bidireccionales empleando Glove como *word embeddings*. Tras probar con distintas configuraciones con esta arquitectura, se han obtenido mejores resultados empleando dos capas de unidades BiLSTM de 200 neuronas cada una.

En la figura 6.3 se puede observar la representación de la arquitectura propuesta. Se puede comprobar que la estructura base es similar a la del modelo reproducido, pero contando en este caso con una estructura neuronal más profunda, al haber introducido dos capas ocultas BiLSTM. De esta manera, se consigue que el modelo aprenda características más específicas de los datos de entrada.

Fuente: elaboración propia.

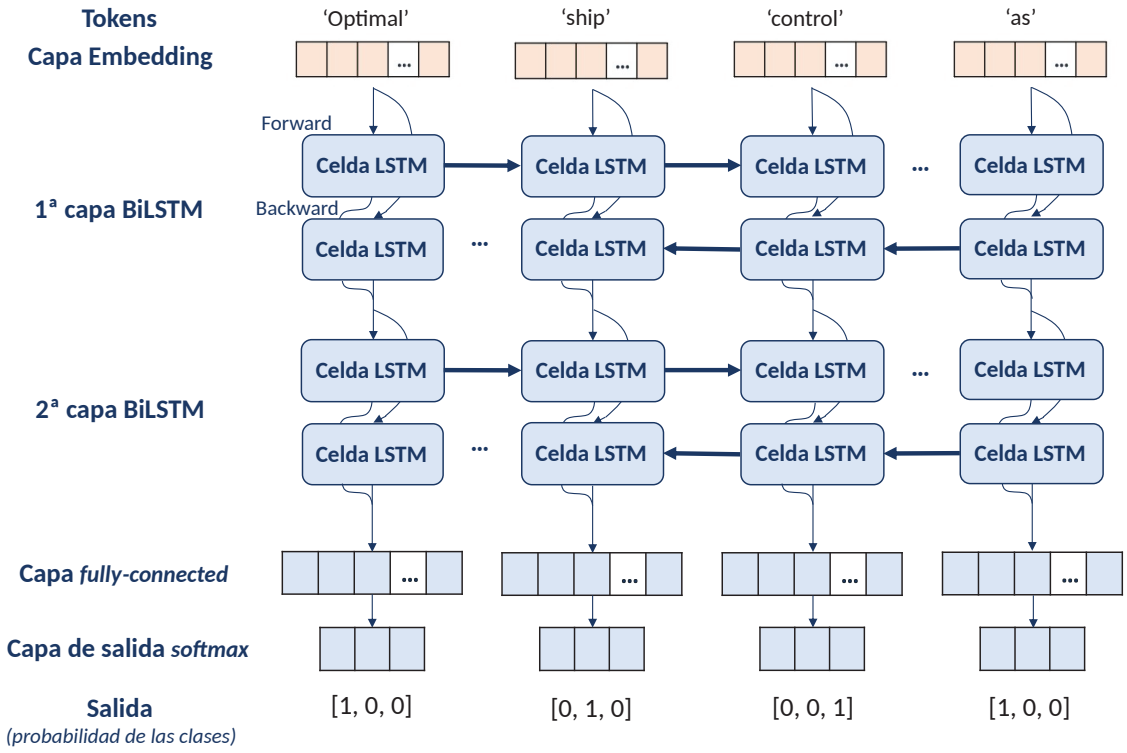


Figura 6.3: Esquema del Modelo 2.

Para llevar a cabo el proceso de selección de los valores de los hiperparámetros, las funciones de activación y coste, y optimizadores, primero se ha tenido en cuenta el estudio realizado en el estado del arte (sección 4). Esto nos ha permitido definir qué características son convenientes dada la arquitectura del modelo diseñado. Posteriormente, se han analizado diferentes versiones del modelo variando cada una de estas características. Por lo tanto, los valores y funciones seleccionados han sido los que mejores resultados han producido para el dataset de entrenamiento empleado, obtenidos mediante un proceso de prueba y error.

En este caso, se ha empleado la técnica de regularización L_2, con un valor de α de 0.01, y un *Dropout* del 25 % (como en el modelo de referencia). Se ha entrenado el

modelo con un tamaño de lote de 32 durante un total de 35 épocas. En cuanto a la función de coste, se ha mantenido el uso de *categorical crossentropy*, al ser un modelo cuyo objetivo es una clasificación multi-clase. Por otro lado, el optimizador empleado ha sido Adam (explicado en el apartado 3.4.2).

Se mantiene el dataset con el que se ha realizado el entrenamiento, el corpus de INSPEC, realizando el mismo preproceso detallado en el apartado 6.2.1.

Al entrenar el modelo descrito se puede comprobar en la figura 6.4 que la función de coste va decrementándose a medida que se ejecutan más épocas. No obstante, respecto al conjunto de validación, se puede observar que la función de coste oscila en torno a un valor más o menos constante tras 5 épocas. Este hecho permite concluir que no es necesario entrenar al modelo durante tantas épocas, aunque, al no aumentar la función de coste, se concluye que no hay sobreajuste de los datos.

Fuente: elaboración propia.

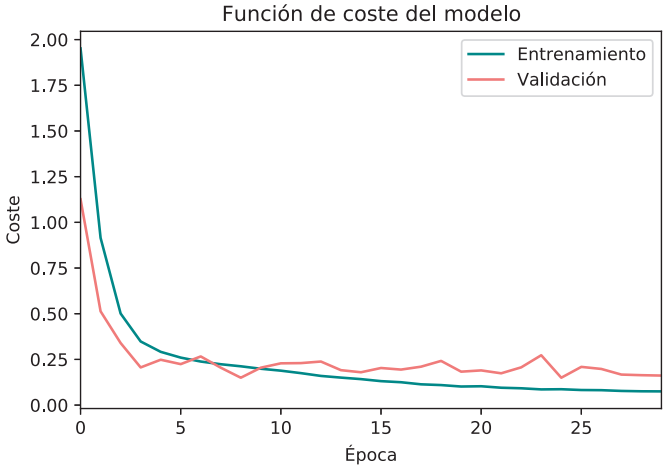


Figura 6.4: Variación de la función de coste en el entrenamiento del Modelo 2.

Los resultados obtenidos con este modelo propuesto se pueden observar en la sección 7.2.2, donde se comprueba que la arquitectura propuesta permite mejorar (alrededor de un 10 % la métrica de *recall* y un 5 % la métrica F1) los resultados obtenidos sobre el conjunto de test de INSPEC.

Capítulo 7

Resultados

En este capítulo se muestran los resultados obtenidos con la implementación de los modelos descritos en el apartado 6.2.2. En la sección 7.1 se introduce la forma de evaluar los rendimientos alcanzados con dichos modelos.

En la implementación de este trabajo, se deben distinguir dos tipos de resultados, en función del corpus sobre el que se han utilizado los modelos creados. En primer lugar, en la sección 7.2 se detallan los resultados obtenidos sobre el dataset etiquetado INSPEC, del cual se pueden obtener unas métricas comparando las terminologías extraídas manualmente y las terminologías seleccionadas por el modelo. Por otro lado, se deben evaluar los resultados obtenidos sobre un corpus del que no se dispone de información inicial relativa a la terminología, como el corpus COVID-19, para lo cual se dedica la sección 7.3.

7.1. Evaluación e interpretación de los resultados

Los modelos de aprendizaje profundo basados en redes neuronales requieren de un proceso iterativo, en el que se van variando los diferentes hiperparámetros y técnicas que se pueden emplear, hasta obtener con la combinación óptima para el conjunto de datos con el que se trata. Para llevar a cabo este proceso iterativo, se deben definir las métricas que permiten comparar los diferentes modelos, con el fin de determinar cuáles son las configuraciones más adecuadas.

Existen distintas métricas para evaluar los resultados de la clasificación de los modelos, por lo que se han escogido aquellas que permiten representar los compromisos hechos en la clasificación. En el caso de los modelos creados en este trabajo, se debe evaluar la clasificación de los tokens de entrada en las tres clases existentes, para determinar si un token pertenece o no a un término.

En este contexto, se entiende que una predicción es verdaderamente positiva si se clasifica una muestra en la clase correspondiente, mientras que será falsamente positiva cuando realmente correspondiera a otra clase. Las predicciones serán falsamente negativas teniendo en cuenta que una de las clases que no se han considerado en la predicción era la clase real de la muestra.

Las métricas empleadas en este trabajo y su interpretación son las siguientes:

- Precisión. Esta métrica representa la proporción de muestras que son verdaderamente positivas, es decir, representa el porcentaje de aciertos al predecir una clase. En concreto, en este trabajo indica el porcentaje de los términos clasificados por el modelo que sí pertenecen al conjunto de términos etiquetados, es decir, seleccionados manualmente por los expertos.
- *Recall*. Indica el ratio de muestras positivas con una clasificación adecuada y, por tanto, como de completos son los resultados. En el contexto de las terminologías, representa el porcentaje de términos que el modelo ha clasificado correctamente sobre la cantidad de términos (correctos y no correctos) que ha clasificado.
- Métrica F1. Se calcula como la media armónica de la precisión y el *recall*, obteniendo así un balance entre las dos métricas. Se utiliza a menudo puesto que simplifica el resultado de un algoritmo de clasificación a una sola métrica.

La definición de estas métricas se puede observar en la ecuación 7.1, donde TP indica las predicciones verdaderamente positivas, FP las falsamente positivas y FN las falsamente negativas.

$$\begin{aligned} precisión &= \frac{TP}{TP+FP} \\ recall &= \frac{TP}{TP+FN} \\ F1 = \frac{2 \cdot precisión \cdot recall}{precisión + recall} &= \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \end{aligned} \tag{7.1}$$

El dataset empleado, tal y como se detalla en la sección 6.1.1, se divide en subconjuntos de entrenamiento, test y validación. Mediante el primero se realiza el entrenamiento de la red neuronal del modelo desarrollado. El conjunto de validación se ejecuta de manera simultánea al de entrenamiento, con el objetivo de determinar los valores de los hiperparámetros más adecuados para dicho modelo, identificando, por ejemplo, problemas de *overfitting* o *underfitting*.

Por último, será el conjunto de test sobre el que se realizarán las métricas mencionadas anteriormente para poder evaluar su funcionamiento y compararlo con los resultados obtenidos en el estado del arte.

No obstante, este procedimiento de evaluación de la terminología solo se puede llevar a cabo sobre un dataset del que se tenga previamente la terminología, para calcular así la precisión, el *recall* y la métrica F1, mediante la comparación de los términos seleccionados por los expertos del dataset, frente a las palabras clave extraídas por el modelo de aprendizaje automático. Estos resultados se exponen en la sección 7.2, indicando los rendimientos obtenidos por los modelos creados en este trabajo sobre el dataset INSPEC, y su objetivo radica en evaluar la bondad de un modelo para la extracción automática de terminología, el cual se pueda emplear para seleccionar los términos de distintos corpus de los que no se obtenga esta información.

