UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aprendizaje automático para la detección de humor en Twitter

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Àngel Andújar Carracedo

Tutor: Lluís Felip Hurtado Oliver

Cotutor: Ferran Pla Santamaría

Curso 2019-2020

Resumen

Actualmente, el estudio del humor desde una perspectiva computacional es un campo que se encuentra en plena exploración en los ámbitos del aprendizaje automático y de la lingüística computacional. Uno de los objetivos de este estudio es comprender qué conceptos son los que llevan a generar la risa en las personas y a partir de esto, crear modelos computacionales capaces de detectar y generar humor.

En este trabajo se aborda la tarea de detección de humor en Twitter a partir de la conformación de un corpus compuesto por tuits de otros conjuntos de datos utilizados en el mismo cometido. Además, se trata la tarea de puntuar cómo de graciosos son los tuits y se realiza el estudio de diferentes personalidades españolas famosas en Twitter. Con el objetivo de alcanzar estas tareas se han utilizado diversas técnicas de aprendizaje automático.

Palabras clave: Aprendizaje automático, detección de humor, humor computacional, procesamiento del lenguaje natural, Twitter.

Resum

Actualment, l'estudi de l'humor des d'una perspectiva computacional és un camp que es troba en plena exploració en els àmbits de l'aprenentatge automàtic i de la lingüística computacional. Un dels objectius d'aquest estudi és comprendre quins conceptes són els que porten a generar el riure en les persones i a partir d'això, crear models computacionals capaços de detectar i generar humor.

En aquest treball s'aborda la tasca de detecció d'humor en Twitter a partir de la conformació d'un corpus compost per tuits d'altres conjunts de dades utilitzats en la mateixa comesa. A més, es tracta la tasca de puntuar com de graciosos són els tuits i es realitza l'estudi de diferents personalitats espanyoles famoses en Twitter. Amb l'objectiu d'aconseguir aquestes tasques s'han utilitzat diverses tècniques d'aprenentatge automàtic.

Paraules clau: Aprenentatge automàtic, detecció d'humor, humor computacional, processament del llenguatge natural, Twitter.

Abstract

Currently, the study of humor from a computational perspective is a field that is in full exploration in the fields of machine learning and computational linguistics. One of the objectives of this study is to understand what concepts are what lead to generating laughter in people and from this, create computational models capable of detecting and generating humor.

This work addresses the task of detecting humor on Twitter based on the formation of a corpus made up of tweets from other data sets used in the same task. In addition, it deals with the task of scoring how funny the tweets are and the study of different famous Spanish personalities on Twitter is carried out. In order to achieve these tasks, various machine learning techniques have been used.

Keywords: Computational humor, humor detection, machine learning, natural language processing, Twitter.

Tabla de contenidos

Contenido

Tabla de figuras

Índice de tablas

1. Introducción

El humor se ha estudiado a lo largo de la historia desde un enfoque cognitivo, lingüístico y psicológico, pero en la actualidad, su investigación desde un punto de vista computacional es un área que se encuentra en plena exploración en los campos del aprendizaje automático y la lingüística computacional (1).

El aprendizaje automático es una rama de la inteligencia artificial cuyo objetivo es crear algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de una gran cantidad de datos proporcionada como ejemplos. Es decir, tratar de desarrollar algoritmos capaces de predecir una conducta futura a partir de lo que ha sucedido en el pasado. Del mismo modo, el aprendizaje automático tiene un amplio abanico de aplicaciones como puede ser los diagnósticos médicos, reconocimiento de imágenes y formas, reconocimiento del habla y lenguaje escrito, robótica, entre otros (2).

La evolución que han experimentado las redes sociales, convirtiéndose en un factor esencial en la comunicación de la sociedad actual (3), ha dado lugar a nuevas fuentes de datos. Estas fuentes, junto a la aparición de mejoras en el campo del aprendizaje automático y a los avances en la capacidad de procesamiento de información, han posibilitado el crecimiento en el área del procesamiento del lenguaje natural. El procesamiento del lenguaje natural ayuda a las máquinas a entender, interpretar y manipular el lenguaje humano como por ejemplo es la tarea de tratamiento de textos escritos o, el procesamiento de un audio de voz trascribiéndose en letras o caracteres.

Este trabajo de final de grado presenta la experimentación con diferentes algoritmos de aprendizaje automático en las siguientes tareas: búsqueda de una solución a la detección de humor de textos en español a partir de un tuit; puntuación de cuán gracioso es dicho tuit; y, por último, el estudio en Twitter de diversos personajes famosos y comprobar qué tan graciosos son a partir de los modelos desarrollados.

1.1. Motivación

Desde un punto de vista personal, la motivación de realizar este trabajo viene dada por un gran interés en el campo de la inteligencia artificial, y en el procesamiento del lenguaje natural. Dicho interés en el ámbito computacional se encuentra unido a mi gusto desarrollado durante la secundaria por la gramática del lenguaje y su importancia para determinar la finalidad y el significado de una oración. Además, la elección de este tema me permite conocer cómo se trabaja en el ámbito del aprendizaje automático y qué herramientas son las más adecuadas para comunicar las máquinas con las personas a partir del uso de lenguas naturales.

Desde una perspectiva profesional, la detección de humor en un texto puede impulsar a entender qué es lo humorístico y qué términos son los responsables de originar la risa. Esto puede ayudar en la evolución de diferentes campos, como en la salud, donde el humor aporta grandes beneficios, por ejemplo en remediar el estrés (4); en el de marketing, debido a que el humor atrae a la gente y capta su atención (5), etc.

1.2. Objetivos

El objetivo principal de este proyecto consiste en el estudio y la experimentación de diferentes algoritmos de aprendizaje automático en la tarea de detección de humor de los usuarios de la red social Twitter, utilizando para ello únicamente el contenido contextual de los tuits. El trabajo se divide en cuatro grandes objetivos que a la vez están compuestos por subobjetivos:

- Creación del corpus que recopile tuits humorísticos.

- Detección de humor.
  - Uso de métodos automáticos de detección de humor.
  - Evaluación de los métodos y análisis de los resultados.

- Cálculo del nivel de humor.
  - Uso de métodos automáticos de cálculo del nivel humor.
  - Evaluación de los métodos y análisis de los resultados.

- Estudio de humor en tuits de personajes famosos.
  - Uso de métodos automáticos de cálculo y de detección de humor.
  - Evaluación de los métodos y análisis de los resultados.

1.3. Estructura

El presente trabajo está constituido por siete capítulos. Seguidamente, se detallan los temas tratados en cada capítulo:

En el **capítulo 1. Introducción**, se realiza la introducción del tema que se aborda a lo largo del proyecto, además de la motivación que ha dado lugar a la elaboración del trabajo y de la descripción de los objetivos a cumplir.

En el **capítulo 2. Estado del arte**, se expone de forma teórica al lector los diferentes modelos de aprendizaje automático que se emplean en el proyecto y sus características principales. Asimismo, se realiza un estudio en el campo del procesamiento del lenguaje natural de diferentes modelos de representación de palabras.

En el **capítulo 3. Tecnologías, problemas e implementación**, se presenta las tecnologías que han sido empleadas en el desarrollo del trabajo, se explica el problema a solucionar y, por último, se expone la implementación que se ha realizado para cada uno de los algoritmos de aprendizaje automático.

En el **capítulo 4, Experimentación**, se realiza la evaluación de los modelos y se muestra un estudio de los resultados obtenidos, además de la exposición de los problemas originados y las medidas tomadas con el fin de paliarlos.

En el **capítulo 5, Relación con los estudios cursados**, se realiza desde una perspectiva introspectiva, un análisis de los conceptos adquiridos a lo largo del grado que se han empleado en la realización del trabajo.

En el **capítulo 6, Conclusiones**, se presenta las conclusiones obtenidas a partir de la experimentación realizada.

En el **capítulo 7, Trabajos futuros**, se exponen diferentes caminos a tomar en próximos trabajos relacionados con la detección del humor.

Asimismo, a estos siete capítulos se referencia la bibliografía que se ha consultado en la realización del trabajo. Finalmente, se incluye un glosario con conceptos específicos y acrónimos relacionados con los ámbitos abordados en el proyecto.

2. Contexto tecnológico

En este apartado se aborda el concepto de aprendizaje automático de una manera exhaustiva. Se presentan los conceptos básicos, los diferentes tipos que existen y se incide en los algoritmos que se utilizan en la experimentación del trabajo. Después en el apartado de procesamiento del lenguaje natural se exponen diferentes modelos de representación de las palabras. Para acabar, se presenta la aplicación del aprendizaje automático al problema de la detección de humor en castellano, así como el estado en el que se encuentra en la actualidad.

2.1. Aprendizaje automático

2.1.1. Conceptos básicos

En el aprendizaje automático los datos se dividen en dos tipos (6), los de entrada que se denominan datos de aprendizaje o entrenamiento y, los de salida. La meta que se persigue es conseguir un modelo que prediga la salida de datos de forma apropiada a partir de nuevos datos de entrada distintos a los de entrenamiento, es decir, que dicho modelo tenga una gran capacidad de generalización. Con respecto a los datos de salida y el tipo de valor que reproducen, el problema que se aborda puede ser de clasificación o de regresión.

Por un lado, en los problemas de clasificación los datos de salida reproducen un valor discreto que se encuentra dentro de un conjunto finito de elementos llamados clases. Así pues, dicho valor discreto se predice mediante una función matemática que realiza el cálculo de sus parámetros a partir de una cierta selección de características sobre los datos de entrada. En clasificación, la calidad del modelo obtenido se evalúa a partir de parámetros como el acierto (*accuracy*), exhaustividad (*recall*), *F1-score* y precisión (*precision*).

Por otro lado, en los problemas de regresión los datos de salida reproducen un valor continuo que es predicho, al igual que en los problemas de clasificación, a partir de las características, la función matemática y el cálculo de los parámetros. En este caso, la calidad de modelo obtenido se calcula a través de métricas como el error cuadrático medio (*MSE*) o el error absoluto medio (*MAE*).

En relación a la selección de características, el modelo clasificador o regresor debe ser capaz de usar aquellas características relevantes, descartando las irrelevantes. Si la cantidad de características o dimensiones insertadas en la entrada es muy grande puede llevar a aumentar la complejidad del algoritmo de aprendizaje. Con el fin de evitar ese problema, se puede reducir la dimensionalidad antes de introducirlas en el modelo con técnicas como pueden ser la extracción o la selección de características.

Asimismo, el ajuste del modelo es otro aspecto a tener en cuenta en el momento de entrenarlo. A la hora de realizar el ajuste de un modelo altamente flexible se debe tener cuidado de no sobreajustar los datos, es decir, evitar que el modelo aprenda detalladamente sobre cada uno de los datos de entrenamiento, debido a que pueden existir algunos que ejerzan ruido y esto derive en que el modelo los aprenda afectando negativamente a su capacidad de generalización.

Sin embargo, también se debe prestar atención al problema de sobregeneralizar los datos, es decir, no disponer de una gran cantidad de datos de entrenamiento con los que entrenar el modelo y por tanto, no ser capaz de generalizar el conocimiento a nuevos datos de entrada.

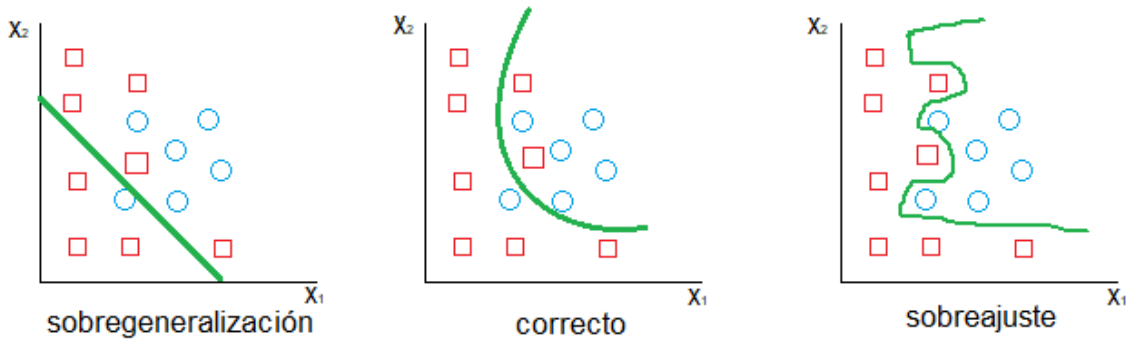

Figura 1. Tipos de ajuste.

El aprendizaje automático normalmente se divide en dos tipos principales como son el aprendizaje supervisado y el aprendizaje no supervisado. Por una parte, en el aprendizaje supervisado los modelos son entrenados con un conjunto de datos donde se conoce de forma completa la información tanto de entrada como de salida. Con este tipo de aprendizaje se pretende que el modelo se vaya entrenando y ajustando a partir de la información otorgada por los datos de salida con el fin de realizar el proceso de generalización adecuadamente.

