Universidad de Valladolid

Grado en Ingeniería Informática

Trabajo de Fin de Grado

Análisis de sentimientos y emociones en redes sociales usando ML

REALIZADO POR:
Alberto Calvo Madurga

TUTELADO POR:
Valentín Cardeñoso Payo

Índice general

Índice de figuras

Índice de cuadros

Resumen

La irrupción de las redes sociales en nuestro día a día como medios de comunicación básicos ha generado un interés creciente por el estudio de la forma de comunicarnos en ellas y cómo nuestros sentimientos y emociones influyen en esa comunicación. En este trabajo se presenta la construcción de un sistema de reconocimiento automático de sentimientos, partiendo de un estudio de los trabajos existentes. De dicho análisis se deriva la estructura general de resolución de este problema y se detallan las distintas fases diferenciadas. Inicialmente se consideran las técnicas de Bag of Words y Words Embeddings para la fase de extracción de características y los algoritmos de Machine Learning SVM, Random Forest y Regresión Logística en una tarea de clasificación supervisada. Se detallan los procedimientos seguidos en todas las etapas del proceso hasta llegar a construir un sistema de clasificación de sentimiento en base a la polaridad positiva o negativa.

Palabras clave: *análisis de sentimientos*, *análisis de emociones*, *machine learning*, *procesamiento del lenguaje natural*

Abstract

The irruption of social networks in our daily life as basic media has generated a growing interest in the study of how we communicate in them and how our feelings and emotions influence that communication. In this paper we present the construction of automatic feelings recognition system, based on the study of existing works. From this analysis, the general structure for solving this problem is derived and the different phases are detailed. Initially the techniques of Bag of Words and Words Embeddings for the feature extraction phase and the Machine Learning algorithms SVM, Random Forest and Logistic Regression are considered in a supervised classification task. The procedures followed in all stages are detailed until a sentimental classification system is built based on positive or negative polarity.

Keywords: *sentiment analysis*, *emotion analysis*, *machine learning*, *natural languaje processing*

Agradecimientos

Me gustaría aprovechar la elaboración de este trabajo que supone el fin de una etapa llena de aprendizaje y buenas experiencias. Agradecer a mis compañeros de carrera por haber compartido este camino intenso a mi lado, apoyándonos y ayudándonos cuando la situación lo ha requerido.

Agradecer también a mi tutor, Valentín, por la buena disposición mostrada y la ayuda prestada en el desarrollo del proyecto, y es que un buen ambiente de trabajo y la confianza para resolver cualquier tipo de duda o hacer cualquier tipo de comentario, es clave y se plasma en un trabajo final de calidad.

Por supuesto agradecer a mis padres, mi hermana y mi familia que son los que me han inculcado los valores del esfuerzo que me han hecho llegar hasta donde estoy ahora mismo, sin ellos nada de esto sería posible.

Para finalizar, agradecer a mi grupo de amigos de siempre, Jugones. Alguien dijo una vez que "Los amigos, son la familia que elegimos", no podría estar más de acuerdo con esta afirmación, sin su apoyo moral constante, estos años tan intensos se hubieran hecho el doble de complicados.

Capítulo 1. Introducción

1.1 Motivación

En este trabajo se presenta un estudio del problema de reconocimiento de emociones en mensajes de texto intercambiados en redes sociales, como aproximación al problema general de análisis y clasificación de emociones y sentimientos a partir de documentos textuales. El enfoque que se da no pretende ser exhaustivo ni definitivo sino que se desea proporcionar una vía para comprender la importancia del problema y sus facetas, a través de la solución, paso a paso, de un problema de clasificación automática de textos cortos extraídos de la red social **Twitter**.

No es ninguna sorpresa decir que las redes sociales han invadido nuestro día a día. Estos sitios facilitan la comunicación entre las personas, intercambiando información vía imágen, vídeo o texto. Con la inmensa cantidad de información compartida en estas plataformas, ¿cómo de útil sería el disponer de un sistema que pudiera reconocer el estado emocional o las intenciones de los usuarios que emiten los mensajes en las redes sociales? Esta tarea se denomina reconocimiento de sentimientos y emociones y es parte del procesamiento del lenguaje natural.

La detección de sentimientos y emociones en redes sociales es uno de los usos más novedosos del procesamiento del lenguaje natural. Cada vez más las empresas y profesionales del marketing utilizan estas tecnologías para saber la percepción que tienen los usuarios sobre una marca, producto o servicio. Para ello se utilizan datos que provienen de redes sociales, como mensajes, comentarios o reacciones y con ellos pretenden extraer información que les permita tomar mejores decisiones de negocio.

Está claro que un sistema construido podrá realizar la tarea de procesamiento del lenguaje natural con un gran acierto, pero por motivos obvios, no toda la información se puede captar de forma acertada. La forma de expresarse del ser humano es cambiante y está llena de detalles que no siempre se pueden procesar correctamente: sarcasmo, ironías, coloquialismos, etc. Esto aumenta sustancialmente la dificultad de construir un sistema de reconocimiento automático de emociones.

Como futuro graduado en el Doble Grado de Ingeniería Informática y Estadística, este trabajo es un buen reto donde se pueden aplicar técnicas aprendidas en multitud de

asignaturas de ambas carreras, como por ejemplo, algoritmos de aprendizaje automático (*Minería de Datos* y *Técnicas de Aprendizaje Automático*), tratamiento del lenguaje (*Gramática y Lenguajes Formales*), visualización de datos (*Computación Estadística o Estadística Descriptiva)* o gestión y organización de proyectos (*Planificación y Diseño de Sistemas Computacionales*).

Del mismo modo, este proyecto junto con el Trabajo de Fin de Grado de Estadística, supone el primer gran trabajo "individual" realizado, con las ganas y el ánimo que esto supone para realizar un buen proyecto del que quizás en un futuro no muy lejano se pueda utilizar en el ámbito laboral.

1.2 Objetivos y tareas

El objetivo principal del trabajo es desarrollar y comparar diversos sistemas de clasificación de emociones a partir de texto en español extraídos de redes sociales.

Las tareas que se van a realizar durante el transcurso del trabajo son:

- **Estudio del problema:** Una primera tarea de estudio del contexto en el que surge el problema, junto con un análisis de los trabajos en el área y la forma en que se aborda la resolución del mismo.

- **Recopilación de datos útiles para entrenamiento de modelos:** Si bien existe una gran abundancia de conjuntos de datos perfectamente preparados en inglés para construir un sistema de reconocimiento de sentimientos propio, en castellano no contamos con esa suerte. Se realiza por tanto una primera tarea de extracción de los datos sin etiquetar directamente de la red social **Twitter**.

- **Etiquetado de documentos a partir de diccionarios:** Ante la ausencia de un conjunto de datos en castellano etiquetado manualmente con el que trabajar, se construye un sistema de etiquetado basado en léxicos de polaridad de sentimientos cuya salida será tratada como la etiqueta real de cada *tweet*.

- **Obtención de modelos de clasificación:** Una de las tareas más importantes y decisivas del trabajo será aplicar los conceptos aprendidos en aprendizaje automático para construir un modelo de clasificación óptimo.

- **Evaluación de resultados:** Finalmente se tomarán decisiones sobre el sistema final a construir en base a la comparación de los modelos construidos, escogiendo así el sistema más óptimo posible.

1.3 Plan de Trabajo

En esta sección se exponen las tareas relativas a la gestión del proyecto ya que la correcta gestión de los esfuerzos es una tarea crítica y determinante en el desarrollo cualquier proyecto, y en consecuencia, del cumplimiento de sus objetivos.

El proyecto tiene lugar como Trabajo de Fin de Grado en el Doble Grado de Ingeniería Informática y Estadística de la Universidad de Valladolid. Tiene una carga de 12 créditos que equivalen a 300h de trabajo. El proyecto, si bien fue acordado tiempo antes, se considera que se inicia el 24 de febrero con una finalización estimada del 1 de julio de 2020. Si nos ajustáramos a estos plazos, se contaría con 18 semanas, donde se estiman 15-20h a la semana de trabajo que equivaldrían a 3-4h diarias si excluyéramos los fines de semana como días laborables.

Una vez se han aclarado los objetivos y tareas a lograr en el proyecto, se define un plan de actuación que irá acompañado de dos diagramas de Gantt que ilustran la planificación inicial 1.1 y el esfuerzo real del trabajo 1.2.

Planificación de tareas

Lo primero que se hace es una introducción a la problemática de la que trata el trabajo, analizando la utilidad de la detección de sentimientos y emociones y la finalidad con la que las empresas y profesionales construyen dispositivos para lograr este fin.

A continuación se da una visión teórica de la estructura genérica de resolución de este problema basándonos en los trabajos realizados en los últimos 10 años en este campo. De esta forma podremos entender mejor la importancia que cada fase aporta al resultado global, desde la obtención de los datos con los que se trabajará hasta la evaluación del prototipo final. En cada fase podemos encontrarnos una gran diversidad de técnicas o métodos que se intentan explicar de forma que el lector pueda tener una visión clara del procedimiento durante el transcurso del documento, y de donde se irá tomando conciencia de las ventajas y facilidades que aportan al experimento. Distintas técnicas son utilizadas para distintos objetivos específicos, sin embargo existen pasos comunes realizados en todos los análisis.

Una vez entendidos y asimilados los conceptos introducidos, se elabora un experimento donde poner en práctica lo aprendido. Se describe la arquitectura del experimento y se explican las técnicas escogidas en cada fase para llevarlo a cabo. Solo cuando ya se ha indagado en las múltiples posibilidades para abordar el problema y se ha asimilado la naturaleza del mismo, podemos decir que contamos con una mayor licencia para decidir que técnicas nos interesan más para el problema específico que queremos resolver.

Llegados a este punto, se define el plan específico que va a determinar el sistema experimental que se va a realizar y el objetivo que se persigue. Nuestro experimento va a centrarse únicamente en *tweets* escritos en castellano, con la problemática que esto pudiera

suponer. Por tanto lo primero que haremos será obtener los datos con los que trabajar. Inicialmente se buscó un conjunto de datos formado por *tweets* escritos en castellano, que fuera relativamente actual y sobre todo que estuviese etiquetado con respecto al sentimiento o emociones que pretenda transmitir. Ante la negativa de esto, se dota de una mayor importancia a esta fase realizando la extracción de los datos directamente de **Twitter** mediante sus APIs y elaborando un sistema de etiquetado basado en léxicos que nos permita clasificar los *tweets* extraídos.

La fase de preprocesamiento se compone de muchas de las técnicas utilizadas en los análisis de textos en redes sociales (tratamiento de emojis, menciones, hashtags, coloquialismos, palabras mal escritas...) haciendo énfasis durante todo el experimento en como influye la técnica de lematización en los resultados. Una vez hecho el preprocesamiento nos disponemos a construir nuestro sistema de etiquetado particular que clasifica en función de un sistema de voto de mayoría de cuatro subsistemas de etiquetado individuales. Se clasificará en tres clases distintas: "positivo", "neutro" y "negativo" en función de la idea general que transmite cada *tweet*. Finalmente trabajaremos únicamente con aquellos *tweets* que no estén clasificados como "neutros", que en la mayoría de los casos significa ausencia de criterio para polarizar a un extremo u otro.

Con el objetivo de analizar por qué existen estos *tweets* que podríamos decir que son nulos en lo que a sentimiento explícito se refiere, se realizar una tarea de análisis descriptivo del conjunto de datos final con el que se va a trabajar, indagando en las diferencias entre los *tweets* que han podido ser etiquetados con respecto a alguna de las dos polaridades y los que no. En dicha tarea diferenciaremos los *tweets* "no nulos" de los *tweets* "nulos" para analizar en ellos características como la longitud de los mismos, los términos utilizados en base al lenguaje español y los emojis utilizados.

La fase de extracción de características irá encaminada a una tarea posterior de análisis supervisado. Se analizan las dos técnicas más comunes utilizadas: **Bag of Words** y **Word Embeddings**. Al igual que otros de los procedimientos que se utilizan en múltiples tipos de análisis, estas dos técnicas son específicas de la tarea del Procesamiento de Lenguaje Natural. En la parte de la construcción del clasificador nos apoyaremos en los trabajos realizados en este campo escogiendo los clasificadores comúnmente utilizados como pueden ser Support Vector Machine, Regresión Logística o un ensemble como es Random Forest.

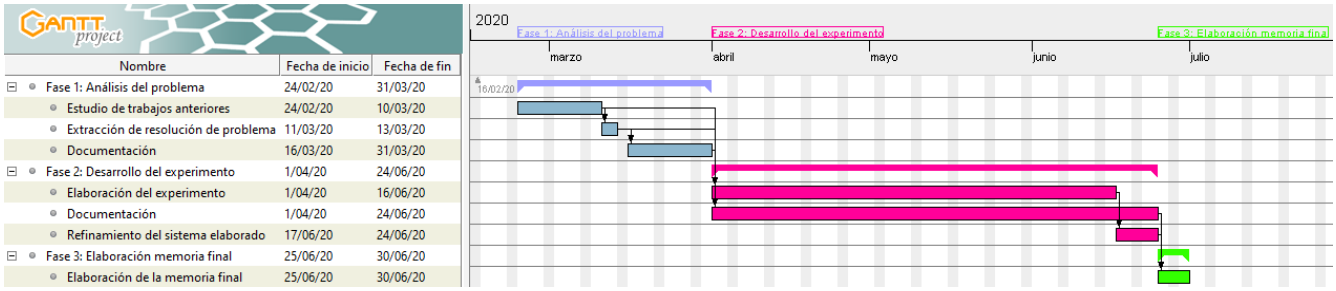
Estas dos fases están unidas mediante un proceso de selección de parámetros que buscan la solución más óptima de cara a un experimento de validación cruzada de 5 folds con el conjunto de entrenamiento. El conjunto de entrenamiento ha sido construido como una partición estratificada 80/20 del conjunto de datos preprocesado sin los *tweets* etiquetados como "neutros".

Una vez se ha seleccionado el clasificador que mejor funciona, se prueba el rendimiento primero con un experimento HoldOut 80/20 con el conjunto de prueba que se separó al principio y posteriormente con un experimento de validación cruzada estratificada de 10 folds con todos los datos. Finalmente conseguimos un sistema que clasifique un *tweet* en base a su polaridad, es decir, como *positivo* o *negativo*.

Con todo el análisis ya realizado, se pretende explicar de la manera más didáctica e ilustrativa posible al lector como se ha procedido. Mostrando los archivos y programas utilizados de manera que se pueda replicar el trabajo y continuar con alguna de las posibles ampliaciones de trabajo que se proponen.

Planificación inicial

El diagrama 1.1 muestra la planificación inicial del proyecto. Se muestran las tareas a abordar divididas en tres fases diferenciadas: análisis del problema, desarrollo del experimento y elaboración de la memoria final.



Nombre	Fecha de inicio	Fecha de fin
Fase 1: Análisis del problema	24/02/20	31/03/20
Estudio de trabajos anteriores	24/02/20	10/03/20
Extracción de resolución de problema	11/03/20	13/03/20
Documentación	16/03/20	31/03/20
Fase 2: Desarrollo del experimento	1/04/20	24/06/20
Elaboración del experimento	1/04/20	16/06/20
Documentación	1/04/20	24/06/20
Refinamiento del sistema elaborado	17/06/20	24/06/20
Fase 3: Elaboración memoria final	25/06/20	30/06/20
Elaboración de la memoria final	25/06/20	30/06/20

Figura 1.1: *Planificación inicial del proyecto*

Durante las dos primeras fases del proyecto, existe una tarea de documentación y refinamiento constante, pero se define claramente puntos de finalización que deben ser cubiertos antes de avanzar de fase. A continuación se muestra el esfuerzo final con los respectivos comentarios de las modificaciones ocurridas.

Esfuerzo real

Tal y como era de esperar, la planificación inicial se ha visto alterada en múltiples puntos ya que es improbable el seguimiento exacto de la misma. Esto se debe tanto a retrasos en la ejecución de las tareas como a la aparición de imprevistos que requieren una adaptación dinámica de la planificación.

En el desarrollo del proyecto se ha mantenido un cuaderno de bitácora donde se han apuntado diariamente los avances producidos. Esto ha ayudado a elaborar un diagrama que muestra el esfuerzo real del proyecto, con las fechas de transición entre fases y tareas. El diagrama 1.2 muestra el esfuerzo real del proyecto con un mayor detalle, donde existe una demora de 15 días hasta el 16 de Julio 2020, fecha límite de entrega del trabajo.

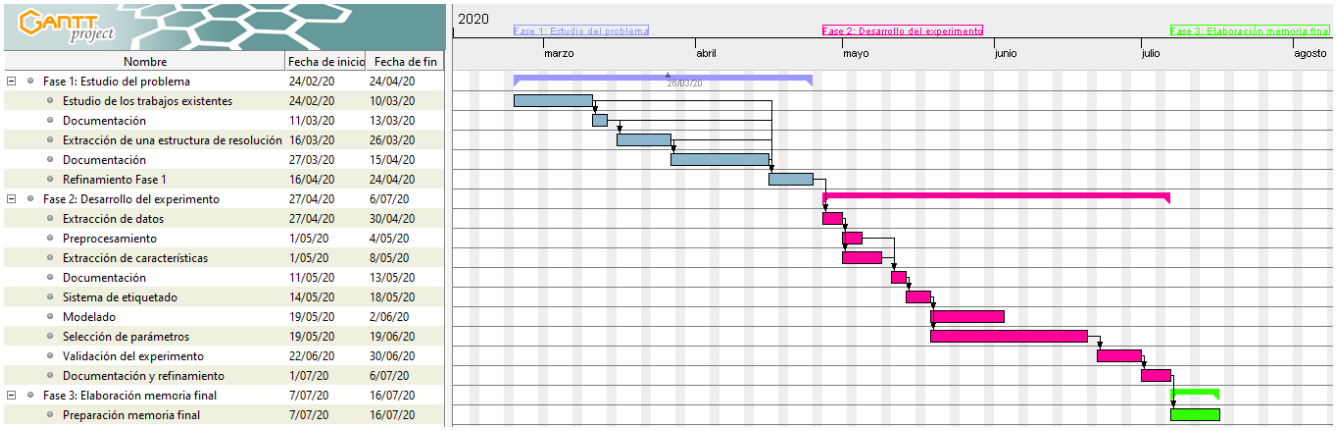


Figura 1.2: *Esfuerzo real del proyecto*

1.4 Estructura

La memoria del trabajo contiene cinco capítulos en los que se desarrollan las partes fundamentales del trabajo. En la primera parte integrada por los **capítulos 1** (Introducción) y **2** (Marco Teórico), se presenta la introducción y el marco teórico en el que surge el proyecto.

En el **capítulo 2** (Marco Teórico) se presenta la problemática que se va a abordar con el contexto natural en el que surge. Se analizan teorías de emociones, que han servido de inspiración para trabajos de gran relevancia. Además se expone el esqueleto básico común que presentan los múltiples trabajos que se realizan en este campo. Este esqueleto define las distintas fases por las que pasan todos los análisis: obtención de datos, preprocesamiento, extracción de características, modelado, diseño del experimento y validación. Especial interés tiene la fase de extracción de características ya que es una etapa específica del Procesamiento del Lenguaje Natural.

En el **capítulo 3** (Desarrollo del experimento) se encuentra la descripción del núcleo central del trabajo y en él se expone detalladamente cada paso y decisión tomada en la construcción de nuestro prototipo. Se describe en secciones diferentes las fases diferentes de construcción que acabamos de enumerar.

El **capítulo 4** (Resultados y discusión) contiene los resultados completos de las pruebas realizadas junto con la explicación de las decisiones finales tomadas y para terminar, en el **capítulo 5** (Conclusiones y trabajo futuro), se exponen las conclusiones finales de la memoria y posibilidades de trabajo futuro, ya bien sea por ser tareas que se salen del ámbito del trabajo, por requerir un tiempo de desarrollo muy elevado o por ser tareas complementarias específicas para determinadas labores.

Capítulo 2. Marco Teórico

2.1 Emociones y comunicación

> El ser humano es social por naturaleza
>
> *Aristóteles*

Sea cual sea el medio de comunicación empleado, el intercambio de información asociado a la comunicación entre seres humanos, se ve influenciado no sólo por la intención que conlleva la generación del mensaje sino por los sentimientos y actitudes del emisor y receptor del mismo. A su vez, la comunicación contribuye a generar en el receptor y en el emisor sentimientos y actitudes, que condicionarán la posterior comunicación. Paul Watzlawick, autor de la **Teoría de la Comunicación Humana** [4], ya planteaba que los problemas de comunicación entre las personas se deben a que no siempre se tienen los mismos puntos de vista. Precisamente el primer axioma de la teoría es "es imposible no comunicar".

Existen múltiples formas en las que se producen estos intercambios, desde vía textual, oral o incluso simplemente con las expresiones faciales. Como es lógico pensar, todas estas variantes comparten características comunes, y a su vez, cada una tiene su campo de estudio con una serie de características específicas. Con la intención de realizar un trabajo acotado a un terreno fijo, nos centraremos en el análisis textual de los sentimientos y emociones.

Ciertamente existe una gran variedad de tipos de documentos, desde el ensayo elaborado tras la reflexión y el estudio, a la comunicación espontánea y casi instantánea de una idea, sentimiento, opinión, crítica o comentario a través de una red social. Con la irrupción de la tecnología en la era en la que vivimos, se presentan con mayor facilidad formas de comunicación que no hace sino facilitar la compartición de nuestros pensamientos y puntos de vista con el mundo.

Esta gran capacidad de conectividad proporcionada por las redes sociales, nos facilita la posibilidad de generar un valor informativo colectivo. Pierre Levy introduce por primera vez en los años 1990s el concepto de *inteligencia colectiva* [3] basándose en que

gracias a la inteligencia humana combinada con el razonamiento reflexivo, el ser humano utiliza el lenguaje como esquema para dar argumentos y producir resultados. Con este concepto en mente, sería de gran utilidad saber si se puede desarrollar un nuevo tipo de instrumento para visualizar el estado de la realidad. ¿Podríamos tener un sistema que observara la realidad física y pudiera proporcionar una visión totalmente fidedigna mediante la inteligencia colectiva de los usuarios de las redes sociales?, ¿que propiedades debería poseer dicho dispositivo?, ¿como de susceptible sería para la manipulación intencionada de la realidad?, ¿que armas tendría para mitigar estos efectos?

Si pudiéramos conseguir este ideal de dispositivo, la intervención del ser humano estaría ayudando a los algoritmos a esquivar la desfiguración de la realidad, ya que estos no tienen la habilidad de reconocer cuando una actividad se corresponde a un comportamiento anormal. A diferencia de los sensores artificiales, los humanos podemos resumir nuestras observaciones de forma que nos permiten crear nuestra propia interpretación de la realidad. Si bien estas interpretaciones pueden estar sesgadas o ser subjetivas, el objetivo es precisamente conseguir que mediante el instrumento desarrollado, estos desafíos queden vencidos.

Queda por tanto planteada la gran problemática del trabajo, que es conseguir entender cómo poder llegar a elaborar sistemas que predigan o clasifiquen textos, en este caso provenientes de las redes sociales, en base a los sentimientos o emociones que estos pretenden transmitir.

2.2 Modelos de emociones

Después de introducir el contexto en el que surge este análisis, debemos entender "qué" es lo que realmente buscamos. Si nos referimos el título encontramos dos conceptos clave como son: *Emoción* y *Sentimiento*. Aunque en ocasiones estos dos términos son tratados como iguales, la detección de emociones es una tarea más laboriosa que la detección de sentimientos. En líneas generales podemos decir que un 'Sentimiento' es el efecto de una 'Emoción' (**Broad** [5]). De esta forma *'Contento'* o *'Triste'* son ejemplos de emociones y *'Positivo'* y *'Negativo'* son los sentimientos asociados correspondientes. Sabiendo esto se ve más claro que la detección de sentimientos se centraría más bien en conocer la polaridad de la actitud de una persona hacia otra persona, evento, cosa, etc., mientras que la detección de emociones trae consigo la tarea de definir un modelo de emociones acorde con teorías psicológicas asociadas. Estos modelos se dividen en dos grandes grupos:

- **Categóricos**: En los que existe una lista finita de categorías de emociones discretas unas de otras. Los modelos utilizados tienen un número de emociones descritas notablemente inferior a los modelos dimensionales.
- **Dimensionales**: En los que se definen unas pocas dimensiones con sus parámetros y cada emoción está asociada a unos valores concretos de estas. Estos modelos son más complejos y por tanto admiten unas estructuras más elaboradas.