Partiendo del modelo seleccionado con el procedimiento mencionado, se procede a la extracción de la terminología del corpus sobre la COVID-19. Puesto que no se pue-

den emplear las métricas de precisión y *recall* para evaluar esta terminología y dado el carácter subjetivo de la extracción de palabras clave, se ha desarrollado la herramienta denominada "Terminologías interactivas" para la comparación de terminologías, creada por Pedro Hernández en paralelo al proyecto *KeyQ*, en el que trabajamos conjuntamente. Esta herramienta tiene en cuenta distintas características relevantes en una terminología, pudiendo calcular distintas métricas relevantes a la hora de evaluar una terminología, las cuales se introducirán en la sección 7.3. Además, el sistema permite que terminólogos expertos del dominio realicen los cambios que consideren necesarios en cada una de las terminologías introducidas, generando de esta manera una comparación entre las terminologías, a partir de lo cual se puede determinar la más adecuada.

En la sección 7.3 se detalla la terminología obtenida sobre el corpus de la COVID-19 de manera automática mediante el modelo propuesto en este trabajo, junto con los resultados obtenidos con la herramienta mencionada.

7.2. Resultados obtenidos sobre el dataset INSPEC

En esta sección se van a analizar y evaluar los resultados obtenidos en la terminología extraída automáticamente por el modelo sobre el conjunto de textos de test del dataset de INSPEC. En este corpus se pueden obtener las métricas de precisión, recall y F1, descritas anteriormente, pudiendo así comparar los rendimientos de los distintos modelos creados para este proyecto, junto con los resultados obtenidos en el estado del arte de la extracción de palabras clave.

Tras un análisis del estado del arte del campo de la extracción de terminología mediante aprendizaje automático (sección 4.3), se ha observado que los resultados de las métricas obtenidos al realizar esta tarea son bastante menores que los resultados obtenidos en otras tareas del procesamiento del lenguaje natural, como la clasificación de textos o el análisis de sentimiento. En concreto, las métricas de precisión, *recall* y F1 del estado del arte relativo a la extracción de términos mediante aprendizaje automático supervisado adquieren valores dentro del rango entre el 30 y el 60 %.

Un hecho a tener en cuenta al obtener los valores de las métricas de evaluación de un modelo de red neuronal es que estos algoritmos son estocásticos. Esto quiere decir que emplean valores aleatorios a la hora de realizar ciertos cálculos, como la inicialización de los pesos de la red, la optimización, o la regularización, por lo que el mismo modelo entrenado sobre los mismos datos puede producir resultados diferentes en cada ejecución.

Existen distintos métodos para reducir la aleatoriedad de los modelos. Los número aleatorios se generan usando un generador de números pseudoaleatorios[1], para lo cual se requiere un valor conocido como semilla para arrancar el proceso. Para asegurar que este valor sea diferente cada vez, los generadores de números aleatorios suelen emplear el tiempo actual en milisegundos. No obstante, se puede establecer un número específico como semilla, para confirmar que se genere la misma secuencia de números aleatorios cada vez que se ejecuta el código.

En la implementación que se ha llevado a cabo en este trabajo, se ha establecido un valor de semilla fijo para tratar de reducir la aleatoriedad. Al emplear Keras se debe establecer una semilla para la librería NumPy, puesto que Keras emplea el generador de números aleatorios de dicha librería. Asimismo, TensorFlow tiene su propio generador de números aleatorios, por lo que también se establece como un valor fijo[2].

No obstante, el empleo de semillas no elimina la aleatoriedad de los modelos neuronales. Existen diversos factores que generan aleatoriedad, como el empleo de distintos núcleos trabajando en paralelo. Debido a esto, para mostrar valores de resultados que sean reales, conviene ejecutar el modelo planteado con los mismos datos varias veces y usar estadísticos para resumir el rendimiento del modelo, y poder así compararlo con otros.

Los siguientes apartados incluyen los resultados obtenidos tras evaluar el conjunto de test del dataset INSPEC ejecutando cada modelo distintas veces, pudiendo aportar así datos estadísticos sobre el rendimiento del modelo.

[1] Un generador de números pseudoaleatorios es una función que genera una secuencia de números aleatorios.

[2] `https://www.tensorflow.org/api_docs/python/tf/random/set_seed`

A continuación, en los apartados 7.2.1 y 7.2.2 se interpretarán los resultados obtenidos en cada uno de los modelos descritos en 6.2.2.1 y 6.2.2.2 tras distintas ejecuciones. Por lo tanto, en estos apartados se hace hincapié en la homogeneidad o no de los resultados al realizar distintas ejecuciones, mediante la observación y el análisis de los estadísticos obtenidos.

En el apartado 7.2.3 se comparan los resultados obtenidos en estos modelos, teniendo en cuenta la distribución de las observaciones de dichos resultados. Asimismo, se relacionarán con los rendimientos alcanzados en el estado del arte.

7.2.1. Resultados del Modelo 1: Reproducción del modelo de Basaldella et al. [65]

En primer lugar, se muestran los resultados obtenidos por el primer modelo propuesto, descrito en el apartado 6.2.2.1 Los resultados indicados se basan en las estadísticas de las métricas de precisión, *recall* y F1 obtenidas tras ejecutar 20 veces cada modelo. Se ha escogido este valor para que los estadísticos obtenidos sean representativos.

En concreto, se han seleccionado los estadísticos de la media aritmética, la mediana, la desviación típica y el coeficiente de variación. La media aritmética es una medida de posición central, por lo que su valor representa el valor resultado del sumatorio de todas las observaciones entre el número total de datos. De manera similar, la mediana representa el valor central al ordenar las observaciones según su valor. La desviación típica o estándar (σ) permite obtener la dispersión de las observaciones y se calcula mediante la ecuación 7.2, donde n es la cantidad de observaciones (en este caso, 20), x_i es el valor de cada una de las observaciones y $\bar{x}$ es el valor de la media aritmética. Por último, el coeficiente de variación es una medida de la distribución de los datos normalizada, por lo que se obtiene como la relación entre la desviación estándar y la media. Gracias a este estadístico, se pueden comparar las dispersiones en las observaciones de distintas poblaciones, como es el caso de las distintas métricas que se van a evaluar: la precisión, el *recall* y la métrica F1.

$$\sigma = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n}}. \tag{7.2}$$

Adicionalmente, se ha obtenido el intervalo de confianza para representar el rango dentro del cual se espera que se encuentren los valores de las métricas en una nueva muestra. Al tener un tamaño de muestra reducido, se calculará el intervalo de confianza según la distribución de probabilidad *T-Student*.

El rango del intervalo de confianza se calcula añadiendo al valor de la media muestral un margen de error, tal y como indica la ecuación (7.3), donde $\bar{x}$ es la media de las observaciones, σ es la desviación típica, n es el número de observaciones y $t_{\alpha/2,n-1}$ es el cuantil[3] de la distribución *T-Student* con una probabilidad de $\alpha/2$ y $n-1$ grados de libertad. Se ha decidido indicar el intervalo de confianza de las poblaciones de las métricas con un nivel de confianza del 95 %, lo cual significa que se puede esperar que el 95 % de las estimaciones serán ciertas.

[3]Los cuantiles son medidas estadísticas de posición que dividen la distribución de una variable en distintas partes, cada una de ellas con el mismo número de frecuencias. Los más habituales son los cuartiles, los deciles y los percentiles.

$$IC = \bar{x} \pm t_{\alpha/2,n-1}\frac{\sigma}{\sqrt{n}}. \tag{7.3}$$

Estos estadísticos, junto con la representación del histograma y del diagrama Box-Plot, conocido también como diagrama de caja y bigotes (*Box and Whisker*), permiten entender la distribución de los valores de las muestras obtenidas.

En el cuadro 7.1 se muestran las métricas de los distintos estadísticos obtenidos tras las 20 ejecuciones mencionadas. La media aritmética permite representar el valor en torno al cual se puede esperar que se obtenga un nuevo resultado ante una nueva ejecución del modelo. No obstante, hay que tener en cuenta la desviación típica, que muestra la dispersión de los datos, con el objetivo de saber si están más centrados en dicha media o más dispersos. Al comparar distribuciones distintas, precisión, *recall* y F1, se ha obtenido el cociente de variación, representando así la dispersión relativa de los conjuntos de valores.

De esta manera, se puede observar en el cuadro 7.1 que los tres estadísticos tienen un coeficiente de variación reducido, bastante menor en el caso de la métrica F1, por lo que la media aporta gran información en cuanto al valor esperado tras una nueva ejecución del modelo. Esta pérdida de variabilidad en los resultados se ha conseguido gracias al empleo de una semilla fija en el generador de números aleatorios del código.

Estadísticos	Precisión	*Recall*	F1
Media (%)	33,88	56,36	42,18
Mediana (%)	33,82	56,55	42,57
Desviación típica (%)	2,37	3,49	1,42
Cociente de variación	0,0699	0,0619	0,0337
Intervalo de confianza (%)	[32,77-34,99]	[54,72-57,99]	[41,51-42,84]

Cuadro 7.1: Estadísticas generales de las métricas (precisión, *recall* y F1) obtenidas tras 20 ejecuciones del Modelo 1.

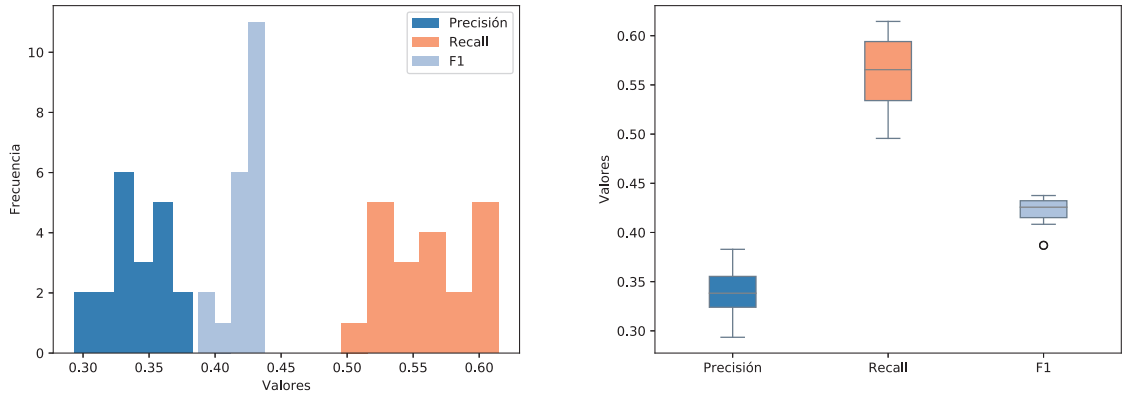
El intervalo de confianza representado en el cuadro 7.1 permite conocer el rango dentro del cual se espera que se encuentre el valor de cada métrica al realizar una nueva ejecución del modelo, con un nivel de confianza del 95 %.