Por otra parte, en el aprendizaje no supervisado los modelos son entrenados con un conjunto de datos donde únicamente se conoce la información de los datos de entrada. El objetivo de este aprendizaje es obtener información sobre la estructura del dominio de salida, es decir, encontrar patrones de interés en los datos.

2.1.2. Algoritmos de aprendizaje automático

Los algoritmos del aprendizaje automático realizan la extracción de conocimiento a partir de los datos de entrada y de los datos de salida también si son supervisados. En este trabajo se han utilizado siguientes:

Árboles de decisión

Un árbol de decisión (2) es un tipo de algoritmo de aprendizaje supervisado en la que el espacio de los datos de entrada se divide en dos o más regiones locales. En este proyecto se ha empleado en la tarea de clasificación.

Este modelo se compone de nodos de decisión internos, nodos de decisión y hojas terminales. Cada nodo de decisión implementa una función de test con resultados discretos que etiquetan las ramas. Por lo tanto, dada una entrada, se comprueba en cada nodo si cumple una cierta condición, y basándose en el resultado de la comprobación sigue por una de las ramas. Este proceso empieza desde el nodo raíz y se itera recursivamente hasta alcanzar un nodo hoja terminal cuyo valor conforma la salida.

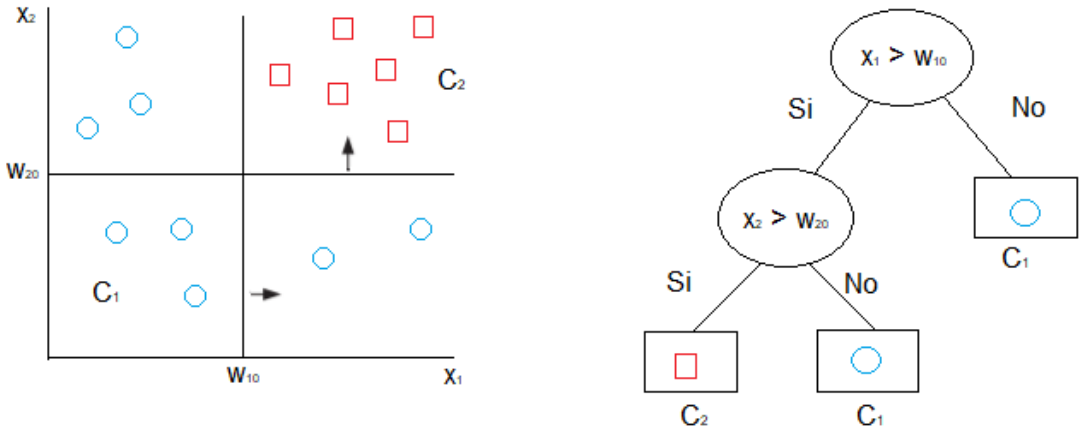

Figura 2. Ejemplo de un conjunto de datos y su correspondiente árbol de decisión. Los nodos ovalados son de decisión y los rectángulos son nodos hoja.

En relación a la función que se emplea para medir la calidad de una partición puede seguir diferentes criterios. Un criterio es el de la impureza de Gini el cual consiste en calcular la frecuencia con la que un elemento escogido al azar del conjunto se etiquetaría incorrectamente si se etiquetase aleatoriamente en relación con la distribución de etiquetas en un subconjunto dado. Otro criterio a seguir es de la ganancia de información que se encarga medir la calidad de una variable a partir de la entropía.

K-vecinos más cercanos

El algoritmo de los K-vecinos más cercanos (*KNN*) es un tipo de algoritmo supervisado en el que el aprendizaje del modelo se efectúa en el mismo momento que se prueban los datos de test. En este trabajo se ha utilizado en tareas de clasificación. El algoritmo consiste en realizar el cálculo de la distancia del nuevo dato de entrada con cada uno de los elementos ya existentes y, una vez ordenados de mayor a menor distancia, se asigna al nuevo dato aquella clase según tenga K elementos más próximos (7).

Máquinas vector soporte

El algoritmo de máquinas vector soporte (*SVM*) fue originalmente diseñado para la clasificación binaria, pero se puede extender su uso en problemas de clasificación multi-clase y de regresión. En este trabajo, se ha empleado este algoritmo a la hora de clasificar si un tuit es humorístico y también, en la tarea de regresión para calcular cómo de gracioso es un tuit.

En cuanto a los problemas de clasificación (6), el algoritmo *SVM* es supervisado y propio de la familia de los clasificadores lineales, ya que, dados unos datos de entrenamiento etiquetados y linealmente separables, el algoritmo determina un hiperplano de separación que clasifica dichos datos entre dos espacios dimensionales donde cada uno de ellos pertenece a una clase propia. El objetivo de esta técnica se basa en determinar un hiperplano separador que maximice el margen de separación entre las clases.

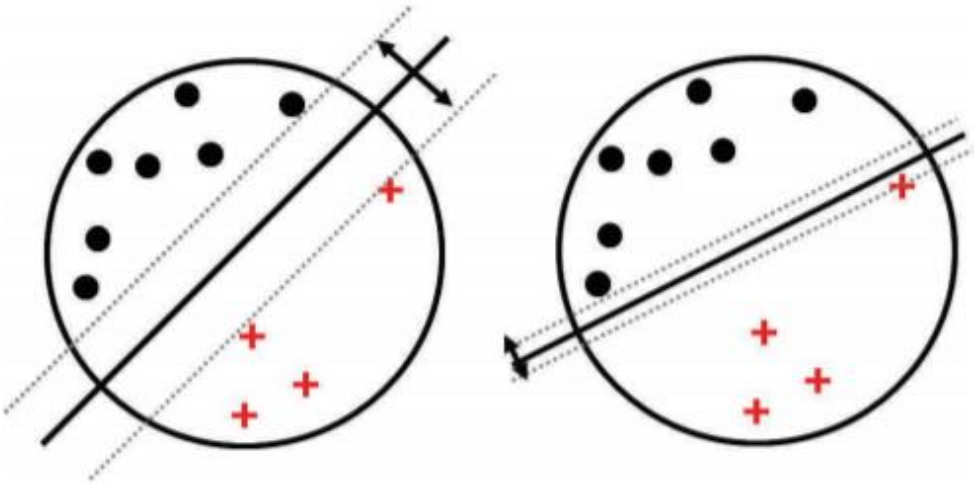

Figura 3. Ilustración de dos hiperplanos separadores con diferentes tamaños de margen (6).

Ahora bien, si los datos que se emplean en el entrenamiento no son linealmente separables, el algoritmo *SVM* no es capaz por sí solo de crear un hiperplano separador para dichas muestras. La solución a este problema (7) consiste en producir una proyección de los datos no linealmente separables a un espacio nuevo donde estos sean linealmente separables a partir de las funciones matemáticas conocidas como núcleos.

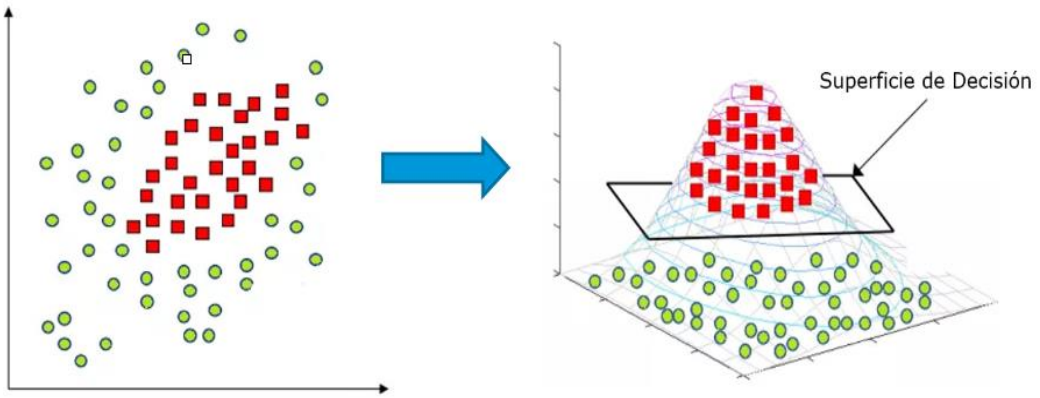

Figura 4. Ilustración donde se muestra como inicialmente un conjunto de datos no linealmente separables con la aplicación de un kernel y la proyección a una nueva dimensión se pueden separar linealmente (8).

Por otro lado, los vectores de soporte de regresión (*SVR*) siguen prácticamente los mismos principios que el *SVM* para clasificación, pero con una única salida que es un número real.

Tanto para clasificación como para regresión las máquinas de vector soporte además de requerir que se especifique como parámetro la función del núcleo (6), se debe especificar el valor de la variable de holgura conocida como C. La variable de holgura C, controla el número y severidad de las violaciones del margen (y del

hiperplano) que se toleran en el proceso de ajuste. Si C es muy bajo, entorno a cero, no se penalizará prácticamente los errores, en cambio, si C es muy alto, entorno a infinito, no se permitirá ningún error. Por lo tanto, se debe tener precaución a la hora de escoger el valor del parámetro C ya que, afectará de forma considerable a la generalización del modelo.

Naïve Bayes

El algoritmo de *Naïve Bayes* (6) se ha utilizado en el proyecto para el problema de clasificación de tuits. Esta técnica se basa en el Teorema de Bayes y supone que las características obtenidas a partir de los datos de entrada son condicionalmente independientes dada la etiqueta de clase. Por tanto, el clasificador de *Naïve Bayes* asume que el impacto de una característica concreta en una clase es independiente de otras características. Habitualmente se utiliza un modelo gráfico para poder capturar la correlación entre características.

Regresión logística

La regresión logística (6) es una técnica de aprendizaje automático empleada en la tarea de clasificación. Se trata de un algoritmo simple que permite interpretar fácilmente los resultados obtenidos a partir de la función logística (Figura 5). Los resultados que se obtienen con este modelo son valores probabilísticos entre cero y uno los cuales, mediante un valor umbral, se asignan a una clase u otra.

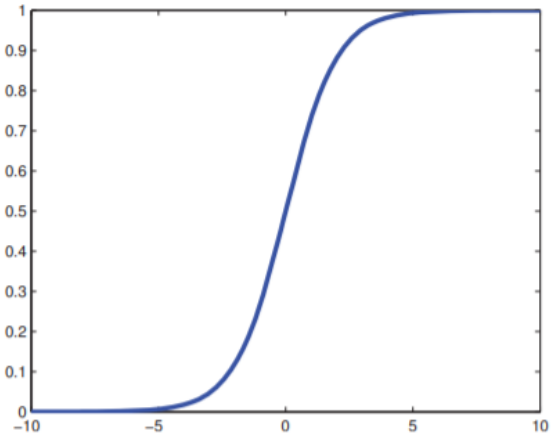

Figura 5. La función logística o sigmoide (6).

Red neuronal

Los modelos de redes neuronales artificiales fueron creados con la intención de recrear en un algoritmo las conexiones y procesos neuronales que existen en los cerebros de las personas. El primer modelo matemático de una neurona artificial fue presentado en el año 1943 por Warren McCulloch y Walter Pitts (9).

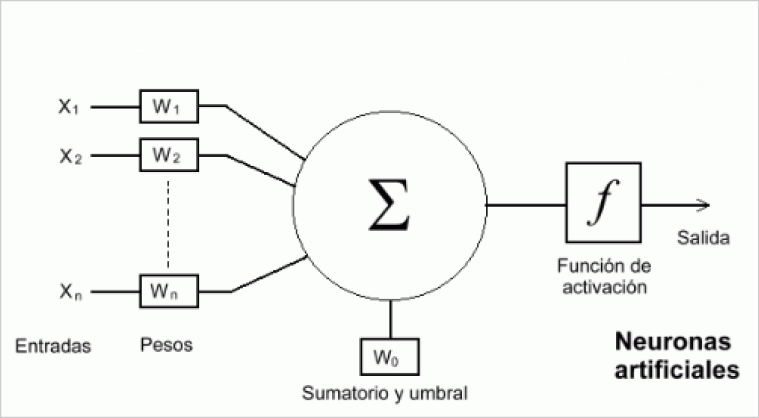

Figura 6. Ejemplo de un modelo de neurona artificial (9).

En cuanto a las funciones de activación aplicadas a las redes neuronales, se encargan de retornar un valor de salida a partir de la entrada recibida. Existen diferentes tipos como la función *Sigmoide, ReLU y Softmax* entre otras.

En la actualidad, con el uso de algoritmos de optimización basados en gradiente y la aparición de librerías específicas empleadas para controlar dichas redes desde un punto de vista computacional por medio de hardware específico que permite el paralelismo, se ha dado lugar a la aparición del aprendizaje profundo.