A lo largo de los años se han elaborado distintos modelos que pretenden explicar cómo se clasifican las emociones de distinta manera y basándose en diferentes experimentos o teorías psicológicas. De esta manera se han construido múltiples modelos que presentan estructuras más o menos complejas como, por ejemplo, el cubo de las emociones de **Lovheim** [6] (modelo dimensional basado en las interacciones de nuestros neurotransmisores; dopamina, noradrenalina y serotonina), modelo de **Shaver** [7] (modelo categórico que establece una jerarquía en forma de árbol donde encontramos varios niveles de emociones), modelo de **Ekman** [8] (modelo categórico que defiende que el ser humano reacciona ante los eventos mediante seis emociones básicas) o la rueda de las emociones de **Plutchik** [9] (modelo dimensional que combina ocho emociones básicas y la forma en la que se relacionan mediante la intensidad de cada una). Más modelos se han construido de forma independiente o utilizando como punto de partida alguno de los ya creados y es evidente pensar que en los trabajos realizados en los últimos años, se puedan adecuar estos modelos al experimento que se va a realizar. Por tanto queda claro que la elección del modelo condiciona la elección de las técnicas a utilizar y por ende el desarrollo del prototipo.

2.3 Las Redes Sociales y las emociones

Las redes sociales son estructuras formadas en Internet por personas u organizaciones que se conectan a partir de intereses comunes. A través de ellas, se crean relaciones entre individuos o empresas de forma rápida, sin jerarquía o límites físicos. En estas estructuras se pueden representar como nodos a los individuos que la forman, y con líneas, las relaciones formadas entre ellos creando así una estructura de red.

Las redes sociales sirven como instrumentos que son utilizados por las personas para expresarse ante eventos de la vida real. Estas expresiones pueden suceder de forma directa o indirecta mediante textos, discursos, imágenes o incluso gestos. El estudio de estos pensamientos para detectar las distintas actitudes o emociones es una tarea laboriosa y es lo que acontece este trabajo. Es especialmente interesante analizar la red social **Twitter**, que es quizás la plataforma más popular de micro-blogging con más de 500M de *tweets* escritos cada día.

Twitter es especial dado que restringe el número de caracteres máximo por *tweet* (entrada básica de texto). Esto lleva al usuario a expresarse utilizando un mayor número de coloquialismos, expresiones informales, abreviaturas o incluso emoticonos. Ya podemos observar como cambiaría el análisis si lo hiciéramos con otra variante de documentos textuales. Los usuarios utilizan esta plataforma para volcar sus opiniones y sentimientos acerca prácticamente cualquier cosa, esto hace idóneo el análisis de esta red social dado el gran volumen de información que proporciona y el amplio abanico de posibilidades de trabajo en él.

2.4 Reconocimiento automático de emociones

La idea del reconocimiento automático de emociones pasa por la construcción de un sistema que pueda recibir un texto como entrada, y tras procesarlo, el sistema genere como salida los sentimientos o emociones que el texto pretende transmitir. Esta es la idea fundamental que lleva consigo el proceso.

Cada sistema desarrollado es diferente y proporciona un tipo de información específica basada en las especificaciones que tenga, es decir, puede que para un mismo texto, un sistema de reconocimiento de emociones A determine únicamente si existe un sentimiento positivo o negativo; un sistema B determine el grado de presencia de una determinada emoción, como podría ser la tristeza, catalogada en valores numéricos del 1 al 100; y un sistema C asigne el emoticono que mejor representa la emoción del mismo.

A lo largo de los años se han efectuado múltiples trabajos de análisis y reconocimiento de sentimientos y emociones, cada uno de ellos con su enfoque, técnicas y objetivos específicos. Mil posibilidades para la elaboración de un prototipo del que aunque existan múltiples diferencias con cualquier otro sistema al que se quiera comparar, podríamos extraer una estructura común de construcción con una serie de etapas definidas. A continuación se presenta cada una de estas etapas con la finalidad de dar a entender al lector la importancia y el por qué de cada fase en el proceso global.

2.4.1 Selección y extracción de datos

Habiéndonos focalizado ya en el análisis textual, existen muchas variantes de expresión en este terreno, desde textos de opinión y de crítica, cartas, entradas en blogs, posts en redes sociales o incluso libros. Cualquiera de estos tipos lleva consigo una carga emocional, más marcada o menos, interesante de analizar y en cada uno de ellos encontraremos diferencias en la forma de redacción.

Sea cual sea el tipo específico de textos con el que vayamos a trabajar, el análisis a realizar estará estrechamente guiado por la existencia o no de un etiquetado de estos con respecto al sentimiento o emociones que cada texto tiene asociado. Este etiquetado puede reducirse a la asignación de uno de los tres posibles valores:1 si el sentimiento es positivo, 0 si es neutral y -1 si es negativo, o ser complejo como para que existan variables con distintas emociones (tristeza, enfado, felicidad...) y para cada una tener una probabilidad de que el texto la contenga.

Una vez se decide el tipo de textos con el que se va a trabajar, existen varias posibilidades de obtención de los mismos. En general podríamos decir que existen dos formas: obtener unos datos que ya hayan sido extraídos y organizados por otra persona o realizar la extracción por nuestra cuenta. Por su gran aportación a este campo vamos a explicar las dos formas de obtención de datos ayudándonos de la red social **Twitter**.

En muchas ocasiones podemos encontrar que alguien haya elaborado ya un conjunto de datos preparado para trabajar directamente con el. Ejemplo de trabajos que han creado su propio corpus directamente de **Twitter** serían: **Pak A, Paroubek P** (2010) [10] crean un corpus de *tweets* diferenciando tres grupos: textos conteniendo emociones positivas, textos conteniendo emociones negativas y textos de carácter objetivo que no expresan emociones; **Dini L, Bittar A** (2016) [11] que crean dos corpus bajo la asunción de que cada *tweet* tiene una connotación emocional o **Mohammad M, Bravo-Marquez F** (2017) [12] que crean cuatro datasets etiquetados con intensidades para las emociones básicas de *enfado, miedo, alegría y tristeza.*

En el caso contrario, si no queremos trabajar con un conjunto de datos ya elaborado, podemos realizar la tarea de extracción por nuestra propia cuenta. Las técnicas más frecuentes de recolección son *web crawling* o *web scraping* o la llamada a las APIs dispuestas por los servicios como puede ser las de **Twitter** o Telegram.

En general, existen muchos conjuntos de datos ya existentes que se utilizan como ejemplos típicos para hacer estos análisis y generalmente estos conjuntos tienen algún sistema de etiquetado asociado. Contar con alguno de ellos nos facilita esta etapa, sin embargo existen características específicas que nos pueden llevar a necesitar extraer los datos por nuestra cuenta, como sería por ejemplo la necesidad de un tipo de textos y un lenguaje específico. De escoger esta última opción, no tendríamos los datos etiquetados inicialmente.

2.4.2 Preprocesamiento: Filtrado y acondicionamiento

El preprocesamiento de los datos es el proceso de limpieza y preparación del texto obtenido en primera instancia para facilitar el trabajo. Este proceso de transformación del texto en algo con lo que un algoritmo pueda trabajar, es muy complicado. Existen gran cantidad de mecanismos que podemos utilizar con el fin de adaptar nuestro conjunto de datos a las necesidades específicas del experimento, siendo las siguientes técnicas las más comunes:

- **Tokenización**: Proceso básico que disemina las oraciones en palabras. Consiste en separar las palabras en entidades denominadas *tokens* con las que trabajaremos. A la vez se puede realizar una tarea de eliminación de signos de puntuación innecesarios.

- **Eliminación de las palabras vacías (*stop words*)**: Estas palabras como pueden ser "de", "el" o "y" son palabras que aparecen con mayor frecuencia y no aportan un significado semántico específico.

- **Conversión de Mayúsculas en Minúsculas**: Esta técnica se utiliza para conseguir una mayor simplicidad en el texto, sin embargo en algún caso se podría cambiar estar perdiendo el significado como por ejemplo de "CIA" (Agencia Central de Inteligencia) a "cia" que es la reducción de la palabra compañía.

- **Stemming**: Los algoritmos de stemming buscan cortar el final o el principio de una palabra teniendo en cuenta una lista de prefijos y sufijos comunes que se pueden encontrar en una palabra conjugada. En la figura 2.1 se muestran varios ejemplos que lo ilustran. Este corte en las palabras no siempre se realiza de manera correcta y es que este enfoque tiene limitaciones.

- **Lematización**: La diferencia frente al "stemming", es que en la lematización sí se tiene en consideración el análisis morfológico de las palabras como observamos en la figura 2.2. Para ello se necesitan diccionarios detallados para que el algoritmo pueda determinar correctamente el lema de la palabra. Una diferencia importante es que un lema es la forma base de todas las formas conjugadas, mientras que un "stem" no lo es.

Forma	Sufijo	Stem
niñas	-as	niñ
niñez	-ez	niñ

Figura 2.1: *Proceso de Stemming*

Forma	Información morfológica	Lema
niñas	Género femenino, plural del sustantivo niño	niño
niñez	Singular del sustantivo niñez	niñez

Figura 2.2: *Proceso de Lematización*

Todas estas herramientas están destinadas a facilitar el análisis textual. La detección y extracción de las emociones sería una tarea relativamente sencilla si las palabras que representan las emociones fueran explícitas en el texto, pero la mayoría de las veces esto no sucede así. Esto ocurre por la cantidad de limitaciones que nos encontramos en el análisis como por ejemplo la ambigüedad de las palabras, el uso de sarcasmo, expresiones como coloquialismos o en el caso de las redes sociales la aparición de los "emojis" o "emoticonos". Este último caso merece especial detalle dado que admite un análisis específico.

Emojis y emoticonos

Los emoticonos son pequeños dibujos creados con signos ortográficos que a menudo se leían inclinando la cabeza como :) o :(y que evolucionan ya en el siglo XXI hacia los emojis que son pequeñas figuras dibujadas con un valor simbólico. Estos dibujos han tomado una gran relevancia en las redes sociales por su simplicidad y facilidad para transmitir una idea, además tienen el valor de la universalidad, ya que se entienden por personas de diferentes culturas y lenguas. Como hemos comentado, estos caracteres son casi exclusivos de las redes sociales dada su naturaleza relajada. En un análisis de sentimientos en las redes sociales se podría tomar la decisión de eliminar directamente los emoticonos

o emojis para facilitar la tarea, sin embargo ejemplos de trabajos nos demuestran resultados significativos como **Shisa M.O, Ayvaz S** (2017) [13] que incluyendo emojis en su análisis de sentimientos en **Twitter** encuentran unos valores mayores para las puntuaciones de los sentimientos constatando que hay una predisposición a utilizar emojis para sentimientos positivos. También **Satapathy et al** (2017) [14] con este proceso de normalización consigue mejorar su sistema de clasificación de sentimientos por un 4 %.

Normalización

La tarea de **normalización** consiste en transformar palabras u oraciones a su forma "canónica". La tendencia en redes sociales es utilizar abreviaturas o expresiones que simbolizan ideas que no se captarían directamente con un tratamiento como palabras reales. Por este motivo, es importante no saltarnos todos estos términos. En la figura 2.3 se describen ejemplos típicos de transformaciones de palabras que ocurren en las redes sociales.

Desafortunadamente, no hay una manera única de normalizar los textos, ya que la forma suele depender de la tarea especifica que se lleva a cabo. No sería lo mismo normalizar textos médicos que mensajes SMS. Existen diccionarios con los que se puede trabajar que intentan mapear estas expresiones o emoticonos con su correspondiente lingüístico.

Crudo	Normalizado
buenass	buenas
tq	te quiero
cc	con copia
:)	sonrisa
:(	tristeza
fb	facebook

Figura 2.3: *Proceso de Normalización*

En resumen, existen muchas técnicas para realizar el preprocesamiento inicial de los datos con los que vamos a trabajar. Algunas de ellas son imprescindibles de realizar y otras dependen más del contexto y el experimento específico que se lleva a cabo. El objetivo es dar una visión crítica de las distintas formas de abordar un conjunto de datos inicial con el fin de dotar al lector de mayor capacidad para tomar decisiones sobre las técnicas a emplear.

2.4.3 Extracción de características

Como hemos comentado en la sección 2.4.2, la detección de emociones y sentimientos no es una tarea directa ya que las palabras que representan una emoción particular no siempre se encuentran de forma explícita en el texto, y aunque lo estuvieran, esto no significaría necesariamente que ese sentimiento es el que se expresa en el texto. No por encontrarnos

la palabra “felicidad” en un texto, la emoción asociada va a ser “felicidad”, y viceversa. Existen multitud de impedimentos que hacen que sea necesario un gran trabajo en esta fase del sistema para no caer en análisis erróneos.

Una vez hemos adecuado y limpiado los datos en crudo para darle un formato con el que podamos empezar a trabajar, es el momento de analizar las palabras del texto. El mapeo de estos datos textuales a vectores con valores numéricos reales con los que los algoritmos puedan trabajar es lo que denominamos **extracción de características**.

Una de las técnicas más simples para representar numéricamente textos es la **Bolsa de Palabras** (Bag of Words, BOW). La Bolsa de Palabras consiste en generar un listado completo de todas las palabras que aparecen en el corpus lingüístico, de forma que para cada texto en particular, se crea un vector numérico asociado donde para cada palabra del corpus se marca con un 1 si dicha palabra está presente en el texto y con 0 si no lo está. Para realizar esta técnica se define el concepto de corpus lingüístico, que es el conjunto total de textos con el que trabajaremos para desarrollar nuestro dispositivo.

Una vez generados los vectores para cada texto individual, lo más común es la técnica de **Frecuencia de Términos-Frecuencia Inversa de Documentos** (Term Frequency-Inverse Document Frequency, TF-IDF). Esta métrica asocia para cada término un valor numérico que expresa como de relevante es dicha palabra para el documento con respecto al conjunto de todos ellos.

Es una relación entre el número de veces que un término aparece en un documento y el número de documentos en el que aparece. Esta técnica, además, se puede utilizar para detectar las *stopwords*, ya que si definiéramos un umbral máximo para el valor IDF, podríamos considerar que las palabras que lo superen son demasiado comunes, y por tanto, son palabras que no aportan sentimiento o emoción a los textos, como podrían ser preposiciones, artículos o pronombres.

La principal desventaja de la técnica BOW es que ignora el orden y, por tanto, el contexto en el que una palabra aparece en el documento y para el procesamiento de lenguaje natural, mantener este contexto pudiera tener una gran importancia. Para solventar este problema, se puede utilizar otro enfoque denominado **Word Embedding**. Esta técnica, pretende crear una representación de las palabras donde las palabras con un significado similar se asocian a vectores que resultan próximos en un espacio vectorial de elevado número de dimensiones.

De nuevo, el objetivo es disponer de una representación numérica de las palabras que facilite el tratamiento posterior por técnicas de clasificación. Mediante comparaciones de las mismas, podríamos detectar que palabras están relacionadas o muestran conceptos similares. Por ejemplo, se podrían representar las palabras en un sistema de coordenadas donde las palabras relacionadas están situadas cerca unas de otras. Una de las técnicas más populares es **Word2Vec**.

Word2Vec está desarrollado por Google [Word2Vec] y utiliza una red neuronal superficial que dado un corpus, analiza las palabras y trata de usar cada una para predecir

que palabras serán vecinas, es decir, que palabras están asociadas a ella o son similares. Existen dos variantes de arquitectura distintas: continuous bag-of-words (CBOW) y continuous skip-gram. Dependiendo de la tarea que se quiere realizar, se tiene que tener en cuenta la arquitectura a utilizar, así como los parámetros que se deben tunear. Los parámetros más importantes son el tamaño de los vectores de las palabras y la ventana, que es la distancia máxima entre la palabra actual y la palabra predicha dentro de un mismo *tweet*. Según las notas del autor [Google Code], CBOW produce resultados más rápidos pero skip-gram se adapta mejor a las palabras no frecuentes.

Más adelante en el documento, se amplia la explicación de esta técnica aplicándola en el experimento que se está desarrollando, de esta forma se entienden los conceptos de mejor manera al ilustrarlos en un ejemplo específico.

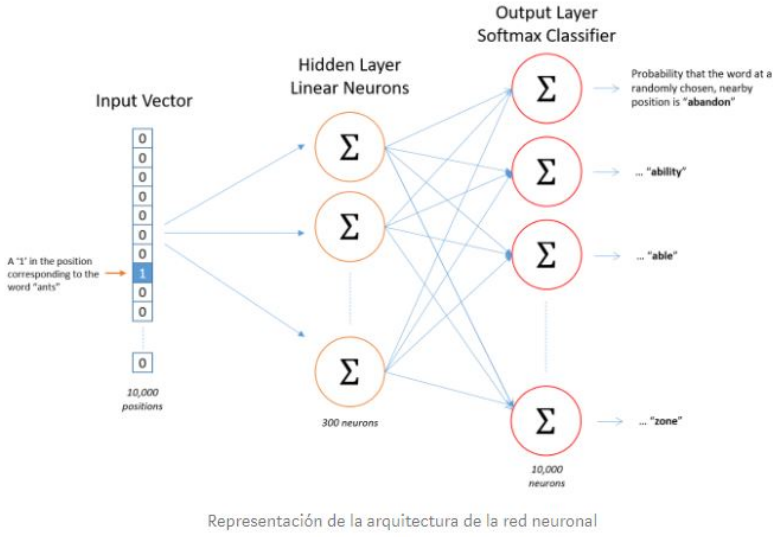


Figura 2.4: *Red Neuronal word2vec*

En resumen, Words Embedding codifica cada palabra en un vector que captura la relación y similitud entre palabras dentro del corpus lingüístico utilizado. Esto ayuda a que variaciones de la palabra en la puntuación, ortografía, etc., automáticamente quedan aprendidas, y por tanto, obtengamos un mayor entendimiento del texto.

Existen muchas técnicas o algoritmos que se pueden aplican en esta fase dependiendo del objetivo que se persigue [15]. En este apartado hemos explicado algunas de las más comunes, pero la finalidad de todas es adaptar la información para que pueda servir como entrada para el modelo o modelos que vamos a crear en nuestro sistema.

2.4.4 Modelado

En la tarea de modelado, se pueden utilizar diversas técnicas de Machine Learning tanto supervisadas como no supervisadas. En ellas se diseña un modelo que entrena un clasificador con una parte del conjunto de datos y después se prueba dicho clasificador con el

resto de los datos.

Para análisis supervisado, se utiliza un conjunto de datos etiquetado con las emociones tanto en la tarea de entrenamiento como en la de prueba. Los ejemplos más comunes podrían ser Naive Bayes, SVM o Árboles de decisión.

En análisis no supervisado, no tenemos el conjunto de datos etiquetado con las clases o emociones a las que pertenece cada texto. La estrategia que se aplica es la de describir los datos intentando encontrar algún tipo de estructura intrínseca y comúnmente oculta en los mismos. De esta forma, se entrena un clasificador que construimos y que se utiliza después para etiquetar los datos de prueba. Los ejemplos más utilizados son las redes neuronales y los métodos de clustering (k-medias, k-vecinos etc).

Ya podemos observar que sería interesante combinar las distintas técnicas de alguna forma para obtener lo mejor de ambas. En este contexto surgen los métodos híbridos que combinan distintas técnicas para alcanzar el máximo nivel de precisión posible. Estas técnicas pueden no ser únicamente de ML, si no que pueden ser basadas en reglas, basadas en palabras clave, basadas en léxicos, etc.

Basándonos en los trabajos y artículos analizados durante la fase de comprensión del problema y estudio de las formas de resolución aplicadas, se ha observado como las técnicas de análisis supervisado son las que más se utilizan. Destaca el uso de Regresión Logística, Support Vector Machine o Naive Bayes. Dada la ausencia de datos etiquetados de calidad en nuestro problema concreto (en castellano), se trabaja con datos extraídos directamente de **Twitter**, es decir, datos que no están etiquetados.

En vez de trabajar con técnicas de análisis no supervisado, se crea un sistema de etiquetado basado en léxicos que "simularía" un etiquetado manual idóneo con el que poder trabajar indistintamente técnicas de análisis supervisado y no supervisado.

2.4.5 Diseño del experimento y Técnicas de Validación

Una vez elegido el modelo, se deberá diseñar cuidadosamente el experimento para entrenar y validar dicho modelo a partir de los datos disponibles. Dependiendo de la cantidad y calidad de los datos de los que se disponga, se pueden utilizar varias técnicas de validación. Si contáramos con una máquina con una mayor potencia computacional, no tendríamos problema con respecto a la cantidad necesaria, ya que se podría extraer directamente de **Twitter** el tamaño deseado de conjunto de datos, sin embargo, este no es el caso. En un experimento idóneo tendríamos suficientes datos como para separar tres conjuntos diferenciados: T el conjunto de entrenamiento, V el conjunto de validación y P el conjunto de prueba. Cada uno sería lo suficientemente grande y estaría seleccionado de forma aleatoria.

Con T y V se construye el clasificador (modelo), y con P evaluaremos el funcionamiento del mismo por ejemplo mediante la tasa de error. Como técnicas clásicas de diseño

del experimento podemos tener: **Holdout estratificado**, que consiste en dividir de forma aleatoria los datos en 2/3 para entrenamiento y 1/3 para prueba. Además para asegurar que las muestras sean representativas, se estratifica para que cada clase esté representada con aproximadamente las mismas proporciones en los dos subconjuntos. En nuestro caso esta estratificación la podríamos hacer si contáramos con un conjunto de datos etiquetado. Este procedimiento se podría repetir varias veces para reducir la variabilidad que representa la partición (**Holdout Repetido**).

Aun con la repetición, este procedimiento no es óptimo ya que los diferentes conjuntos de prueba se solapan. Para evitar este problema podemos efectuar una **Validación cruzada (XV)**. Consiste en repartir los datos en k subconjuntos del mismo tamaño. Cada subconjunto será utilizado como prueba y el resto para entrenamiento. De nuevo podemos tanto estratificar los subconjuntos como repetir varias veces este proceso. Con XV conseguimos un diseño muy poco variable pero a su vez mucho más costoso. Dependiendo del volumen de datos con el que trabajemos y la complejidad del modelo, será necesario tomar una decisión que consiga un compromiso entre funcionamiento y tiempo de ejecución.

Con respecto a la forma de evaluarla calidad del experimento, la forma clásica para evaluar el funcionamiento de un modelo cuya tarea es la clasificación, es la tasa de error sobre el conjunto de prueba. Esta se calcula como el porcentaje de predicciones correctas sobre el total de predicciones realizadas. Esta medida de validación es buena, pero realiza la suposición de que los costes de los errores son los mismos y la distribuciones de clases están equilibradas, y dependiendo de la tarea específica que se lleve a cabo, esto podría ser fatal. Supongamos que estamos realizando una tarea de predicción de *tweets* pertenecientes a terroristas. Es obvio pensar que no es igual de importante predecir de forma errónea un *tweet* como peligroso, como no predecir un *tweet* peligroso.

Podríamos entonces utilizar otras métricas de validación de modelos más elaboradas y específicas. De las más comunes utilizadas son: *precission*, *recall* y *F1-Measure* (ver ecuación 2.4.1). Ambas tres están basados en la elaboración de la matriz de confusión que etiqueta las predicciones como *verdadero positivo*, *verdadero negativo*, *falso positivo* y *falso negativo*.

La métrica *precission* mide la calidad del modelo en la tarea de clasificación y *recall* da información sobre la cantidad que el modelo es capaz de identificar. La métrica *F1* se utiliza para combinar ambas medidas en un solo valor, por tanto es práctica al hacer más fácil la comparación del rendimiento combinado. El valor F1 asume que la importancia de de la calidad y la cantidad es la misma, pero esto no tiene por qué ser así. De esta manera surge la métrica F2 que introduce un factor que regula la importancia. Encontramos ejemplos de uso de estas métricas en los trabajos [11], [16], [17] o [18]

$$precision = \frac{TP}{TP + FN}; \quad recall = \frac{TP}{TP + FN}; \quad F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.4.1}$$

La validación, por tanto, es una tarea que proporciona criterios objetivos para com-

probar el funcionamiento de nuestros modelos. Esta tarea da pie a posibles refinamientos que mejoren la calidad del experimento.

Capítulo 3. Desarrollo del experimento

3.1 Introducción

Este capítulo pretende proporcionar una solución a la problemática planteada del análisis de emociones y sentimientos. Se exponen los pasos conceptuales específicos seguidos junto con las herramientas utilizadas y los razonamientos que nos han llevado a elegir o desestimar las técnicas.

De nuevo haremos énfasis en el objetivo que se desea conseguir: la creación de un sistema experimental que pueda clasificar un *tweet* como positivo o negativo en base al sentimiento que transmite. Para ello explicaremos con detalle cada una de las fases por las que hemos pasado hasta llegar a un prototipo de una calidad lo suficientemente aceptable como para poder ser utilizado.