En las figura 7.1 se puede observar el diagrama BoxPlot y el histograma del conjunto de datos. Un histograma representa la frecuencia de las observaciones de cada uno de los valores de una población. En el caso de la figura 7.1a, se pueden observar tres poblaciones correspondientes a las distintas métricas (precisión, *recall* y F1) cada una representada con un color distinto. En el eje x se representan los valores de las observaciones obtenidas en las ejecuciones del modelo, es decir, los valores alcanzados en las distintas métricas, mientras que en el eje y figuran las frecuencias absolutas. Las barras del histograma se representan mediante intervalos, pudiendo así analizar la frecuencia de cada uno de estos intervalos.

Por tanto, con el histograma (figura 7.1a) se puede observar la variabilidad de los resultados obtenidos en las distintas ejecuciones. Cada una de las métricas tiene una distribución distinta. Por ejemplo, la precisión y el *recall* son bi-modales, es decir, hay dos intervalos que tienen una mayor frecuencia en comparación con el resto, mientras que la métrica F1 es uni-modal, al haber un pico que se corresponde con el intervalo que tiene una frecuencia absoluta mayor al resto de intervalos.

El pico del histograma de los datos de F1 está desplazado a la derecha y su frecuencia es mucho mayor que el resto de valores, lo cual implica una notable asimetría negativa. Además, se puede observar que los valores que toma esta métrica están mucho más juntos, por lo que tienen una desviación típica menor, como se puede comprobar en el cuadro 7.1. Por otro lado, los resultados de la precisión y el *recall* están más dispersos, dando lugar a una desviación típica y a un coeficiente de variación mayores.

Fuente: elaboración propia.



(a) Histograma.

(b) Diagrama BoxPlot.

Figura 7.1: Diagramas de los valores estadísticos de las métricas (precisión, *recall* y F1) obtenidos con 20 ejecuciones del Modelo 1. La subfigura (a) muestra el histograma y la subfigura (b) el diagrama de BoxPlot.

De manera similar, el diagrama BoxPlot (7.1b) muestra la distribución de los resultados, pudiendo observar la dispersión y la simetría de las observaciones de cada población. Al igual que en el histograma, se han reproducido tres poblaciones (precisión, *recall* y F1), cada una de ellas en un color distinto y referenciadas en el eje x. En este diagrama, el eje y representa los valores de las observaciones obtenidas en las ejecuciones del modelo, es decir, los valores alcanzados en las distintas métricas.

En cada una de las poblaciones se puede distinguir una caja rectangular, cuya longitud viene determinada por la diferencia entre los valores del el primer y del tercer cuartil[4], definida como recorrido intercuartílico. Estas cajas están divididas por una línea recta que representa el valor de la mediana, la cual coincide con el segundo cuartil. Las líneas que aparecen a continuación de las cajas se denominan bigotes, indicando los valores inferiores al primer cuartil y superiores al tercero. También

[4]Los cuartiles son unas medidas estadísticas de posición que dividen los valores de las observaciones de una población en cuatro grupos, cada uno de ellos con el mismo número de observaciones.

pueden aparecer representados datos atípicos mediante pequeños círculos (como en el caso de la métrica F1), cuyos valores se encuentran alejados del resto. Se pueden observar algunas características ya mencionadas con el histograma, como que la variabilidad de los valores es mucho mayor en la precisión y en el *recall* que en la métrica F1, al tener un tamaño de la caja y de los bigotes mayor. Además, en ninguna de las cajas la mediana se encuentra en el centro, por lo que los datos no son simétricos. En el caso de la métrica F1, la línea de la mediana se encuentra en desplazada hacia arriba, lo cual coincide con la asimetría negativa contemplada en el histograma.

7.2.2. Resultados del Modelo 2: Doble capa BiLSTM usando los *word embeddings* de Glove

En este apartado se detallan los estadísticos que se pueden extraer de los resultados obtenidos tras ejecutar 20 veces el modelo con la arquitectura mejorada propuesta (denominado Modelo 2), detallada en el apartado 6.2.2.2. Se ha escogido el mismo valor de reproducciones que en el apartado anterior, con el objetivo de poder realizar una comparativa equitativa y que los estadísticos obtenidos sean representativos.

La mejora que incluye el Modelo 2 está determinada por el empleo de dos capas BiLSTM formadas por 200 neuronas cada una. Otra diferencia respecto al modelo reproducido consiste en el optimizador empleado en el entrenamiento, los autores emplearon RMSProp, mientras que en el modelo propuesto se ha usado el optimizador Adam, ya que, como se introdujo en el apartado 3.4.2, ha permitido obtener mejores resultados en el estado del arte.

A continuación, se muestran en el cuadro 7.2 y en la figura 7.2 los estadísticos utilizados en el apartado anterior 6.2.2.1, obtenidos en este caso tras 20 ejecuciones del Modelo 2. Se puede comprobar que el coeficiente de variación de las tres poblaciones (precisión, *recall* y F1) es bastante reducido, siendo menor en la métrica F1, al igual que ocurría con los estadísticos obtenidos en el Modelo 1 (cuadro 7.1). Sin embargo, el coeficiente de variación es todavía menor en los estadísticos del Modelo 2, lo cual indica que la variación que se puede esperar de los resultados es menor en este caso, es decir, se puede esperar que al realizar una nueva ejecución del modelo se obtengan valores de las métricas similares a las medias representadas en el cuadro 7.2.

Esta característica también se puede comprobar mediante el intervalo de confianza, el cual representa un rango de valores reducido dentro del cual se puede esperar que se encuentre una nueva estimación con un nivel de confianza del 95 %.

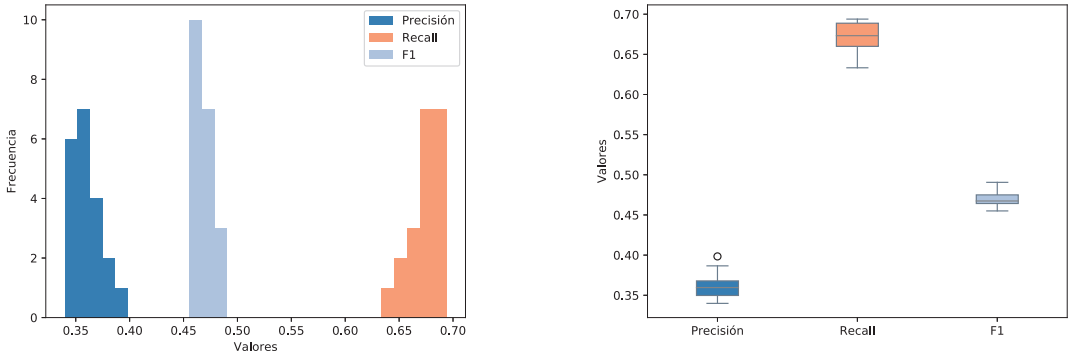
Estadísticos	Precisión	*Recall*	F1
Media (%)	36,11	67,34	46,97
Mediana (%)	35,96	67,32	46,74
Desviación típica (%)	1,48	1,67	0,98
Cociente de variación	0,0415	0,0252	0,0207
Intervalo de confianza (%)	[35,41-36,81]	[66,55-68,14]	[46,51-47,43]

Cuadro 7.2: Estadísticas generales de las métricas (precisión, *recall* y F1) alcanzadas con 20 ejecuciones del Modelo 2.

Al igual que con los resultados del Modelo 1, se ha generado el histograma y el diagrama BoxPlot de las observaciones obtenidas tras las distintas ejecuciones del Modelo 2, con la finalidad de poder observar la distribución de los resultados obtenidos.

El histograma (figura 7.2a) permite comprobar cómo es la variabilidad de los resultados obtenidos en las ejecuciones de cada métrica. De manera similar al Modelo 1, el histograma de la precisión y del *recall* son bi-modales, ya que existe dos intervalos cuya frecuencia absoluta supera a la del resto de intervalos. No obstante, los intervalos que tienen una mayor frecuencia son simultáneos, lo cual podría considerarse como una tendencia uni-modal. En el caso de la precisión, se observa una asimetría positiva, al haber más valores separados de la media por el lado derecho, ocurriendo al contrario con el *recall*, que presenta una asimetría negativa. Respecto a la métrica F1 se puede incidir en el hecho de que los valores se distribuyen entre unos pocos intervalos, lo cual indica una desviación típica y un cociente de variación reducidos, como se puede observar en el cuadro 7.2.

Fuente: elaboración propia.



(a) Histograma.

(b) Diagrama BoxPlot.

Figura 7.2: Diagramas de los valores estadísticos de las métricas (precisión, *recall* y F1) calculados tras 20 ejecuciones del Modelo 2. La subfigura (a) representa el histograma y la subfigura (b) el diagrama de BoxPlot.

De manera similar, el diagrama BoxPlot representado en la figura 7.2b permite obtener prácticamente las mismas conclusiones extraídas del histograma. Las asimetrías de la precisión y el *recall* se perciben mediante la distinta longitud de los bigotes y de las secciones de cada una de sus cajas, siendo el bigote más largo el superior en el caso de la precisión (asimetría positiva) y el bigote inferior en el *recall* (asimetría negativa). El hecho de que la caja y los bigotes del diagrama de la métrica F1 sean menores indica el reducido valor de la desviación típica, al tener unas observaciones menos dispersas. En la población obtenida de esta métrica también se observa una asimetría positiva, al estar desplazada la línea de la mediana hacia la sección inferior.

7.2.3. Comparativa de los resultados obtenidos en los distintos modelos

Tras haber introducido los resultados obtenidos en los dos modelos implementados en este trabajo, teniendo en cuenta la dispersión y homogeneidad de los resultados al realizar distintas ejecuciones de los modelos, en este apartado 7.2.3 se comparan dichos resultados relacionándolos también con los rendimientos logrados mediante modelos similares del estado del arte.

En la literatura, se han desarrollado distintas arquitecturas neuronales para la resolución de la tarea de la extracción automática de terminologías. El estudio realizado en este campo (descrito en el apartado 4.3.2), ha permitido contemplar que los resultados de las métricas obtenidos al realizar esta tarea son bastante menores que los rendimientos adquiridos en modelos destinados a la resolución de otras tareas del procesamiento del lenguaje natural, como la clasificación de textos o el análisis de sentimiento. En concreto, las métricas de precisión, *recall* y F1 del estado del arte relativo a la extracción de términos mediante aprendizaje automático supervisado adquieren valores dentro del rango entre el 30 y el 60 %.

En el cuadro 7.3 se han representado los resultados obtenidos en distintos modelos basados en arquitecturas BiLSTM empleando los *word embeddings* de Glove. Se muestran, en primer lugar, los resultados obtenidos por los autores del artículo de Basaldella et al. [65], con una arquitectura formada por una única capa BiLSTM con 300 neuronas.

Asimismo, se pueden observar los resultados obtenidos en este trabajo al reproducir la arquitectura del artículo mencionado, indicando el intervalo de confianza de cada métrica con un nivel de confianza del 95 % calculado con 20 observaciones de las ejecuciones del modelo. La implementación de este modelo se ha desarrollado en el apartado 6.2.2.1, donde se detalla tanto su estructura, formada básicamente por una única capa BiLSTM con 300 neuronas, como el proceso llevado a cabo para su entrenamiento. Se puede observar que los valores obtenidos con el modelo reproducido (Modelo 1) incluyen a los resultados proporcionados por los autores, lo cual permite concluir que el modelo se ha conseguido reproducir adecuadamente.