A diferencia de las primeras redes, en la actualidad con el aprendizaje profundo se manipulan redes con más capas y más número de neuronas por capa formando estructuras más flexibles y se realizan computaciones más duraderas sobre conjuntos de datos más grandes.

Perceptrón multicapa

El perceptrón (2) es el elemento básico de procesamiento. Tiene entradas que pueden provenir de datos de entrada o pueden ser salidas de otros perceptrones. Un perceptrón que solo tiene una única capa de pesos presenta el problema de no poder resolver problemas discriminantes no lineales ni problemas no lineales de regresión.

No obstante, esta limitación no afecta en la retroalimentación de redes con capas intermedias u ocultas entre las capas de entrada y salida. Por ello, si se utiliza para clasificación, como el perceptrón multicapa, puede realizar la implementación de funciones discriminantes no lineales y, si se usa para regresión, puede aproximar las funciones no lineales de la salida.

El perceptrón multicapa (2) es un modelo de red neuronal artificial que forma parte del aprendizaje supervisado. Este modelo se consta de un conjunto de neuronas repartidas en diferentes capas interconectadas entre sí. Aunque el número de capas puede variar según el problema y la cantidad de datos que se posea, siempre existe una capa de entrada y una de salida, las demás se denominan capas ocultas. La capa de entrada recibe la información obtenida de los datos de entrada y propaga dicha información por la red hasta llegar a la capa de salida.

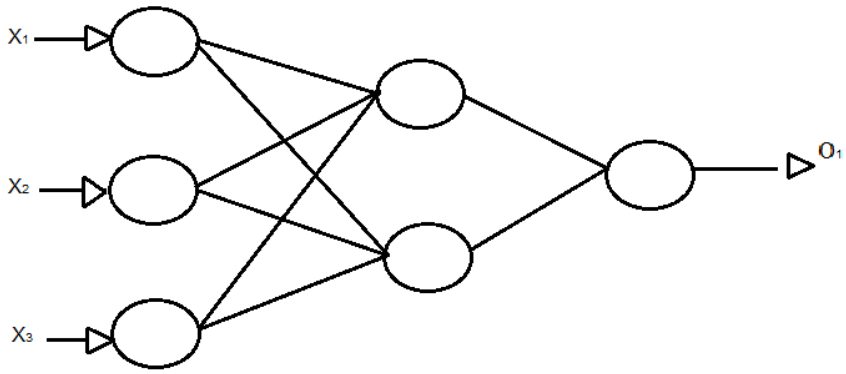

Figura 7. Perceptrón multicapa con una capa oculta.

Así pues, una vez que la estructura de la red neuronal se define, los únicos parámetros que se pueden modificar para obtener comportamientos diferentes son los pesos entre diferentes capas. Por tanto, el objetivo es encontrar los pesos de las conexiones que proporcionan la mejor predicción posible. Para conseguir aprender dichos pesos, se emplea el algoritmo de propagación hacia detrás (*BackPropagation*) utilizando las muestras etiquetadas que han sido proporcionadas en la entrada.

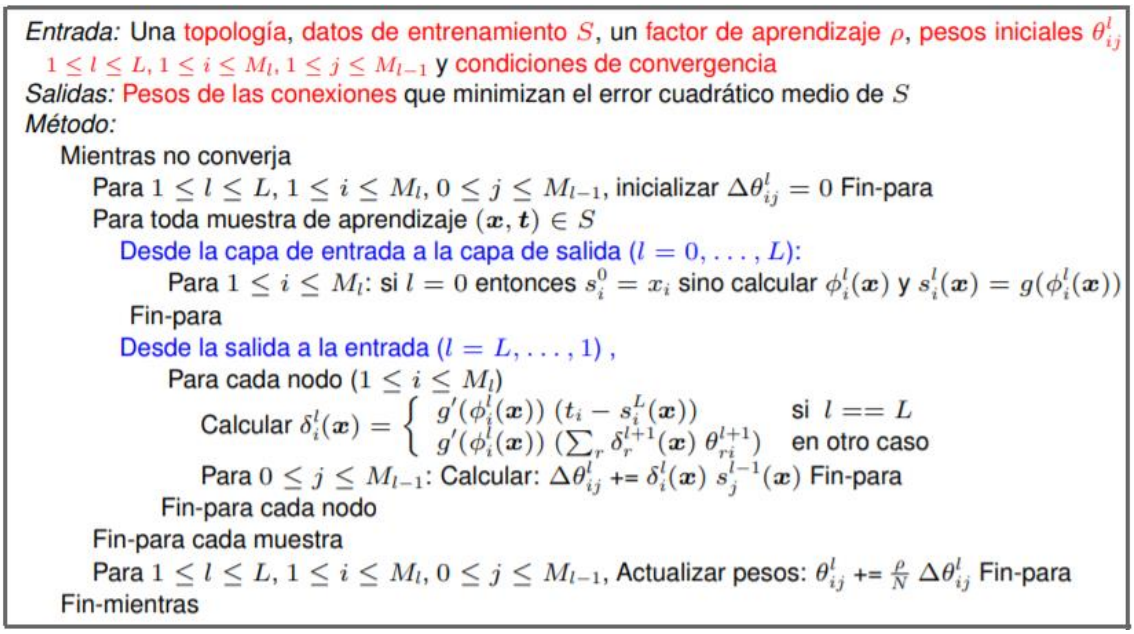

Figura 8. Algoritmo de propagación hacia detrás (BackPropagation).

2.2. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural permite recopilar una gran cantidad de ejemplos procedentes de las redes sociales y almacenarlos en un corpus. Estos datos se pueden utilizar con el fin de entrenar un modelo de aprendizaje automático y que este sea capaz de realizar la generalización de una tarea dada. Para ello, se debe realizar una extracción de las características de dichos datos a partir de un proceso de análisis.

Con el fin de realizar la extracción de características, una gran cantidad de sistemas y técnicas de procesamiento del lenguaje natural emplean las palabras como unidad básica y se realiza una representación de estas mediante vectores numéricos. A continuación, se presenta diversos tipos de representaciones:

2.2.1. One-hot

La representación *one-hot* consiste en crear un vector del tamaño del vocabulario para cada palabra existente en él, donde cada elemento del vector comprende la representación de una palabra o *token* de dicho vocabulario. Es decir, el elemento de un vector que represente a una palabra dada tiene un valor de uno mientras que todos los demás elementos existentes en el vocabulario, y por tanto en el vector, tendrán un cero. Un ejemplo con un vocabulario formado por cuatro palabras, V = {partido, de, tenis, pelota} se expone en la Tabla 1.

Palabra	Vector			
	partido	de	tenis	pelota
partido	1	0	0	0
de	0	1	0	0
tenis	0	0	1	0
pelota	0	0	0	1

Tabla 1. Ejemplo de representación one-hot.

Hay que resaltar que la representación vectorial *one-hot* no aporta información sobre el contexto de las palabras al no presentar ninguna relación entre ellas. Además, presenta problemas de dimensionalidad si el tamaño del vocabulario es grande y de dispersión debido a la gran cantidad de ceros que contiene.

2.2.2. Bolsa de palabras

La representación basada en bolsa de palabras, o comúnmente conocida como *bag of words* (*BOW*), consiste en construir un vector del tamaño del vocabulario existente en un documento dado. Este vector estará compuesto por los diferentes *tokens* del texto, cuyo valor dependerá de si aparece en la oración que se está analizando con un uno, o si no aparece con un cero.

Un ejemplo explicativo con las frases «Hoy es el partido de tenis de Pablo» y «Juan jugará mañana el partido» y, por tanto, el vocabulario está formado por diez palabras o *tokens*, V = {Hoy, es, el, partido, de, tenis, Pablo, Juan, jugará, mañana} se muestra en la siguiente Tabla 2.

Oración	Vocabulario									
	hoy	es	el	partido	de	tenis	Pablo	Juan	jugará	mañana
Hoy es el partido de tenis de Pablo	1	1	1	1	1	1	1	0	0	0
Juan jugará mañana el partido	0	0	1	1	0	0	0	1	1	1

Tabla 2. Ejemplo de representación BOW.

También se puede emplear la bolsa de palabras mostrando la frecuencia total de las palabras mediante la cuenta del número de veces de aparición (Tabla 3). El problema de esta variante es que da más importancia a aquellos *tokens* que aparecen con mayor frecuencia en los documentos, los cuales suelen ser palabras que no aportan información para conocer el contexto. Por ello, la bolsa de palabras se suele utilizar junto a la ponderación *Tf-idf*, ya que permite atenuar la influencia de los *tokens* que aparecen en muchos documentos.

Oración	Vocabulario									
	hoy	es	el	partido	de	tenis	Pablo	Juan	jugará	mañana
Hoy es el partido de tenis de Pablo	1	1	1	1	2	1	1	0	0	0
Juan jugará mañana el partido	0	0	1	1	0	0	0	1	1	1

Tabla 3. Ejemplo de representación BOW por frecuencia.

Uno de los inconvenientes de este modelo de representación es que no captura las relaciones entre los *tokens* de las palabras lo que repercute a que la comprensión del contexto sea baja. Para solucionar este problema se usa la bolsa de palabras con secuencias de *n-tokens*, conocidas como *n-gramas*. Esto permite crear una ventana de contexto entre una cierta cantidad de *tokens* consecutivos adquiriendo así información entre las relaciones de las palabras y por tanto tener una mayor comprensión del contexto.

Por último, se debe señalar que el modelo de bolsa de palabras presenta un problema de dimensionalidad cuando el tamaño del vocabulario es muy grande y también supone problemas de dispersión en el vector de representación al contener muchos ceros.

2.2.3. CBOW

El modelo de representación continua de la bolsa de palabras (10,11), utiliza una red neuronal con una sola capa oculta. Este sistema, dado un contexto compuesto por las palabras vecinas de la palabra central que se busca, realiza la predicción de la palabra en cuestión. Por ejemplo, suponiendo la frase «Pepe tiene mucha sed» y dada la entrada de un contexto formado por el siguiente conjunto de palabras {Pepe, mucha, sed} en dicho modelo, lo que se pretende es que este realice la predicción de forma correcta de la palabra que falta, en este caso la palabra es «tiene».

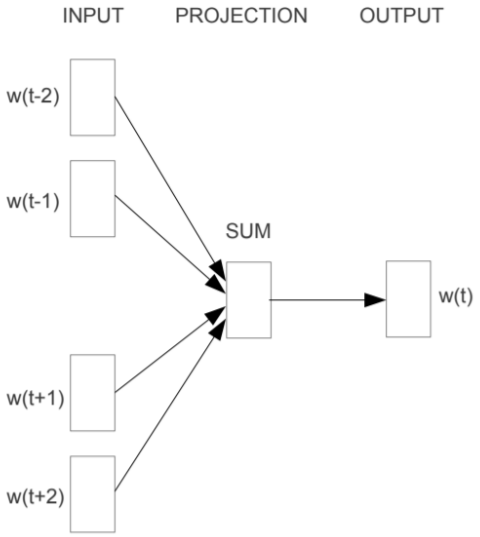

Figura 9. Arquitectura del modelo CBOW (10).

Por tanto, *CBOW* recibe como entrada la ventana definida con el conjunto de palabras del contexto de la palabra central que se busca, representadas cada una de ellas en forma de vector *one-hot*. Estas palabras en la capa oculta son proyectadas y sumadas sobre un vector de proyección conectado con la capa de salida. En la salida se obtiene la palabra central del contexto que el modelo ha predicho.

2.2.4. Skip-Gram

El modelo de representación *Skip-Gram* (12) utiliza una red neuronal con una única capa oculta. Este modelo, dado el vocabulario originado con las palabras del conjunto de oraciones, pretende entrenar la red neuronal con cada una de las frases existentes con el fin de que, dada una palabra, el modelo proporcione para cada una de las palabras del vocabulario la probabilidad de que estas sean vecinas de la palabra en cuestión. Hay que señalar que *Skip-Gram* permite ajustar el tamaño de ventana con el fin de especificar el número de palabras a tener en cuenta en cada contexto.

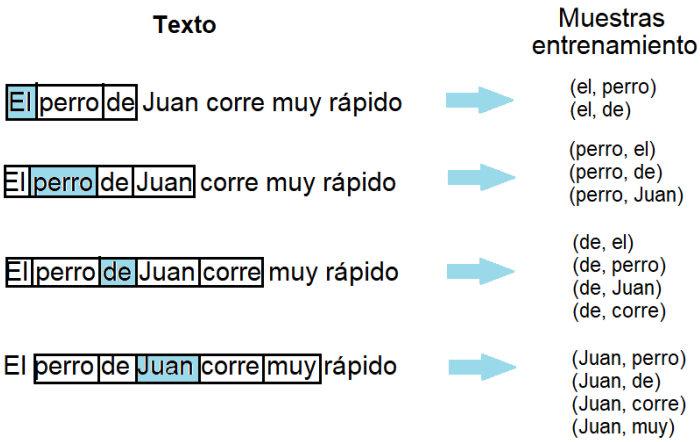

Figura 10. Skip-Gram con tamaño de ventana igual a dos.