Además se realiza una doble tarea de detallar los programas y archivos necesarios para la realización del sistema, con las consiguientes explicaciones sobre su orden de ejecución y funcionamiento. La parte más técnica de puesta en marcha se puede ver en el apéndice C.

3.2 Arquitectura del sistema experimental

La figura 3.1 muestra la arquitectura del proceso de elaboración del sistema de clasificación de *tweets*, cuyas fases se detallan y explican en las siguientes secciones.

3.3 Selección de datos

Si bien podemos encontrar multitud de conjuntos de datos ya elaborados que nos permitan comenzar a elaborar nuestro sistema, se toma la decisión de realizar la extracción de los

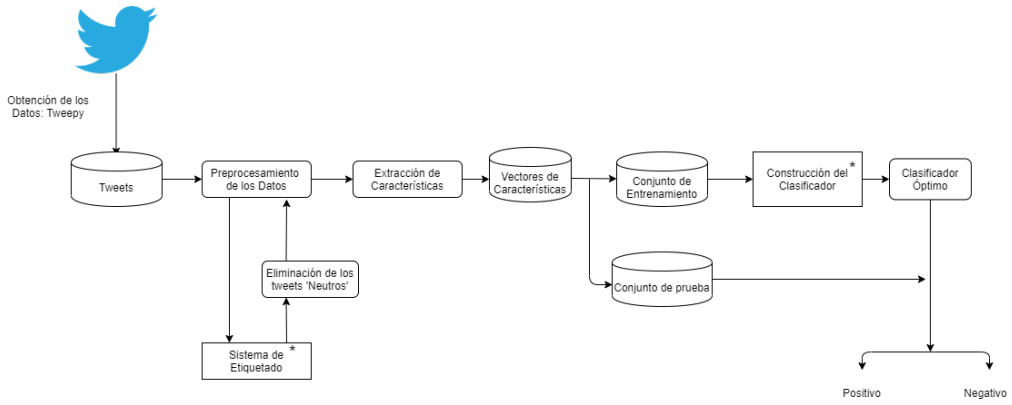


Figura 3.1: *Arquitectura del Sistema de clasificación de tweets*

mismos de forma propia por los siguientes motivos:

- Existe un porcentaje muy pequeño de conjuntos de datos formados únicamente por *tweets* escritos en castellano. En la fase de búsqueda se encuentran la mayor parte de conjuntos en inglés, que es el lenguaje principal en el que los análisis de este campo se suelen realizar.

- Aún encontrando algún conjunto de datos en castellano, no se encuentran etiquetados con algún tipo de marcador que indique el sentimiento o emoción asociada.

- Los conjuntos de datos que nos pudiéramos encontrar, también han sido extraídos por otra persona de alguna manera, como por ejemplo siguiendo una temática específica o recogiendo todos los *tweets* publicados por un conjunto de personas previamente seleccionado. Este rango de especificidad de los datos es una variable que podríamos necesitar tener controlada por si queremos apuntar nuestro análisis en una dirección específica.

- Los conjuntos de datos, con ciertas excepciones, cuentan con un grado de antigüedad. Esto no es necesariamente un problema, sin embargo dada la naturaleza cambiante de las redes sociales, ya que constantemente se generan nuevas formas de expresarse siguiendo las tendencias, nos convendría contar con un conjunto de datos actualizado. Además sabiendo que nos vamos a centrar en la red social **Twitter**, en Septiembre de 2017 ya se empezó a testear el aumento de caracteres límite de 140 a 280, por lo que puede ser que solo nos valgan conjuntos posteriores a esa fecha.

- Para completar el último punto, cuando se extraen manualmente los datos con los que se van a trabajar, ganas el poder decidir que es lo que quiero y que es lo que no. Las Redes Sociales son un reflejo de la realidad, y por tanto las tendencias se actualizan constantemente.

Una vez hemos tomado la decisión de extraer los datos por nuestra cuenta y analizando las posibilidades de como realizar esta tarea, se decide utilizar la API de **Twitter**. Las API son la forma en la que los programas informáticas realizan entre si solicitudes de

información. La API de **Twitter** ofrece un amplio acceso a los datos públicos de la red social.

El primer paso que tenemos que realizar para acceder a la API es registrar una aplicación, que en este caso va destinada a la extracción de un conjunto de *tweets* del que se van a basar nuestros análisis. Este paso se puede realizar desde su página web *developer.twitter.com/en/docs* y tras rellenar la información necesaria te dará unas claves de autenticación con las que poder acceder a los servicios de las API. Estas claves son: *consumer_key*, *consumer_secret*, *access_token* y *access_token_secret*.

Una vez hemos completado esta parte, la extracción de los datos la haremos mediante Tweepy, que es una librería de Python para acceder a la API de **Twitter**. El programa twitter_scrapper.py será el encargado de realizar la tarea de extracción de los datos. Lo primero que tenemos que hacer es configurar nuestra cuenta para poder acceder a la API correctamente, para ello:

1. Autenticarnos mediante *OAuthHandler*, proporcionando la clave y clave secreta de consumidor (*consumer_key* y *consumer_secret*).
2. Proporcionar nuestro token de acceso para poder trabajar con la API mediante el método set_access_token, que recibirá nuestro token de acceso y su clave (*access_token* y *access_token_secret*).
3. Una vez tenemos la configuración hecha bajo el nombre de *auth*, crearemos una instancia de la clase "API" a la que le pasaremos la configuración como parámetro y con la que comenzaremos a trabajar.

Para comenzar la extracción de los *tweets* tendremos que crear un Cursor que itera a través de timelines, listas de usuarios, mensajes... Este Cursor contiene muchos parámetros a configurar. Como parámetro principal se le indica que lo que va a iterar es una colección de *tweets* que coincidan con una búsqueda específica.

La situación actual extraordinaria que estamos viviendo a Marzo, 2020 me lleva a que los datos extraídos estén relacionados con el Covid-19. Esto se consigue pasando como argumento el método api.search donde le indicamos que queremos *tweets* que contengan la palabra Covid sin contar los retweets (q="Covid -filter:retweets), que estén escritos en español (lang="es") y que puedan tener hasta 280 caractéres correspondientes a la extensión de los caracteres límite previamente comentada (tweet_mode="extended").

El número de datos a extraer escogido es 15000, ya que es una cantidad aceptable para trabajar comparando con los trabajos analizados en la fase de investigación. Es una cantidad elevada pero que no requiere demasiada potencia de cómputo y por tanto permitirá trabajar desde el ordenador personal sin necesidad de acceder a algún servicio de la UVa.

Nuestro iterador recoge objetos "Tweet" que contienen muchos atributos como pueden ser el texto, su identificador, el identificador del usuario, la fecha en la que se escribió,

los hashtags incluidos etc. Nosotros queremos recoger simplemente el texto escrito, es decir, el "Tweet" propiamente dicho. Para almacenar esta información se construye un DataFrame ayudados por una de las librerías más comunes en Python, **Pandas**. A lo largo del trabajo se utilizará en muchas fases dada su facilidad de uso, flexibilidad y rapidez en las tareas de análisis de datos.

Uno de los problemas iniciales que nos encontramos es que **Twitter** limita las peticiones por ventanas. Cada ventana consta de 15 minutos y en cada ventana se pueden realizar 15 peticiones, por lo que para poder conseguir solucionar este problema podemos configurar un parámetro de nuestra instancia de la clase API que es wait_on_rate_limit, que activándolo, permite esperar automáticamente en cada ventana para poder hacer las peticiones.

Gracias a este parámetro podemos automatizar la extracción de los datos que nos ha llevado entre dos y tres horas en completar la tarea. Es por esto que almacenaremos los conjuntos de datos en un archivo externo que podremos cargar y descargar en cualquier momento y el cual a partir de ahora nos referiremos como **tweet15k.csv**

Con la ayuda de el programa hemos realizado la extracción y almacenado de los datos. Como bien hemos descrito, existen muchas posibilidades de configuración de los parámetros que nos ayudarían a poder extraer datos apuntando en otras direcciones para futuras ampliaciones del análisis.

3.4 Preprocesamiento de los datos

Una vez extraído el conjunto de datos con el que vamos a trabajar, podemos ver antes de comenzar a preprocesarlo la pinta que tiene.

	Tweet
160	@Xaviercitt0 sisi es aplazado por covid y en l...
161	@eldesconcierto cuidado¡¡el covid-19 acelera l...
162	Me encanta lo luchadores q son mis niños con c...
163	#Covid_19 #endthelockdownnow #id2020 #C...
164	@eldiarioes Y que? Si no cuesta dinero al erar...
165	A saber que llevará la vacuna contra el covid ...
166	Dinamarca declara controlada la COVID-19 y cre...

Figura 3.2: *Conjunto de datos inicial*

Observamos elementos clásicos de los mensajes de las redes sociales, y más específicamente de **Twitter**, que merecen un tratamiento especial, como menciones, hashtags,

palabras mal escritas o emojis. No debemos perder de vista que esta etapa está destinada a conseguir que los datos estén correctamente preparados como para que los algoritmos de extracción de características puedan trabajar con ellos.

Esta fase del proceso está cubierta por el programa preprocessing_data.py que contiene una función preprocessing_tweet_es() que recibe una cadena de caracteres, es decir un *tweet* individual, y aplica todas las técnicas que se describen a continuación en orden.

Para esta etapa es fundamental el uso de las **expresiones regulares**, que son las que nos van a ayudar a identificar patrones que deseamos eliminar o modificar. Para ello utilizaremos el módulo **re** de Python que está específicamente destinado al trabajo con expresiones regulares. La función re.sub() recibe dos parámetros principalmente: el primero es el patrón que vamos a buscar y el segundo es por lo que la vamos a sustituir. En caso de dejar este segundo parámetro vacío, estaríamos simplemente eliminando aquellas cadenas que coincidan con el patrón indicado. Los pasos seguidos son por tanto:

- **Eliminación de URLs:**

  ```
  text=re.sub('((www\.[^\s]+)|(https?://[^\s]+))', '', text)
  ```

  Es común que un *tweet* lleve consigo un enlace a una página externa. Esto no nos sirve en nuestro análisis, y por tanto, lo eliminaremos. Son enlaces válidos aquellos que comienzan por www., https:// o http:// y terminan con el primer espacio que se encuentre.

- **Eliminación de saltos de línea:**

  ```
  text=re.sub('\n','',text)
  ```

  Los saltos de línea se codifican como \n y podemos eliminarlos ya que podrían causar problemas.

- **Eliminación de menciones:**

  ```
  text=re.sub('@\s','@',text)
  text=re.sub('(@[^\s]+)|(w/)', '', text)
  ```

  Las menciones están formadas por una @ seguida de el nombre de un usuario, de esta forma, al incluirlo en un *tweet*, el usuario al que se ha mencionado recibe una notificación. Se tiene en cuenta también posibles problemas de *tweets* que contienen menciones mal escritas, como las que tienen un espacio entre @ y el nombre del usuario o aquellas que en vez de utilizar @, utilizan w/, que es la forma abreviada de poner with.

- **Eliminación de Hashtags**

  Los hashtags son una cadena de caracteres formada por una almohadilla (#) precedida de caracteres que no sean un espacio. En las redes sociales se utilizan para representar temas. De esta forma, los usuarios pueden indicar que están hablando sobre algo incluyendo un hashtag relacionado.

  Una de las posibilidades planteadas para la extracción, era haber hecho la búsqueda por hashtags, que se podría utilizar asociado a la lista de tendencias de temas en **Twitter**. Esto se podría plantear como una posible ampliación del trabajo.

  Un hashtag si que incluye información valiosa, por lo que únicamente se elimina es la almohadilla-

  ```
  text = re.sub(r'#([^\s]+)', r'\1', text)
  ```

- **Eliminación de los signos de puntuación:**

  En el módulo **string** podemos encontrar una lista de los signos de puntuación existentes a los que les añadiremos ¿ y ¡ ya que la lista está construida para el idioma inglés.

  Es muy importante realizar esta fase después de la eliminación de URLS, menciones y hashtags, ya que las tres contienen signos de puntuación que son necesarios para su identificación. Una vez hemos hecho estas técnicas, descomponemos la cadena con la que estamos trabajando en caracteres de longitud 1, eliminamos los signos de puntuación, y volvemos a unir los caracteres restantes.

  ```
  signos_puntuacion=string.punctuation+'¿¡'
  text=[char for char in text if char not in signos_puntuacion]
  text=''.join(text)
  ```

- **Conversión de mayúsculas a minúsculas**

  Este paso es definitivo para poder tratar como iguales a dos palabras que solo se distingan en tener caracteres iguales en mayúsculas o minúsculas.

  ```
  text = text.lower()
  ```

Finalmente, antes de dar por concluido el preprocesamiento de cada *tweet*, tenemos que recordar que todos los datos que hemos extraído han sido escogidos por presentar la palabra "covid" en su interior, por tanto la eliminaremos junto con sus posibles variantes ya que no añaden información alguna.

```
text=re.sub('covid19','',text)
text=re.sub('covid 19','',text)
text=re.sub('covid','',text)
text=re.sub('coronavirus','',text)
```

El proceso en general se aplica recorriendo el conjunto de datos y aplicando *tweet* a *tweet* todas las técnicas. El orden está definido por razones evidentes como la que hemos explicado refiriéndonos a los signos de puntuación.

Mención especial aparte requiere la técnica de la "lematización", que es el proceso lingüístico que dada una forma flexionada (es decir conjugada, en plural/singular, en femenino/masculino, etc.) pretende hallar el lema correspondiente. Es una fase muy importante en el procesamiento del lenguaje natural ya que necesitamos poder detectar que "aborreciérais" y "aborrezco" o "asqueroso" y "asquerosas" transmiten la misma idea. Además, en general, los léxicos disponibles para realizar tareas de procesamiento del lenguaje natural contienen los términos lematizados, como es en nuestro caso que se detallará a continuación.

Vamos a utilizar la libería **spacy**, que es una biblioteca para procesamiento avanzado de lenguaje natural. Incluye modelos preentrenados para predecir etiquetas PoS (Part of Speech), dependencias sintácticas o entidades propias. Esta librería proporciona modelos para muchos lenguajes, de entre ellos, el castellano. El modelo que vamos a utilizar se denomina *es_core_news_sm*. Este modelo es una Red Neuronal Convolucional multitarea entrenada con los corpus **AnCora**, que contiene 500k palabras procedentes de textos periodísticos, y **WikiNER**, que utiliza la Wikipedia y DBpedia para detectar entidades propias a nivel de palabra (Londres, Andrés Martínez, Imperio Romano...). En definitiva, utilizaremos esta biblioteca para generar el lema de cada término individual resultante del preprocesamiento inicial de los datos.

Términos	**Completo**
Tweet Original	En @elconfidencial: Islas contra Península: la guerra del verano covid entre los principales destinos turísticos https://t.co/4XUQzBmsfX,
Tweet Preprocesado	["islas","península","guerra", "verano","principales","destinos","turísticos"]
Tweet Lematizado	["isla","península","guerra", "verano","principal","destino","turístico"]

Cuadro 3.1: *Ejemplo de preprocesamiento y lematización.*

En la tabla 3.2, se muestran los tiempos de ejecución para la fase del preprocesado aplicada al conjunto de datos de 15000 *tweets* diferenciando entre aplicar lematización o no.

Fase	**Sin lematización**	**Con lematización**
Preprocesamiento	1.19 seg	172.91 seg

Cuadro 3.2: *Tiempo de ejecución de la fase de preprocesado.*

Si no aplicamos el proceso de lematización, el tiempo de preprocesado es prácticamente despreciable, sin embargo aplicando lematización obtenemos un tiempos de ejecución para la fase de preprocesado de 172,91 segundos. Es una gran diferencia de tiempo ya que causa una demora notable cuando se aplique el proceso global. Con los experimentos que se realicen a continuación, se determinará si merece la pena aplicar esta técnica.

3.5 Etiquetado de los *tweets*

Si bien ya hemos preprocesado los datos para darle un mejor formato, no los tenemos etiquetados con respecto al sentimiento que pretenden transmitir. Esto es un problema, ya que para la posterior fase en la que entrenaremos los modelos de clasificación, necesitamos saber a que clase pertenece cada *tweet*, y por tanto, es necesaria una tarea de etiquetado de los datos. Explorando las técnicas existentes, nos encontraremos múltiples aplicaciones o paquetes que podrían servirnos, pero además del etiquetado, hacen todo el proceso de análisis de un texto (extracción de características y modelado) con un par de órdenes o clicks y esa no es la finalidad del trabajo. Construiremos un sistema de etiquetado con el que clasificar nuestros datos de una forma lo suficientemente precisa como para asumir que esas etiquetas sean la realidad de los *tweets*.

Se proponen varias formas de etiquetado de los *tweets*. En líneas generales, se intenta crear un sistema de etiquetado basándonos en el uso de léxicos separando en términos y en emojis. Como ya hemos comentado anteriormente, los mensajes en redes sociales, y más especialmente en el caso de **Twitter**, van cargadas de coloquialismos, expresiones mal escritas o emojis. Especial atención presentaremos en estos últimos, etiquetando así los datos en base tanto a los términos utilizados como a los emojis empleados. Se busca el compromiso mediante alguna función matemática que tomando la información de unos y otros, produzca como salida, la etiqueta asociada a cada sentimiento.

Finalmente se muestra una relación del esfuerzo de ejecución que ha llevado esta fase con el objetivo de resaltar la importancia de la misma.

Listado de léxicos

Los léxicos con los que trabajaremos son los siguientes:

- **Sentiment Polarity Lexicons (SPL)** [19]: Creación de léxicos de alta calidad para 136 lenguajes mediante la construcción de una red de grafos de conocimiento inmensa. El trabajo queda validado comprobando el funcionamiento de sus léxicos en un análisis de 2000 figuras históricas en artículos de Wikipedia en 30 lenguajes diferentes.

  Para cada idioma se crean dos léxico de términos, uno con sentimiento positivo y otro con sentimiento negativo. Ciertamente, para muchos de los lenguajes cubiertos, se generan léxicos realmente pequeños, pero afortunadamente, no es el caso de los asociados al castellano. Según un análisis comparativo con el resto de los léxicos, el castellano es el quinto lenguaje con más palabras con sentimiento asociado con **4275**, presentando uno de los ratios de positividad más bajos siendo este de **0.36**.

  En la práctica contamos con dos ficheros separados a los que nos referiremos en nuestro proyecto como *./lexicos/spl/negative_words_es* y *./lexicos/spl/positive_words_es*

Terminos	Tamaño
Positivos	1555
Negativos	2720
Total	4275

Cuadro 3.3: *Distribución del léxico Sentiment Polarity Lexicons (es)*

- **Emoji Sentiment Ranking v1.0** [20]: Léxico de emojis elaborado a partir del etiquetado por parte de 83 personas de 1.6 millones de *tweets* como positivo, neutral o negativo en 14 lenguajes europeos. Un 4 % de los *tweets* contenían emojis y finalmente, se genera el primer léxico de sentimientos de emojis compuesto por los **751** emojis más frecuentes utilizados. Además sus creadores constatan que no se observan diferencias significativas entre los rankings de los emojis entre los 13 lenguajes, por tanto, se propone este ranking como un recurso para el análisis de sentimientos independiente del lenguaje europeo.

  Para cada emoji obtenemos información variada de donde nos quedaremos: *unicode codepoint*, que es el código identificativo del emoji, *occurences*, que es el número de apariciones total del mismo, y *pos*, *neg* y *neut*, que contienen el número de veces que han aparecido en *tweets* anotados como positivos, negativos o neutros.

Char	Image [twemoji]	Unicode codepoint	Occurrences [5...max]	Position [0...1]	Neg [0...1]	Neut [0...1]	Pos [0...1]	Sentiment score [-1...+1]	Sentiment bar (c.i. 95%)
😂		0x1f602	14622	0.805	0.247	0.285	0.468	0.221	
❤		0x2764	8050	0.747	0.044	0.166	0.790	0.746	
♥		0x2665	7144	0.754	0.035	0.272	0.693	0.657	
😍		0x1f60d	6359	0.765	0.052	0.219	0.729	0.678	

Figura 3.3: *Versión Web de Emoji Sentiment Ranking*

En la práctica contamos con un ficheros al que nos referiremos en nuestro proyecto como *./lexicos/emoji/Emoji Sentiment Ranking 1.0*.

- **Léxico ML-SentiCon** [21]: Conjunto de léxicos de polaridades semánticas a nivel de lemas para inglés, castellano, catalán, gallego y euskera. Los léxicos, han sido generados a partir de una mejora del método utilizado para generar SentiWordNet. Este método está basado en una estructura de capas. Cada léxico está formado por ocho capas ordenadas de la primera a la octava, de manera que las capas posteriores contienen todos los lemas de los anteriores y añaden nuevos. Conforme se va bajando de capa, se van rebajando las restricciones para que el número de lemas que las cumplen vaya aumentando capa tras capa. Los requisitos tienen que ver con la fiabilidad de que sean indicadores de positividad y negatividad.

  El léxico está evaluado manualmente. Los cuatro primeros niveles han sido revisados etiquetando cada entrada como correcta o incorrecta. Para los cuatro niveles

inferiores se ha revisado una muestra aleatoria representativa de cada uno de los niveles de forma que se garantice un error muestral menor al 5 % en la estimación de proporción de elementos correctos. Los resultados de las pruebas que se efectuaron son los siguientes

Capa	Prec.	Tam.
1	97.73 %	353
2	97.20 %	642
3	94.95 %	891
4	93.06 %	1138
5	91.75 %	1779
6	86.09 %	2849
7	77.69 %	6625
8	61.29 %	11918

Cuadro 3.4: *Estimación del porcentaje de lemas con la polaridad correcta (Prec.) y número de lemas totales (Tam.) de cada una de las capas del lexicon ML-SentiCon*

En la versión actual disponible, varían un poco los números, ya que no son 11918 si no **11542** los elementos incluidos. De ellos **5568** tienen polaridad positiva y **5974** negativa, por lo que observamos una proporción casi igual de un **49.3 %** de tasa positivo/negativo. Como nosotros vamos a trabajar a nivel de palabra y no de palabra o expresión, tendremos que reducir el léxico a nuestras necesidades. Además existen lemas repetidos, de los que nos quedaremos con el primero que aparece en la lista, ya que corresponde a las capas más elevadas done las restricciones son más duras. Tras realizar estas dos tareas, nos queda un léxico final con **8565** lemas, de ellos **4356** son positivos y **4404**, obteniendo así una tasa de 49.7 % de positivo/negativo.

Terminos	Completo	Reducido
Positivos	5568	4356
Negativos	5974	4404
Proporción P/N	49.3 %	49.7 %
Total	11542	8760

Cuadro 3.5: *Distribución de clases en el léxico ML-SentiCon adaptado.*

En la práctica contamos con un ficheros al que nos referiremos en nuestro proyecto como *./lexicos/mlsenticon/senticon-limpio*.

- **iSOL** [23]: Léxico generado a partir de una traducción automática del inglés al español del léxico creado por Bing Liu [24], generando así, el recurso SOL (*Spanish Opinion Lexicon*). Se realiza una correción manual de SOL dando lugar a **iSOL**. La inflexión morfológica española puede generar hasta cuatro posibles palabras de un adjetivo inglés, dos para el género (masculino o femenino) y dos para el número (singular o plural). También se siguió la filosofía de Bing Liu y se introducen en la

lista palabras mal escritas o no pertenecientes al Diccionario de la Real Academia Española (DRAE).

Finalmente iSOL está compuesto inicialmente por **8135** palabras, de las cuales **2509** son positivas y **5626**. Realizando una labor de revisión de los datos, nos encontramos cuatro casos de palabras repetidas: "partidarios" y "simplificada" aparecen con signos opuestos por lo que no las vamos a considerar en nuestro léxico particular, y "daños" y "engaños" aparecen repetidas con el mismo signo, por lo que se eliminan las repetidas simplemente. De esta forma nuestro léxico asociado resultante consta de **8129** palabras de las cuales **2507** son positivas y **5622** obteniendo así una proporción positivo/negativo del 30.8 %.

Terminos	**Completo**	**Reducido**
Positivos	2509	2507
Negativos	5626	5622
Proporción P/N	30.8 %	30.84 %
Total	8135	8129

Cuadro 3.6: *Distribución de clases en el léxico iSol adaptado.*

En la práctica contamos con un ficheros al que nos referiremos en nuestro proyecto como *./lexicos/crisol/isol*.

- **CRiSOL** [23]: Léxico generado combinando información de iSOL y SentiWordNet. Se añaden a iSOL las puntuaciones de polaridad de SentiWordNet. iSOL está formado por formas en general, ya que tanto lemas como algunas de sus derivaciones lo constituyen. Sin embargo, SentiWordNet, es un recurso compuesto por conceptos en inglés, por lo que es necesario un recurso auxiliar para enlazar ambos, el MCR.