En el cuadro 7.3 también se han incluido los resultados obtenidos con el modelo descrito en el apartado 6.2.2.2, denominado Modelo 2, cuya estructura cuenta con dos capas BiLSTM formadas por 200 neuronas cada una. Otra diferencia importante a destacar respecto al Modelo 1 consiste en el empleo del optimizador Adam en el entrenamiento del algoritmo, el cual se ha observado que permite obtener mejores resultados en la literatura del ámbito del procesamiento del lenguaje, como se menciona en el apartado 3.4.2. Al igual que con el Modelo 1, se han indicado en el cuadro 7.3 los límites del intervalo de confianza de cada métrica calculados con 20 observaciones, con un nivel de confianza del 95 %.

El cuadro 7.3 permite extraer algunas conclusiones relativas al empleo de modelos neuronales con una arquitectura basada en redes LSTM bidireccionales junto con los *word embeddings* de Glove como representación vectorial de las palabras. Se observa que el valor de *recall* es bastante mayor que el de precisión. En la sección 7.1 se precisaron los significados de estas métricas, así como las ecuaciones que permiten calcularlas. En la ecuación (7.4) se vuelve a introducir la formulación de las métricas

Modelos	Precisión (%)	*Recall* (%)	F1 (%)
Modelo del artículo [65]	34,0	57,8	42,8
Modelo 1 (Intervalo Confianza)	[32,77-34,99]	[54,72-57,99]	[41,51-42,84]
Modelo 2 (Intervalo Confianza)	[35,41-36,81]	[66,55-68,14]	[46,51-47,43]

Cuadro 7.3: Comparativa de las métricas de los modelos basados en arquitecturas BiLSTM con Glove *embeddings*. Se representan los resultados proporcionados por el artículo de referencia [65], además de los resultados obtenidos en los modelos implementados en este trabajo.

de precisión y *recall*, para facilitar su comprensión.

$$\begin{aligned} precisión &= \frac{TP}{TP+FP} \\ recall &= \frac{TP}{TP+FN} \end{aligned} \tag{7.4}$$

Se puede comprobar que la diferencia entre ambas métricas radica en el segundo elemento del denominador, puesto que en la precisión se tienen en cuenta los falsos positivos (FP), mientras que en el *recall* se consideran los falsos negativos (FN). Por tanto, es evidente que en estos modelos los falsos positivos superan a los falsos negativos.

Esta afirmación tiene sentido en la tarea de extracción de terminología, puesto que los falsos positivos se corresponden con aquellos términos que el modelo selecciona de manera automática, pero que no coinciden con ningún término extraído manualmente por los expertos que desarrollan la terminología del dataset etiquetado. La coincidencia entre términos debe ser total para que se considere una predicción verdaderamente positiva (TP), lo cual es relativamente difícil al tratarse del lenguaje natural. Esta dificultad ocurre debido a que es posible que un término predicho por el modelo tenga el mismo significado que uno seleccionado manualmente, pero la formulación del lenguaje varíe en aspectos morfosintácticos, como podría ser el orden de las palabras o la forma léxica de las palabras empleadas. A menudo, este problema se trata de reducir al representar los tokens mediante la raíz de las palabras, evitando así posibles divergencias entre las palabras debidas a su terminación, como, por ejemplo, el empleo de plural o singular. No obstante, esta técnica no consigue resolver el problema de manera completa, aunque sí lo reduzca.

Por tanto, como se puede advertir en el cuadro 7.3 el reducido valor de la precisión genera que la métrica F1, la cual se calcula como la media aritmética de la precisión y el *recall*, no alcance valores mayores como los que se obtienen con el *recall*.

En el estado del arte, se ha analizado la tarea de extracción de términos con el objetivo de estudiar detalladamente la dificultad de este problema que genera unos rendimientos menores en las métricas de evaluación en comparación con otras tareas del procesamiento del lenguaje natural. Se ha podido observar que existen varios factores relacionados con el corpus sobre el que se realiza la tarea que afectan a la dificultad de la selección de terminologías [49]. Por ejemplo, una mayor longitud del corpus del que se pretenden extraer las palabras clave genera que la dificultad de la tarea aumente, al existir más candidatos a términos. Un factor que puede facilitar la tarea consiste en el empleo de un corpus estructurado, por ejemplo, formado por

artículos científicos, ya que se los términos pueden aparecer con mayor probabilidad en determinadas posiciones, como en el abstract inicial. En general, el empleo de artículos científicos en el corpus de extracción de terminologías simplifica la tarea, ya que muchos de los términos en un documento están relacionados entre ellos.

Debido a esto, el empleo del dataset INSPEC, el cual se basa en artículos de revistas científicas, para el entrenamiento, prueba y validación del modelo ha permitido obtener unos resultados estables. Adicionalmente, debido a que el corpus sobre el que se ha utilizado el modelo está compuesto por artículos científicos (corpus sobre la COVID-19), la elección del dataset INSPEC permite que el formado de los documentos con los que se ha entrenado y validado el modelo neuronal sea similar al de los documentos sobre los que se predice una nueva terminología.

A pesar de las dificultades mencionadas relacionadas con la extracción de términos, los resultados que se han conseguido adquirir mediante los modelos implementados en este Trabajo de Fin de Máster, han alcanzado a los resultados obtenidos en los modelos similares analizados en el estado del arte, tal y como se ha comentado en el cuadro 7.3. Se debe destacar que los resultados con los que se comparan directamente los rendimientos obtenidos en el desarrollo de este trabajo son aquellos adquiridos en la literatura mediante modelos basados en redes BiLSTM, empleando representaciones vectoriales fijas, como Glove y Word2vec. El empleo de representaciones más complejas, como los *word embeddings* contextuales, ha permitido a algunos autores obtener grandes avances en el desempeño de tareas de aprendizaje automático en el dominio del procesamiento del lenguaje y, en concreto, en la extracción de terminologías. En el apartado 4.3.2 se han comparado las últimas arquitecturas implementadas en el estado del arte para la selección de términos, pudiendo observar que el empleo de las representaciones vectoriales del modelo pre-entrenado de BERT junto con una capa CRF consigue un resultado de la métrica F1 en torno al 59 %. Por lo tanto, se ha determinado una línea de trabajo futura que consiste en el uso de los *word embeddings* de BERT junto con una capa CRF para mejorar los modelos implementados en ese Trabajo de Fin de Máster.

7.3. Resultados obtenidos sobre el corpus COVID-19

En esta sección se van a analizar y evaluar los resultados obtenidos al emplear los modelos implementados en este trabajo para predecir la terminología automáticamente sobre un corpus del cual no se dispone de una terminología seleccionada por expertos, pero cuya extracción puede ser de gran utilidad. Este es el caso de los corpus de Airbus y de la COVID-19, los cuales se introdujeron en los apartados 6.1.2 y 6.1.3. En concreto, se muestran los resultados alcanzados con el corpus sobre la COVID-19, al tratarse de un dataset público.

Las terminologías extraídas automáticamente sobre estos documentos, al no existir un conjunto de términos seleccionados por terminólogos, no pueden ser evaluadas mediante las métricas de precisión, *recall* y F1, definidas en la sección 7.1. Para solucionar este inconveniente, se ha desarrollado una herramienta interactiva que permite tratar con grandes corpus y sus terminologías asociadas. Este sistema ha sido desarrollado por Pedro Hernández, en paralelo al proyecto *KeyQ* en el cual hemos trabajado de manera conjunta. En términos generales, el sistema proporciona una interfaz interactiva sobre la que se puede indicar tanto el corpus como la terminología asociada con los que se quiere trabajar. Asimismo, aporta información adicional relacionada con las terminologías al calcular distintas técnicas de puntuación consolidadas sobre los términos, como tf-idf, RAKE y C-Value. Otra característica de esta herramienta consiste en el hecho de que permite que expertos del dominio trabajen con las palabras clave, editándolas, contextualizándolas y comparándolas, con la finalidad de optimizarlas y obtener conclusiones.

Las técnicas de puntuación mencionadas han sido empleadas en el estado del arte para evaluar y comparar distintas terminologías. En primer lugar, la puntuación tf-idf (*Term Frequency - Inverse Document Frequency*) [47] mide la importancia de una palabra o conjunto de palabras para un documento dentro de una colección, calculando un valor para cada uno de los términos basándose en los documentos del corpus en los que aparece. Se obtiene la frecuencia relativa de los términos en cada documento, denominada $tf(t,d)$; es decir, el número de veces que un término aparece en un texto, así como la frecuencia inversa del documento (designado como $idf(t)$), que indica qué tan inusual o común es ese término dentro del conjunto de documentos.

Estos conceptos están definidos mediante la ecuación (7.5), donde t indica un término, d un documento, N el número total de documentos del corpus y $n(t)$ representa el número de documentos en los que aparece un término t. Cuanto mayor es el valor obtenido de tf-idf, mayor es la trascendencia del término dentro del documento sobre el que se ha calculado.

$$\begin{aligned} tf(t,d) &= \frac{freq(t,d)}{\sum_i freq(t_i,d)} \\ idf(t) &= ln\left(\frac{N}{n(t)}\right) \\ tf-idf(t,d) &= tf(t,d)\cdot idf(t) \end{aligned} \tag{7.5}$$

Otra de las métricas utilizadas en la evaluación de terminologías es el C-Value, un estadístico que combina información lingüística y estadística [78], cuya aplicación se ha centrado en terminologías de origen técnico, al dotar a los términos más técnicos una mayor puntuación [79]. Su definición se indica en la ecuación (7.6), donde t_1 es el término cuyo C-Value se quiere calcular, $|t_1|$ es el número de palabras en t_1, $f(t_1)$ es la frecuencia de t_1 en un documento, S_1 es el conjunto de términos que contienen a t_1 y t_2 es un término perteneciente al conjunto S_1. Esta métrica acentúa la habitual métrica de frecuencia relativa de un término, lo cual la hace sensible a los términos contenidos, es decir, aquellas palabras clave que forman parte de otros términos con una longitud mayor, favoreciendo así a los términos multi-palabra.

$$CValue(t_1) = \begin{cases} \log_2(|t_1|) \cdot f(t_1) & \text{si } t_1 \text{ no contenido} \\ \log_2(t_1|) \cdot \left(f(t_1) - \frac{1}{|S_1|}\sum_{t_2 \in S_1} f(t_2)\right) & \text{si } t_1 \text{ contenido} \end{cases} \tag{7.6}$$

Por último, se introduce la métrica conocida como RAKE [80], cuyo nombre viene de *Rapid Automatic Keyword Extraction*, debido a que se trata no solo de una puntuación de terminologías, sino también de un método de extracción de palabras clave que tiene una velocidad de ejecución elevada. Este método emplea *stopwords* y delimitadores de frases para detectar candidatos a términos. A partir de estos candidatos, calcula una matriz de co-ocurrencia, donde se indica el número de veces que cada palabra aparece junto con otra da las palabras candidatas dentro de un término. A partir de esta matriz, se obtiene la métrica Rake, la cual beneficia a los términos compuestos por un mayor número de palabras, al dotarles de una mayor puntuación. Estas palabras clave serán, por tanto, más específicas, al contener más palabras que permitan concretar o delimitar el significado del término.