En lo relativo a los componentes de la red neuronal cabe resaltar que la entrada será una palabra representada como un *one-hot* vector y por tanto será un vector con un tamaño igual al del vocabulario. Por otro lado, la capa oculta no utiliza una función de activación, pero sí emplea una función de salida. La capa de salida de la red neuronal será un vector con el mismo tamaño que el de entrada, el cual en este caso representará las probabilidades de que cada una de las palabras sea vecina a la palabra considerada en la entrada.

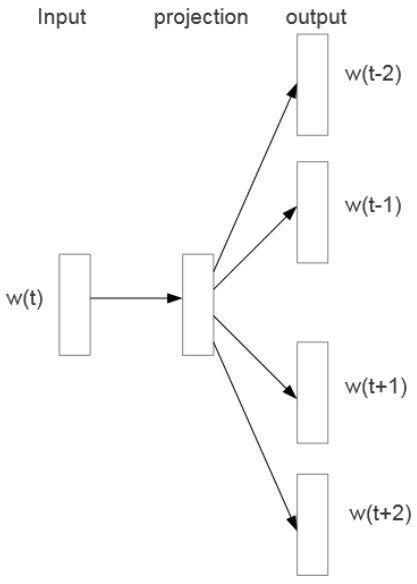

Figura 11. Arquitectura del modelo Skip-Gram (12).

2.2.5. Actualidad PLN

En la actualidad el procesamiento del lenguaje natural ha experimentado un gran avance gracias a un nuevo modelo de red neuronal de representación del lenguaje natural creado por el equipo de Google llamado *BERT* (13).

BERT es bidireccional a diferencia de los modelos de lenguaje anteriores a él, los cuales son unidireccionales como el de la bolsa de palabras. Los modelos unidireccionales para entender el contexto de una palabra solo pueden mover la ventana de contexto con n-palabras en una sola dirección.

En cambio, *BERT* es capaz de contemplar la oración completa en ambas direcciones con el fin de entender el contexto de una palabra. Además de esto, este modelo usa transformadores que se centran en los pronombres y conectores y su relación con las demás palabras con el fin de entender el contexto de una oración y saber qué tema se está tratando.

Asimismo, este nuevo modelo está diseñado para preentrenar representaciones profundas bidireccionales de muestras de textos no etiquetadas condicionando conjuntamente el contexto de ambas direcciones en todas las capas. Como resultado, el modelo BERT previamente entrenado se puede ajustar con una sola capa de salida adiciona con el objetivo de crear modelos de vanguardia para una gran variedad de tareas en el ámbito del procesamiento del lenguaje.

2.2.6. Tarea HAHA

La tarea *HAHA* es una competición cuya primera edición fue realizada en el año 2018. Dicha tarea, consiste en clasificar tuits en español como humorísticos o no, y en determinar cómo de graciosos son. Para ello, en la tarea *HAHA*, se otorga un corpus de tuits anotados por humanos para clasificarlos mediante un sistema de votación.

En cuanto a la detección de humor en textos en español y el empleo del modelo *BERT* en este cometido, se va a hablar acerca del trabajo del equipo ganador de la competición *HAHA*@IberLEF 2019 (14).

Con respecto al equipo ganador Adilism (15) para la tarea de clasificación utilizaron el modelo de lenguaje no supervisado *BERT* junto a algunas modificaciones con el fin de evitar el sobreajuste del modelo en el entrenamiento. Además de esto, emplearon un modelo de *Naïve Bayes* con la obtención de características a partir de unigramas y

bigramas con *Tf-idf* y combinaron sus predicciones con las del modelo *BERT* con regresión logística para obtener la solución final.

Team	F1	Precision	Recall	Accuracy
adilism	**82.1**	79.1	**85.2**	**85.5**
Kevin & Hiromi	81.6	80.2	83.1	85.4
bfarzin	81.0	78.2	83.9	84.6
jamestjw	79.8	79.3	80.4	84.2
INGEOTEC	78.8	75.8	81.9	82.8
BLAIR GMU	78.4	74.5	82.7	82.2
UO UPV2	77.3	78.0	76.5	82.4
vaduvabogdan	77.2	72.9	82.0	81.1
UTMN	76.0	75.6	76.5	81.2
LaSTUS/TALN	75.9	77.4	74.5	81.6
Taha	75.7	**81.0**	71.1	82.2
LadyHeidy	72.5	74.4	70.8	79.1
Aspie96	71.1	67.8	74.9	76.3
OFAI–UKP	66.0	58.8	75.3	69.8
acattle	64.0	68.3	60.2	73.6
jmeaney	63.6	61.3	66.1	70.5
garain	59.3	49.1	74.8	59.9
Amrita CEN	49.5	47.8	51.4	59.1
random baseline	44.0	39.4	49.7	50.5

Figura 12. Resultados de la competición HAHA 19 en la tarea de clasificación (14).

En el cometido de determinar cómo de gracioso es un tuit usaron el mismo modelo *BERT*, exceptuando que se cambió la función de pérdida utilizando la pérdida cuadrática media, y combinaron las predicciones con un modelo *LightGBM* que empleaba las mismas características que el de *Naïve Bayes* en la tarea previa.

Team	RMSE
adilism	**0.736**
bfarzin	0.746
Kevin & Hiromi	0.769
jamestjw	0.798
INGEOTEC	0.822
BLAIR GM	0.910
LaSTUS/TALN	0.919
UTMN	0.945
acattle	0.963
Amrita CEN	1.074
garain	1.653
Aspie96	1.673
OFAI–UKP	1.810
random	2.455

Figura 13. Resultados de la competición HAHA 19 en la tarea de regresión (14).

3. Tecnología, problema e implementación

En esta sección se explica el entorno y las librerías que se han utilizado, se propone el problema a resolver y los diferentes procesos que se han realizado con el fin de implementar una solución.

3.1. Tecnologías utilizadas

El lenguaje de programación que se ha empleado en el desarrollo de la fase de experimentación de este trabajo es Python. Este es un lenguaje interpretado de tipado dinámico cuyo objetivo principal es conseguir un código legible y fácil de entender. Además, es multiparadigma y multiplataforma.

Asimismo, Python posee un gran abanico de librerías para trabajar en el campo del procesamiento del lenguaje natural y en el del aprendizaje automático. A continuación, se exponen las principales herramientas que se han utilizado.

Freeling

Freeling (16) es una librería de código abierto para el procesamiento multilingüe, que proporciona una amplia gama de funcionalidades de análisis para varios idiomas. Entre ellas, se ha empleado el módulo *analyzer* en el proceso de extracción de características de los tuits.

Keras

Keras (17) es una librería que se ha empleado en este trabajo junto a *Tensorflow*. Esta herramienta facilita la creación de modelos de redes neuronales y también aporta utilidades para realizar el entrenamiento y testeo de estas.

MySQL

MySQL es un sistema de gestión de base de datos relacional que ofrece un gran abanico de herramientas. En este trabajo se ha utilizado para almacenar el corpus con todos los tuits y su información asociada. Además, se ha empleado la funcionalidad de *MySQL Connector* en Python con el fin de acceder al corpus y obtener los datos que se requieran.

NLTK

NLTK (18) es una plataforma que provee una gran cantidad de herramientas para el procesamiento del lenguaje natural. Además, permite acceder a múltiples recursos, como corpus etiquetados, con información disponible, proporcionados por *WordNet*.

NumPy

NumPy (19) es una librería que dispone de una gran cantidad de funcionalidades para operar con matrices y vectores.

Scikit-learn

Scikit-learn (20) es un conjunto de herramientas de aprendizaje automático y algoritmos para resolver problemas de agrupamiento, clasificación, regresión, etc.

Twint

Twint es una herramienta de datos en Twitter que permite recopilar tuits e información sobre usuarios ofreciendo criterios de búsqueda sin necesidad de utilizar la aplicación de Twitter.

3.2. Problema

A lo largo de la historia se ha estudiado el humor desde un punto de vista cognitivo, lingüístico y psicológico (5), además en la actualidad se ha transformado en un dominio activo de investigación dentro del aprendizaje automático y la lingüística computacional (14). No obstante, desde la perspectiva computacional la determinación del humor con el fin de reconocerlo y generarlo se encuentra lejos de poderse realizar correctamente.

Además, con el objetivo de comprender qué es lo humorístico y qué términos son los responsables de originar la risa, se debe realizar una definición sencilla del humor y cómo decidir si un tuit es humorístico. Para ello, en este trabajo se entiende que todo tuit es humorístico si proviene de una cuenta cuya intención es ser graciosa. Por otro lado, con respecto a valorar cómo de gracioso es un tuit se tiene en cuenta aquellos tuits que se ha decidido que son humor y la puntuación que han recibido.

3.3. Implementación

En este apartado se exponen los procesos que se han realizado para desarrollar diferentes modelos capaces de cumplir la tarea para la que se les ha asignado. Entre los procesos se encuentra la creación del corpus, la extracción de las características de

dichos tuits y la creación de los diferentes modelos de aprendizaje automático junto a los procesos de entrenamiento y evaluación.

3.3.1. Creación del corpus

Para obtener un modelo de aprendizaje automático se requiere de datos de entrenamiento etiquetados según la clase a la que pertenezcan. En este caso, para entrenar un modelo capaz de realizar la tarea de detección de humor los datos de entrenamiento se corresponden con tuits etiquetados como humorísticos o como no. En este proyecto los tuits escogidos para formar el conjunto de datos de entrenamiento se han recopilado de tres fuentes diferentes.

La primera fuente, de la cual se han adquirido parte de los tuits, proviene del trabajo *Is This a Joke? Detecting Humor in Spanish Tweets* (21) donde los tuits se han etiquetado como humor si proceden de cuentas humorísticas. No obstante, los tuits procedentes de cuentas humorísticas, además, han sido sometidos a un proceso de votación con el fin de controlar si realmente son graciosos. Para realizar la votación se creó una aplicación para el sistema operativo de Android y una página web donde cualquier usuario que se registrase podía indicar si un tuit era humorístico otorgándole una puntuación del uno al cinco o, indicar que no lo era.

Una vez realizada las votaciones, se realiza un filtrado de estos tuits para decidir su etiquetado final. El filtrado de un tuit se calcula mediante la división de los votos que ha recibido como gracioso entre el número de votos totales recibidos. Si el resultado de esta división es mayor que 0,6 el tuit se etiqueta como humorístico, al contrario, si el valor es menor de 0,3, se añade como no humorístico. Hay que señalar que si el valor se encuentra entre 0,6 y 0,3 el tuit se descarta del corpus.

Por lo tanto, a partir del proceso de filtrado de los tuits procedentes de cuentas humorísticas se ha recopilado una cantidad de 35.891 tuits, donde 29.673 se han etiquetado como no humorísticos y 6.218 como humorísticos.

Figura 14. Proporción de tuits en el corpus obtenido del trabajo Is This a Joke? Detecting Humor in Spanish Tweets (21).

Por otro lado, la segunda fuente de donde en este proyecto se ha obtenido datos de entrenamiento es el corpus que se utilizó en la competición *HAHA*@IberLEF 2018 (22). Dicho corpus está compuesto por 20.000 tuits donde su etiquetado se realizó a partir de un sistema de votos donde un tuit podía ser votado como no humorístico o, en cambio, como humorístico indicando con una puntuación del uno al cinco cómo de gracioso era. Con esto, aquel tuit que tenga al menos cinco votos con alguna puntuación indicando que es gracioso se etiqueta como humorístico, al contrario, si un tuit tiene al menos tres votos como no gracioso se etiqueta como no humorístico.

De este conjunto de datos se ha obtenido 16.255 tuits y no 20.000 debido a que algunos ya existían en el corpus de la primera fuente. De dichos tuits, 5.650 se etiquetan como humorísticos y 10.605 como no.

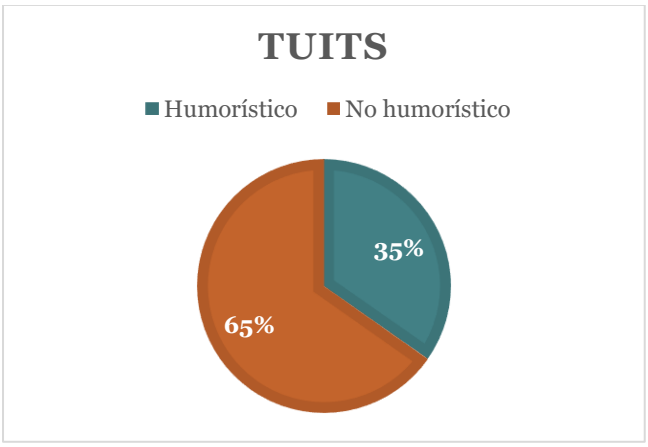

Figura 15. Proporción de tuits en el corpus de la tarea HAHA 18 (22).