  MCR (*Multilingual Central Repository*), es un recurso que se puede usar en procesos semánticos que necesitan una cantidad grande de conocimiento lingüístico. MCR integra versiones diversas de WordNet para diferentes lenguas (inglés, español, vasco, catalán y gallego). Se construyen *synsets* que se enlazan mediante un índice entre lenguas (*InterLingual Index-ILI*. Este índice es la clave para conectar lenguajes, haciendo posible ir de una palabra de un idioma a otras similares a esta en otros idiomas.

  El proceso de creación de CRiSOL comienza obteniendo los lemas de las palabras de iSOL y utilizando MCR encontrar el *ILI* asociado mediante la heurística de primera búsqueda. Una vez hecho esto, se recupera de SentiWordNet los valores de polaridad asociados al *ILI*.

  En la Figura 3.4 disponible en el artículo, observamos un diagrama que explica el proceso de generación de CRiSOL. Finalmente el recurso está formado por las **8135** palabras existentes en iSOL, de las cuales **4434** tienen categoría morfológica y puntuaciones de polaridad.

  Igual que con el léxico iSOL, nos encontramos con una palabra repetida con distinta polaridad, y dos repetidas con el mismo signo. Procederemos igual: eliminamos los dos casos de la palabra repetida con distinta polaridad ("partidarios") ya que

presenta los valores (0,1,0) que se corresponden con "neutro" en ambas palabras repetidas; y uno de los casos de las palabras repetidas con misma polaridad ("daños" y "engaños"). De esta forma nos quedamos con un léxico final formado por **4430** palabras.

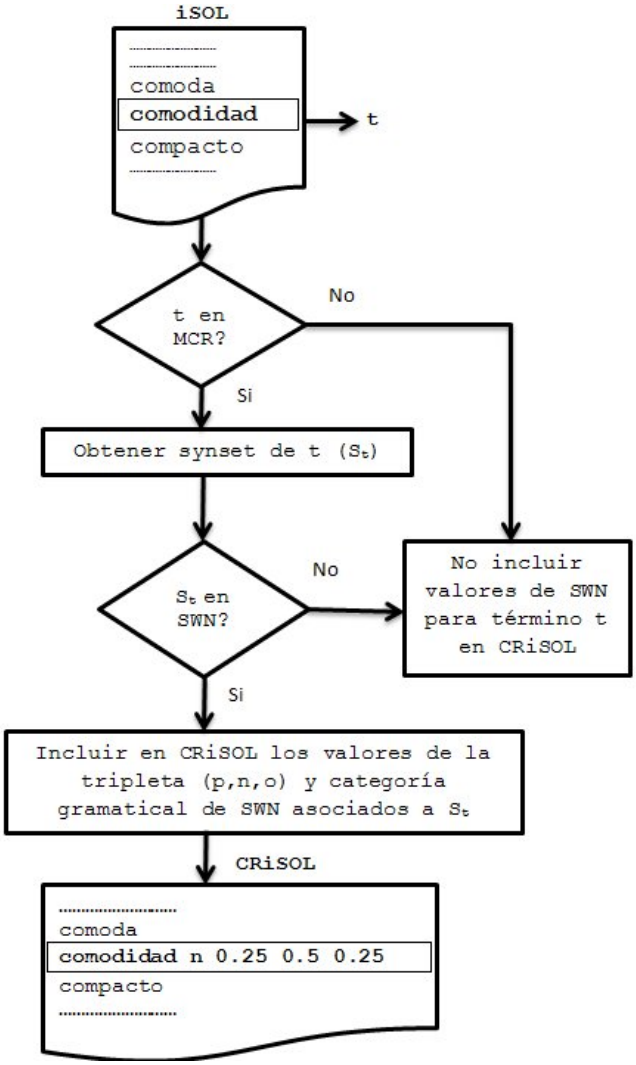


Figura 3.4: *Construcción del léxico CRiSOL*

Términos	Completo	Reducido
Total	4434	4430

Cuadro 3.7: *Número de formas en el léxico CRiSol adaptado.*

En la práctica contamos con un ficheros al que nos referiremos en nuestro proyecto como *./lexicos/crisol/crisol*.

Estos son los 5 léxicos con los que vamos a trabajar: cuatro asociados a términos en español (castellano), y uno asociado a emojis (multilenguaje). Cada léxico ha sido creado utilizando técnicas diferentes y con objetivos diferentes, por tanto, una buena forma de

proceder sería intentar combinar los resultados de utilizar cada uno por separado para así compensar los posibles fallos de un léxico con los aciertos del resto.

Sistema de etiquetado

El procedimiento que se ha seguido, es la construcción de 4 sistemas de etiquetado separados, cada uno utilizando uno de los 4 léxicos correspondientes a términos y el correspondiente a emojis. La decisión de utilizar el mismo diccionario de emojis, viene de comprobar que está ampliamente aceptado su funcionamiento y su uso por trabajos en este mismo campo. En la fase anterior habíamos construido para cada *tweet*, un vector de elementos individuales que podían ser o términos individuales o emojis con el que trabajaremos ahora. Con el fin de poder comprobar la efectividad de incluir el proceso de lematización en el sistema, se realizará un proceso de etiquetado completo con los *tweets* sin lematizar y otro proceso completo con los *tweets* lematizados.

Se describen a continuación el funcionamiento de los cuatro subsistemas por separado:

- **Subsistema 1: Sentiment Polarity Lexicons (SPL) + Emoji S.Ranking**.
  El léxico SPL, está compuesto por dos ficheros que muestran una relación de palabras asociadas con una etiqueta "positiva" y "negativa" respectivamente sin indicar el grado de la misma. Del mismo modo el diccionario de emojis nos proporciona la proporción de veces que un emoji se utiliza en un mensaje positivo o negativo, por lo tanto, realizaremos un mapeo a "positivo" o "negativo", en función de cual de los dos valores es mayor.

  De esta forma obtenemos cuatro valores por cada *tweet* con valores numéricos enteros que se describen como: *términos_positivos*, que contiene un número entero mayor o igual que 0 correspondiente al número de términos en el *tweet* que aparecen en el léxico etiquetados como "positivos"; *términos_negativos*, que contiene un número entero mayor o igual que 0 correspondiente al número de términos en el *tweet* que aparecen en el léxico etiquetados como "negativo"; *emojis_positivos*, que contiene un número entero mayor o igual que 0 correspondiente al número de emojis en el *tweet* que aparecen en el diccionario de emojis con una mayor proporción de uso en textos "positivos"; y *emojis_negativos*, que contiene un número entero mayor o igual que 0 correspondiente al número de emojis en el *tweet* que aparecen en el diccionario de emojis con una mayor proporción de uso en textos "negativos".

  Para dictaminar la polaridad de los *tweets* en el conjunto de dato, se calcula si predomina el sentimiento positivo o el negativo. El algoritmo de etiquetado asignará clase "positiva" si la suma de términos y emojis positivos es mayor a la de negativos; y asignará "negativo" en caso contrario. De existir igualdad en ambos,

etiquetaremos como “neutro” momentáneamente.

Algoritmo 1: *Etiquetado del subsistema de etiquetado 1 (SPL+Emoji Sentiment Ranking)*

Entrada: Tweet preprocesado[i]
if *(terminos_positivos[i]+emojis_positivos[i]) > (terminos_negativos[i]+emojis_negativos[i])* **then**
  etiqueta_tweet[i]=“positivo” ;
else if *(terminos_positivos+emojis[i]_positivos[i]) < (terminos_negativos[i]+emojis_negativos[i])* **then**
  etiqueta_tweet[i]=“negativo” ;
else
  etiqueta_tweet[i]=“neutro” ;
Salida: Etiqueta del tweet[i] en el subsistema 1

Los resultados que arroja este subsistema con los 15000 *tweets* se muestran en la tabla 3.8.

	Tamaño	
Clasificación	**Sin Lematización**	**Con Lematización**
Positivos	5102	6535
Neutro	5286	3630
Negativo	4612	4835

Términos positivos únicos	819 de 1555 (52.7 %)	770 de 1555 (49.5 %)
Términos negativos únicos	1153 de 2720 (42.4 %)	1086 de 2720 (39.9 %)
Tweets sin polaridad	3820 (25.47 %)	1446 (9.64 %)
NºTérminos positivos	10433	20512
NºTérminos negativos	12061	18663
Total de términos	22494	39175

Cuadro 3.8: *Clasificación de los tweets con el sistema de etiquetado que utiliza el léxico Sentiment Polarity Lexicons (es).*

La tabla 3.8, recoge los valores resultantes del sistema que utiliza el léxico SPL para etiquetar, y se observa claramente, como los datos que han pasado por el proceso de lematización están mejor etiquetados, ya que existe una reducción muy notable de *tweets* que quedan etiquetados como neutros, que recordemos que puede significar una igualdad de proporción de elementos de ambas polaridades o la ausencia de términos o emojis correspondientes a los léxicos. Se mantiene una proporción razonable de *tweets* etiquetados como “positivos” y como “negativos” siendo mayor la primera, y el número de términos que aparecen al aplicar lematización con respecto a no hacerlo prácticamente se duplica pasando de **22494** a **39175**.

En los diagramas 3.5 y 3.6, se muestra un listado de los 10 términos que más se han utilizado para clasificar como “positivo” o “negativo” utilizando este subsistema.

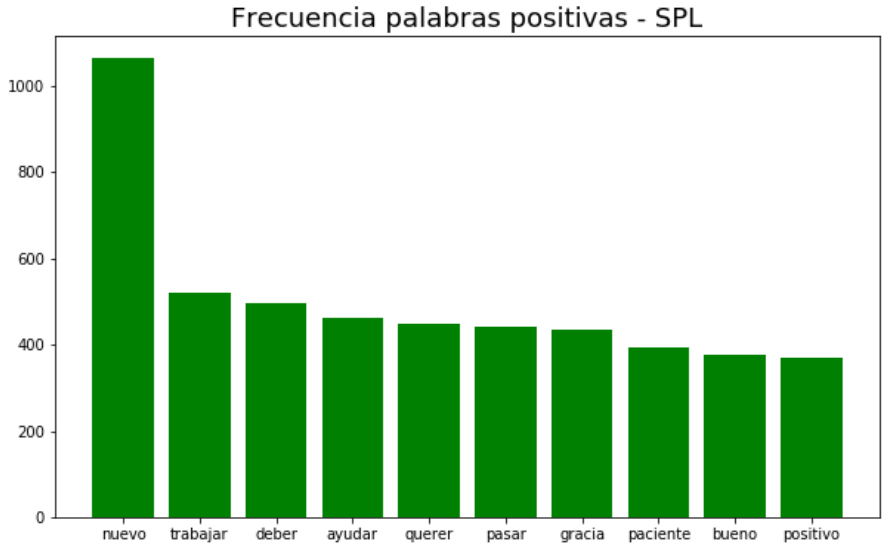


Figura 3.5: *10 términos positivos más utilizados*

- **Subsistema 2: ML-SentiCon + Emoji S.Ranking**
  El léxico ML-SentiCon, presenta una relación de palabras o términos individuales asociados con un valor numérico entre -1 y 1. Los valores de -1 a 0 corresponden al grado de polaridad de los términos negativos y los valores de 0 a 1 corresponden corresponden al grado de polaridad de los términos positivos, siendo los más cercanos a -1 los más negativos y los más cercanos a 1 los más positivos.

  En este caso para cada *tweet* individual se tienen cuatro valores: *max_termino_positivo*, que contiene el grado de polaridad entre 0 y 1 correspondiente al término del *tweet* que mayor carga positiva tenga en base a los términos incluidos en el léxico ; *max_término_negativo*, que contiene el grado de polaridad entre 0 y -1 correspondiente al término del *tweet* que mayor carga negativa tenga en base a los términos incluidos en el léxico; *max_emoji_positivo*, que contiene un valor entre 0 y 1 correspondiente a la proporción de textos positivos en los que aparece el emoji que mayor carga positiva tenga en base al diccionario de emojis; y *max_emoji_negativo*, que contiene un valor entre 0 y 1 correspondiente a la proporción de textos negativos en los que aparece el emoji que mayor carga negativo tenga en base al diccionario de emojis.

  Aplicamos la misma fórmula que en el caso anterior, teniendo en cuenta que ahora tenemos valores reales de 0 a 1 y de 0 a -1 en las variables asociadas a los términos, y de 0 a 1 en las variables asociadas a los emojis, mientras que antes teníamos valores enteros entre 0 y en adelante para las cuatro variables.

  De nuevo el algoritmo de etiquetado posterior asignará clase “positiva” si la suma de los elementos positivos es mayor que la de los negativos y ”negativo” en caso contrario. De existir igualdad en ambos, etiquetaremos de nuevo como “neutro”.

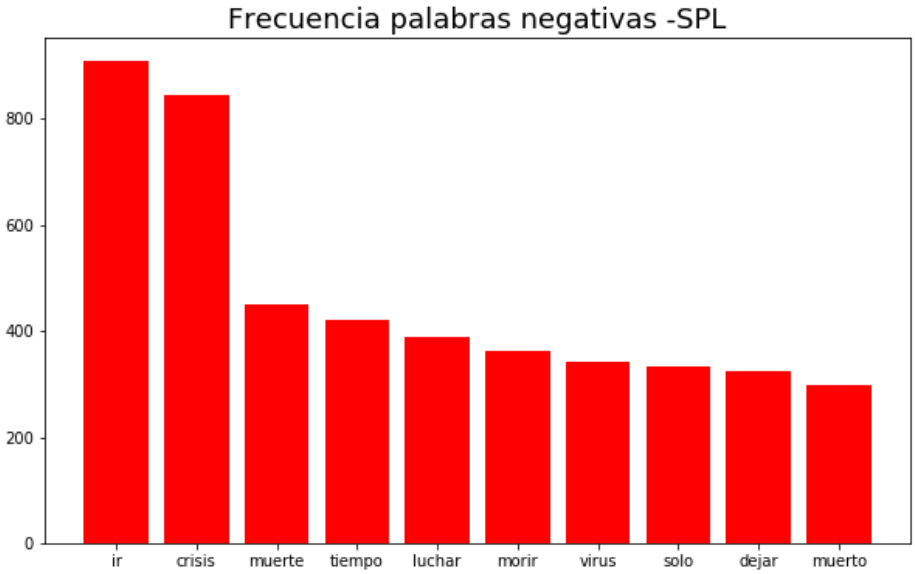


Figura 3.6: *10 términos negativos más utilizados*

Algoritmo 2: *Etiquetado del subsistema de etiquetado 2 (ML-SentiCon+Emoji Sentiment Ranking)*

Entrada: Tweet preprocesado[i]
if *(max_term_positivo[i]+max_emoji_positivo[i]) > (max_term_negativo[i]+max_emoji_negativo[i])* **then**
  etiqueta_tweet[i]="positivo" ;
else if *(max_term_positivo[i]+max_emoji_positivo[i]) < (max_term_negativo[i]+max_emoji_negativo[i])* **then**
  etiqueta_tweet[i]="negativo" ;
else
  etiqueta_tweet[i]="neutro" ;
Salida: Etiqueta del tweet[i] en el subsistema 2

Para este sistema, observamos en la tabla 3.9, como claramente se utiliza un léxico que tiende a clasificar los textos de forma positiva, ya que para el mismo conjunto de *tweets*, el sistema de etiquetado que acabamos de comentar, mantenía una proporción mucho más balanceada. De nuevo el aplicar el proceso de lematización consigue dejar de clasificar como "neutros" a **1500** *tweets* que en su mayoría, pasan a ser etiquetados como "positivos". También el número de términos utilizados para etiquetar aumenta casi en un 50 %, pasando de **12896** a **17815**. Es un número mucho más reducido ya que únicamente se contabiliza el término máximo encontrado por cada *tweet*.

En los diagramas 3.7 y 3.8, se muestran un listado de los 10 términos que más se han utilizado para clasificar como "positivo" o "negativo" utilizando este subsistema.

	Tamaño	
Clasificación	**Sin Lematización**	**Con Lematización**
Positivos	7235	8568
Neutro	5078	3579
Negativo	2687	2853

Términos positivos únicos	902 de 4356 (20.7 %)	859 de 4356 (19.7 %)
Términos negativos únicos	606 de 4404 (13.8 %)	679 de 4404 (15.4 %)
Tweets sin polaridad	5078 (33.85 %)	3544 (23.62 %)
Nº Términos positivos	8789	11837
Nº Términos negativos	4107	5348
Total de términos	12896	17815

Cuadro 3.9: *Clasificación de los tweets con el sistema de etiquetado que utiliza el léxico ML-SentiCon.*

- **Subsistema 3: iSOL + Emoji S.Ranking**.
  El léxico iSOL, es una relación de palabras o términos asociados con una etiqueta "positiva" o "negativa" sin indicar el grado de la misma. Del mismo modo el diccionario de emojis nos proporciona la proporción de veces que un emoji se utiliza en un mensaje positivo o negativo, por lo tanto, realizaremos un mapeo a "positivo" o "negativo", en función de qué valor aparece más veces.

  De esta forma, obtenemos cuatro valores por cada *tweet* con valores numéricos enteros que se describen como: *términos_positivos*, que contiene un número entero mayor o igual que 0 correspondiente al número de términos en el *tweet* que aparecen en el léxico etiquetados como "positivo"; *términos_negativos*, que contiene un número entero mayor o igual que 0 correspondiente al número de términos en el *tweet* que aparecen en el léxico etiquetados como "negativo"; *emojis_positivos*, que contiene un número entero mayor o igual que 0 correspondiente al número de emojis en el *tweet* que aparecen en el diccionario de emojis con una mayor proporción de uso en textos "positivos"; y *emojis_negativos*, que contiene un número entero mayor o igual que 0 correspondiente al número de emojis en el *tweet* que aparecen en el diccionario de emojis con una mayor proporción de uso en textos "negativos".

  Para dictaminar la polaridad de los *tweets* en el conjunto de datos, se calcula si predomina el sentimiento positivo o el negativo. En el algoritmo de etiquetado, se asignará clase "positiva" si la suma de términos y emojis positivos es mayor a la de negativos y "negativo" en caso contrario. De existir igualdad en ambos,

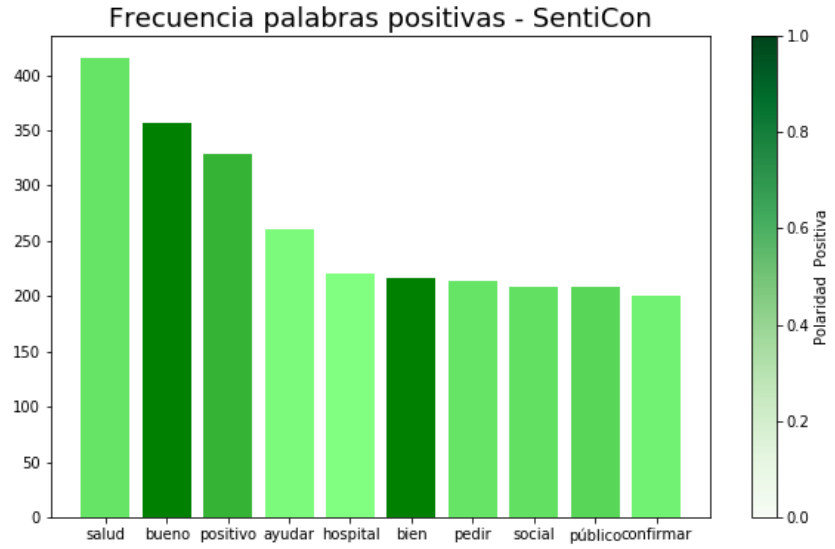


Figura 3.7: *10 términos positivos más utilizados*

etiquetaremos como "neutro" momentáneamente.

Algoritmo 3: *Etiquetado del subsistema de etiquetado 3 (iSOL+Emoji Sentiment Ranking)*

Entrada: Tweet preprocesado[i]
if *(terminos_positivos[i]+emojis_positivos[i]) > (terminos_negativos[i]+emojis_negativos[i])* **then**
  etiqueta_tweet[i]="positivo" ;
else if *(terminos_positivos[i]+emojis_positivos[i]) < (terminos_negativos[i]+emojis_negativos[i])* **then**
  etiqueta_tweet[i]="negativo" ;
else
  etiqueta_tweet[i]="neutro" ;
Salida: Etiqueta del tweet[i] en el subsistema 3

Los resultados que arroja este subsistema con los 15000 *tweets* se muestran en la tabla 3.10.

En el sistema que utiliza el léxico iSOL, la diferencia no es tan abismal como en el sistema anterior, sin embargo, se observa claramente como hay alrededor de 600 *tweets* que dejan de ser clasificados como "neutros" cuando hemos aplicado la lematización, y casi en su totalidad, pasan a ser etiquetados como "negativos". El número de términos que se encuentran en el léxico al realizar el proceso de lematización, de nuevo aumenta en aproximadamente **2500** casos.

En los diagramas 3.9 y 3.10, se muestran un listado de los 10 términos que más se han utilizado para clasificar como "positivo" o "negativo" utilizando este subsistema.

- **Subsistema 4: CRiSOL + Emoji S.Ranking**

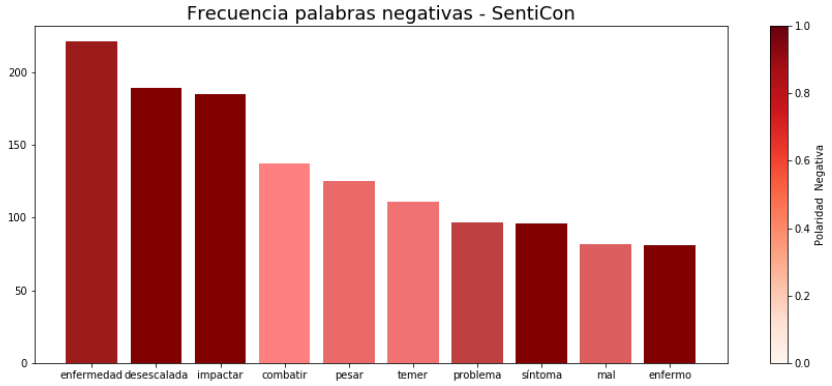


Figura 3.8: *10 términos negativos más utilizados*

	Tamaño	
Clasificación	**Sin Lematización**	**Con Lematización**
Positivos	5256	5312
Neutro	5801	5194
Negativo	3943	4494

Términos positivos únicos	847 de 2507 (33.8 %)	484 de 2507 (19.3 %)
Términos negativos únicos	1168 de 5622 (20.8 %)	788 de 5622 (14 %)
Tweets sin polaridad	4476 (29.84 %)	3819 (25.46 %)
Nº Términos positivos	8890	9783
Nº Términos negativos	8895	10667
Total de términos	17845	20450

Cuadro 3.10: *Clasificación de los tweets con el sistema de etiquetado que utiliza el léxico iSOL.*

El léxico CRiSOL, presenta un listado de palabras o términos individuales, y cada uno de ellos, tiene asociados tres valores correspondientes al grado de pertenencia a sentimiento "positivo", "negativo" y "neutro". Además, la suma de estos tres valores es igual a 1.

Para este caso, cada *tweet* individual tiene de nuevo cuatro valores: *max_termino_positivo*, que contiene el grado de polaridad entre 0 y 1 correspondiente al término del *tweet* que mayor carga positiva tenga en base a los términos incluidos en el léxico; *max_término_negativo*, que contiene el grado de polaridad entre 0 y -1 correspondiente al término del *tweet* que mayor carga negativa tenga en base a los términos incluidos en el léxico; *max_emoji_positivo*, que contiene un valor entre 0 y 1 correspondiente a la proporción de textos positivos en los que aparece el emoji que mayor carga positiva tenga en base al diccionario de emojis; y *max_emoji_negativo*, que contiene un valor entre 0 y 1 correspondiente a la proporción de textos negativos en los que aparece el emoji que mayor carga negativo tenga en base al diccionario de emojis. En este caso pueden existir términos del léxico que presenten valores su-

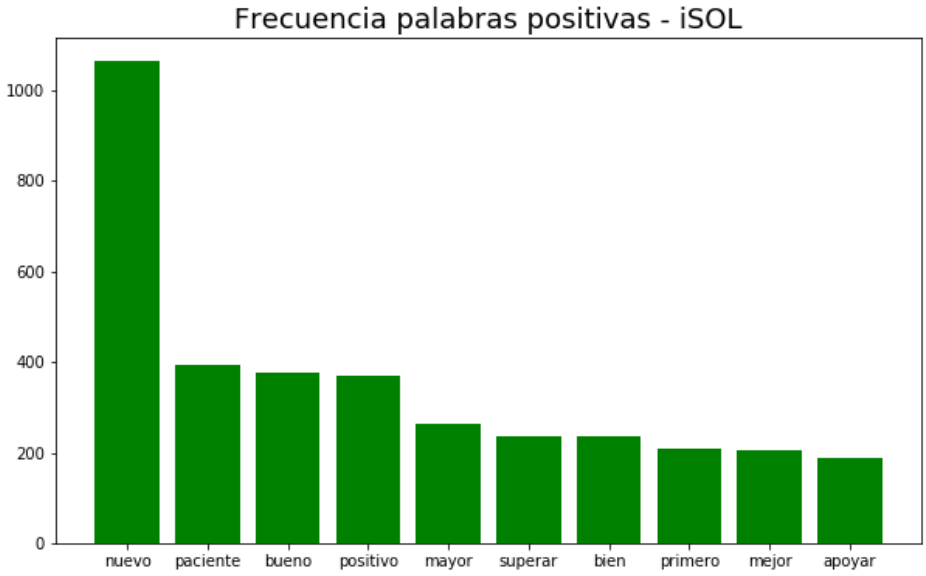


Figura 3.9: *10 términos positivos más utilizados*

periores a cero tanto en "positivo" como en "negativo", dado a que su pertenencia a una clase u a otra es ambigua.