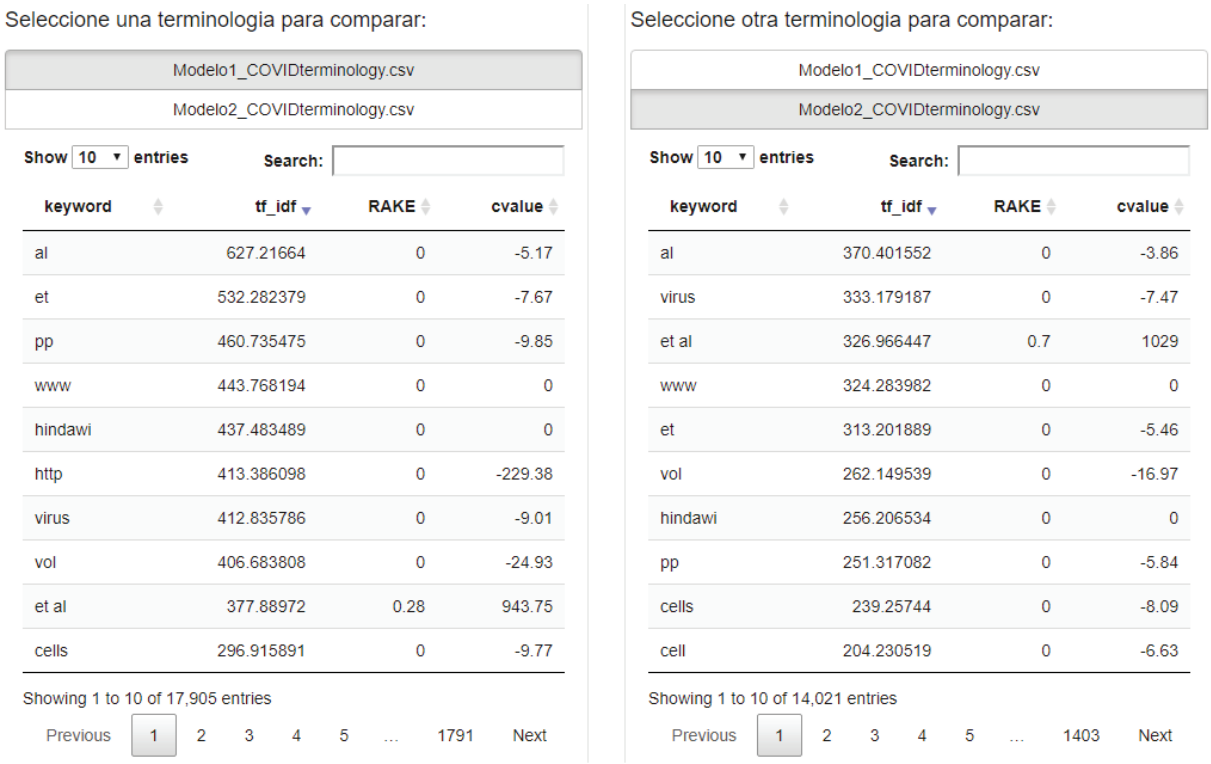
A continuación, se muestran las capturas de la herramienta "Terminologías interactivas" que permite comparar las palabras clave de las distintas terminologías en base a las métricas explicadas en esta sección. Mediante las capturas que se van a representar se realiza la comparativa entre los términos seleccionados por el Modelo 1 (reproducción del paper de referencia detallado en el apartado 6.2.2.1) y por el Modelo 2 (mejora del modelo anterior explicado en el apartado 6.2.2.2) sobre el corpus de la COVID-19.

En primer lugar, en la figura 7.3 aparecen los 10 primeros términos de cada una de las terminologías seleccionadas ordenados en orden descendente según la métrica de tf-idf de cada una de las palabras clave. Como se puede observar, esta métrica favorece a los términos más cortos, es decir, formados por un número menor de palabras, ya que, al tener una mayor frecuencia de aparición en los documentos, se incrementa la parte de tf (ecuación 7.5). La incorporación del idf genera que la métrica tf-idf no dote de mayor importancia a los términos más frecuentes que aparecen en todos los documentos, puesto que, por lo general, las palabras más frecuentes serán palabras vacías o *stopwords*.

La lista con los términos con mayor puntuación tf-idf de la figura 7.3 permite comprobar que esta métrica no incluye *stopwords* pero sí beneficia a palabras específicas del dominio del que trata el corpus, como "*virus*" o "*cells*". No obstante, aparecen térmi-

nos con una alta puntuación de tf-idf que no aportan gran información sobre el tema del corpus, por lo que no deberían considerarse términos, pero que sí son propios del formato de los documentos por los que está formado el corpus de la COVID-19, es decir, artículos científicos, como son las palabras seleccionadas de "*et al*" (al citar un paper con más de un autor), "*vol*" o "*pp*" (volumen y página de los artículos).

Fuente: elaboración propia.



Seleccione una terminologia para comparar:

Modelo1_COVIDterminology.csv

Modelo2_COVIDterminology.csv

Show 10 entries Search:

keyword	tf_idf	RAKE	cvalue
al	627.21664	0	-5.17
et	532.282379	0	-7.67
pp	460.735475	0	-9.85
www	443.768194	0	0
hindawi	437.483489	0	0
http	413.386098	0	-229.38
virus	412.835786	0	-9.01
vol	406.683808	0	-24.93
et al	377.88972	0.28	943.75
cells	296.915891	0	-9.77

Showing 1 to 10 of 17,905 entries

Previous 1 2 3 4 5 … 1791 Next

Seleccione otra terminologia para comparar:

Modelo1_COVIDterminology.csv

Modelo2_COVIDterminology.csv

Show 10 entries Search:

keyword	tf_idf	RAKE	cvalue
al	370.401552	0	-3.86
virus	333.179187	0	-7.47
et al	326.966447	0.7	1029
www	324.283982	0	0
et	313.201889	0	-5.46
vol	262.149539	0	-16.97
hindawi	256.206534	0	0
pp	251.317082	0	-5.84
cells	239.25744	0	-8.09
cell	204.230519	0	-6.63

Showing 1 to 10 of 14,021 entries

Previous 1 2 3 4 5 … 1403 Next

Figura 7.3: Captura de la herramienta comparativa de terminologías. Muestra los términos con mejor puntuación tf-idf obtenidos por los modelos 1 y 2 sobre el corpus COVID-19.

De manera similar, la figura 7.4 muestra los 10 términos con mayor C-Value, en orden descendente, de las terminologías extraídas por los modelos 1 y 2 desarrollados en este trabajo.

Analizando la ecuación (7.6) se puede deducir que esta métrica favorece a los términos multi-palabra, ya que, la incorporación del logaritmo del número de palabras del término genera que la puntuación C-Value sea nula si el término está formado por una única palabra. Este hecho se ve reflejado en la figura 7.4 puesto que no aparece ningún término basado en una sola palabra.

Asimismo, el C-Value dota con un mayor valor a los términos que son específicos y que representan por sí mismos una información concreta del dominio. La manera en la que se realiza esto viene dada por la ecuación (7.6), donde se favorecen aquellas palabras clave que no están contenidas en ningún otro término y se perjudica a los términos que están contenidos en otros, sobre todo si estos otros términos tienen una frecuencia elevada en los documentos.

En la figura 7.4 se puede comprobar que los términos que tienen un mayor C-Value son aquellos formados por dos palabras y que son específicos del dominio del corpus sobre el que se ha extraído la terminología, como "*public health*", "*repiratory syndrome*" o "*local transmission*".

En cuanto a la comparativa que se puede realizar sobre los distintos términos seleccionados por el Modelo 1 y por el Modelo 2, se observa que muchos de los términos con mayor C-Value aparecen en ambas terminologías, los cuales se corresponden con palabras clave del dominio de la enfermedad COVID-19. Sin embargo, el Modelo 1 ha seleccionado como término "*e u*", lo cual es una predicción falsamente positiva que no aparece en la terminología del Modelo 2.

Fuente: elaboración propia.

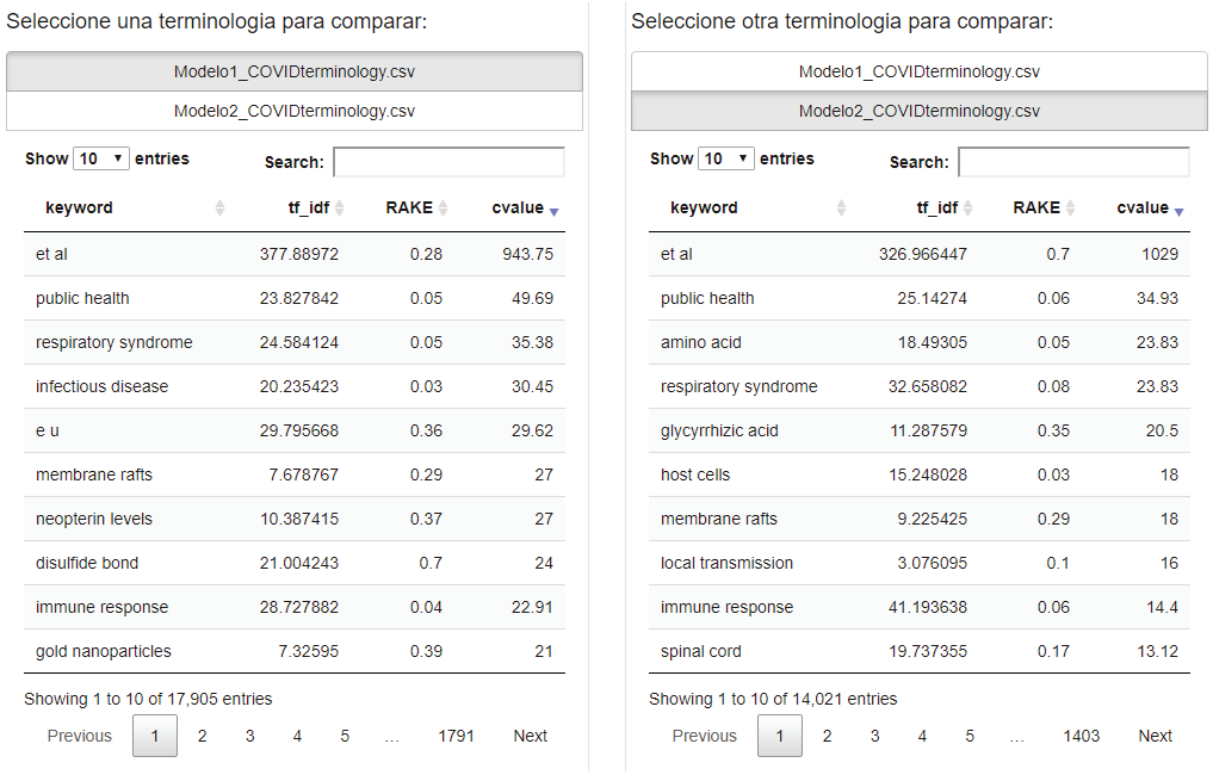


Figura 7.4: Captura de la herramienta comparativa de terminologías. Representa las palabras clave con mayor C-Value de las terminologías extraídas por el Modelo 1 y por el Modelo 2 sobre el corpus COVID-19.