La última fuente de datos es el corpus que se utilizó en la competición *HAHA*@IberLEF 2019 (14). Dicho corpus está compuesto por 30.000 tuits donde su etiquetado se realizó a partir del mismo sistema de votación que se ha comentado en la segunda fuente. De este total de tuits, se ha obtenido 15.156 tuits debido a que el resto se encontraban en los otros corpus. De dichos tuits, 6.116 se etiquetan como humorísticos y 9.040 como no.

Figura 16. Proporción de tuits en el corpus de la tarea HAHA 19 (14).

Con la agrupación de los tuits obtenidos de las diferentes fuentes se ha construido para este proyecto un corpus formado por la cantidad de 67.302 tuits, en los que 18.002 se han etiquetado como humorísticos y 49.354 como no.

Figura 17. Proporción de tuits en el corpus utilizado para este trabajo.

En cambio, para la tarea de predecir cómo de gracioso es un tuit se han utilizado únicamente los conjuntos de datos proporcionados por las competiciones *HAHA*@IberLEF 2018 y *HAHA*@IberLEF 2019. En este caso la etiqueta para cada uno de los tuits se corresponde con la media de la puntuación obtenida por los votos y por lo tanto solo se tendrán en cuenta aquellos tuits que son etiquetados como humor. Así

pues, el corpus de esta tarea está formado por un total de 10.498 tuits con una puntuación de humor asignada a cada uno.

3.3.2. Extracción de características

Asimismo, se ha aprovechado y adaptado a las necesidades de este proyecto la funcionalidad de extracción de características obtenida del trabajo que abordó la tarea de detección de humor de textos en español *Is This a Joke? Detecting Humor in Spanish Tweets* (21) en el año 2016.

Este trabajo realiza la extracción de 23 características basándose en un estudio realizado sobre qué factores caracterizan el humor y realizando para cada tipo de característica un cálculo específico relacionado con las palabras que componen el tuit.

Las características que se obtienen de los tuits son las siguientes:

- **Presencia de animales**: se emplea un diccionario formado por una gran cantidad de nombres de animales y variantes de estos con errores ortográficos usuales. Se realiza el cálculo de esta característica mediante la cuenta de palabras que forman el tuit y se encuentran en dicho diccionario y normalizándola a partir de la longitud de palabras del tuit.
- **Jerga sexual**: se utiliza un diccionario formado por palabras relacionadas con la temática sexual, el cálculo se hace de la misma forma que la anterior característica, pero con el uso de este diccionario.
- **Primera y segunda persona**: mediante la herramienta *Freeling*, se intenta captar palabras que hagan referencia a la primera persona. Luego se realiza el cálculo mediante la división del número de tokens en primera persona y la raíz cuadrada del número de *tokens* total del tuit. La característica de segunda persona se calcula de igual manera.
- **Distancia temática**: se pretende estimar la cercanía del contenido del tuit con temas potenciales de los chistes. Entre estos se han tenido en cuenta lo tema relacionados con **adivinanzas**, **animales**, **atlantes**, **chistes cortos** y **otros** donde cada uno es una característica particular. El valor de estas se calcula a partir de la probabilidad de que un tuit pertenezca a dicho tema computada por un clasificador previamente construido.
- **Diálogo**: se indica la existencia de diálogo en el tuit mediante la búsqueda de símbolos utilizados normalmente como puede ser la raya, el símbolo de resta, viñetas, etc.

- **Pregunta-respuesta**: se cuenta la cantidad de preguntas que se responden en un tuit.
- **Palabras clave**: se emplea un diccionario formado por términos frecuentes en chistes. Se realiza el cálculo de esta característica mediante la cuenta de palabras que forman el tuit y dicho diccionario y normalizándolo a partir de la longitud de palabras del tuit.
- ***Links***: se detecta la presencia de enlaces a sitios web en el tuit. La presencia de estos indica que un tuit no es humorístico, ya que solo se tienen en cuenta los tuits compuestos por texto únicamente.
- **Antónimos**: a partir del uso de la herramienta *WordNetCorpusReader* del módulo *nltk.corpus* de la librería *NLTK* se obtienen los *synsets* para cada *token* de la oración. A partir de los *synsets*, se extraen los diferentes antónimos de cada *token* y se comprueba la existencia de los antónimos en la oración. La cantidad de pares de antónimos que hay se divide por la raíz cuadrada del número de *tokens* en el tuit.
- ***Hashtags***: se calcula la cantidad de *hashtags* existentes.
- **Exclamación**: se cuenta la cantidad de signos de exclamación de un tuit y se dividen entre la raíz cuadrada del número de *tokens* que componen el tuit.
- **Palabras en mayúscula**: se calcula la cantidad de palabras escritas completamente en mayúscula y se divide por la raíz cuadrada de la cantidad de *tokens*.
- **Negación**: se cuenta la cantidad de veces que aparece la palabra «no» en el tuit y se divide por el número de *tokens*.
- **Palabras fuera de los diccionarios**: se intenta obtener la presencia de palabras que existen fuera del vocabulario que se encuentran en textos humorísticos debido a su originalidad. Se implementan cuatro tipos diferentes de características *OOV* donde la diferencia se encuentra en los diccionarios que utilizan. Por tanto, se añade una característica para cada diccionario OOV: ***OOV Freeling-Google***, ***OOV Freeling***, ***OOV Freeling-Wiktionary***, ***OOV Wiktionary***.
- **Palabras no españolas**: se calcula la cantidad de palabras que tienen caracteres fuera del vocabulario español en un tuit y se normaliza entre la cantidad de *tokens*.

3.3.3. Creación de los modelos

Árbol de decisión

Para realizar la implementación del modelo de clasificación de un árbol de decisión se ha utilizado el método *DecisionTreeClassifier* del módulo *sklearn.tree* de la librería *Scikit-learn*. Además, este método permite especificar distintos parámetros, uno de ellos es la función para medir la calidad de la partición. Otro parámetro a destacar comprende la estrategia utilizada para elegir la partición de cada nodo.

K-vecinos más cercanos

El algoritmo de clasificación de los K-vecinos más cercanos se ha implementado a través del método *KNeighborsClassifier* del módulo *sklearn.neighbors* de la librería *Scikit-learn.* También se permite especificar el número de vecinos que usar como parámetro.

Máquina vector soporte

En cuanto al modelo *SVM* con la aplicación de diferentes tipos de núcleos, se ha realizado su implementación a partir de los métodos *LinearSVC* y *SVC* del módulo *sklearn.svm* de la librería *Scikit-learn*.

Por un lado, el método *SVC* está basado en la librería *libsvm*. Este método permite especificar por parámetro el tipo de núcleo que se quiere aplicar ya sea lineal, polinomial, *rbf* o *sigmoid,* además del valor de la variable de holgura C entre otros. Por otro lado, el método *LinearSVC* es similar al *SVC* con núcleo lineal, pero se basa en la librería *liblinear* por lo que tiene una mayor flexibilidad a la hora de elegir las penalizaciones y funciones de pérdida y de escalar de una mejor forma con un gran tamaño de muestras.

Asimismo, en relación con efectuar la implementación del modelo de regresión de los vectores de soporte regresión se han utilizado los métodos *LinearSVR* y *SVR* del módulo *sklearn.svm* de la librería *Scikit-learn*. Estos métodos tienen las mismas características y opciones que los mencionados en el párrafo anterior.

La elección de los parámetros de estos modelos se realiza en el apartado de experimentación con el fin de comparar qué función kernel y configuración de la variable de holgura son las más adecuadas para las tareas en cuestión.

Naïve Bayes

Con el fin de desarrollar la implementación del modelo de clasificación de Naïve Bayes se han utilizado los métodos de GaussianNB y MultinomialNB del módulo *sklearn.naive_bayes* de la librería *Scikit-learn*.

Regresión logística

El algoritmo de clasificación de regresión logística se ha implementado mediante el método *LogisticRegression* del módulo *sklearn.linear_model* de la librería *Scikit-learn*. Este método permite especificar por parámetro el valor de la variable de holgura C entre otros.

Perceptrón multicapa

Con el propósito de implementar el modelo de clasificación del perceptrón multicapa se han utilizado los métodos *Activation*, *BatchNormalization*, *Dense*, *Dropout* e *Input* del módulo *keras.layers* de la librería *Keras*.

En primer lugar, se utiliza el método *Input* para indicar el número de neuronas de la capa de entrada que es igual al número de características de los datos. Después, mediante un bucle limitado por el número de capas que se desea se usa la instrucción *Dense* para añadir una capa oculta a la capa previa indicando además el número de neuronas que se dispone en esa capa oculta. Dentro del mismo bucle y a continuación del método *Dense,* se emplea *BatchNormalization* con el fin de normalizar los datos de la capa oculta en cuestión antes de que se les aplique la función de activación ReLU mediante la instrucción *Activation*.

Cabe señalar que el método *Dropout* consiste en desactivar un número de neuronas de una capa de forma aleatoria en cada iteración en el entrenamiento de la red neuronal, obligando así a que las neuronas que son cercanas no aprendan patrones que puedan dar lugar al sobreajuste del modelo.

Por último, una vez creadas el número de capas ocultas necesarias, se añade una capa de salida con la cantidad de neuronas equivalente a la de clases y su función de activación *Softmax* mediante la instrucción *Dense*. Ahora bien, mediante el método

Model del módulo *keras.models* se unifica la estructura de capas del modelo neuronal y además, con la función *Model.compile* se indica como parámetros el tipo de optimizador, pérdida y métricas a usar.

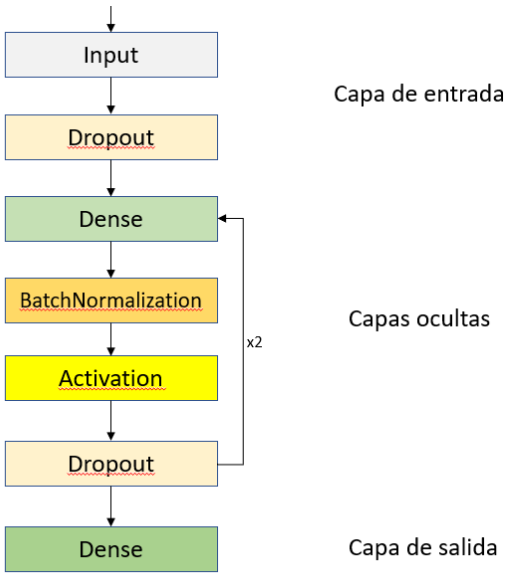

Figura 18. Arquitectura de un modelo de clasificación del perceptrón multicapa con dos capas ocultas.

De modo similar a la arquitectura del modelo de clasificación del perceptrón multicapa se construye la del modelo de regresión. No obstante, el modelo de regresión varía en el número de neuronas de la capa de salida debido a que, al devolver un valor continuo, solo tiene una neurona con una función de activación lineal.

3.3.4. Entrenamiento de los modelos

Una vez construido el conjunto de datos, se ha realizado la extracción de características y se han creado los modelos, el siguiente paso es entrenar dichos modelos.

Con respecto a los modelos cuya implementación se ha desarrollado a partir de la librería *Scikit-learn,* se utiliza el método *fit* asociado a cada modelo para realizar el entrenamiento de este. La instrucción *fit* recibe como parámetros los datos de entrenamiento, es decir, por un lado, las características asociadas a los tuits de entrenamiento y por otro, las etiquetas de clase a la que pertenecen si el modelo es de clasificación, o el valor asociado, si el modelo es de regresión.

En relación con los modelos neuronales se ha empleado el método *fit* de la librería *Keras* para el entrenamiento. En este método, además de recibir como parámetro los datos de entrenamiento, admite la introducción del tamaño del *batch* que indica la frecuencia con la que se actualiza el gradiente y el número de épocas para apuntar la cantidad de veces que se entrena el modelo sobre las muestras dadas. Hay que señalar que en el perceptrón multicapa de clasificación se debe codificar mediante *one-hot* las etiquetas de clase de entrenamiento debido a que la librería *Keras* lo requiere.

3.3.5. Evaluación de los modelos

Después del proceso de entrenamiento se debe realizar una evaluación del modelo adquirido con el objetivo de valorar de qué forma realiza la función para la que ha sido desarrollado. Según la tarea que se trate, se emplean unas métricas u otras debido a que el tipo de datos obtenido por la predicción de un clasificador es diferente al obtenido por un modelo de regresión.

Por una parte, los modelos desarrollados a partir de la librería *Scikit-learn*, utilizan la instrucción *predict* la cual recibe como parámetro los datos de test y retorna la predicción de las clases pertenecientes en clasificación o los valores perenecientes en caso de regresión. Una vez que se obtiene las predicciones se comparan con las etiquetas de test para poder evaluar el modelo. Por otra parte, los modelos realizados a partir de la librería *Keras*, emplean el método *predict* que funciona de forma similar al de la otra librería.