De nuevo, el algoritmo de etiquetado posterior asignará clase "positiva" si la suma de los elementos positivos es mayor que la de los negativos y "negativo" en caso contrario. De existir igualdad en ambos, etiquetaremos de nuevo como "neutro".

Algoritmo 4: *Etiquetado del subsistema de etiquetado 4 (CRiSOL + Emoji Sentiment Ranking)*

Entrada: Tweet preprocesado[i]
if *(max_term_positivo[i]+max_emoji_positivo[i]) > (max_term_negativo[i]+max_emoji_negativo[i])* **then**
  etiqueta_tweet[i]="positivo" ;
else if *(max_term_positivo[i]+max_emoji_positivo[i]) < (max_term_negativo[i]+max_emoji_negativo[i])* **then**
  etiqueta_tweet[i]="negativo" ;
else
  etiqueta_tweet[i]="neutro" ;
Salida: Etiqueta del tweet[i] en el subsistema 4

Los resultados que arroja el último de los subsistemas, con los 15000 *tweets*, se muestran en la tabla 3.11.

Este último subsistema es el que menos *tweets* consigue clasificar como "positivo" o "negativo". Esto podría deberse a que el léxico que se utiliza, es de un tamaño reducido (de unas 4000 palabras). Sin embargo, el léxico SPL también tiene unas características similares, y consigue etiquetar un número mucho mayor de *tweets*.

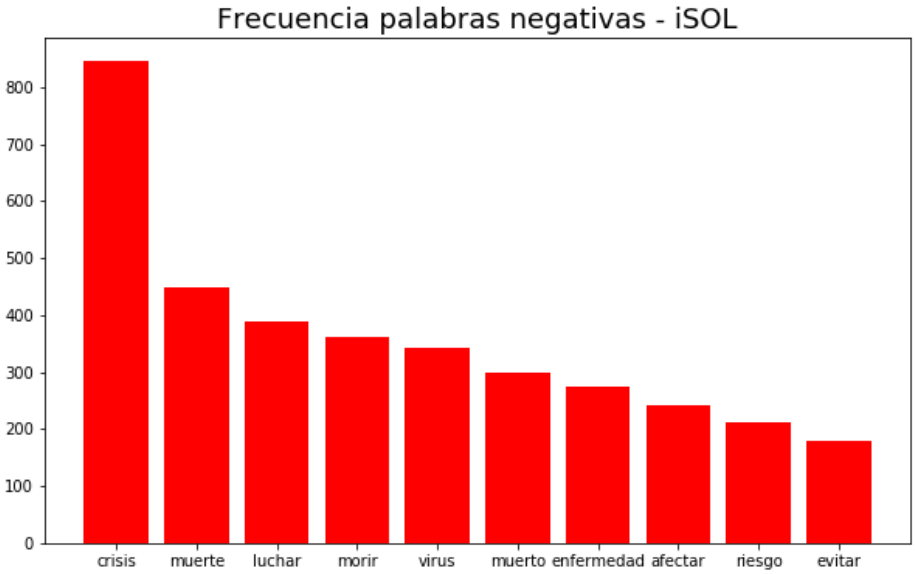


Figura 3.10: *10 términos negativos más utilizados*

De nuevo el aplicar el proceso de lematización, se consigue clasificar **540** *tweets* más de una forma balanceada: 308 como positivos y 232 como negativos. El número de términos totales encontramos en este caso se reduce mínimamente (253 términos). Esto se puede deber tanto al sistema de lematización, como a la construcción de este léxico.

Las figuras 3.11 y 3.12, muestran un listado de los 10 términos que más se han utilizado para clasificar como "positivo" o "negativo" utilizando este subsistema.

Una conclusión clara que hemos sacado observando el funcionamiento de estos cuatro sistemas de etiquetado individuales, es que el proceso de lematización es clave para poder encontrar un mayor número de términos de nuestro conjunto de datos en los léxicos utilizados. Esto se debe, a que los léxicos, han sido elaborados siendo conscientes de que lo más interesante era extraer para cada palabra el lema, y este sería el que apareciera en la lista. De esta forma, mediante el proceso de lematización, cualquier usuario conseguiría unos mejores resultados utilizando su léxico que si no aplicara dicho proceso.

Etiquetado final

En el diagrama 3.13, se muestra un esquema conceptual del sistema descrito en esta sección.

Hemos conseguido etiquetar mediante cuatro sistemas diferentes, un mismo *tweet* cuatro veces como "positivo", "negativo" o "neutro". Para obtener la etiqueta final asociada

	Tamaño	
Clasificación	**Sin Lematización**	**Con Lematización**
Positivos	4535	4843
Neutro	6834	6294
Negativo	3631	3863

Términos positivos únicos	594 de 1514 (39.2 %)	342 de 1514 (22.59 %)
Términos negativos únicos	578 de 2916 (19.8 %)	354 de 2916 (12.13 %)
Tweets sin polaridad	6834 (45.56 %)	6294 (41.96 %)
NºTérminos positivos	5278	5261
NºTérminos negativos	5363	5127
Total de términos	10641	10388

Cuadro 3.11: *Clasificación de los tweets con el sistema de etiquetado que utiliza el léxico CRiSOL*

a cada *tweet*, calcularemos la moda de las etiquetas resultantes y esta será la que se considerará como definitiva, antes de pasar a las siguientes fases. Sin embargo, al calcular la moda para cada uno de los *tweets*, pueden suceder dos posibilidades:

- **Existe una única moda**: En este caso será la clasificación final del *tweet*
- **Existen dos modas**: El número de sistemas utilizados es cuatro, por lo que dos sistemas pueden etiquetar un *tweet* de una misma forma, y los dos restantes de otra misma forma. Dentro de esta situación, nos encontraremos tres distintas formas de actuar:
  - Dos modas son "positivo" y dos modas son "neutro": La clasificación del *tweet* será como "positivo".
  - Dos modas son "negativo" y dos modas son "neutro": La clasificación del *tweet* será como "negativo".
  - Dos modas son "positivo" y dos modas son "negativo": La clasificación del *tweet* será como "neutro". Se considera que no existe fuerza suficiente como para poder clasificar correctamente ese *tweet*, y por tanto, no será utilizado en el conjunto de datos final que se utilice para entrenar nuestro modelo de clasificación como vamos a explicar a continuación.

Los resultados finales del sistema de etiquetado se muestran en la tabla 3.12

Durante todo este proceso, la ausencia de sentimiento positivo y negativo, se ha etiquetado como "neutro", sin embargo el sistema de análisis de sentimientos que se desea construir pretende ser dicotómico, es decir, dado un *tweet* como entrada, la salida resultante será "positivo" o "negativo", en función de el sentimiento al que más se acerque. Por este motivo, se reduce el conjunto de datos a únicamente aquellos que han sido

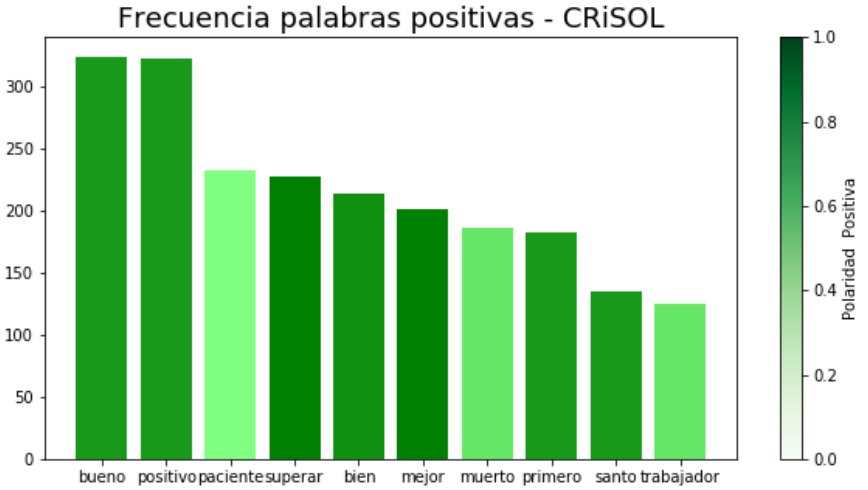


Figura 3.11: *10 términos positivos más utilizados*

	Tamaño	
Clasificación	**Sin Lematización**	**Con Lematización**
Positivos	5588	6493
Neutro	5565	4427
Negativo	3847	4080
N°Términos	73306	87198
N°Emojis	4963	4963
N°Elementos	**78269**	**92161**

Cuadro 3.12: *Clasificación final de los tweets mediante la moda de las clasificaciones de los sistemas individuales.*

etiquetados por el sistema final como "positivo" o "negativo". De esta forma conseguimos un conjunto final que consta de **10573** *tweets*, de los cuales **6493** son considerados "positivos" y **4080** "negativos".

En las tablas 3.14 y 3.15, se muestran los tiempos de ejecución del sistema de etiquetado desglosado por subsistema para los datos lematizados y sin lematizar.

Si consideramos el tiempo de ejecución global, contando con el etiquetado final para esta fase (tanto con lematización como sin lematización), obtenemos el resultado de la tabla 3.16.

Análisis del conjunto de datos

Antes de eliminar los *tweets* considerados como "neutros" o "nulos" con respecto tener un grado de polaridad hacia un sentimiento positivo o negativo, vamos a indagar un poco en

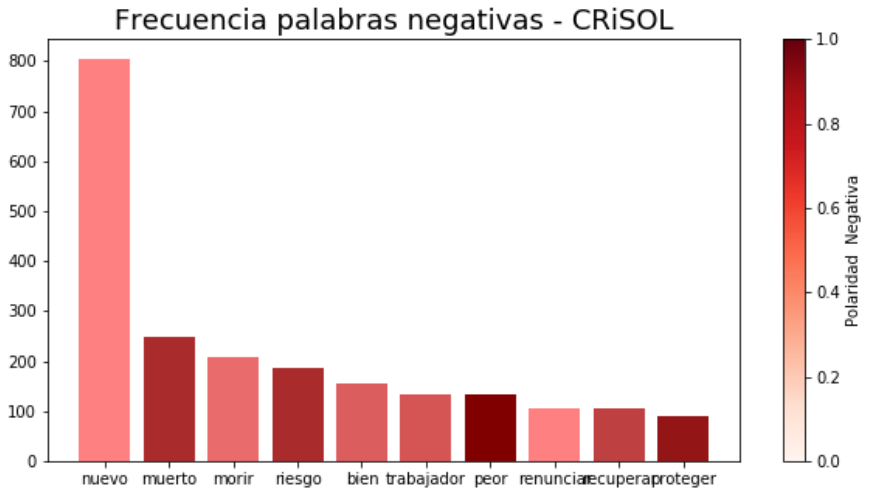


Figura 3.12: *10 términos negativos más utilizados*

Léxico de Términos	Positivas	Negativas	Total
SPL	20512	18663	39175
ML-SentiCon	11837	5348	17185
iSOL	9783	10667	20450
CRiSOL	5261	5127	10388
Total	47393	39805	87198

Cuadro 3.13: *Análisis de los términos utilizados en el etiquetado.*

las características de nuestro corpus lingüístico ya etiquetado. Inicialmente contábamos con **15000** *tweets* compuestos por **390955** elementos individuales antes de ser preprocesados.

Como vamos a trabajar con los *tweets* que han sido etiquetados como "positivo" o "negativo", se pretende buscar una explicación para los motivos que lleven a ciertos *tweets* a ser considerados "neutros". Evidentemente, por la naturaleza de los léxicos utilizados, existen casos en los que pueden aparecer elementos de ambas polaridades que al compararse, no se pueda tener una certeza de que uno de ellos tenga más fuerza que el otro. Es el caso de los *tweets* neutros propiamente dichos. Sin embargo el caso más interesante de analizar, es cuando no encontramos ningún término o emoji que nos pueda proporcionar un grado de polaridad para un *tweet* específico. Estos son los *tweets* que denominaremos "nulos".

Lo más lógico, sería que esto se debiera a la longitud de los mismos, ya que cuantos más caracteres o elementos contiene un *tweet*, más posibilidades existen de que alguno de los elementos contenga cierta polaridad. En la tabla 3.17 observamos claramente como las sospechas se confirman.

Existe una diferencia clara entre las longitudes mencionadas. La longitud media de

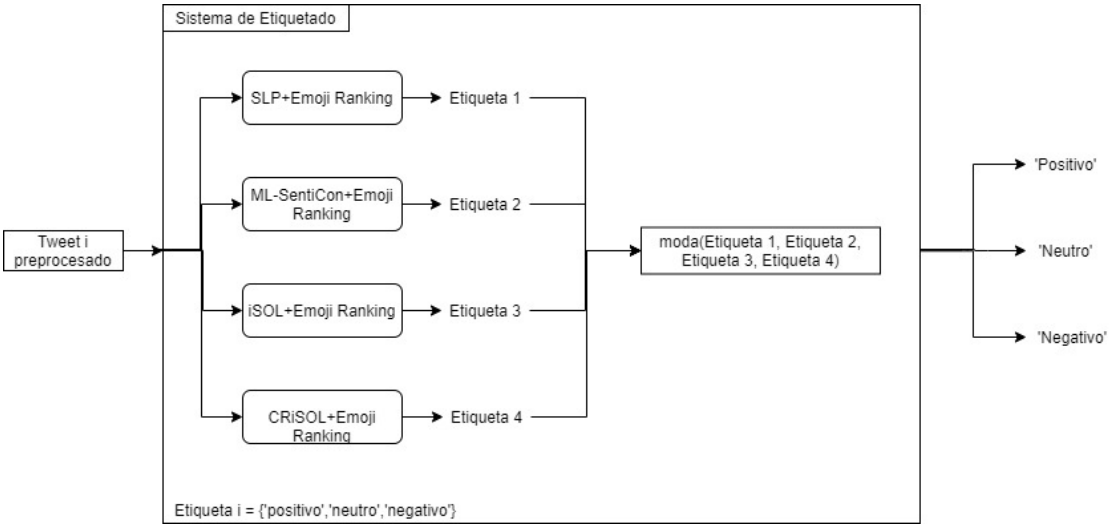


Figura 3.13: *Arquitectura del sistema de etiquetado*

	Cargado del léxico	**Tratamiento de emojis**	**Cálculo variables correspondientes a las palabras**	**Etiquetado**
SPL	0.01 seg	6.78 seg	378.32 seg	1.41 seg
ML-SentiCon	4.91 seg	7.97 seg	447.48 seg	2.16 seg
iSOL	0.58 seg	6.78 seg	366.34 seg	1.41 seg
CRiSOL	1.74 seg	7.97 seg	322.71 seg	2.01 seg

Cuadro 3.14: *Tiempos de ejecución del sistema de etiquetado (con lematización)*

los *tweets* "nulos", es aproximadamente, la mitad que la de los "no nulos". Además, comparando el conjunto completo con el conjunto eliminando los nulos, vemos que la longitud media incluso aumenta de 26 a 27.

Al aplicar el preprocesamiento, se reducen a la mitad las longitudes de los *tweets* en los tres grupos casi en una proporción exacta. Esto se debe a la eliminación de URLs, eliminación de StopWords, etc. En las figuras 3.14 se presentan unos diagramas de cajas donde se observa claramente como existen diferencias en la naturaleza de los *tweets* nulos con el resto.

De la misma forma, podemos observar como las proporciones de elementos en los *tweets*, se mantienen en los distintos grupos y tras el preprocesamiento. La tabla 3.18 muestra los resultados

Igualmente, se puede dar el caso de que *tweets* con longitudes de 1 palabra tengan un mayor contenido sentimental que uno con 200, por tanto es interesante también observar los términos utilizados. Ya hemos comentado que los *tweets* se han extraído de **Twitter** indicando que únicamente queremos los etiquetados como escritos en español, ahora vamos a indagar un poco más en los términos utilizados en los mismos.

	Cargado del léxico	Tratamiento de emojis	Cálculo variables correspondientes a las palabras	Etiquetado
SPL	0.01 seg	6.73 seg	361.8 seg	1.42 seg
ML-SentiCon	5.08 seg	8.13 seg	416.67 seg	2.23 seg
iSOL	0.52 seg	6.22 seg	369.43 seg	1.47 seg
CRiSOL	1.99 seg	8.13 seg	299.43 seg	2.04 seg

Cuadro 3.15: *Tiempos de ejecución del sistema de etiquetado (sin lematización)*

	Sin lematización	Con lematización
Tiempo	1660.9 seg	1766.79 seg

Cuadro 3.16: *Tiempos de ejecución del sistema de etiquetado completo*

El **CREA** (Corpus de Referencia del Español Actual) [28], creado por la Real Academia Española, presenta unos corpus con las palabras más utilizadas en los textos en español, junto con las medidas de frecuencia de las mismas. El corpus escrito cuenta con casi 140000 documentos de donde el 49 % procede de libros, otro 49 % de prensa y un 2 % de material misceláneo. Además el 50 % del material procede de España y la otra mitad de América. Podemos acceder a las 1000, 5000 o 10000 formas más frecuentes, así como la lista total que contiene 737799 formas. Vamos a analizar los términos de nuestro corpus en función de los corpus del CREA de 10000 formas y de una reducción de la lista total a las formas con una frecuencia mayor a 50, resultando en 77729 formas.

De nuevo vamos a comprobar la naturaleza de los *tweets* en base a los tres grupos que se están analizando: "completos", "nulos" y "no nulos". La tabla 3.19 muestra los valores asociados.

Si nos fijamos en los resultados referidos al corpus de 10000 formas, si que observamos una diferencia clara entre los *tweets* nulos y no nulos, destacando que solo la mitad de los términos en los *tweets* nulos tendrían una correspondencia con los términos utilizados en el lenguaje español según nuestro corpus. En general los porcentajes de cobertura son solo del 55-65 %, por tanto, tiene sentido utilizar el siguiente corpus más completo, donde ya se igualan las diferencias y vemos que en general el 82 % de las palabras utilizadas en nuestro corpus corresponden al lenguaje español.

Especial atención merece el tratamiento de los emojis, que ya hemos comentado que tienen una gran importancia en los textos de las redes sociales. Contamos con un léxico que ha recogido 721 emojis con la proporción de veces que han aparecido en textos etiquetados como "positivos", "neutros" o "negativos".

Si hacemos un primer recuento de los emojis que aparecen en nuestro corpus, nos encontramos con **7227**, de los cuales, **4963** aparecen en el léxico. Esto se debe a que existen múltiples emojis que se corresponden con letras, símbolos o signos de puntuación que no debieran contener sentimiento alguno. No obstante, por las características de los lenguajes y las frecuencias de caracteres en los mismos, pudiera darse el caso de que algún

	Tamaño	Sin Preproc	Con Preproc
Completo	15000	26.06	13.08
No Nulos	13848	27.04	13.63
Nulos	1152	14.34	6.54

Cuadro 3.17: *Longitudes medias de los tweets*

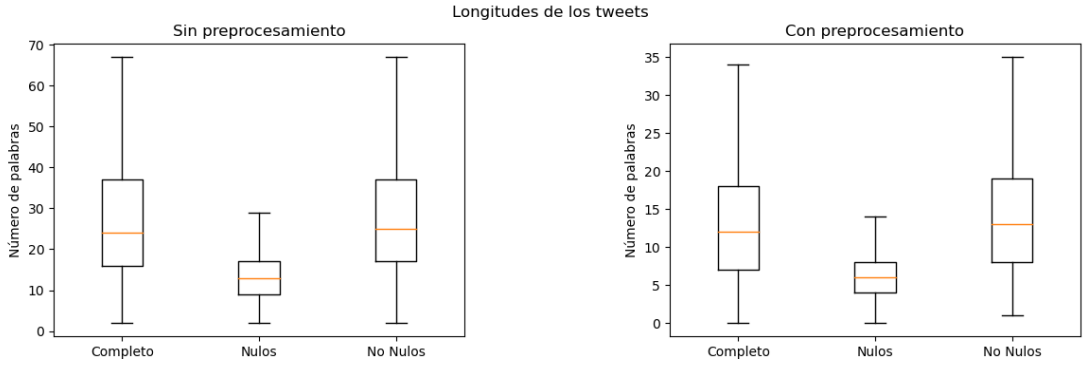


Figura 3.14: *Longitudes medias de los tweets sin y con preprocesamiento*

símbolo en particular si que aparezca en mayor medida asociada a cierta polaridad.

Siguiendo con el análisis, descubrimos que en **12279** de los 15000 *tweets* del corpus, no aparece ningún emoji, o visto de la forma contraria, en el **18.14 %** de los *tweets* recogidos aparece al menos un emoji o emoticono. Basándonos en los trabajos [26], que trabaja con dos conjuntos de 2.15 y 1.5 millones de *tweets*, donde en ambos existe un 15 % de *tweets* que presentan emojis con una media de 2.4 emojis por *tweet* frente a 2.6 emojis de media en nuestro corpus, y en [27], que trabaja con un dataset de 12 millones de *tweet*, con una frecuencia de 19 % de *tweets* conteniendo al menos un emoji, podemos asumir que estamos trabajando con una muestra lo suficiente representativa como para trabajar la influencia de los emojis.

Una vez nos centramos en los *tweets* que contienen al menos un emoji, podemos construir el gráfico 3.15, que muestra la distribución de los emojis en los *tweets* que contienen al menos uno.

Existen 55 *tweets* que presentan un número de emojis mayor o igual a 10, lo cual podría parecer no tener sentido, pero observando alguno de ellos, se entiende lo que ocurre, por lo que hemos comentado anteriormente de las letras o signos de puntuación que se incluyen como emojis. La figura 3.16, contiene un ejemplo claro de un *tweet* que contiene 39 emojis, siendo prácticamente todas letras, mientras quizás solo uno de ellos sea realmente considerado como emoji, y sea el único presente en el léxico con el que trabajamos. Este ejemplo es el *tweet* #1308 en nuestro conjunto de datos inicial.

Podemos confirmar nuestras sospechas, si analizamos la distribución únicamente para los emojis del corpus que aparecen en nuestro léxico de emojis asociados a cierta polaridad.

	Tamaño	Sin Preproc	Con Preproc
Completo	15000	390955	196271
No Nulos	13848	374433	188735
Nulos	1152	16522	7536

Cuadro 3.18: *Longitudes de los tweets*

		Corpus con 10000 formas		Corpus con 77729 formas	
	Tamaño	NºTérminos	% Cubierto	NºTérminos	% Cubierto
Completo	196271	120427	63.70 %	154419	81.68 %
No Nulos	188735	116268	64.04 %	148835	81.97 %
Nulos	7536	4159	55.58 %	5584	74.62 %

Cuadro 3.19: *Análisis de correspondencia real de los términos del corpus con los del corpus español de la RAE*

En la figura 3.17, se ve muy claro como se reducen drásticamente los *tweets* con una gran cantidad de emojis. Si contamos en conjunto los *tweets* donde aparecían más de 5 emojis, pasan de ser de 228 a 97 si solo contabilizamos los emojis que aparecen en nuestro léxico.

De hecho, si nos fijamos únicamente en los emojis que aparecen en nuestro léxico, las cifras cambian como se muestra en la tabla 3.20. Se reduce el número de *tweets* que contienen emojis, así como el número de emojis y la media de emojis por *tweet*.

	NºTweets con Emojis	Nº Emojis	Media por tweet
Completo	2721 (18.14 %)	7227	2.65
Solo en léxico	2327 (15.54 %)	4963	2.13

Cuadro 3.20: *Comparativa de las cifras de los emojis en el léxico utilizado y en global*

3.6 Extracción de características

En esta fase, vamos a considerar las dos principales técnicas comentadas anteriormente: Bag of Words, con TF-IDF, y Word Embeddings, con Word2Vec. Para ello, nos ayudaremos de paquetes específicos como son: **gensim**, que es una biblioteca preparada para el procesamiento de lenguaje natural, y **sklearn**, que es probablemente la biblioteca más potente para aprendizaje automático.