Por último, en la figura 7.5 se muestra la lista de los 10 términos con mayor puntuación RAKE de cada una de las terminologías. Tal y como se ha mencionado al definir el método de extracción de terminologías RAKE, este método favorece a los términos más específicos, los cuales tienden a ser los que están formados por un mayor número de palabras. Esta característica se puede ver en los términos con un valor RAKE más alto, ya que todos ellos tienen una longitud mayor de tres palabras, tratándose de palabras clave muy concretas, como "*glycyrrhizin modified sweetened beverage products*" o "*torilidis fructus ethanol extract*".

Una desventaja de emplear la métrica de RAKE para seleccionar los mejores términos es que, al favorecer las palabras clave más específicas, a menudo se dota con una puntuación alta a un conjunto de palabras tan específico que no tiene por qué ser considerado como término. Se puede ver un ejemplo de estas predicciones erróneas en la figura 7.5, como la selección de "*infected number 1200 g4 1000 g3*" como término por el Modelo 1, o la extracción de "*observers could add comments*", por el Modelo 2.

Fuente: elaboración propia.

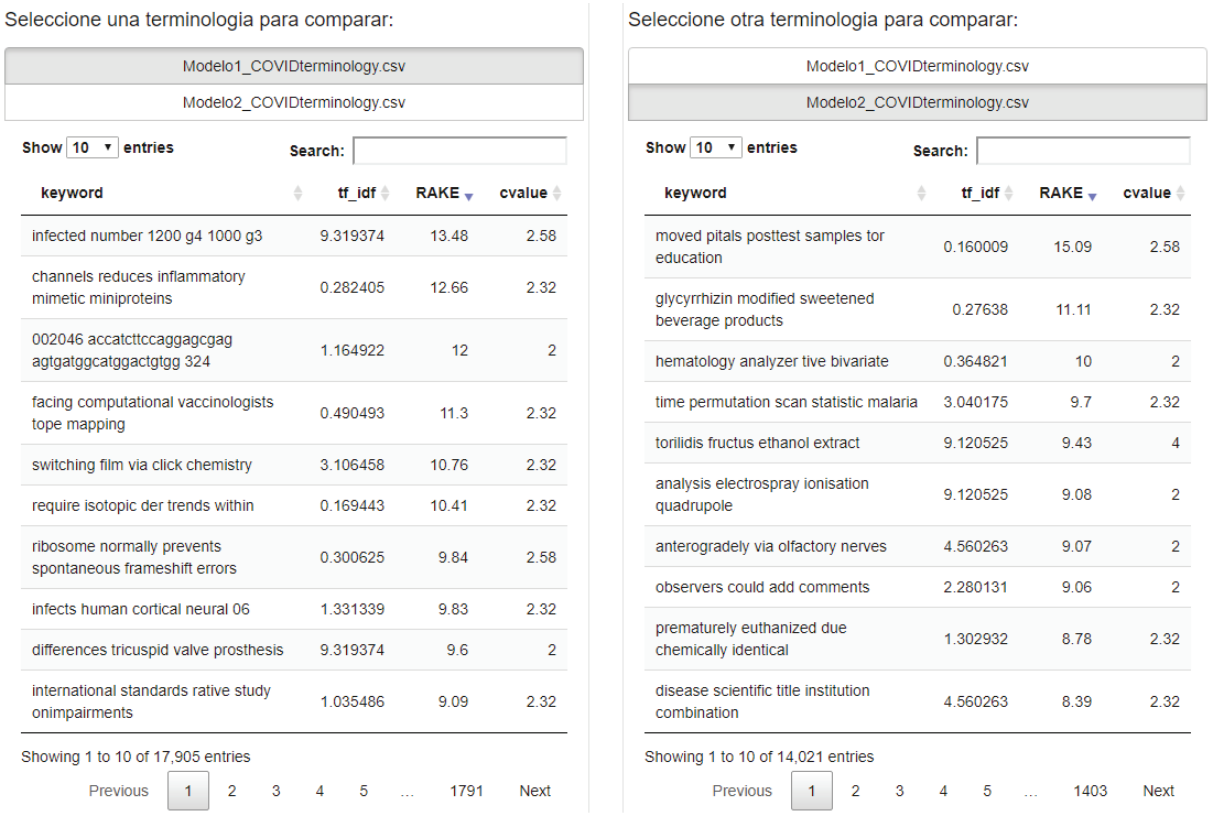


Figura 7.5: Captura de la herramienta comparativa de terminologías. Aparecen los términos con mejor puntuación RAKE seleccionados por los modelos 1 y 2 sobre el corpus COVID-19.

Tras analizar los términos obtenidos por cada uno de los modelos implementados en este trabajo mediante la herramienta comparativa denominada "Terminologías interactivas" se puede afirmar que la evaluación de los términos mediante las métricas de tf-idf, RAKE y C-Value, permite valorar y seleccionar las palabras clave más adecuadas en función de distintas características.

Como se ha podido observar, la métrica de tf-idf favorece los términos más frecuentes, mientras que la métrica RAKE beneficia a los términos más específicos. Por otro lado, el C-Value tiene en cuenta tanto la frecuencia de los términos, como el hecho de si están contenidos en otros términos más concretos. Como se ha podido examinar en la figura 7.4, los términos con mayor C-Value captan información importante del dominio sin ser demasiado específicos. Por lo tanto, se puede deducir que el análisis en conjunto de los resultados obtenidos por las métricas del C-Value, tf-idf y

RAKE aportan una gran cantidad de información para evaluar la terminología de un dominio concreto.

Se debe destacar el hecho de que una terminología debe ser evaluada por un experto en el dominio. Para ello, la herramienta comparativa de terminologías permite que terminólogos modifiquen y evalúen las distintas terminologías para obtener conclusiones más precisas. La comparativa realizada en este apartado sirve para examinar los resultados proporcionados por los modelos de extracción de términos implementados en este trabajo en base a las métricas proporcionadas.

Adicionalmente, la herramienta se ha empleado para evaluar y comparar las terminologías obtenidas por dos métodos distintos: modelos basados en redes neuronales, para lo cual se ha empleado el Modelo 2 desarrollado en este trabajo, y modelos basados en expresiones regulares. Esta comparativa se trata de la contribución principal del paper elaborado junto con Pedro Hernández y Mariano Rico para la conferencia *MNLP 2020 (4th IEEE Conference on Machine Learning and Natural Language Processing)*.

Quiero destacar que, a fin de garantizar la reproducibilidad de los experimentos realizados, tanto el artículo como los datos, código, y resultados experimentales llevados a cabo se encuentran disponibles en el siguiente enlace: `http://keyq.linkeddata.es`.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

En este Trabajo de Fin de Máster se ha llevado a cabo el desarrollo de un modelo de aprendizaje automático de extracción de términos, cuya finalidad consiste en la detección de las palabras o expresiones clave de un texto. La implementación desarrollada en este trabajo se ha centrado en los modelos de aprendizaje automático más empleados en los últimos años, debido a su gran potencial: el aprendizaje profundo o *Deep Learning* de las redes neuronales.

La motivación del desarrollo de este modelo ha sido la mejora de la herramienta de búsqueda sobre documentos de dominios específicos, creada en el proyecto *KeyQ* llevado a cabo en el *AI.nnovation Space* de la Universidad Politécnica de Madrid. La utilización de los términos (simples o compuestos) extraídos a partir de los documentos que forman el corpus sobre el que se pone en funcionamiento la herramienta de búsqueda aporta un gran valor añadido, ya que, al tratarse de dominios específicos, añaden información que permite que el usuario realice consultas más específicas.

La implementación del modelo neuronal de predicción de terminologías se ha llevado a cabo tras un exhaustivo análisis de los distintos modelos creados en el estado del arte, lo cual ha permitido partir de una base de conocimiento sólida. Debido a esto, el Modelo 1 creado en este trabajo se ha centrado en la reproducción de una de las arquitecturas producidas en el estado del arte, basada en redes LSTM bidireccionales junto con los *word embeddings* de Glove.

Sobre este modelo se han medido sus rendimientos siguiendo las métricas más habituales en los problemas de clasificación de aprendizaje supervisado sobre el conjunto de datos de prueba. Se ha decidido aportar los resultados de estas métricas tras 20 ejecuciones del modelo, obteniendo así datos y diagramas estadísticos que representan la dispersión de los valores de las métricas tras las distintas ejecuciones. Este hecho conviene ser destacado ya que, dada la aleatoriedad que presentan los algoritmos basados en redes neuronales, no sería preciso aportar un único valor de las métricas. Los estadísticos obtenidos han demostrado que el Modelo 1 ha reproducido correctamente los resultados del artículo de referencia del estado del arte, en el cual se indicaba un valor de F1 de 42,8 %, tras haber obtenido en la reproducción llevada a cabo en este trabajo un intervalo de confianza cuyos límites se corresponden con 41,51 y 42,84 %, con un nivel de confianza del 95 %. Por lo tanto, se puede espe-

rar que una nueva ejecución del modelo implementado genere un rendimiento de la métrica F1 dentro del intervalo mencionado, con una probabilidad del 95%.

Tras la implementación satisfactoria del Modelo 1, se desarrolló un nuevo modelo (denominado Modelo 2) con una arquitectura más profunda, al contar con dos capas ocultas de tipo BiLSTM, lo cual permite que el modelo aprenda características específicas de los datos de entrada. Adicionalmente, se ha empleado el optimizador Adam, el cual combina las técnicas de RMSProp y Momentum, al haber demostrado un mejor rendimiento en la literatura científica. Al igual que con el modelo anterior, se han mostrado los estadísticos de los resultados obtenidos con 20 ejecuciones, lo cual ha permitido observar que la variación de las métricas entre distintas ejecuciones es incluso menor en el Modelo 2 que en el 1. En este caso, los valores obtenidos en la métrica F1 oscilan entre el 46,51 y el 47,43% con un nivel de confianza del 95%, lo cual demuestra que la arquitectura propuesta mejora el rendimiento en la tarea de extracción de terminologías.

Esto permite que concluir que el uso de una red neuronal más profunda, junto con el empleo de un optimizador más completo, han dado lugar a un modelo con una mejor generalización al predecir la terminología de un conjunto de datos distinto al de entrenamiento. Cabe destacar que el aumento de la profundidad y la complejidad de la red no siempre conllevan unos mejores resultados, puesto que a menudo genera que el modelo se adapte demasiado a los datos del entrenamiento, produciéndose el fenómeno de *overfitting*. No obstante, en la etapa de validación del modelo implementado en este trabajo se ha observado que el empleo del modelo sobre un conjunto de datos distinto del entrenamiento no ha desembocado en un sobreajuste, ya que la función de coste disminuye o se mantiene constante a medida que se entrena el modelo, como se ha podido comprobar en la figura 6.4 del apartado 6.2.2.2.