Ahora bien, en cuanto a los modelos de clasificación binaria a partir de la predicción se distingue como valores de clase positivos a los tuits etiquetados como humorísticos y como valores de clase negativa, aquellos etiquetados como no humorísticos. Con esto se define cuatro categorías para los ejemplos clasificados: verdadero positivo, falso positivo, verdadero negativo y falso negativo.

Así pues, se comprende como verdadero positivo aquella muestra clasificada correctamente como positiva. En cambio, se entiende como falso positivo aquel ejemplo clasificado incorrectamente como positivo. Por otro lado, aquella muestra que se clasifica como negativa de forma correcta se concibe como verdadero negativo. Por último, se considera falso negativo aquel ejemplo clasificado incorrectamente como negativo.

A partir de estas cuatro categorías de clasificación se puede realizar el cálculo de las métricas utilizadas para realizar la evaluación de los modelos de clasificación: acierto (*accuracy*), exhaustividad (*recall*), *F1-score* y precisión (*precision*).

El acierto mide los ejemplos que se han clasificado correctamente siendo *vp* verdadero positivo, *vn* verdadero negativo, *fp* falso positivo y *fn* falso negativo.

$$Acierto = \frac{vp + vn}{vp + vn + fp + fn}$$

La exhaustividad (*recall*) calcula la capacidad del clasificador de encontrar muestras positivas.

$$Recall = \frac{vp}{vp + fn}$$

La precisión mide la calidad del clasificador a la hora de predecir casos positivos.

$$Precision = \frac{vp}{vp + fp}$$

La métrica *F1-score* se utiliza para combinar las medidas de *precision* y *recall* en un solo valor.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Hay que destacar que la medida de acierto es engañosa en casos donde el número de muestras para cada clase está desbalanceado como bien sucede en el corpus de este proyecto. Por ello, con el fin de comparar la calidad de los modelos se comprende como referente la medida *F1-score* debido a que es más representativa que el acierto y funciona tanto si las clases están balanceadas o no.

Por otro lado, en cuanto a la evaluación de los modelos de regresión se ha usado la métrica del error cuadrático medio (*RMSE*). El error cuadrático medio mide la diferencia cuadrada entre las predicciones y el objetivo para cada muestra y luego realiza la raíz cuadrada del promedio de esos valores. Cuanto mayor sea este valor peor es el modelo. Se obtiene de la siguiente forma siendo y_i el resultado real esperado e y_i' la predicción obtenida:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2}$$

4. Experimentación

En esta sección se presenta el proceso de experimentación que se ha desarrollado para cada uno de los objetivos planteados. Este proceso consiste en realizar la partición de los datos almacenados en el corpus para el proceso de entrenamiento y el de prueba, en entrenar los diferentes modelos probando diversas configuraciones y finalmente en realizar su evaluación. Una vez obtenido el mejor modelo de cada tipo, se comparan los resultados proporcionados por cada uno de ellos y se escoge el mejor modelo para utilizarlo en la tarea de estudio de los tuits de diferentes personajes famosos.

4.1. Detección de humor

Con respecto a la tarea de detectar si un tuit es humorístico se debe señalar que se ha empleado el corpus de tuits compuesto por las tres fuentes de las cuales se han obtenido los datos presentados el punto 3.3.1. Asimismo, de este conjunto de datos se ha utilizado el 80% de los tuits para realizar el entrenamiento de los modelos y el 20% restante para la evaluación. La métrica F1-score es la utilizada para la comparación de la calidad de los modelos de clasificación.

Para realizar la separación del conjunto de datos en dos subconjuntos se ha utilizado el método *train_test_split* de la librería *Scikit-learn* empleando siempre la misma semilla de generación con el fin de que todos los modelos se entrenen y evalúen con los mismos conjuntos.

4.1.1. Desarrollo de la experimentación

Durante la experimentación de esta tarea se han hecho diversas pruebas de los modelos modificando sus parámetros con el objetivo de obtener el mejor modelo posible para cada uno de ellos.

En primer lugar, se ha realizado el entrenamiento del modelo basado en arboles de decisión utilizando por un lado la función para medir la calidad de las particiones basado en el criterio de impureza Gini y, por otro lado, usando dicha función con el criterio de la entropía para la ganancia de información. Además, se usa la estrategia que elige la mejor partición posible de cada nodo.

Árbol de decisión				
Función de calidad	**Acierto**	***F1-score***	**Precisión**	***Recall***
Gini	81,96%	66,52	66,00%	**67,06%**
Entropía	**82,38%**	**67,53**	**66,55%**	66,53%

Tabla 4. Resultados de la experimentación con diferentes criterios para medir la función de calidad del modelo árbol de decisión.

Con relación a los resultados de la Tabla 4 se contempla que el modelo con el criterio de entropía para la ganancia de información en la función de calidad proporciona mejores resultados de *F1-score* que el que sigue el criterio de impureza Gini.

En segundo lugar, el entrenamiento del algoritmo de clasificación de los K-vecinos más cercanos se ha efectuado a partir de la variación del número de vecinos a considerar y la función de pesos a utilizar, ya sea con pesos uniformes donde todas las muestras en cada vecindario son ponderadas por igual o, con pesos influidos por la distancia en los que los vecinos más cercanos de la muestra consultada tienen una mayor influencia que los vecinos que están más lejos.

K-vecinos más cercanos					
Función de pesos	**Número de vecinos**	**Acierto**	***F1-score***	**Precisión**	***Recall***
Uniformes	5	84,35%	67,88	78,22%	59,95%
	10	85,74%	67,55	86,20%	55,54%
	25	86,00%	68,07	87,23%%	55,81%
	50	85,97%	67,55	88,44%	54,65%
Distancia	5	86,08%	71,26	79,56%	**64,53%**
	10	87,10%	72,42	84,54%	63,34%
	25	**87,52%**	**72,52**	88,12%	61,62%
	50	87,50%	72,14	**89,27%**	60,53%

Tabla 5. Resultados de la experimentación del modelo KNN en base al número de vecinos y al tipo de función de pesos.

A la vista de los resultados de la Tabla 5 se observa una diferencia notable entre el uso de la función de pesos uniformes y por distancia, debido a que esta última tiene unos mejores resultados en la métrica *F1-score.* Además, junto a la función por distancias, el mejor resultado se obtiene con el uso de 25 vecinos a tener en cuenta para la clasificación.

En tercer lugar, el entrenamiento del modelo de máquina vector soporte se ha desarrollado a partir de la aplicación de diversos tipos de núcleo junto a diferentes valores en la variable de holgura C.

SVM					
Núcleo	**C**	**Acierto**	***F1-score***	**Precisión**	***Recall***
Lineal	0,01	85,23%	77,73	86,91%	74,31%
	0,1	85,27%	77,83	86,84%	74,43%
	1	85,26%	77,82	86,82%	74,43%
	10	85,22%	77,76	86,73%	74,37%
Poliomial *con grado 3*	0,01	85,14%	77,32	**87,50%**	73,80%
	0,1	85,59%	78,33	87,44%	74,87%
	1	85,81%	78,85	87,28%	75,45%
	10	86,00%	79,25	87,29%	75,89%
Sigmoid	0,01	75,77%	67,98	68,73%	67,42%
	0,1	73,82%	66,41	66,49%	66,33%
	1	73,21%	65,87	65,84%	65,90%
	10	73,18%	65,84	65,81%	65,88%
RBF	0,01	85,31%	77,96	86,70%	75,82%
	0,1	85,83%	79,09	86,76%	75,82%
	1	86,09%	79,55	87,00%	76,30%
	10	**86,20%**	**79,84**	86,83%	**76,66%**

Tabla 6. Resultados del modelo de máquina vector soporte alternando el núcleo empleado y los valores de C.

Como se puede contemplar a través de la Tabla 6 de resultados, el mejor modelo de máquinas vector soporte es aquel que se emplea junto al núcleo *RBF* y con un valor de C igual a 10. Hay que señalar que el resultado con el uso del núcleo polinomial de tercer grado con el mismo valor de C es prácticamente similar.

En cuarto lugar, con respecto al modelo de *Naïve Bayes* se ha realizado el entrenamiento de la variante del modelo gaussiano el cual asume que los datos de entrada tienen una distribución normal, y también de la variante del modelo multinomial el cual asume que los datos de entrada siguen una distribución multinomial.

Naïve Bayes				
Variante	**Acierto**	***F1-score***	**Precisión**	***Recall***
Gaussiana	81,25%	**66,43**	64,67%	**68,28%**
Multinomial	**84,49%**	62,29	**88,49%**	48,06%

Tabla 7. Resultados del modelo de clasificación Naïve Bayes en su variante gaussiana y multinomial.

A partir de los resultados de la Tabla 7 se puede apreciar como el algoritmo *Naïve Bayes* gaussiano realiza el proceso de clasificación mejor que el multinomial. Esto se debe a que el uso del modelo gaussiano es adecuado cuando los datos son continuos

como lo son las características que se utilizan para el entrenamiento. El empleo del modelo multinomial es apropiado cuando los datos tienen valores discretos.

En quinto lugar, el algoritmo de clasificación de regresión logística se ha entrenado mediante la variación de la variable de holgura C.

Regresión logística				
C	**Acierto**	***F1-score***	**Precisión**	***Recall***
0,01	85,46%	65,69	**88,98%**	52,06%
0,1	**85,68%**	66,95	87,38%	54,26%
1	85,66%	**67,08**	86,84%	**54,65%**
10	85,64%	67,03	86,79%	54,59%

Tabla 8. Resultados del modelo de clasificación de regresión logística con diferentes valores de C.

A través de la Tabla 8 de resultados se puede ver como el modelo de regresión logística con el valor de la variable C igual a uno proporciona los mejores resultados.

Por último, el entrenamiento del modelo de red neuronal del perceptrón multicapa se ha llevado a cabo a partir del uso de diferentes números de capas ocultas, de la elección de la cantidad de neuronas que las componen y también de la variación del tamaño del *batch.* En lo que se refiere al número de neuronas de la capa de entrada cabe destacar que es equivalente al número de características extraídas, es decir 23 y, por otro lado, con respecto al número de neuronas en la capa de salida hay una para cada clase.

Además, dado que la métrica de evaluación que se ha elegido como referencia para comparar los modelos es la *F1-score*, al no incorporar la librería *Keras* ninguna monitorización durante la fase de entrenamiento para dicha métrica, se calcula justo después de entrenar una época con el conjunto de validación y se guarda el mejor modelo. Por tanto, en este modelo a diferencia de los demás, se ha utilizado un 60% de los datos para entrenamiento, un 20% para validación y el otro 20% restante para el testeo.

Perceptrón Multicapa						
Número de Capas ocultas	**Número de neuronas**	**Batch**	**Acierto**	***F1-score***	**Precisión**	***Recall***
1	[11]	36	86,12%	79,88	86,29%	76,84%
2	[14|6]	32	86,24%	79,99	86,65%	76,89%
2	[18|10]	32	86,27%	80,10	86,53%	**77,05%**
2	[17|10]	36	**86,77%**	**80,16**	**86,75%**	76,95%
3	[17|10|2]	36	86,03%	79,59	86,54%	76,44%

Tabla 9. Mejores resultados obtenidos en la fase de experimentación del modelo de clasificación perceptrón multicapa.

Con respecto a los resultados presentados en la Tabla 9 por el modelo de clasificación del perceptrón multicapa se contempla como el modelo que está compuesto por dos capas ocultas proporciona mejores resultados que con una o con tres. Así pues, la mejor combinación del perceptrón multicapa se compone por un número de 17 neuronas en la primera capa oculta y diez en la segunda además con un valor de 36 con tamaño del *batch*.

4.1.2. Recopilación de resultados

Una vez realizado el entrenamiento de los diferentes modelos de clasificación y obtenidos los resultados a partir de la evaluación de sus predicciones, a continuación, se realiza una comparación de la mejor versión de cada modelo con los demás.

Modelo	**Acierto**	***F1-score***	**Precisión**	***Recall***
Árbol de decisión	82,38%	67,53	66,55%	66,53%
KNN	**87,52%**	72,52	**88,12%**	61,62%
SVM	86,20%	79,84	86,83%	76,66%
Naïve Bayes	81,25%	66,43	64,67%	68,28%
Regresión logística	85,66%	67,08	86,84%	54,65%
Perceptrón multicapa	86,77%	**80,16**	86,75%	**76,95%**

Tabla 10. Recopilación del mejor resultado de cada modelo de clasificación.

A partir de los resultados de la Tabla 10 se observa como los modelos de árbol de decisión, *Naïve Bayes* y regresión logística no consiguen alcanzar en la métrica *F1-score* un valor del 70%. Este aspecto se puede deber a que al ser modelos sencillos no sean capaces de detectar las interrelaciones entre los diferentes datos.