Ambos métodos de extracción de características tienen una serie de parámetros a configurar, por tanto, es necesaria realizar una etapa de selección de parámetros, que se combinará con la selección de modelos que se explica en la sección siguiente.

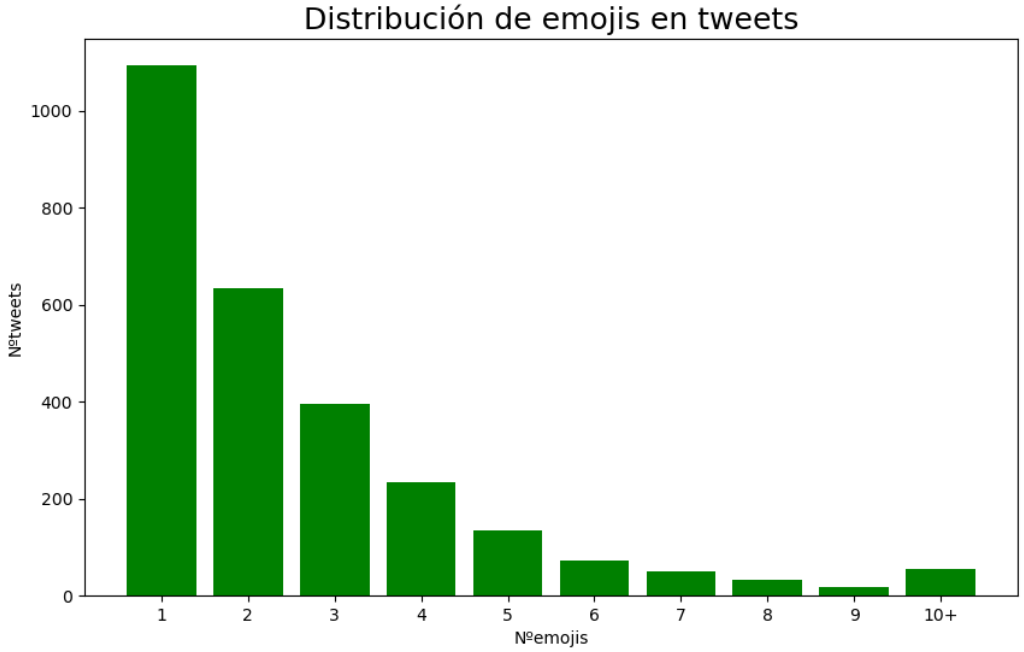


Figura 3.15: *Frecuencia de emojis en los tweets que presentan mínimo uno*

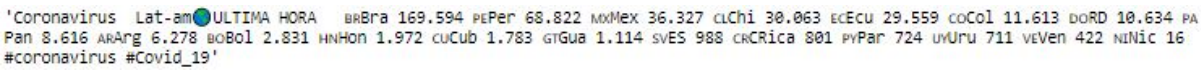


Figura 3.16: *Tweet #1308 con cantidad atípica de "emojis"*

3.6.1 Bag of Words

Para realizar BoW con TF-IDF, crearemos una instancia **TfidfVectorizer** que encontramos en sklearn.feature_extraction.text. Esta, se crea recibiendo los parámetros de configuración iniciales, y posteriormente, mediante un método de su clase fit_transform(), recibe el conjunto de datos preprocesado para construir el array asociado. Este array tiene como dimensiones el número de *tweets* que forma el conjunto de datos y el número de términos máximo que se ha establecido previamente. En la figura 3.18, se observa un ejemplo ilustrativo de como funciona esta técnica.

Con respecto a los parámetros para la construcción de BoW, algunos los fijaremos desde el primer momento como podría ser la eliminación de stopwords en español (stopwords="es") o establecer que si una palabra aparece en más del 70 % de los *tweets* del conjunto de datos, no se considerará como posible característica (max_df=0.7). Los dos parámetros más importantes que determinar son: el tamaño de la bolsa de palabras (max_features), es decir el número de palabras que se van a incluir en ella, y el mínimo número de veces que tiene que aparecer una palabra en el corpus para que pueda ser

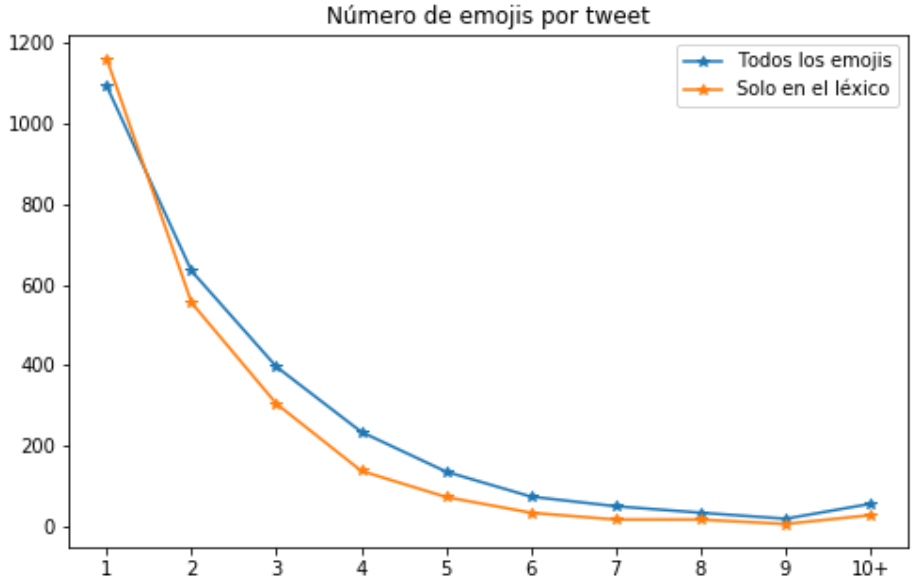


Figura 3.17: *Comparativa frequencia de emojis únicamente en el léxico*

considerada en la construcción de la bolsa de palabras (min_df). Ambos parámetros se decidirán más tarde en una tarea de selección de parámetros conjunta con la elaboración del modelo de clasificación.

Antes de entrar en la selección de parámetros, tenemos que indagar más en la frecuencia de las palabras de nuestro corpus para tener un criterio mayor en como seleccionar los mismos. La tabla 3.21, muestra como cambian estos valores conforme modificamos los distintos parámetros que tunearemos.

Min DF	1	2	3	4	5	10
Términos únicos	18758	7877	5445	4357	3709	2232

Cuadro 3.21: *Comparativa del número de palabras únicas consideradas en el modelo BoW conforme al mínimo de veces que estas tienen que aparecer en el corpus*

Los resultados, muestran como es obvio que el parámetro min_df tiene una gran influencia en el número de términos que se consideran, aunque quizás esto no se traduzca en diferentes resultados más adelante. Existen aproximadamente 10000 palabras que aparecen solo una vez en todo el corpus, que sabiendo que se proviene de una extracción directa de las redes sociales, pueden ser expresiones, coloquialismos o directamente palabras mal escritas que no tendrían por qué ser importantes para el modelado de nuestro sistema.

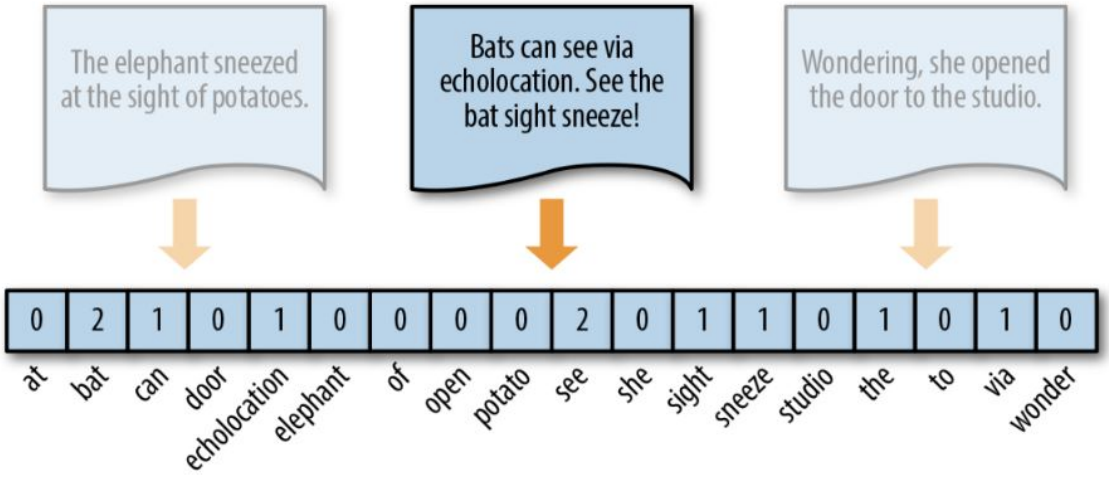


Figura 3.18: *Funcionamiento de Bag of Words*

3.6.2 Word embeddings (Word2Vec)

Para realizar Word Embedding con Word2Vec, podemos construir nuestro propio modelo, llamando a **Word2Vec** que encontramos en gensim.models, al que le pasaremos el corpus de datos preprocesado junto con una serie de parámetros que tenemos que definir. Los más importantes para definir serían: el tamaño de los vectores de palabras (size), la ventana (window), que es la distancia máxima entre la palabra actual y la palabra predicha dentro del mismo *tweet*, el número mínimo de veces que tiene que aparecer un corpus para que pueda ser considerada en la construcción de los vectores (min_count) y en relación con la velocidad para entrenar el modelo se puede definir el parámetro que define los hilos a utilizar (workers).

Además, existen dos variantes de arquitectura de Word2Vec como son: **CBOW** (Continuous Bag of Words) o **Skip Gram**. CBOW, recibe como entrada una ventana de palabras que rodean a la palabra a predecir y como salida se desea obtener la palabra que mejor se ajusta a dicho contexto. En esta variante el orden de las palabras del contexto, no influyen en la predicción. En la variante Skip Gram, se tiene como entrada la palabra actual para predecir la ventana que lo rodea, es decir, el contexto de dicha palabra. Esta arquitectura pondera las palabras situadas más cerca en el documento más fuertemente que aquellas que no lo están. Podemos observar en la figura 3.19, este funcionamiento.

Según las notas del autor [Google Code], CBOW es más rápido pero Skip Gram consigue un mejor trabajo para palabras infrecuentes, por tanto, deberíamos probar ambas arquitecturas para ver cual resulta mejor en nuestro experimento. Esto se consigue fácilmente con la variable "sg", que por defecto realiza la variante CBOW, pero si la ponemos con valor 1 utilizará la variante Skip Gram.

Es importante también analizar como se realiza eel entrenamiento de la red neuronal. Existen dos posibilidades como son: **softmax jerárquico** y **negative sampling** que ayudan al funcionamiento de la misma. Tenemos que tener en cuenta que la última capa

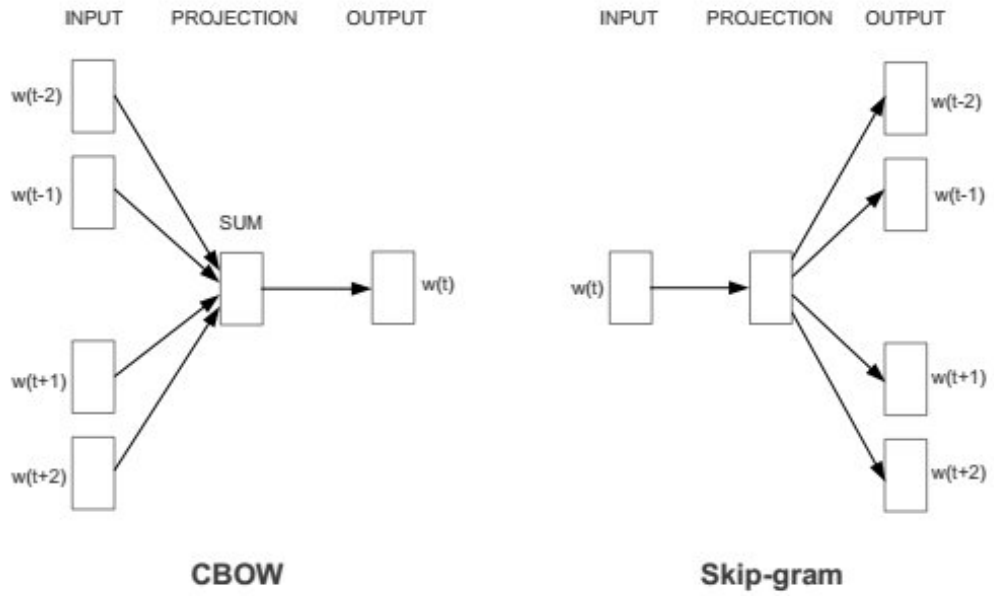


Figura 3.19: *Arquitecturas de Word2Vec CBOW y Skip Gram*

de la red utiliza la función softmax, y que las muestras de entrenamiento, son parejas de palabras seleccionadas en base a su proximidad de ocurrencia. Existen varios problemas con W2V como son que para cada muestra de entrenamiento, solo los pesos de la palabra objetivo obtienen una actualización significante. El resto de las palabras que no son el objetivo, reciben actualización marginal o nula. Además, para cada muestra de entrenamiento, la función softmax calcula todas las probabilidades finales, lo cual es una operación muy costosa. Teniendo todo esto en cuenta, **negative sampling**, permite modificar solo un pequeño porcentaje de los pesos. Esto se consigue seleccionando aleatoriamente un conjunto k de palabras negativas, es decir palabras que no están en el contexto, de las cuales solo se actualizarán los pesos junto con la palabra objetivo. Lógicamente esta aproximación es más rápida ya que se ahorra gran cantidad de cálculo computacional.

La otra variante que se puede utilizar es el **softmax jerárquico**. Esta metodología, reduce la complejidad computacional de $O(V)$ a $O(log_2 V)$. Esto se debe, a que se utiliza un árbol binario donde las hojas representan las probabilidades de las palabras. Cada palabra del vocabulario tiene un camino desde la raíz hasta su hoja que representa la probabilidad. Esta aproximación, según las notas del autor produce mejores resultados si utilizamos palabras poco frecuentes mientras que negative sampling produce mejores resultados con vectores de baja dimensión y palabras frecuentes. La selección de una técnica u otra, se modela con la variable "hs" (hierarchical softmax) que puesta a 1 significa que utilizas softmax jerárquico, y puesta a 0, utilizas negative sampling con el número de palabras negativas seleccionadas en la variable "negative".

Volviendo a la selección de parámetros, se tomará la decisión de que el número mínimo de veces que tenga que aparecer una palabra en el corpus para ser considerada sea 1

(min_count=1), ya que no estamos trabajando con un conjunto de datos inmenso como para suponer que todas las palabras van a aparecer más de una vez. Con respecto al número de "trabajadores", el número por defecto es 3, y al construir nuestro modelo, el tiempo de ejecución es suficientemente rápido como para no preocuparnos por él. El tamaño de la ventana y el tamaño de los vectores de palabras, serán elegidos en la tarea de construcción de parámetros combinado con la elaboración de los modelos de clasificación. Siguiendo las notas del autor, los valores más recomendados para el tamaño de ventana serían 10 para skip-gram y 5 para CBOW, con respecto a la dimensionalidad, se recomienda establecerla entre 100 y 1000. Además probaremos las dos técnicas de entrenamiento seleccionadas para ver si claramente una funciona mejor que la otra.

Una vez hemos construido el modelo con los vectores de palabras, vamos a construir los vectores asociados a cada *tweet*. Siguiendo con la técnica más habitual utilizada en los distintos trabajos analizados, lo haremos calculando la media de los vectores de cada palabra de un *tweet* individual. Hay que tener en cuenta que si solo entrenáramos nuestro modelo con palabras que tengan que aparecer más de una vez en el corpus, o bien entrenamos solo con el conjunto de entrenamiento, se van a producir casos en los cuales las palabras pueden no estar incluidas en el modelo, y por tanto, no tener una representación vectorial, por tanto no se podrían considerar.

3.7 Modelado

Una vez tenemos los vectores numéricos construidos en la fase de extracción de características, nos disponemos a seleccionar una serie de algoritmos de clasificación supervisados cuyo funcionamiento comprobaremos, conjuntamente, con el de las técnicas de extracción de características, ya que también vamos a tener que determinar ciertos parámetros en los distintos algoritmos. Probaremos el funcionamiento de tres algoritmos supervisados que son comúnmente utilizados en el análisis de sentimientos: **SVM**, **Random Forest** y **Regresión Logística**.

Support Vector Machine (SVM)

Está ampliamente aceptado el funcionamiento de SVM para tareas de clasificación binaria como es nuestro caso. Las máquinas de vector soporte construyen un hiperplano óptimo en forma de superficie de decisión tal que el margen de separación entre las dos clases en los datos, se amplíe al máximo. En la figura 3.20, observamos una ilustración del problema con dimensión 2 como es nuestro caso.

Ya que vamos a clasificar en función de dos clases opuestas, "positivo" y "negativo", utilizaremos el kernel lineal que está destinado al aprendizaje de dos clases.

$$K\left(x_1, x_2\right)) = x_1^\top x_2 \tag{3.7.1}$$

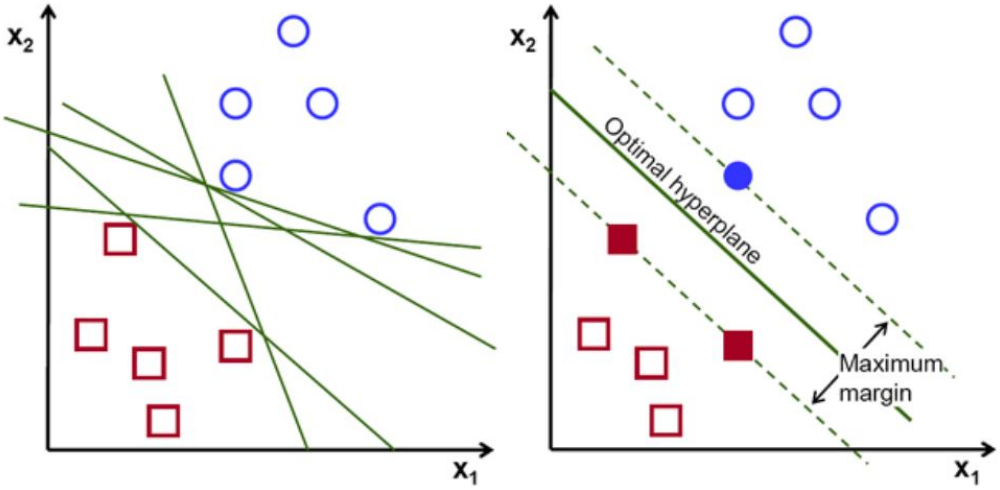


Figura 3.20: *Funcionamiento de SVM en dimensión 2*

La implementación de este algoritmo es muy sencilla utilizando la librería **sklearn** ya introducida anteriormente. Con la orden *SVC(kernel="linear")*, construimos un clasificador SVM con kernel lineal. En nuestro caso, el único parámetro adicional a configurar, es el parámetro de regularización *C*. Este parámetro, indica cuanto queremos evitar la clasificación errónea en el conjunto de entrenamiento. Para valores grandes de C, se construirá un hiperplano con un margen menor si este consigue clasificar un mayor número de puntos de entrenamiento correctamente. Del mismo modo un valor pequeño construirá un hiperplano con un margen mayor incluso si clasifica erróneamente más puntos.

El clasificador, a continuación, se ajusta con la función fit(), pasándole como argumentos: los datos de entrenamiento, que corresponden a los vectores numéricos que hemos construido en la fase de extracción de características, y las clases de cada *tweet* en el conjunto de entrenamiento. Después de esto, con la función predict(), se le pasa como argumento los datos del conjunto de test y proporciona como salida, las clases predichas que compararemos con las originales para ver el funcionamiento.

Random Forest

Random Forest es un ensemble utilizado tanto para clasificación como para regresión. En nuestro caso, lo utilizamos para clasificación entre dos clases posibles. Su funcionamiento dentro de los ensembles es bastante intuitivo. Se construyen un número determinado de árboles de decisión en la fase de entrenamiento, y se calcula la salida final del ensemble como la moda de las salidas de cada uno de los árboles de decisión como se puede ver en la figura 3.21. Random Forest, corrige la querencia a sobreajustar que tienen los árboles de decisión por separado y son una de los algoritmos más utilizados en ML.

Para la construcción, utilizaremos el módulo **sklearn.ensemble**, donde podemos construir un clasificador RandomForestClassifier() del que tenemos que decidir el parámetro más importante que es el tamaño (n_estimators), es decir, el número de árboles de

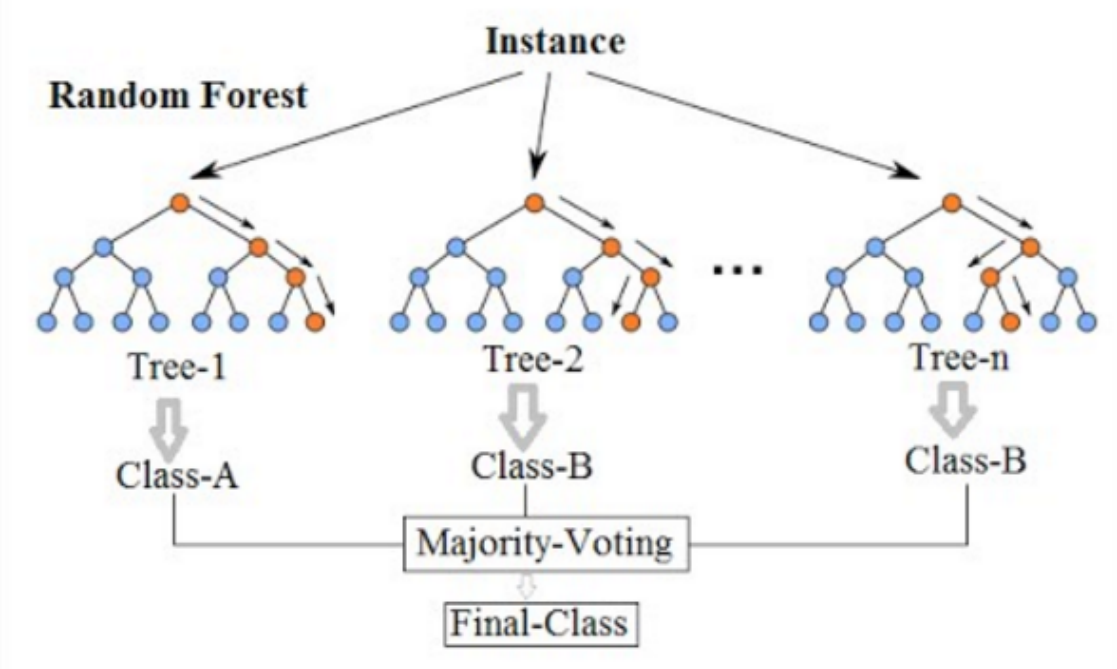


Figura 3.21: *Funcionamiento de Random Forest*

decisión que va a construir para el resultado final.

Regresión Logística

La Regresión Logística, es un tipo de análisis de regresión específico que predice el resultado de una variable categórica, en este caso de dos categorías, en función de las variables independientes o predictoras. Su funcionamiento, se basa en el uso de la función de enlace **logit**. La función 3.7.2, es la responsable de hacer corresponder la salida de un modelo de regresión simple, que puede no estar acotada entre [0,1], a este dominio probabilístico, para poder después clasificar al valor más elevado.

$$\text{logit}\,(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i} \tag{3.7.2}$$

El modelo Regresión Logística se calcula minimizando la función de coste MSE (Mean Squared Error) 3.7.3. Esto se consigue, actualizando los parámetros de cualquier forma que reduzca el valor de MSE. Sin embargo, cuanto más grandes son los parámetros, más alta es la posibilidad de que nuestro modelo sobreajuste los datos. Para solucionar esto, se puede añadir el parámetro de regularización λ que vemos en la ecuación 3.7.4. Este parámetro, penaliza valores grandes, ya que cada vez que un parámetro es actualizado para ser significativamente grande, el valor de la función de coste se aumentará por este

término, y como resultado, será penalizado, y actualizado a un valor menor.

$$J = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta \left(X^{(i)} \right) - y^{(i)} \right)^2 \tag{3.7.3}$$

$$J = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta \left(X^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \tag{3.7.4}$$

La implementación sigue de nuevo el mismo esquema que los algoritmos anteriormente comentados. Tenemos una función LogisticRegression() que se encuentra en el módulo **sklearn.linear_model**. Esta función, recibe los parámetros de configuración deseados. En nuestro caso, igual que en SVM, realizaremos una tarea de selección de parámetros para el inverso del parámetro de regularización C ($C = 1/\lambda$). Con esta función crearemos el clasificador que posteriormente ajustaremos con la función fit() pasándole como argumentos el conjunto de entrenamiento y las etiquetas de entrenamiento.