Respecto a los resultados obtenidos en las métricas por los modelos mencionados, se debe destacar que, al igual que ocurre con los modelos desarrollados en el estado del arte de la extracción de términos, los rendimientos son bastante reducidos en comparación con los resultados sobre otras tareas del procesamiento del lenguaje natural. Uno de los motivos principales por los que estos resultados son tan bajos es debido a que estas métricas se realizan mediante criterios de estricta correspondencia, es decir, para que una predicción se considere como verdadera positiva, el término extraído debe coincidir exactamente con uno de los términos etiquetados del dataset.

Asimismo, cabe señalar que, en todos los casos, el *recall* es bastante mayor que la precisión. Por ejemplo, en el caso del Modelo 2, la media de la precisión es del 36,11%, mientras que la de *recall* es del 67,34%. Esto ocurre debido a que las predicciones que resultan ser falsamente positivas superan a las falsas negativas, lo cual tiene sentido en la extracción de términos, ya que muchos términos se evalúan como falsos positivos (términos que el modelo selecciona pero que en la terminología etiquetada manualmente no aparecen) incluso aunque exista un término con el mismo significado en la terminología etiquetada. Este hecho es consecuencia de la incapacidad de estas métricas de medir que una predicción sea correcta cuando no coincide con ningún término etiquetado, por tener un orden distinto o ser un subconjunto del término etiquetado, por ejemplo, aunque ambos tengan el mismo significado semántico. Por tanto, el limitado valor de la métrica F1 ocurre al haber muchas predicciones falsamente positivas, lo que genera que la precisión se reduzca.

Los resultados obtenidos en este trabajo permiten comprobar que el empleo de una red neuronal más profunda mejora el rendimiento de los modelos del estado del arte basados en el uso de las representaciones de Glove junto con redes LSTM bidireccionales, hasta donde se tiene constancia. Esta comparativa se puede ver en el cuadro 7.3, donde se destaca la gran mejora del *recall*, en torno al 10%, y de la métrica F1, alrededor de un 5%. El hecho de que el incremento de la F1 sea menor ocurre debido a que la mejora conseguida en la precisión es insuficiente. No obstante, los rendimientos obtenidos por el Modelo 2 no superan a los mejores del estado del arte en la extracción de términos, ya que no se han utilizado todas las estructuras o características usadas en los distintos modelos de la literatura, como los *word embeddings* contextuales. Como se ha mencionado, en este trabajo se ha optado por la representación de los tokens mediante Glove, debido a que el balance entre los resultados que alcanzan y las dimensiones de sus vectores es muy productivo y ventajoso.

Después de evaluar los rendimientos obtenidos por los modelos desarrollados y concluir que el Modelo 2 desempeña satisfactoriamente la tarea de extracción automática de términos, se ha utilizado dicho modelo para ejecutar el objetivo final de este Trabajo de Fin de Máster: la predicción de la terminología asociada al corpus de Airbus, perteneciente al proyecto de *KeyQ*, y al corpus sobre la COVID-19, con la finalidad de incorporarlos en la herramienta de búsqueda sobre dominios específicos creada en el proyecto.

Los análisis llevados a cabo en el capítulo 7 de Resultados, permiten deducir que el modelo desarrollado puede tener un buen rendimiento en la generalización sobre otros corpus, siempre que estén redactados en inglés, el formato de los documentos sea similar al de los artículos científicos y el dominio sobre el que traten sea parecido, al no detectar indicios de *overfitting* u otros problemas habituales en algoritmos de aprendizaje automático supervisado. No obstante, se ha empleado un sistema de evaluación y comparativa de terminologías elaborado en el proyecto *KeyQ*, con el propósito de analizar y examinar los términos extraídos automáticamente por los modelos implementados. El estudio de las terminologías realizado sobre este sistema permite obtener algunas conclusiones.

- El modelo neuronal implementado en este trabajo genera terminologías de manera automática sobre distintos corpus de dominios específicos.
- Algunos de los términos seleccionados por el modelo no se corresponden con términos correctos, al incluir palabras o números que no deberían formar parte de las palabras clave, lo cual era de esperar dado el limitado valor de la precisión.
- El uso del sistema de evaluación permite ordenar los términos en base a distintas métricas relevantes en las terminologías, como el C-Value, tf-idf o RAKE, obteniendo así distintas clasificaciones de los mejores términos, en función de las características que se requieran en la terminología.
- Asimismo, a través del sistema, se pueden eliminar o modificar aquellos términos que se consideren erróneos, incompletos o redundantes, generando de esta manera una terminología filtrada más precisa.
- En términos generales se puede concluir que el uso del Modelo 2 creado en este trabajo, junto con el sistema de análisis de terminologías, permite elaborar una terminología adecuada sobre un dominio específico sin tener que recurrir a expertos terminólogos que realicen esta tarea de manera manual.

Estas conclusiones permiten confirmar que, tanto el objetivo general, como los objetivos específicos introducidos en el el capítulo 2 se han cumplido mediante el desarrollo de este trabajo.

Cabe destacar que este Trabajo de Fin de Máster ha permitido realizar varias contribuciones, desde el estudio y análisis exhaustivo del estado del arte en el campo del procesamiento del lenguaje natural, haciendo hincapié en la literatura del aprendizaje automático y la extracción de terminologías. La contribución principal consiste en el desarrollo del modelo neuronal capaz de extraer términos en dominios específicos, el cual, junto con la herramienta comparativa de terminologías, genera unos buenos resultados.

Adicionalmente, la aportación del trabajo desarrollado en este documento ha permitido redactar un artículo científico junto con Pedro Hernández y Mariano Rico para la conferencia *MNLP 2020 : 4th IEEE Conference on Machine Learning and Natural Language Processing*. En este artículo se realiza una comparativa sobre la extracción de terminología basada en distintos enfoques: modelos basados en redes neuronales, para lo cual se ha empleado el Modelo 2 desarrollado en este trabajo, y modelos basados en expresiones regulares.

8.2. Líneas de trabajo futuro

Finalmente, en esta sección se introducen las líneas de trabajo futuro que pueden aportar un valor añadido al trabajo desarrollado, tras haber analizado los resultados obtenidos.

Tal y como se ha podido concluir tras el análisis del estado del arte, la extracción automática de términos se trata de un campo en el que se han conseguido grandes avances en los últimos años, aunque aún sigue siendo una de las tareas de procesamiento del lenguaje natural con los rendimientos más bajos.

En este trabajo, los modelos implementados para resolver esta tarea se han basado en las aportaciones de la literatura científica. Se ha optado por el empleo de representaciones vectoriales de las palabras que no tienen en cuenta el contexto en el que se encuentran, como los *word embeddings* de Glove, debido a que permiten alcanzar buenos resultados en las tareas del procesamiento del lenguaje, sin requerir excesivos recursos, como la memoria RAM o el tiempo necesario para entrenar, validar y predecir terminologías con los modelos.

No obstante, el uso de representaciones vectoriales contextuales, basados en modelos pre-entrandos más complejos, como BERT, ELMo, RoBERTa o ERNIE (descritos en el apartado 4.2.2) ha permitido mejorar los rendimientos de los modelos en distintas tareas del procesamiento del lenguaje, entre las que se incluye la extracción de términos. Por ello, una de las líneas que se pretenden seguir tras el desarrollo llevado a cabo en este trabajo consiste en implementar los modelos explicados en el apartado 6.2.2 con algunas de las representaciones vectoriales contextuales como BERT o RoBERTa, al ser estas unas de las representaciones que han conseguido las mejores métricas en el estado del arte.

De manera similar, se pretende mejorar la estructura de los modelos modificando la última capa de la arquitectura por una estructura CRF, puesto que los estudios

realizados en el estado del arte justifican que su implementación incrementa las métricas de los resultados obtenidos en tareas como el reconocimiento del nombre de entidades (NER) o la extracción de términos.

Por otro lado, la herramienta de evaluación y comparación de terminologías, detallada en la sección 7.3 se quiere emplear para que un grupo de términólogos editen distintas terminologías, eliminando, modificando o añadiendo términos. Las terminologías que se van a emplear corresponden a aquellas obtenidas mediante este trabajo, así como las extraídas mediante expresiones regulares, estudiados e implementados en en paralelo al proyecto *KeyQ*. De esta forma, los expertos podrán obtener conclusiones más precisas en cuanto a los modelos de extracción de palabras clave que mejores resultados obtienen en los dominios específicos que se traten.

Capítulo 9

Referencias

[1] G. T. Doran y col., «There'sa SMART way to write management's goals and objectives», *Management review*, vol. 70, n.º 11, págs. 35-36, 1981.

[2] W. S. McCulloch y W. Pitts, «A logical calculus of the ideas immanent in nervous activity», *The bulletin of mathematical biophysics*, vol. 5, n.º 4, págs. 115-133, 1943.

[3] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[4] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[5] S. Hochreiter y J. Schmidhuber, «Long short-term memory», *Neural computation*, vol. 9, n.º 8, págs. 1735-1780, 1997.

[6] X. Glorot e Y. Bengio, «Understanding the difficulty of training deep feedforward neural networks», en *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, págs. 249-256.

[7] S. Ioffe y C. Szegedy, «Batch normalization: Accelerating deep network training by reducing internal covariate shift», *arXiv preprint arXiv:1502.03167*, 2015.

[8] C. C. Aggarwal, «Neural networks and deep learning», *Springer*, vol. 10, págs. 978-3, 2018.

[9] D. P. Kingma y J. Ba, «Adam: A method for stochastic optimization», *arXiv preprint arXiv:1412.6980*, 2014.

[10] A. M. Turing, «On computable numbers, with an application to the Entscheidungsproblem», *J. of Math*, vol. 58, n.º 345-363, pág. 5, 1936.

[11] C. E. Shannon, «A mathematical theory of communication», *The Bell system technical journal*, vol. 27, n.º 3, págs. 379-423, 1948.

[12] N. Chomsky, «Three models for the description of language», *IRE Transactions on information theory*, vol. 2, n.º 3, págs. 113-124, 1956.

[13] S. K. K. Langer y S. K. Langer, *An introduction to symbolic logic*. Dover Publications New York, 1953.

[14] R. O. Duda, P. E. Hart, N. J. Nilsson y G. L. Sutherland, «Semantic network representations in rule-based inference systems», en *Pattern-directed inference systems*, Elsevier, 1978, págs. 203-221.

[15] J. Giménez y L. Marquez, «Fast and accurate part-of-speech tagging: The SVM approach revisited», *Recent Advances in Natural Language Processing III*, págs. 153-162, 2004.