Por otro lado, la técnica *KNN* consigue mejores resultados que los modelos anteriores pero, aun así, no alcanza a los modelos más complejos como el perceptrón multicapa y *SVM* los cuales tienen una mayor capacidad de detectar las relaciones entre características y datos. Estos dos últimos obtienen resultados prácticamente similares, aunque el perceptrón multicapa proporciona un valor superior de la medida *F1-score* convirtiéndose así en el mejor modelo desarrollado para la tarea de detección de humor.

Por último, con el fin de realizar una comparación con los modelos de clasificación que se desarrollaron en la competición *HAHA* 19, se ha utilizado el modelo perceptrón multicapa con los parámetros que han otorgado los mejores resultados en la

experimentación de este trabajo empleando únicamente los datos de entrenamiento y testeo de dicha competición.

***HAHA* 19**				
Equipo	**Acierto**	***F1-score***	**Precisión**	***Recall***
Adilism	**85,5%**	**82,1**	79,1%	**85,2%**
Perceptrón multicapa	78,5%	75,2	**80,7%**	74,0%

Tabla 11. Comparación del mejor modelo de clasificación obtenido durante la experimentación con el mejor resultado en la competición HAHA 19.

Como se puede observar en la Tabla 11, el modelo obtenido en este proyecto no logra alcanzar los mejores valores de la métrica *F1-score* que se obtuvieron en la tarea de clasificación. El equipo Adilism utilizó para esta tarea una combinación del modelo neuronal *BERT* y el modelo multinomial de *Naïve Bayes*. También se puede contemplar que la mayor diferencia entre el modelo del perceptrón multicapa y el modelo ganador se encuentra en la métrica de *Recall*.

4.2. Cálculo del nivel de humor

En cuanto al cometido de detectar cómo de humorístico es un tuit se debe destacar que se ha utilizado el corpus de tuits compuesto por las dos fuentes de las cuales se han obtenido los datos como se plantea en el final del punto 3.3.1. Además, de este conjunto de datos se ha utilizado el 80% de los tuits para realizar el entrenamiento de los modelos y el 20% restante para la evaluación cuyos subconjuntos se han obtenido de igual forma que los de clasificación. La métrica del error cuadrático medio (*RMSE*) es la utilizada para la comparación de la calidad de los modelos de regresión. Por último, resaltar que la puntuación de los tuits se comprenderá entre uno y cinco, siendo cinco la mejor valoración posible.

4.2.1. Desarrollo de la experimentación

Al igual que en la tarea anterior, en el proceso de experimentación del cálculo del nivel de humor de los tuits etiquetados como humorísticos se han desarrollado diversas pruebas de los modelos modificando sus componentes con el fin de conseguir el mejor modelo posible para cada variante.

Por un lado, el entrenamiento del modelo de vector de soporte regresión se ha desarrollado mediante la aplicación de diversos tipos de núcleo junto a diferentes valores en la variable de holgura C.

SVC		
Núcleo	**C**	***RMSE***
Lineal	0,01	0,655
	10	0,654
Polinomial *con grado 3*	0,01	0,659
	0,1	0,658
RBF	0,01	0,657
	0,1	**0,652**

Tabla 12. Resultados obtenidos por el modelo vector de soporte regresión.

A partir de la Tabla 12 se contempla que la diferencia entre la aplicación del núcleo al vector de soporte regresión es mínima en cuanto a los valores obtenidos en la métrica *RMSE*. No obstante, el núcleo *sigmoid* en esta tarea proporcionaba resultados muy altos en comparación con los demás, por lo que no se han plasmado en la tabla. El mejor modelo se consigue a partir de la variable de holgura con un valor de 0,1 con el uso del núcleo *RBF*.

Por otro lado, el entrenamiento del modelo de regresión del perceptrón multicapa se ha realizado a partir del uso de diferentes números de capas ocultas y de la cantidad de neuronas en cada una de ellas. También se ha variado el tamaño del *batch.* Hay que resaltar que, a diferencia del modelo de clasificación, en la arquitectura de este modelo de regresión hay solo una neurona en la capa de salida. La capa de entrada se compone de 23 neuronas, es decir, por el número de características extraídas de los tuits.

Asimismo, puesto que la métrica de evaluación que se ha escogido como referencia para comparar los modelos es la *RMSE*, al no incorporar la librería *Keras* ninguna monitorización durante la fase de entrenamiento para dicha métrica, se calcula justo después de entrenar una época con el conjunto de validación y se almacena el mejor modelo. Por tanto, con este modelo, a diferencia del anterior, se han utilizado un 60% de los datos para entrenamiento, un 20% para validación y el otro 20 restante para el testeo.

Perceptrón multicapa			
Número de Capas ocultas	**Número de neuronas**	***Batch***	***RMSE***
1	[15]	20	0,650
2	[17,2]	20	0,649
2	[40,100]	20	**0,648**
3	[23,40,15]	20	0,649

Tabla 13. Mejores resultados obtenidos en la fase de experimentación del modelo de regresión del perceptrón multicapa

Con los resultados de la Tabla 13 se observa como, a diferencia del modelo de clasificación, en el ejercicio de regresión el perceptrón multicapa obtiene mejores resultados con el empleo de una mayor cantidad de neuronas en las capas ocultas que el número de neuronas en la entrada. En este modelo el uso de dos capas ocultas proporciona resultados superiores que con otra cantidad. El mejor modelo que se ha conseguido utiliza un número de 40 neuronas en la primera capa oculta y 100 en la segunda además de un tamaño del *batch* con un valor igual a 20.

4.2.2. Recopilación de resultados

Justo después de desarrollar el entrenamiento de los distintos modelos de regresión y adquiridos los resultados a través de la evaluación de sus predicciones, se efectúa un contraste de la mejor versión del modelo vector de soporte regresión con la del modelo neuronal perceptrón multicapa.

Modelo	***RMSE***
SVC	0,652
Perceptrón multicapa	**0,648**

Tabla 14. Recopilación de los mejores resultados de cada modelo de clasificación

A la vista de los resultados presentados en la Tabla 14 se observa como la diferencia entre los dos modelos es muy baja y, al igual que en el trabajo de clasificación, el perceptrón multicapa proporciona el mejor resultado en esta tarea de regresión.

Finalmente, con el objetivo de realizar una comparación con los modelos de regresión que se presentaron en la competición *HAHA* 19, se ha utilizado el modelo perceptrón multicapa con los parámetros que han conseguido los mejores resultados en la experimentación de esta tarea, empleando únicamente los datos de entrenamiento y testeo con la puntuación que se proporciona en la competición.

***HAHA* 19**	
Equipo	**RMSE**
Adilism	0,736
Perceptrón Multicapa	**0,664**

Tabla 15.Comparación del mejor modelo de regresión obtenido durante la experimentación con el mejor resultado en la competición HAHA 19.

A partir de la Tabla 15 se puede observar que el modelo desarrollado en este proyecto proporciona mejores resultados en la métrica *RMSE* que el modelo ganador de la competición. Uno de los motivos por los que esto puede suceder es por método de extracción de las características de los tuits utilizados en este proyecto.

4.3. Estudio del humor en los tuits de personajes famosos

Con el fin de realizar el estudio del humor en los tuits escritos por personajes famosos se han empleado tanto el mejor modelo de clasificación como el de regresión obtenidos en las tareas anteriores, es decir, el perceptrón multicapa. Por un lado, se utiliza el modelo de clasificación para categorizar qué tuits son graciosos y, por el otro lado, se emplea el modelo de regresión para indicar cómo de graciosos son dichos tuits.

Asimismo, se ha decidido que las personalidades elegidas pertenezcan al campo del entretenimiento. Por ello, se ha realizado la selección de aquellos personajes que tengan una gran cantidad de seguidores en la red social de Twitter y por tanto, tengan una gran capacidad de influencia en la sociedad.

La primera persona escogida se trata de Ana Morgade Pérez. Además de ser una de las humoristas españolas con más seguidores, se ha elegido debido a que su carrera profesional siempre ha estado ligada al humor.

La segunda personalidad elegida es Dani Mateo Patau. Su selección se debe a que suele emplear la ironía y el humor en muchos de sus artículos o en sus programas a la hora de plasmar la actualidad.

Por último, el tercer personaje seleccionado para el estudio del humor en sus tuits es Jordi Évole Requena. El principal motivo para su selección es que es el periodista español con mayor número de seguidores y además de periodista, Jordi también ha trabajado como cómico en diferentes proyectos por lo que puede aportar al estudio datos interesantes.

4.3.1. Desarrollo de la experimentación

En primer lugar, se recopilan los últimos 500 tuits de cada una de las cuentas en Twitter a través de la herramienta de agrupación de tuits *Twint.* Después de la recopilación, se extraen las características relativas a cada tuit para que los modelos puedan realizar las predicciones.

A continuación de la extracción de características, se utiliza el perceptrón multicapa de clasificación y realiza su predicción etiquetando cada tuit como humor o no. En el momento en que se ha obtenido la clasificación completa de los tuits, se realiza el filtrado de aquellos que han sido etiquetados como graciosos y se utiliza el perceptrón multicapa de regresión para que obtenga el pronóstico de la puntuación asociada a cada uno de los tuits filtrados.

4.3.2. Análisis de los resultados

En primer lugar, en cuanto a los tuits recopilados de la cuenta de Jordi Évole en Twitter cabe señalar que el modelo de clasificación ha etiquetado 55 tuits como graciosos y los 445 restantes como no, además, aquellos que son graciosos el modelo de regresión ha asignado una puntuación.

Ahora bien, entre los tuits etiquetados como humor, encontramos una gran cantidad de ellos que se corresponden con diálogos que Jordi suele tener con alguna personalidad pública durante la entrevista que realiza. Por ejemplo, uno de los tuits con diálogo que tiene una puntuación de 2:

«Fortnite, Instagram y el confinamiento:

-¿Quién lleva mejor el confinamiento vosotros o vuestros padres?

-Nosotros. Nuestros padres están muy agobiados. Tienen mucho que hacer y además estamos nosotros en casa.

Berta y Jacobo en #LoDeQuédateEnCasa6»

Además, un aspecto de los tuits que han sido etiquetados como graciosos y no forman parte de un diálogo, es porque suelen contener signos de exclamación o de interrogación, algún *hashtag*, o palabras groseras. Un ejemplo de este tipo de tuits es el siguiente con una puntuación de 2,1:

«En esta serie estoy aprendiendo argot taleguero. Y político. ¿Qué es “mojar pista”? »

Por otro lado, al respecto de la puntuación de los tuits el modelo de regresión proporciona unos valores que en un rango entre 1,84 y 2,2. Así pues, el tuit de Jordi Évole con una mayor puntuación se corresponde con:

«-Oriol, ¿sabes qué dirá Torra?

-No tengo ni idea.

Con este diálogo fuera de cámaras empezó la entrevista más rara de mi vida. “Junqueras, una entrevista con historia”. Mañana veremos un avance en el estreno de @LoDeEvole La Sexta. 21:25…»

En segundo lugar, con relación a los tuits procedentes de la cuenta de Ana Morgade el modelo ha clasificado 42 de ellos como divertidos y como no los 448 restantes. La mayoría de tuits destacan por contener signos de exclamación, *hashtags*, palabras en mayúsculas o alguna que otra palabra ordinaria. Como ejemplo de esto se puede contemplar el siguiente tuit que ha sido puntuado con 1,92:

«Conclusión: NINGUNA. LA VIDA ES ASÍ, UNA MIERDA QUE NO SE EXPLICA. Queréis respuestas? YO TAMBIÉN JODER! Y aquí me quedo, en la basura de la incertidumbre. Qudedaos aquí conmigo, en la zona que no cubre. #GraciasPorVenir #Lavavajillas »

En lo relativo a la puntuación que se ha asignado a los tuits, los valores se encuentran entre 1,8 y 2,3, siendo el mejor tuit de Ana Morgade el siguiente:

«¿Que por qué no llevo cristales en las gafas?»

En último lugar, en lo que respecta a los tuits recopilados de la cuenta de Dani Mateo el modelo de clasificación ha etiquetado 17 de los 500 tuits como graciosos. Los tuits humorísticos contienen signos de exclamación, *hashtags*, palabras en mayúsculas o palabras groseras. Un ejemplo es el siguiente tuit con una valoración de 1,96:

«Prueba la dieta que ha cambiado la vida de millones de personas en todo el mundo y que tú abandonarás a los dos días sintiéndote no solo un puto gordo sino también un fracasado! Ahora con un 20% menos de autoestima!»

La puntuación proporcionada por el modelo de regresión proporciona valores comprendidos entre 1,93 y 2,16, siendo el tuit con una valoración superior el siguiente:

«3 globos de oro para "Érase una vez en Hollywood"? Entonces cuántos se merece "Pulp Fiction"? 27? No estoy de acuerdo. Yo me aburrí. Que se los quiten!!! (Le grita al ordenador. Muy loco. Se levanta y da vueltas. Nadie le escucha. El gato le observa...)»