3.8 Selección de parámetros

Vamos a dividir el conjunto de datos en entrenamiento y prueba, donde este último no se va a utilizar hasta la fase de validación. El conjunto de entrenamiento, consta de **8458** *tweets* y el de prueba de **2115**, estando ambos estratificados por la clase, que recordemos, mantiene una proporción de un 61 % de positivos.

Tenemos que realizar selección de parámetros en dos fases diferentes. Primero se decide la configuración de los parámetros de los modelos para la fase de extracción de características, es decir, para BoW, con TF-IDF, y Word2Vec, en sus dos variantes CBOW y Skip Gram. Después tenemos que comprobar la configuración de los parámetros de los algoritmos de clasificación (SVM, Random Forest y Regresión Logística).

Para realizar esta labor, construiremos dos experimentos distintos como vemos en la figura 3.22. El primero, utilizará como extracción de características BoW con TF-IDF y como algoritmos de clasificación los tres mencionados; y el segundo utilizará como extracción de características Word2Vec, con las dos variantes mencionadas, y como algoritmos de clasificación también los tres clasificadores que hemos comentado. Se busca la combinación óptima de técnicas que proporcione el mejor resultado con el conjunto de entrenamiento.

Los posibles parámetros de la fase de extracción de características se introducen manualmente. Sin embargo para los relacionados con los algoritmos de clasificación, existe el módulo **sklearn.model_selection** con una función clave que es **GridSearchCV**. Este función recibe un estimador junto con una lista de parámetros de configuración a explorar, y realiza una búsqueda exhaustiva con todas las posibles combinaciones de los

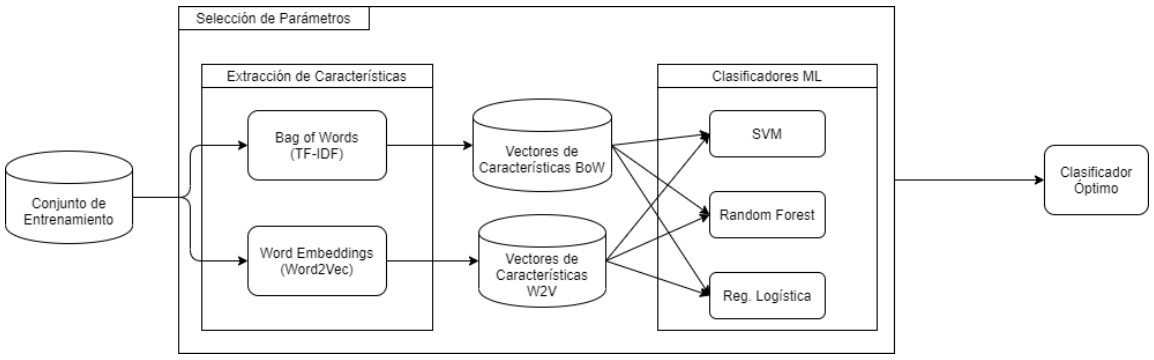


Figura 3.22: *Construcción del clasificador óptimo*

mismos seleccionando cuales producen mejores resultados.

Para validar, se le especifica que diseño de experimento se desea realizar. En nuestro caso, una validación cruzada estratificada de 5 folds generada con la función StratifiedK-Fold() y que se pasa a la función como parámetro. Además, la métrica de validación, si no se le especifica lo contrario, será la tasa de acierto (sklearn.metrics.accuracy_score) para tareas de clasificación y el R^2 (sklearn.metrics.r2_score) para tareas de regresión.

Además de comprobar por defecto la tasa de acierto (accuracy), cambiaremos la métrica de validación a la **medida F1 ponderada** (scoring="f1-weighted"). Esta métrica, como hemos explicado anteriormente, es un compromiso entre las métricas *precission* y *recall*, que se centran en evaluar la calidad y cantidad de los elementos clasificados y es comúnmente utilizada para tareas de procesamiento de lenguaje natural. La parte de la ponderación viene de que las clases no mantienen una proporción 50/50.

En las tablas 3.22, 3.23 y 3.24, se muestran los resultados de los estimadores que mejores resultados dan en cada uno de los dos experimentos construidos; y en las tablas 3.25 y 3.26, tenemos una relación de los tiempos de ejecución. Como la búsqueda es exhaustiva, el tiempo de ejecución aumenta exponencialmente con el número de parámetros incluidos a seleccionar, por tanto, hay que tomar decisiones de antemano sobre la configuración del estimador. Los tiempos para Word2Vec son similares independientemente de la arquitectura que se utilice, por tanto se indican únicamente los tiempos que surgen de la utilización de CBOW.

Claramente obtenemos unos mejores resultados utilizando BoW como opción para extraer características. Recordemos que la proporción de nuestros datos es de 61.4 % de *tweets* "positivos", de forma que si quisiéramos construir un sistema ficticio, que etiquetara todos los *tweets* que recibiera de entrada como "positivo", conseguiríamos una tasa de acierto de 61.4 %.

Fijándonos en Word2Vec como extracción de características, claramente observamos unos resultados mejores con la arquitectura Skip Gram. Las notas del autor ya nos habían hecho intuir que esto podría ser así, ya que funciona mejor para palabras poco frecuentes, y como hemos analizado anteriormente, una gran cantidad de los términos de nuestro corpus aparecen solo una vez. Con esto sabido, vamos a apartar ya la posibilidad de

		Clasificador								
Extr.Caract: BoW		**SVM**			**Random Forest**			**Reg.Logística**		
Tamaño	**MinDF**	**C**	**acc**	**F1**	**N_Est**	**acc**	**F1**	**C**	**acc**	**F1**
250	1	1	0.77	0.77	1000	0.78	0.78	10	0.77	0.77
	3	1	0.77	0.77	300	0.78	0.78	10	0.77	0.77
	5	1	0.77	0.77	1000	0.78	0.78	10	0.77	0.77
	10	1	0.77	0.77	300	0.78	0.78	10	0.77	0.77
500	1	1	0.81	0.81	1000	0.81	0.81	1	0.81	0.81
	3	1	0.81	0.81	1000	0.81	0.81	1	0.81	0.81
	5	1	0.81	0.81	500	0.81	0.81	1	0.81	0.81
	10	1	0.81	0.81	300	0.81	0.81	10	0.81	0.81
1000	1	1	0.84	0.84	1000	0.83	0.84	10	0.84	0.84
	3	1	0.84	0.84	300	0.83	0.84	10	0.84	0.84
	5	1	0.84	0.84	1000	0.83	0.84	10	0.84	0.84
	10	1	0.84	0.84	1000	0.83	0.84	1	0.83	0.84

Cuadro 3.22: *Selección de parámetros con BoW (TF-IDF) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

		Clasificador								
Extr.Caract: W2V (CBOW)		**SVM**			**Random Forest**			**Reg.Logística**		
Tamaño	**Ventana**	**C**	**acc**	**F1**	**N_Est**	**acc**	**F1**	**C**	**acc**	**F1**
200	5	10	0.67	0.75	300	0.7	0.76	10	0.67	0.74
	10	10	0.67	0.75	200	0.7	0.76	10	0.68	0.75
	15	10	0.68	0.75	300	0.7	0.76	10	0.68	0.75
400	5	10	0.66	0.75	300	0.7	0.76	10	0.67	0.75
	10	10	0.67	0.75	100	0.7	0.76	10	0.67	0.75
	15	10	0.67	0.75	200	0.7	0.76	10	0.68	0.75
600	5	1	0.66	0.75	300	0.69	0.76	10	0.67	0.74
	10	1	0.67	0.75	200	0.7	0.76	10	0.67	0.75
	15	1	0.66	0.75	300	0.69	0.76	10	0.67	0.75
800	5	1	0.66	0.75	300	0.7	0.75	10	0.67	0.74
	10	1	0.67	0.75	200	0.69	0.76	10	0.67	0.75
	15	1	0.66	0.75	200	0.69	0.76	10	0.68	0.74

Cuadro 3.23: *Selección de parámetros con Word2Vec (CBOW) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

		Clasificador								
Extr.Caract: W2V (Skip Gram)		**SVM**			**Random Forest**			**Reg.Logística**		
Tamaño	**Ventana**	**C**	**acc (ns)**	**acc (hs)**	**N_Est**	**acc (ns)**	**acc (hs)**	**C**	**acc (ns)**	**acc (hs)**
200	5	10	0.74	0.79	100	0.76	0.80	10	0.74	0.78
	10	10	0.76	0.79	300	0.78	0.80	10	0.76	0.78
	15	10	0.77	0.80	300	0.79	0.80	10	0.76	0.80
400	5	10	0.74	0.78	200	0.76	0.80	10	0.73	0.78
	10	10	0.76	0.79	200	0.78	0.80	10	0.76	0.79
	15	10	0.76	0.81	200	0.78	0.80	10	0.76	0.80
600	5	10	0.73	0.78	100	0.76	0.79	10	0.73	0.78
	10	10	0.76	0.80	300	0.78	0.80	10	0.75	0.79
	15	10	0.76	0.81	200	0.79	0.80	10	0.76	0.80
800	5	10	0.73	0.78	200	0.76	0.80	10	0.72	0.78
	10	10	0.75	0.80	300	0.77	0.80	10	0.75	0.79
	15	10	0.76	0.80	200	0.78	0.80	10	0.76	0.79

Cuadro 3.24: *Selección de parámetros con Word2Vec (SkipGram) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

Extr.Caract: BoW			**Clasificador**		
Tamaño	**MinDF**	**Tiempo**	**SVM**	**Random Forest**	**Reg.Logística**
250	1	0.35 seg	179.73 seg	376.55 seg	0.79 seg
	3	0.34 seg	185.28 seg	354.48 seg	0.8 seg
	5	0.32 seg	183.87 seg	399.23 seg	0.81 seg
	10	0.33 seg	194.76 seg	359.54 seg	0.78 seg
500	1	0.32 seg	301.52 seg	858.13 seg	1.21 seg
	3	0.34 seg	317.56 seg	852.6 seg	1.27 seg
	5	0.37 seg	303.21 seg	734.97 seg	1.21 seg
	10	0.37 seg	299.83 seg	709.91 seg	1.22 seg
1000	1	0.36 seg	567.72 seg	1642.55 seg	1.91 seg
	3	0.41 seg	555.9 seg	1554.57 seg	1.94 seg
	5	0.34 seg	559.39 seg	1630.23 seg	1.91 seg
	10	0.34 seg	560.01 seg	1593.56 seg	2.04 seg

Cuadro 3.25: *Tiempos de ejecución selección de parámetros con BoW (TF-IDF) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

Extr.Caract: Word2Vec				Clasificador		
Tamaño	**Ventana**	**Tiempo**	**C.Medias**	**SVM**	**RandomForest**	**Reg.Log**
200	5	1.66 seg	1.16 seg	192.74 seg	278.10 seg	9.31 seg
	10	1.74 seg	1.1 seg	226.74 seg	285.29 seg	11.59 seg
	15	2.19 seg	1.3 seg	220.95 seg	276.38 seg	10.52 seg
400	5	2.15 seg	1.2 seg	397.83 seg	402.02 seg	17.21 seg
	10	2.44 seg	1.24 seg	403.3 seg	440.46 seg	23.1 seg
	15	3.81 seg	1.71 seg	483.35 seg	471.37 seg	22.56 seg
600	5	3.78 seg	1.57 seg	678.83 seg	569.72 seg	26.3 seg
	10	4.64 seg	1.46 seg	624.52 seg	528.68 seg	28.28 seg
	15	4.1 seg	1.56 seg	586.65 seg	518.92 seg	28.42 seg
800	5	3.56 seg	1.46 seg	712.71 seg	549.26 seg	29.29 seg
	10	3.4 seg	1.27 seg	703.26 seg	524.21 seg	32.87 seg
	15	4.23 seg	1.5 seg	683.99 seg	516.83 seg	33.98 seg

Cuadro 3.26: *Tiempos de ejecución selección de parámetros con Word Embeddings (Word2Vec) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

utilizar CBOW. Sin embargo, con Skip Gram, vemos que para ventanas más grandes, obtenemos unos mejores resultados, por tanto, seguiremos comprobando si obtenemos una mejoría en nuestros resultados si seguimos ampliando estos valores.

También, una vez que queda claro que existe una arquitectura peor, vamos a comprobar como si existen diferencias significativas dependiendo de la forma de entrenamiento de la red Word2Vec, es decir, comprobar si existe diferencia entre utilizar "negative sampling" y "hierarchical softmax".

Utilizando BoW (TF-IDF) como extracción de características, observamos claramente que a mayor tamaño de la bolsa de palabras, mejores valores obtenemos. Ya con el menor tamaño probado (250 palabras) obtenemos un valor del 77-78 % tanto en la tasa de acierto como en la medida F1, que es casi 20 puntos por encima que el baseline. Para el mayor tamaño probado, 1000 palabras, obtenemos unos valores de 83-84 %, más de 20 puntos por encima del baseline. Sin embargo, no parece que el parámetro que indica el mínimo de veces que una palabra tiene que aparecer en el corpus para ser considerada, tenga influencia en los valores. A su vez los tres clasificadores obtienen unos resultados que crecen en la misma proporción, por tanto se podría intentar combinarlos de alguna forma para construir el clasificador final. Ante la evidencia de que tamaños más grandes en el modelo BoW puedan producir mejores resultados, seguimos probando distintas combinaciones esta vez con la variable min_df reducida a 1 o 3, ya que no se han mostrado evidencias de que sea un parámetro influyente.

Con respecto a los tiempos de ejecución, recordemos que son tiempos que surgen de la evaluación exhaustiva de los clasificadores con distintos parámetros, buscando optimizar los mismos. Esto quiere decir, que una vez seleccionemos los parámetros definitivos, los tiempos de construcción en la tarea de modelado se reducirán. En general SVM y

		Clasificador								
Extr.Caract: BoW		**SVM**			**Random Forest**			**Reg.Logística**		
Tamaño	**MinDF**	**C**	**acc**	**F1**	**N_Est**	**acc**	**F1**	**C**	**acc**	**F1**
1500	1	1	0.85	0.85	1000	0.85	0.84	10	0.85	0.85
	3	1	0.85	0.85	500	0.85	0.85	10	0.85	0.85
2000	1	1	0.85	0.86	1000	0.85	0.85	10	0.87	0.86
	3	1	0.86	0.86	500	0.85	0.85	10	0.86	0.86
2500	1	1	0.86	0.87	1000	0.85	0.85	10	0.87	0.87
	3	1	0.87	0.87	1000	0.85	0.85	10	0.87	0.87

Cuadro 3.27: *Selección de parámetros con BoW (TF-IDF) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

especialmente Random Forest, son con diferencia los clasificadores que mas esfuerzo han requerido. Los tiempos de construcción tanto de BoW, como de el modelo W2V y el clasificador de Regresión Logística son despreciables en comparación.

En los nuevos resultados con BoW, se ha fijado para SVM el inverso del parámetro de regularización en 1, ya que ha sido el valor que ha resultado mejor en todas las comprobaciones sin ser un valor límite; y para regresión logística, comprobaremos aún este parámetro con valores mayores de 10, pero los mejores resultados se obtienen con C=10. Los resultados obtenidos se muestran en la tabla 3.27 y sus tiempos de ejecución en 3.28

Con respecto a W2V, vemos que el tamaño del modelo no parece ser demasiado relevante y sin embargo la ventana parece influir mucho obteniéndose mejores valores cuanto más grande es. Con respecto a los algoritmos de clasificación, Random Forest no parece demasiado claro que tenga un número de estimadores idóneo. Para SVM y Regresión Logística, le habíamos dado una parrilla de valores para el inverso del parámetro de regularización, donde el valor máximo era 10 y es el que mejores resultados ha producido en cada caso. Con este análisis hecho, vamos a comprobar finalmente para diferentes tamaños del modelo si ampliando incluso más la ventana y el inverso del parámetro de regularización se obtienen mejores resultados. Los resultados los vemos en la tabla 3.29 y sus tiempos de ejecución en 3.30

Tras esta segunda tarea de selección de parámetros, comprobamos que utilizando Word2Vec (SkipGram) como extracción de características, Random Forest es claramente el clasificador que peor rendimiento tiene. SVM y Regresión Logística producen valores de 85-86 % en su combinación óptima, que se produce cuando el tamaño del modelo Word2Vec es 800 y se utiliza "softmax jerárquico" en el entrenamiento, sin embargo, esto sucede cuando le damos valores muy grandes del inverso del parámetro de regularización a ambos clasificadores, que nos pueden estar haciendo caer en sobreajuste. Este parámetro permite márgenes de error elevados para alcanzar mayores tasas en la clasificación con el conjunto de entrenamiento. Además dado que estas labores no se están realizando con una gran máquina computacionalmente hablando, los tiempos de construcción de ambos clasificadores con valores de este parámetro más grande incluso se hacen inmanejables.

Extr.Caract: BoW			Clasificador		
Tamaño	**MinDF**	**Tiempo**	**SVM**	**Random Forest**	**Reg.Logística**
1500	1	0.36 seg	282.14 seg	2791.02 seg	0.65 seg
	3	0.37 seg	457.09 seg	2383.69 seg	0.64 seg
2000	1	0.35 seg	620.3 seg	2987.36 seg	0.88 seg
	3	0.38 seg	358.75 seg	3033.83 seg	0.8 seg
2500	1	0.45 seg	469.08 seg	3733.38 seg	0.96 seg
	3	0.4 seg	502.92 seg	3826.37 seg	1.04 seg

Cuadro 3.28: *Tiempos de ejecución selección de parámetros con BoW (TF-IDF) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

		Clasificadores								
Extrac. Caract: W2V (SkipGram)		**SVM**			**Random Forest**			**Reg.Logística**		
Tamaño	**Ventana**	**C**	**F1 (ns)**	**F1 (hs)**	**NºEst**	**F1 (ns)**	**F1 (hs)**	**C**	**F1 (ns)**	**F1 (hs)**
200	15	10000	0.80	0.80	400	0.78	0.8	10000	0.81	0.81
	20	10000	0.80	0.80	200	0.79	0.8	1000000	0.80	0.8
	25	10000	0.80	0.80	100	0.79	0.8	10000	0.81	0.81
400	15	10000	0.83	0.83	200	0.78	0.8	1000000	0.83	0.84
	20	10000	0.83	0.83	200	0.78	0.8	100000	0.83	0.84
	25	10000	0.83	0.83	400	0.79	0.8	100000	0.83	0.83
600	15	10000	0.84	0.84	100	0.78	0.8	1000000	0.84	0.85
	20	10000	0.84	0.85	100	0.79	0.8	100000	0.83	0.85
	25	10000	0.84	0.84	400	0.78	0.8	100000	0.84	0.85
800	15	10000	0.84	0.86	300	0.78	0.8	100000	0.85	0.86
	20	10000	0.84	0.85	400	0.79	0.81	100000	0.85	0.85
	25	10000	0.84	0.85	400	0.79	0.81	100000	0.85	0.86

Cuadro 3.29: *Selección de parámetros con W2V (Skip Gram) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

Extr.Caract: Word2Vec				Clasificador		
Tamaño	**Ventana**	**Tiempo**	**C.Medias**	**SVM**	**RandomForest**	**Reg.Log**
200	15	4.29 seg	0.61 seg	1687.23 seg	584.36 seg	248.91 seg
	20	5.28 seg	0.63 seg	1722.89 seg	372.4 seg	199.79 seg
	25	7.08 seg	0.65 seg	1992.12 seg	355.28 seg	221.2 seg
400	15	9.53 seg	0.93 seg	2239.21 seg	532.64 seg	596.5 seg
	20	11.78 seg	0.78 seg	2447.72 seg	523.11 seg	708.13 seg
	25	10.73 seg	0.79 seg	2504.94 seg	635.1 seg	805.73 seg
600	15	14.49 seg	0.94 seg	1967.98 seg	803.59 seg	1103.37 seg
	20	17.2 seg	0.98 seg	2441.16 seg	725.65 seg	1062.76 seg
	25	18.42 seg	0.98 seg	2847.48 seg	728.16 seg	1196.5 seg
800	15	15.87 seg	0.84 seg	2543.05 seg	795.12 seg	2437.87 seg
	20	24.42 seg	0.84 seg	3579.02 seg	745.77 seg	1804.82 seg
	25	25.81 seg	0.84 seg	3843.2 seg	779.2 seg	2700.26 seg

Cuadro 3.30: *Tiempos de ejecución selección de parámetros con Word2Vec (Skip Gram) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

Por otro lado, utilizando BoW, alcanzamos unos valores de 85-87 % con cualquiera de los tres clasificadores tanto en la tasa de acierto como en la métrica F1 ponderada. Primero comprobaremos si parece que estemos cometiendo sobreajuste utilizando W2V para la extracción de características con el conjunto de prueba reservado para comprobar el rendimiento de nuestro sistema. Si este es el caso, nos decantaremos por BoW como técnica para la extracción de características, ya que no sospechamos que se produzca sobreajuste y además proporciona unos valores mínimamente mejores para la métrica de validación F1 ponderada con la que estamos trabajando. A continuación se describe el clasificador final que se construirá.

Clasificador final

Como acabamos de comentar, utilizando BoW con tamaño 2500 en la fase de extracción de características, conseguimos un valor de la métrica F1 de 85-87 % con el conjunto de entrenamiento en los tres algoritmos probados. Podemos probar el funcionamiento de aplicar un clasificador final que se base en un sistema de votación de mayoría puro entre los tres clasificadores probados, con el conjunto de validación que hemos separado desde el principio.

Los clasificadores utilizados en base a los resultados obtenidos en la fase de evaluación con selección de parámetros son:

- Support Vector Machine con kernel linear y parámetro de regularización (C=1.0).
- Random Forest con tamaño (número de estimadores) 1000

- Regresión Logística con parámetro de regularización (C=10.0).
- Sistema conjunto que aplica un sistema de votación por mayoría con los 3 clasificadores

Además vamos a probar utilizando W2V, en su versión Skip Gram, con tamaño del modelo 800 y ventana 15, y los tres clasificadores junto con un sistema de votación de mayoría como el que acabamos de mencionar. En este caso los clasificadores específicos son:

- Support Vector Machine con kernel linear y parámetro de regularización (C=10000.0).
- Random Forest con tamaño (número de estimadores) 800
- Regresión Logística con parámetro de regularización (C=100000.0).
- Sistema conjunto que aplica un sistema de votación por mayoría con los 3 clasificadores

Capítulo 4. Resultados y Discusión

4.1 Resultados

Para comprobar el funcionamiento del sistema construido, vamos a diseñar dos experimentos distintos para los que observaremos diferentes métricas de las que sacar las conclusiones sobre que modelos proporcionan los mejores resultados. Los diseños son:

- **HoldOut estratificado 80/20**: Desde el primer momento, hemos reservado el conjunto de datos de prueba que no se utilizaría en la selección de modelos y sobre el que se probaría el funcionamiento. El conjunto de entrenamiento ha contado con **8458** *tweets* y el conjunto de prueba con **2115** *tweets*. Ambos conjuntos están estratificados según la clase, es decir, sobre su correspondencia a sentimiento "positivo" y "negativo".
- **Validación cruzada estratificada con K=10 folds**: La validación cruzada palía el efecto de la partición aleatoria creada en un experimento HoldOut. Se divide el conjunto de datos en 10 folds, de los cuales, 7 contienen **1057** *tweets* y 8 1058 *tweets*. De nuevo, cada fold está estratificado por la etiqueta de la clase. En cada experimento se utilizan 9 de los 10 subconjuntos para entrenamiento y el restante para validación.

Las métricas de validación utilizadas que ya han sido introducidas anteriormente son:

- **Accuracy**: Mide el porcentaje de casos que el modelo ha acertado. Cuando las clases no están balanceadas, se recomienda no utilizar esta métrica ya que si nuestra base hubiese tenido 90 % de positivos y nuestro modelo clasificara todo como positivos, estaríamos acertando un 0 % de los negativos. Se propone utilizar mejor las siguientes métricas.
- **"Precission" Ponderada**: Mide la habilidad de un clasificador de no predecir como positivo una muestra que sea negativa y viceversa. Se calcula la métrica para cada clase y luego se realiza la media ponderada.
- **"Recall" Ponderada**: Mide la habilidad del clasificador de clasificar correctamente todas las muestra positivas y lo mismo para las negativas. Se calcula la métrica para cada clase y luego se realiza la media ponderada.

- **Medida F1 Ponderada**: Es la media ponderada de "precission" y "recall", donde ambas métricas contribuyen de igual forma. Esta métrica es la más deseable para comparar modelos.