[16] A. M. Dai y Q. V. Le, «Semi-supervised sequence learning», en *Advances in neural information processing systems*, 2015, págs. 3079-3087.
[17] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent y S. Bengio, «Why does unsupervised pre-training help deep learning?», *Journal of Machine Learning Research*, vol. 11, n.º Feb, págs. 625-660, 2010.
[18] D. Bahdanau, K. Cho e Y. Bengio, «Neural machine translation by jointly learning to align and translate», *arXiv preprint arXiv:1409.0473*, 2014.
[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser e I. Polosukhin, «Attention is all you need», en *Advances in neural information processing systems*, 2017, págs. 5998-6008.
[20] C. D. M. Jeffrey Pennington Richard Socher. (2014). GloVe: Global Vectors for Word Representation, dirección: `https://nlp.stanford.edu/projects/glove/` (visitado 24-01-2020).
[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado y J. Dean, «Distributed representations of words and phrases and their compositionality», en *Advances in neural information processing systems*, 2013, págs. 3111-3119.
[22] S. Ruder. (2018). NLP's ImageNet moment has arrived, dirección: `https://ruder.io/nlp-imagenet/` (visitado 24-01-2020).
[23] R. Socher, M. Ganjoo, C. D. Manning y A. Ng, «Zero-shot learning through cross-modal transfer», en *Advances in neural information processing systems*, 2013, págs. 935-943.
[24] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee y L. Zettlemoyer, «Deep contextualized word representations», *arXiv preprint arXiv:1802.05365*, 2018.
[25] J. Howard y S. Ruder, «Universal language model fine-tuning for text classification», *arXiv preprint arXiv:1801.06146*, 2018.
[26] A. Radford, K. Narasimhan, T. Salimans e I. Sutskever, «Improving language understanding by generative pre-training», *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.
[27] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, «Bert: Pre-training of deep bidirectional transformers for language understanding», *arXiv preprint arXiv:1810.04805*, 2018.
[28] I. Beltagy, A. Cohan y K. Lo, «Scibert: Pretrained contextualized embeddings for scientific text», *arXiv preprint arXiv:1903.10676*, 2019.
[29] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So y J. Kang, «BioBERT: a pre-trained biomedical language representation model for biomedical text mining», *Bioinformatics*, vol. 36, n.º 4, págs. 1234-1240, 2020.
[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer y V. Stoyanov, «Roberta: A robustly optimized bert pretraining approach», *arXiv preprint arXiv:1907.11692*, 2019.
[31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei e I. Sutskever, «Language models are unsupervised multitask learners»,
[32] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov y Q. V. Le, «Xlnet: Generalized autoregressive pretraining for language understanding», en *Advances in neural information processing systems*, 2019, págs. 5754-5764.
[33] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le y R. Salakhutdinov, «Transformer-xl: Attentive language models beyond a fixed-length context», *arXiv preprint arXiv:1901.02860*, 2019.

[34] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian y H. Wu, «Ernie: Enhanced representation through knowledge integration», *arXiv preprint arXiv:1904.09223*, 2019.

[35] Y. Zhang, N. Zincir-Heywood y E. Milios, «World wide web site summarization», *Web Intelligence and Agent Systems: An International Journal*, vol. 2, n.º 1, págs. 39-53, 2004.

[36] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning y E. Frank, «Improving browsing in digital libraries with keyphrase indexes», *Decision Support Systems*, vol. 27, n.º 1-2, págs. 81-104, 1999.

[37] A. Hulth y B. B. Megyesi, «A study on automatically extracted keywords in text categorization», en *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2006, págs. 537-544.

[38] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin y C. G. Nevill-Manning, *Domain-Specific Keyphrase Extraction, to appear in: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[39] S. Beliga, «Keyword extraction: a review of methods and approaches», *University of Rijeka, Department of Informatics, Rijeka*, págs. 1-9, 2014.

[40] M. Dostal y K. Ježek, «Automatic keyphrase extraction based on NLP and statistical method», 2011.

[41] C. D. Manning, P. Raghavan y H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008.

[42] A. Hulth, «Improved automatic keyword extraction given more linguistic knowledge», en *Proceedings of the 2003 conference on Empirical methods in natural language processing*, Association for Computational Linguistics, 2003, págs. 216-223.

[43] S. Beliga, A. Meštrović y S. Martinčić-Ipšić, «An overview of graph-based keyword extraction methods and approaches», *Journal of information and organizational sciences*, vol. 39, n.º 1, págs. 1-20, 2015.

[44] Y. Ying, T. Qingping, X. Qinzheng, Z. Ping y L. Panpan, «A graph-based approach of automatic keyphrase extraction», *Procedia Computer Science*, vol. 107, págs. 248-255, 2017.

[45] B. Baldwin, C. Doran, J. C. Reynar, M. Niv, B. Srinivas y M. Wasson, «EAGLE: An Extensible Architecture for General Linguistic Engineering.», en *ANLP*, 1997, pág. 23.

[46] S. Petrov, D. Das y R. McDonald, «A universal part-of-speech tagset», *arXiv preprint arXiv:1104.2086*, 2011.

[47] J. S. Justeson y S. M. Katz, «Technical terminology: some linguistic properties and an algorithm for identification in text», *Natural language engineering*, vol. 1, n.º 1, págs. 9-27, 1995.

[48] M. Rico, P. Calleja, P. Martın y E. Montiel, «Extracting terminologies in the legal domain: a syntactic pattern-based approach for Spanish», 2019.

[49] K. S. Hasan y V. Ng, «Automatic keyphrase extraction: A survey of the state of the art», en *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, págs. 1262-1273.

[50] S. D. Gollapalli y X.-l. Li, «Keyphrase extraction using sequential labeling», *arXiv preprint arXiv:1608.00329*, 2016.

[51] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin y C. G. Nevill-Manning, «Kea: Practical automated keyphrase extraction», en *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, IGI global, 2005, págs. 129-152.

[52] P. D. Turney, «Learning algorithms for keyphrase extraction», *Information retrieval*, vol. 2, n.º 4, págs. 303-336, 2000.
[53] R. Alzaidy, C. Caragea y C. L. Giles, «Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents», en *The world wide web conference*, 2019, págs. 2551-2557.
[54] D. Sahrawat, D. Mahata, M. Kulkarni, H. Zhang, R. Gosangi, A. Stent, A. Sharma, Y. Kumar, R. R. Shah y R. Zimmermann, «Keyphrase Extraction from Scholarly Articles as Sequence Labeling using Contextualized Embeddings», *arXiv preprint arXiv:1910.08840*, 2019.
[55] A. Hulth, J. Karlgren, A. Jonsson, H. Boström y L. Asker, «Automatic keyword extraction using domain knowledge», en *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, 2001, págs. 472-482.
[56] W. Ni, T. Liu y Q. Zeng, «Extracting keyphrase set with high diversity and coverage using structural SVM», en *Asia-Pacific Web Conference*, Springer, 2012, págs. 122-133.
[57] L. Page, S. Brin, R. Motwani y T. Winograd, «The pagerank citation ranking: Bringing order to the web.», Stanford InfoLab, inf. téc., 1999.
[58] R. Mihalcea y P. Tarau, «Textrank: Bringing order into text», en *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, págs. 404-411.
[59] G. Erkan y D. R. Radev, «Lexrank: Graph-based lexical centrality as salience in text summarization», *Journal of artificial intelligence research*, vol. 22, págs. 457-479, 2004.
[60] A. Bougouin, F. Boudin y B. Daille, «Topicrank: Graph-based topic ranking for keyphrase extraction», 2013.
[61] S. Danesh, T. Sumner y J. H. Martin, «Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction», en *Proceedings of the fourth joint conference on lexical and computational semantics*, 2015, págs. 117-126.
[62] X. Wan y J. Xiao, «Single Document Keyphrase Extraction Using Neighborhood Knowledge.», en *AAAI*, vol. 8, 2008, págs. 855-860.
[63] A. Judea, H. Schütze y S. Brügmann, «Unsupervised training set generation for automatic acquisition of technical terminology in patents», en *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical Papers*, 2014, págs. 290-300.
[64] R. Wang, W. Liu y C. McDonald, «Using word embeddings to enhance keyword identification for scientific publications», en *Australasian Database Conference*, Springer, 2015, págs. 257-268.
[65] M. Basaldella, E. Antolli, G. Serra y C. Tasso, «Bidirectional lstm recurrent neural network for keyphrase extraction», en *Italian Research Conference on Digital Libraries*, Springer, 2018, págs. 180-187.
[66] Z. Huang, W. Xu y K. Yu, «Bidirectional LSTM-CRF models for sequence tagging. CoRR abs/1508.01991 (2015)», *arXiv preprint arXiv:1508.01991*, 2015.
[67] C. Zhang, «Automatic keyword extraction from documents using conditional random fields», *Journal of Computational Information Systems*, vol. 4, n.º 3, págs. 1169-1180, 2008.
[68] S. N. Kim, O. Medelyan, M.-Y. Kan y T. Baldwin, «Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles», en *Proceedings of the 5th International Workshop on Semantic Evaluation*, 2010, págs. 21-26.

[69] W.-t. Yih, J. Goodman y V. R. Carvalho, «Finding advertising keywords on web pages», en *Proceedings of the 15th international conference on World Wide Web*, 2006, págs. 213-222.

[70] W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim y X. Li, «Topical keyphrase extraction from twitter», en *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, Association for Computational Linguistics, 2011, págs. 379-388.

[71] Z. Liu, W. Huang, Y. Zheng y M. Sun, «Automatic keyphrase extraction via topic decomposition», en *Proceedings of the 2010 conference on empirical methods in natural language processing*, Association for Computational Linguistics, 2010, págs. 366-376.

[72] O. Medelyan, E. Frank e I. H. Witten, «Human-competitive tagging using automatic keyphrase extraction», en *Proceedings of the 2009 conference on empirical methods in natural language processing*, 2009, págs. 1318-1327.

[73] S. D. Gollapalli y C. Caragea, «Extracting keyphrases from research papers using citation networks», en *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[74] I. Augenstein, M. Das, S. Riedel, L. Vikraman y A. McCallum, «Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications», *arXiv preprint arXiv:1704.02853*, 2017.

[75] S. Bird, E. Klein y E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. .º'Reilly Media, Inc.", 2009.

[76] A. Mikołajczyk y M. Grochowski, «Data augmentation for improving deep learning in image classification problem», en *2018 international interdisciplinary PhD workshop (IIPhDW)*, IEEE, 2018, págs. 117-122.

[77] A. Adhikari, A. Ram, R. Tang y J. Lin, «Docbert: Bert for document classification», *arXiv preprint arXiv:1904.08398*, 2019.

[78] K. Frantzi, S. Ananiadou y H. Mima, «Automatic recognition of multi-word terms:. the c-value/nc-value method», *International journal on digital libraries*, vol. 3, n.º 2, págs. 115-130, 2000.

[79] S. Ananiadou, «A methodology for automatic term recognition», en *COLING 1994 Volume 2: The 15th International Conference on Computational Linguistics*, 1994.

[80] S. J. Rose, W. E. Cowley, V. L. Crow y N. O. Cramer, *Rapid automatic keyword extraction for information retrieval and analysis*, US Patent 8,131,735, mar. de 2012.