Como conclusión, a partir del estudio realizado a las tres personalidades, se ha contemplado como la aparición de características en los tuits como son los diálogos, *hashtags*, símbolos de exclamación e interrogación, el uso de mayúsculas y de palabras groseras tienen un gran peso en el modelo de clasificación, ya que la existencia de estas propiedades hace que el modelo determine como humor un tuit sin tener en cuenta el contexto del que se está tratando realmente.

Otro aspecto obtenido a partir del estudio es que, al tener una gran parte de los tuits humorísticos las mismas características de las que se ha hablado en el párrafo anterior y estas tener una mayor influencia en el modelo de clasificación, el modelo de regresión ha proporcionado puntuaciones con valores similares siendo el menor valor un 1,8 y el mayor un 2,3.

5. Relación con los estudios cursados

A lo largo del grado se han adquirido una gran cantidad de conceptos y conocimientos de las asignaturas cursadas. Así pues, la mayor parte de las ideas utilizadas en este trabajo han sido aprendidas en las asignaturas procedentes de la rama de computación.

En primer lugar, cabe señalar que asignaturas como percepción (PER) junto a aprendizaje automático (APR) han otorgado los conocimientos esenciales en el ámbito tanto teórico como práctico de los diferentes modelos de aprendizaje empleados en el proyecto, además de presentar algunas representaciones aplicables a los textos.

En segundo lugar, la asignatura de sistemas de almacenamiento y recuperación del texto (SAR) ha proporcionado los conceptos necesarios para realizar el tratamiento y procesado de texto, también diferentes formas de representación del texto y además, ha impulsado el aprendizaje y uso de Python en dichas tareas. También esta materia ha llevado al empleo de algunas herramientas de procesamiento del lenguaje natural.

Con relación a asignaturas que no son propias de la rama de computación, la disciplina de gestión de proyectos (GPR) ha aportado los conocimientos necesarios para realizar la planificación y la ejecución del proyecto de forma eficiente para que el proyecto consiga desarrollar los objetivos planteados en el tiempo estimado.

Por otro lado, la materia de estructura de datos y algoritmos (EDA) ha otorgado los conocimientos esenciales a la hora de programar para emplear diferentes estructuras de datos y algoritmos.

Por último, se debe señalar que a lo largo de la realización de este trabajo se han ido desarrollando un gran número de competencias transversales como «aplicación y pensamiento práctico», «aprendizaje permanente», «diseño y proyecto» o «planificación y gestión del tiempo».

6. Conclusiones

En el presente trabajo se han utilizado técnicas de aprendizaje automático para obtener modelos que permiten detectar el humor de los usuarios a través de los mensajes que publican en la red social Twitter. Para ello, previamente al desarrollo del modelo hemos introducido los conceptos necesarios de los modelos de aprendizaje automático y el procesamiento del lenguaje natural.

Para construir un sistema capaz de realizar la detección de humor, en primer lugar, se ha construido un conjunto de datos. Con el fin de realizar la recopilación del corpus hemos obtenido los tuits de otros tres conjuntos de datos que se habían utilizado en la tarea de detección de humor. Uno de los problemas que nos hemos encontrado es que existían tuits que se repetían en los diferentes conjuntos, por lo que se ha realizado un control de estos y se han descartado los duplicados.

Así pues, el proceso de extracción de características se compone de la obtención de 23 características diferentes a partir del formato del texto y los temas que se tratan en el contenido del tuit.

Una vez extraídas las características, se ha realizado la experimentación de diferentes modelos de aprendizaje automático en la tarea de detección de humor, cuyo mejor clasificador desarrollado es el modelo neuronal del perceptrón multicapa. Asimismo, se ha realizado una comparación de los resultados obtenidos por nuestro modelo con los de la competición *HAHA*@IberLEF 2019 y hemos comprobado que nuestro modelo no consigue alcanzar los mejores resultados.

Después del desarrollo de los modelos de clasificación, se ha realizado la experimentación en la tarea de asignar una puntuación de humor a los tuits etiquetados como graciosos. Los mejores resultados los ha proporcionado de nuevo el modelo neuronal del perceptrón multicapa y, al igual que en la tarea de clasificación, se ha hecho una comparación de los resultados conseguidos por nuestro sistema con los de la competición *HAHA*@IberLEF 2019 y hemos comprobado que nuestro modelo de percepción mejora los resultados del ganador de la tarea.

Además, hemos hecho el estudio de humor en los tuits procedentes de tres personajes españoles famosos aplicando nuestros modelos de clasificación y de regresión. En este estudio hemos podido comprobar que el modelo de clasificación no

funciona completamente bien debido a que la aparición de ciertos elementos puede llevar a etiquetar siempre como humor el tuit que los contenga.

Por otro lado, el modelo de regresión tiende a puntuar los tuits con valores similares por lo que creemos que es debido en parte al modelo de clasificación que tiene más en cuenta ciertas características como puede ser la aparición de hashtags, exclamaciones o interrogaciones y mayúsculas y suele etiquetar como graciosos tuits con elementos similares.

Por último, se debe destacar que el desarrollo de este proyecto ha llevado a comprender con una mayor profundidad el funcionamiento de forma teórica de los algoritmos de aprendizaje automático, además de aprender a implementarlos de una forma diferente a la vista en el grado mediante el empleo de diferentes librerías en el entorno de Python.

7. Trabajos futuros

En este trabajo se ha decidido tratar el objetivo principal de detección de humor utilizando la extracción de 23 características y se ha comprobado que los resultados de los modelos obtenidos mediante el entrenamiento con dichas características no son tan buenos como los alcanzados mediante el empleo del modelo de lenguaje no supervisado *BERT.* Por ello, una futura línea de investigación sería abordar la tarea de clasificación con el uso del modelo *BERT* combinado con otros modelos de aprendizaje automático.

Otro aspecto a considerar es que el corpus que se ha empleado en este trabajo tenía una diferencia muy grande entre la cantidad de tuits no humorísticos y los humorísticos. En un futuro proyecto, sería interesante equiparar el número de tuits de ambas clases con el fin de obtener modelos capaces de realizar la función de clasificación y regresión con una mayor calidad.

8. Bibliografía

1. GARCÍA, DANIEL. Los algoritmos del humor. *La Vanguardia* [en línea]. 2017. [Consulta: 14 junio 2020]. Disponible en: https://www.lavanguardia.com/tecnologia/20170408/421527558553/algoritmos-humor.html.

2. ALPAYDIN, ETHEM. *Introduction to machine learning*. 2. Cambridge: MIT Press, 2014. ISBN 978-0-262-01243-0

3. KEMP, SIMON. Digital 2020: 3.8 billion people use social media - We Are Social. *We Are Social* [en línea]. 2020. [Consulta: 14 junio 2020]. Disponible en: https://wearesocial.com/blog/2020/01/digital-2020-3-8-billion-people-use-social-media

4. LOSADA, ANALÍA & LACASTA, MARISOL. Sentido del Humor y sus Beneficios en Salud. *Journal of Clinical and Health Psychology* [en línea]. 2019, 12 (1).1-21 [Consulta: 14 junio 2020]. ISSN 1850-6216. Disponible en: https://www.researchgate.net/publication/334749609_Sentido_del_Humor_y_sus_Beneficios_en_Salud_Title_Sense_of_Humor_and_its_health_benefits

5. REYES ROJAS, MÓNICA. Reseña de "Psicología del humor: un enfoque integrador" de MARTIN, ROD. *Revista Latinoamericana de Psicología* [en linea]. 2010, 42(2), 328-334 [Consulta: 14 junio 2020]. ISSN: 0120-0534. Disponible en: https://www.redalyc.org/articulo.oa?id=80515381014

6. MURPHY, KEVIN. *Machine learning. A probabilistic perspective*. Cambridge: The MIT Press, 2013. ISBN: 978-0-262-01802-9

7. GIROLAMI, MARK. & ROGERS, SIMON. *A First Course in Machine Learning, Second Edition.* 2. New York: CRC Press 2016. ISBN 9781498738484

8. MARTINEZ HERAS, JOSÉ. Máquinas de Vectores de Soporte (SVM) - IArtificial.net. *IArtificial.net* [en línea]. 2020. [Consulta: 14 junio 2020]. Disponible en: https://iartificial.net/maquinas-de-vectores-de-soporte-svm/.

9. SANCHO CAPARRINI, FERNANDO. Redes Neuronales: una visión superficial - Fernando Sancho Caparrini. Cs.us.es [en línea]. 2010. [Consulta: 14 junio 2020]. Disponible en: http://www.cs.us.es/~fsancho/?e=72.

10. MIKOLOV, TOMAS & CHEN, KAI & CORRADO, G.S & DEAN, JEFFREY. 2013. Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.

11. KARANI, DHRUVIL. Introduction to Word Embedding and Word2Vec. Medium [en línea]. 2020. [Consulta: 14 junio 2020].Disponible en: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

12. MIKOLOV, TOMAS & SUTSKEVER, IlYA & CHEN, KAI & CORRADO, G.S & DEAN, JEFFREY. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems.* 2013, 26. [Consulta: 14 junio 2020]. Disponible en: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

13. DEVLIN, JACOB, CHANG, MING-WEI, LEE, KENTON, TOUTANOVA, KRISTINA. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volumen 1 (Long and Short Papers).* 2019, 1. 4171–4186 [Consulta: 14 junio 2020]. Disponible en: https://www.aclweb.org/anthology/N19-1423

14. HAHA - Humor Analysis based on Human Annotation - IberLEF 2019. *Fing.edu.uy* [En línea]. 2020. [Consulta: 14 junio 2020]. Disponible en: https://www.fing.edu.uy/inco/grupos/pln/haha/index.html

15. ISMAILOV, ADILZHAN.: Humor Analysis Based on Human Annotation Challenge at IberLEF 2019: First-place Solution. *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019)*. CEUR Workshop Proceedings, CEUR-WS, Bilbao, Spain 2019.

16. LLUÍS PADRÓ. Analizadores Multilingües en FreeLing Linguamatica. 2011,3 (2). 13-20.

17. CHOLLET, FRANÇOIS and others.: Keras. 2015. Disponible en: https://keras.io

18. STEVEN BIRD, EWAN KLEIN, EDWARD LOPER, DAN GARRETTE, PETER LJUNGLÖF, JOEL NOTHMAN, MIKHAIL KOROBOV, STEVEN BIRD, ALEXIS DIMITRIADIS. The nltk project. Disponible en: http://www.nltk.org.

19. OLIPHANT, TRAVIS: A guide to NumPy. 2006. USA: Trelgol Publishing. Disponible en: http://www.numpy.org/

20. PEDREGOSA FABIAN, VAROQUAX GAËL, GRAMFORT ALEXANDER, MICHEL VINCENT, THIRION BERTRAND, GRISEL OLIVER, BLONDEL MATHIEU, PRETTENHOFER PETER, WEISS RON, DUBOURG VINCENT, VANDERPLAS JAKE, PASSOS ALEXANDRE, COURNAPEAU DAVID, BRUCHER MATTHIEU, PERROT MATTHIEU, y DUCHESNAY ÉDOUARD. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*. 2011, 12. 2825–2830.

21. CASTRO, SANTIAGO, CUBERO, MATÍAS, GARAT, DIEGO AND MONCECCHI, GUILLERMO. Is This a Joke? Detecting Humor in Spanish Tweets. *Ibero-American Conference on Artificial Intelligence [En línea]*. 2016, 10022. 139-150 [Consulta: 14 junio 2020]. DOI 10.1007/978-3-319-47955-2_12

22. CASTRO, SANTIAGO & CHIRUZZO, LUIS & ROSA, AIALA & GARAT, DIEGO & MONCECCHI, GUILLERMO. (2018). A Crowd-Annotated Spanish Corpus for Humor Analysis. *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media [En línea]*. 2018, 7-11 [Consulta: 14 junio 2020]. DOI 18653/v1/W18-3502.

Glosario

- ***Entropía***: medida de incertidumbre que proporciona la pureza de una colección arbitraria de ejemplos.
- ***Función de activación ReLU***: transforma los valores de entrada suprimiendo los valores negativos y dejando los valores positivos de la misma forma que los recibe.
- ***Función de activación sigmoide***: transforma los valores de entrada a una escala de valores entre el cero y el uno, donde los valores muy altos tienden de manera asintótica a uno y los valores muy bajos tienden de forma asintótica a cero.
- ***Función de activación softmax***: transforma los valores de salida en un conjunto de probabilidades de tal forma que la suma de todas las salidas debe ser igual a uno.
- ***Ruido***: muestras atípicas de un conjunto de datos.
- ***Token***: extracción de un elemento de una oración que se corresponde con una palabra o con un símbolo.
- ***Wordnet***: es una base de datos léxica que reúne palabras en conjuntos de sinónimos llamados *synsets*, proporcionando definiciones cortas y generales y almacenando las relaciones semánticas entre los conjuntos de sinónimos.