Bag of Words

La tabla 4.1 corresponde a los resultados obtenidos con el diseño HoldOut en las métricas de validación planteadas.

	Accuracy	**Precission**	**Recall**	**F1-Weighted**
SVM	87 %	88 %	87 %	87 %
Random Forest	85 %	85 %	85 %	85 %
Reg.Logística	87 %	88 %	87 %	88 %
Voto por Mayoría	88 %	88 %	88 %	88 %

Cuadro 4.1: *Validación del clasificador final con BoW con diseño HoldOut estratificado 80/20*

En la fase de selección de parámetros habíamos conseguido unos valores para la métrica F1 ponderada con el conjunto de entrenamiento de 85-87 %, por tanto, podemos dar por buenos los resultados que conseguimos en este experimento, y por ende, que la construcción del sistema es correcta.

La tabla 4.2 contiene los resultados del diseño de validación cruzada. El tiempo de ejecución que ha llevado este experimento ha sido de **6838 seg**, del cual la construcción de BoW es despreciable, así como el tiempo de ajuste del regresor logístico.

	Accuracy	**Precission**	**Recall**	**F1-Weighted**
SVM	87.34 %	87.74 %	87.34 %	87.45 %
Random Forest	85.72 %	86.18 %	85.72 %	85.85 %
Reg.Logística	87.50 %	87.70 %	87.50 %	87.56 %
Voto por Mayoría	87.92 %	88.26 %	87.91 %	88.01 %

Cuadro 4.2: *Validación del clasificador final con BoW con diseño de validación cruzada estratificada de 10 folds*

En ambos diseños se observa claramente que utilizar Random Forest como clasificador proporciona unos resultados ligeramente peores. Además, el sistema de voto por mayoría, no tiene la mejoría esperada. Si bien en ambos casos presenta los mejores resultados, no son significativamente mejores que utilizar SVM o Regresión Logística por individual, esto nos hace sospechar que ambas técnicas estén funcionando de forma similar y este es el motivo que lleva a que los resultados sean prácticamente iguales. De todas formas, si nos basamos únicamente en los mejores resultados, tendríamos que hablar de que los proporciona el sistema de voto por mayoría.

Word2Vec - Skip Gram

Repetimos las mismas pruebas con diseño HoldOut con el conjunto de validación reservado primero. Los resultados se muestran en la tabla 4.3

	Accuracy	Precission	Recall	F1-Weighted
SVM	86 %	86 %	86 %	86 %
Random Forest	81 %	83 %	81 %	81 %
Reg.Logística	80 %	81 %	80 %	80 %
Voto por Mayoría	84 %	85 %	84 %	84 %

Cuadro 4.3: *Validación del clasificador final con Word2Vec con diseño HoldOut estratificado 80/20*

Observamos claramente unos peores resultados en comparación con el sistema anterior. Para la Regresión Logística se ve claramente como hemos sobreajustado, ya que se ha reducido un 5 % la métrica F1 en comparación con los valores resultantes de trabajar solo con el conjunto de entrenamiento. Además, Random Forest claramente produce los peores resultados y el sistema de votación no llega a mejorar las tasas de SVM, donde sospechábamos que podríamos estar cometiendo sobreajuste. Igualmente SVM no ha conseguido producir mejores resultados con Word2Vec como con Bag of Words.

Vamos a probar por último el diseño de validación cruzada estratificada de 10 folds con el conjunto de datos entero para definitivamente desestimar el uso de Word2Vec en nuestro sistema de clasificación. Los resultados obtenidos tras más de 5 horas de ejecución se muestran en la tabla 4.4.

	Accuracy	Precission	Recall	F1-Weighted
SVM	85.80 %	85.94 %	85.80 %	85.84 %
Random Forest	80.88 %	82.16 %	80.88 %	81.23 %
Reg.Logística	85.87 %	86.04 %	85.87 %	85.92 %
Voto por Mayoría	86.14 %	86.36 %	86.14 %	86.21 %

Cuadro 4.4: *Validación del clasificador final con Word2Vec con diseño de validación cruzada estratificada de 10 folds*

En este caso, ya hemos conseguido unas métricas del 86 % utilizando el sistema de voto por mayoría. Sin embargo, los resultados son peores si los comparamos con sus homólogos habiendo utilizado BoW para la extracción de características.

4.2 Discusión

Una vez hemos concluido el trabajo, debemos mirar atrás para analizar los pasos y decisiones tomadas a lo largo de la construcción de nuestro sistema de clasificación de sentimientos en base a su polaridad negativa o positiva.

Primero, contábamos con 15000 *tweets* sin etiquetar extraídos de **Twitter** utilizando su API. Después se les realiza una tarea de preprocesado incluyendo el proceso de lematización, de modo que cada *tweet* se convierte en una lista de elementos individuales (palabras, lemas, emojis...). Una vez tenemos los datos de esta manera, se construye un sistema de etiquetado que poder aplicar a los *tweets*, de manera que la clasificación que esta de, se considere real como para seguir procediendo con las siguientes fases.

El sistema de etiquetado consta de cuatro subsistemas que clasifican un *tweet* como "positivo", "negativo" y en ausencia de motivos para clasificar como alguna de estas dos clases, se etiquetará como "neutro". Finalmente, para cada *tweet*, se obtiene la etiqueta global calculando la **moda** de las etiquetas de los subsistemas, y en caso de existir un empate queda definida también la forma de proceder.

Una vez hemos clasificado nuestro conjunto de datos hemos obtenido: **6493** *tweets* "positivos", **4427** "neutros" y **4080** "negativos". Como el sistema que se quiere construir pretende detectar la polaridad de un texto, no vamos a considerar los *tweets* "neutros" de aquí en adelante, y por tanto, el conjunto de datos final con el que se trabaja consta de **10573** *tweets*.

A continuación, se ha realizado una tarea extensa de selección de parámetros, comprobando el funcionamiento tanto de las técnicas de la fase de extracción de características (BoW y Word Embeddings), como de clasificadores ampliamente utilizados en el procesamiento de lenguaje natural (SVM, Random Forest y Regresión Logística). Los resultados nos hacen ver que BoW funciona mejor que Word2Vec, y los tres clasificadores parecen funcionar de forma similar, no obstante, en la validación final se mantienen las pruebas con ambas técnicas de extracción para comprobar si estamos en lo cierto.

Comprobamos el rendimiento del sistema construido mediante dos diseños experimentales: primero con un HoldOut estratificado 80/20 con el conjunto de validación que habíamos reservado para observar el funcionamiento global del sistema construido; y posteriormente, con una validación cruzada estratificada de 10 folds con todos los datos. Los resultados finales nos dicen que hemos conseguido unas métricas de casi un 90 % utilizando BoW en la extracción de características, y que el sistema de voto por mayoría, mejora ligeramente los resultados que individualmente conseguimos con SVM y Regresión Logística. Quizás como la mejora es tan ligera, se podría llegar a tomar la decisión de construir el sistema únicamente con SVM por razones de potencia computacional.

Los resultados obtenidos se consideran muy aceptables teniendo en cuenta las limitaciones del trabajo: primero, por el hecho de que se trabaja con un conjunto de datos no preparado, ya que ha sido extraído manualmente con lo que esto supone, y por tanto, ha sido necesaria la elaboración de un sistema de etiquetado con el que establecemos una "ground truth", y segundo, porque el trabajo de modelado y selección de parámetros, está siendo llevado a cabo con una máquina que no es especialmente potente en lo que a cómputo se refiere.

Capítulo 5. Conclusiones y Trabajo Futuro

5.1 Conclusiones

En este trabajo, se ha realizado una labor de acercamiento al campo del procesamiento de lenguaje natural, y más específicamente, al análisis de sentimientos en redes sociales analizando los trabajos realizados en este campo en los últimos años. En este primera tarea, ya nos hemos cerciorado de que existe la posibilidad real de construir un sistema propio de reconocimiento de sentimientos de un nivel de calidad propio de un proyecto importante como es un Trabajo de Fin de Grado.

Durante el desarrollo del TFG, se ha pretendido explicar con detalle las distintas fases de la manera más ilustrativa y didáctica posible, para conseguir que el lector que sea nuevo en la materia, pueda comprender con más sencillez los procedimientos utilizados.

En la tarea de la programación, prácticamente la totalidad del trabajo ha sido elaborado con Python. Es un lenguaje extremadamente versátil, donde se han utilizado librerías potentes en las distintas etapas del proceso, destacando la utilización de: *nltk*, para el procesamiento del lenguaje, *tweepy*, para la extracción de datos de **Twitter**, hasta librerías más comúnmente utilizadas como *sklearn*, *matplotlib* o *pandas*.

Con respecto al sistema experimental elaborado, se han encontrado impedimentos como la ausencia de un conjunto de datos inicial ideal. Esto nos ha llevado a tomar la decisión de construir un sistema de etiquetado propio basado en léxicos, cuyos resultados, se han utilizado como "ground truth" para la posterior tarea de entrenamiento de modelos. Los léxicos utilizados están elaborados manualmente por organismos y profesionales en la materia, por lo que los resultados procedentes del sistema de etiquetado construido se consideran una opción acertada y de calidad con la que trabajar.

El sistema construido que mejor funciona es el que utiliza **Bag of Words** como técnica de extracción, y clasifica en función de un sistema de voto por mayoría pura donde intervienen tres de los clasificadores más utilizados en este campo: **SVM**, **Random Forest** y **Regresión Logística**. Para dicho sistema, se han alcanzado resultados de un 88 % en la medida F1 ponderada y en la tasa de acierto, que nos hacen reforzar la idea de

que se ha completado un buen trabajo. No obstante sería ideal para un trabajo futuro, poder contar con conjuntos de datos ya etiquetados manualmente con los que poder comprobar el funcionamiento real del prototipo desarrollado.

La realización de este trabajo, nos ha permitido aplicar y profundizar técnicas aprendidas a lo largo de la carrera como las correspondientes a asignaturas como: *Técnicas de Aprendizaje Automático*, *Gramáticas y lenguajes formales*, *Minería de Datos* o *Planificación y Diseño de Sistemas Computacionales*. Además el estudio simultáneo del Grado en Estadística, ha sido de gran ayuda gracias al enfoque que se realiza en las tareas de análisis de datos.

Finalmente, dadas las restricciones temporales para la elaboración del proyecto y el carácter introductorio del mismo, existen posibles ampliaciones del trabajo que quedan planteadas a continuación. Estas posibles vías de ampliación, contemplan tanto la mejora de los resultados conseguidos, como el aumento de la certeza con la que podemos catalogar nuestro sistema construido en un primer acercamiento, como un sistema competitivo.

5.2 Trabajo Futuro

El trabajo desarrollado, se ha centrado en un tipo de comunicación (texto) y en una red social específica (**Twitter**). Existen etapas del proceso que están destinadas específicamente a ello, como es el preprocesamiento de los datos. Sin embargo, modificando esta fase, ya podríamos estar hablando de una posible forma de extrapolar el análisis a otras formas de comunicación textual, como podría ser, textos de opinión, libros, artículos de crítica, entradas en blogs, cartas, etc. Del mismo modo nos hemos centrado en la red social más importante de compartición de microtextos, que es **Twitter**, pero se podría analizar otras redes sociales donde la comunicación vía texto cobra gran importancia, como puede ser Facebook o Telegram.

Con respecto al idioma de los datos analizados, se ha tomado la decisión de trabajar con textos en castellano aún sabiendo los impedimentos que esto conllevaba, como por ejemplo, la falta de conjuntos de datos etiquetados manualmente, que nos hubieran permitido destinar un mayor esfuerzo a la parte de modelado y no a la elaboración de un sistema de etiquetado cuyas clasificaciones se traten como reales para realizar el análisis. Sería interesante ampliar el trabajo hacia otros idiomas tanto para poder probar la influencia que la lengua tiene en los análisis, como para poder probar técnicas de traducción o transferencia léxica.

Ya hemos comentado las limitaciones en la potencia computacional de la máquina donde se han realizado las labores de recolección de datos y construcción de los modelos de clasificación utilizados. Sería idóneo, poder extender el número de modelos probados, y ampliar el conjunto de datos de trabajo para mejorar el rendimiento del sistema experimental construido. Se propone además, el estudio de técnicas semisupervisadas que partan de pequeños conjuntos de datos etiquetados manualmente, dadas las dificultades

anteriormente mencionadas, para extenderlos luego. De esta manera podríamos confirmar la calidad del sistema de etiquetado construido en primera instancia, así como mejorar el mismo gracias al etiquetado manual que se aplica.

Bibliografía

[1] Wang D, Szymanski B.K, Abdelzaher T, Heng Ji, Kaplan L (2018). *The age of social sensing.* IEEE Computer Society.

[2] Sailunaz K, Dhaliwal M, Rokne J, Alhajj R (2018). *Emotion detection from text and speech: a survey.* Social Network Analysis and Mining.

[3] Lévy P (2013). *The Semantic Sphere 1: Computation, Cognition and Information Economy.* Hoboken, NJ:Wiley.

[4] Watzlawick P, Helmick Beavin J, D.Jackson D. (1985) *Teoría de la Comunicación Humana.*

[5] C.D.Broad (1954). *Emotion and sentiment.* Journal of Aesthetics and Art Criticism 13(2):203-214.

[6] Lövheim H. (2011). *A new three-dimensional model for emotions and monoamine neurotransmitters.* Medical hypotheses. 78. 341-348.

[7] Shaver P, Schwartz J, Kirson D, O'Connor C (1987). *Emotion knowledge: Further exploration of a prototype approach.* Journal of Personality and Social Psychology, 52(6), 1061–1086.

[8] Ekman P (1992) *An argument for basic emotions.* Cognition and Emotion 6(3-4):169-200.

[9] Plutchik R (1980). *Emotion: a psychoevolutionary synthesis.* Harper and Row, New York.

[10] Pak A, Paroubek P (2010). *Twitter as a Corpus for Sentiment Analysis and Opinion Mining.* LREC.

[11] Dini L, Bittar A (2016). *Emotion Analysis on Twitter: The Hidden Challenge.* LREC.

[12] Mohammad S.M, Bravo-Marquez F (2017). *Emotion intensities in tweets.* Proceedings of the sixth Joint Conference on Lexical and Computational Semantics.

[13] Shiha M.O, Ayvaz S (2017). *The Effects of Emoji in Sentiment Analysis.* International Journal of Computer and Electrical Engineering vol. 9, no. 1, pp. 360-369.

[14] Satapathy R, Guerreiro C, Chaturvedi I, Cambria E (2017). *Phonetic-Based Microtext Normalization for Twitter Sentiment Analysis.* 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 407-413.

[15] Wang B, Wang A, Chen F, Wang Y, Kuo C. (2019). *Evaluating word embedding models: Methods and experimental results.* APSIPA Transactions on Signal and Information Processing, 8, E19. doi:10.1017/ATSIP.2019.12.

[16] Jain V.K, Kumar S, Fernandes S.L (2017). *Extraction of emotions from multilingual text using intelligent text processing and computational linguistics.* J. Comput. Sci., 21, 316-326.

[17] Kang X, Ren F, Wu Y. (2018) *Exploring latent semantic information for textual emotion recognition in blog articles.* in IEEE/CAA Journal of Automatica Sinica, vol. 5, no. 1, pp. 204-216.

[18] Yat Mei Lee S, Chen Y, Huang C. (2010). *A text-driven rule-based system for emotion cause detection.* In Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text (CAAGET '10). Association for Computational Linguistics, USA, 45–53.

[19] Chen Y, Skiena S. (2014). *Building Sentiment Lexicons for All Major Languages.* In ACL (2) (pp. 383-389).

[20] Kralj Novak P, Smailovic J, Sluban B, Mozetic I (2015). *Sentiment of Emojis*, PLoS ONE 10(12).

[21] Cruz F. L, Troyano J. A, Pontes B, Ortega F. J. (2014) *ML-SentiCon: A multilingual, lemma-level sentiment lexicon.* Expert Systems with Applications, vol. 41, nº 13, pp. 5984-5994.

[22] Cruz F.L, Troyano J.A, Pontes B, Ortega F.J. (2014) *Building layered, multilingual sentiment lexicons at synset and lemma levels, Expert Systems with Applications.*

[23] Molina González M.D, Martínez Cámara E., Martín Valdivia M.T. (2015). *CRiSOL: Opinion Knowledge-base for Spanish.* Procesamiento Del Lenguaje Natural, 55, 143-150.

[24] Hu M., Liu B. (2004) *Mining and summarizing customer reviews.* Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery, Data Mining (KDD-2004, full paper), Seattle, Washington, USA, Aug 22-25, 2004.

[25] Ghelani S. (2019) *From Word Embeddings to Pretrained Language Models - A new Age in NLP - Part 1*, Towars Data Science.

[26] Preisendorfer M. (2018). *Social Media Emoji Analysis, Correlations and Trust Modeling.* 10.13140/RG.2.2.25466.18888.

[27] Ljubesic N, Fiser D. (2016). *A Global Analysis of Emoji Usage.* WAC@ACL.

[28] REAL ACADEMIA ESPAÑOLA: Banco de datos (CORDE). *Corpus diacrónico del español.* <http://www.rae.es> 12/06/2020.

[29] Gupta V, Karnick H, Bansal A, Jhala P. (2016) *Product Classification in E-Commerce using Distributional Semantics.*

[30] Gandhi R. (2018). *Support Vector Machine - Introduction to Machine Learning Algorithms*, Towards Data Science.

[31] Matesanz M. (2018). *Análisis de sentimiento en Twitter con herramientas de Big Data.*

Apéndice A. Selección de parámetros con Bag of Words

En la tabla A.1, se muestran los resultados finales completos de la selección de parámetros con BoW como extracción de características.

La recolección de los valores de la tabla para los diferentes parámetros, ha sido la tarea que más esfuerzo computacional ha llevado. Esto se debe al uso de la función GridSearchCV(), ya introducida en el documento, que realiza una búsqueda exhaustiva entre los parámetros que recibe como argumento. Se ha combinado la búsqueda de los clasificadores óptimos con la búsqueda de los parámetros óptimos en el modelo Bag of Words de extracción de características.

		Clasificador					
Extr.Caract: BoW		**SVM**		**Random Forest**		**Reg.Logística**	
Tamaño	**MinDF**	**C**	**F1**	**N_Est**	**F1**	**C**	**F1**
5	1	0.1	0.47	500	0.47	0.1	0.47
	3	0.1	0.47	1000	0.47	0.1	0.47
	5	0.1	0.47	1000	0.47	0.1	0.47
25	1	100	0.59	500	0.58	100	0.58
	3	100	0.59	500	0.59	100	0.58
	5	100	0.59	500	0.58	100	0.58
50	1	1	0.65	1000	0.67	100	0.66
	3	1	0.65	1000	0.66	100	0.66
	5	1	0.65	500	0.67	100	0.66
100	1	1	0.71	500	0.73	100	0.72
	3	1	0.71	1000	0.73	100	0.72
	5	1	0.71	1000	0.73	100	0.72
	10	1	0.71	500	0.73	100	0.72
250	1	1	0.77	1000	0.78	10	0.77
	3	1	0.77	300	0.78	10	0.77
	5	1	0.77	1000	0.78	10	0.77
	10	1	0.77	300	0.78	10	0.77
500	1	1	0.81	1000	0.81	1	0.81
	3	1	0.81	1000	0.81	1	0.81
	5	1	0.81	500	0.81	1	0.81
	10	1	0.81	300	0.81	10	0.81
1000	1	1	0.84	300	0.84	10	0.84
	3	1	0.84	1000	0.84	10	0.84
	5	1	0.84	500	0.84	10	0.84
	10	1	0.84	500	0.84	10	0.84
1500	1	1	0.85	1000	0.84	10	0.85
	3	1	0.85	500	0.85	10	0.85
2000	1	1	0.86	1000	0.85	10	0.86
	3	1	0.86	500	0.85	10	0.86
2500	1	1	0.87	1000	0.85	10	0.87
	3	1	0.87	1000	0.85	10	0.87

Cuadro A.1: *Selección de parámetros con BoW (TF-IDF) como extracción de características*

Apéndice B. Selección de parámetros con Word2Vec (Skip-Gram)

En la tabla B.1, se muestran los resultados finales completos de la selección de parámetros con Word2Vec con la arquitectura Skip Gram como extracción de características y softmax jerárquico para el entrenamiento de la red neuronal.

La recolección de los valores de la tabla para los diferentes parámetros, ha sido la tarea que más esfuerzo computacional ha llevado. Esto se debe al uso de la función GridSearchCV(), y introducida en el documento, que realiza una búsqueda exhaustiva entre los parámetros que recibe como argumento. Se ha combinado la búsqueda de los clasificadores óptimos con la búsqueda de los parámetros óptimos en el modelo de extracción de características Word2Vec en su variante de arquitectura Skip Gram.

Extr.Caract: W2V (Skip Gram)		Clasificador					
		SVM		Random Forest		Reg.Logística	
Tamaño	**Ventana**	**C**	**F1**	**N_Est**	**F1**	**C**	**F1**
5	5	0.1	0.47	300	0.68	10	0.49
	10	10	0.54	300	0.71	10	0.62
	15	1	0.64	400	0.71	10	0.67
25	5	10	0.72	300	0.72	10	0.72
	10	10	0.72	400	0.77	10	0.72
	15	1	0.72	200	0.77	10	0.72
50	5	10	0.73	200	0.76	10	0.72
	10	10	0.74	100	0.78	10	0.74
	15	10	0.75	200	0.78	10	0.75
100	5	10	0.74	400	0.76	10	0.74
	10	10	0.76	400	0.78	10	0.75
	15	10	0.76	400	0.78	10	0.76
200	5	10	0.79	100	0.80	10	0.78
	10	10	0.79	300	0.80	10	0.78
	15	10	0.80	300	0.80	10	0.80
	20	10000	0.80	200	0.80	1000000	0.80
	25	10000	0.80	100	0.80	10000	0.81
400	5	10	0.78	200	0.80	10	0.78
	10	10	0.79	200	0.80	10	0.79
	15	10	0.81	200	0.80	10000	0.83
	20	10000	0.83	200	0.80	100000	0.84
	25	10000	0.83	400	0.80	100000	0.83
600	5	10	0.78	100	0.76	10	0.78
	10	10	0.80	300	0.80	10	0.79
	15	10000	0.84	200	0.80	100000	0.85
	20	10000	0.85	100	0.80	100000	0.85
	25	10000	0.84	400	0.80	100000	0.85
800	5	10	0.78	200	0.80	10	0.78
	10	10	0.80	300	0.80	10	0.79
	15	10000	0.86	200	0.80	100000	0.86
	20	10000	0.85	400	0.81	100000	0.85
	25	10000	0.85	400	0.81	100000	0.86

Cuadro B.1: *Selección de parámetros con Word2Vec (SkipGram) como extracción de características*

Apéndice C. Estructura de archivos del proyecto y puesta en marcha del mismo

En este apéndice, se pretende explicar paso por paso las órdenes que habría que ejecutar para obtener los resultados, analíticas y gráficos mostrados en el trabajo de forma que el lector pueda replicar el proceso o realizar los cambios pertinentes para utilizar este trabajo como base para una posible ampliación.

Para una mejor explicación, vamos a suponer que se desea rehacer el proceso completo en todas sus fases con nuevos datos extraídos.

C.C.1 Requisitos y preparación del entorno

Es necesario contar con Python versión 3 instalado, y si se está trabajando en una máquina Windows, será necesario tenerlo como variable del sistema para poder ejecutar Python en cualquier momento desde la CMD de Windows.

El proyecto entero está en mi repositorio en GitHub directo para ser clonado y poder trabajar con él. Para poder acceder a el tendremos que tener instalado Git en nuestra máquina y tendremos que estar logueados correctamente en el mismo.

Una vez tenemos estas dos cosas, en la carpeta deseada, tendremos que ejecutar el comando para clonar el proyecto.

```
$ git clone https://github.com/AlbCalv/sentiment-analysis
```

Se ha utilizado la librería pipreqs para generar las dependencias de librerías necesarias para el funcionamiento del proyecto. Estas se encuentran en el archivo **.requirements.txt**. Para instalar las librerías basta con ejecutar la siguiente orden:

```
$ pip install -r requirements.txt
```

C.C.2 Estructura de directorios

La estructura de directorios en que se organizan los ficheros de datos y programas del proyecto, es la que se muestra en la Figura C.1. El directorio **./data** contiene todo lo referido al trabajo realizado. Se podría replicar partiendo del archivo de datos **./data/tweet15.csv**. El directorio **./datos_nuevos** está destinado a la ejecución de todas las etapas con un archivo de datos nuevo que sería el archivo **./datos_nuevos/tweetXXXX.csv** como se va a detallar a continuación.

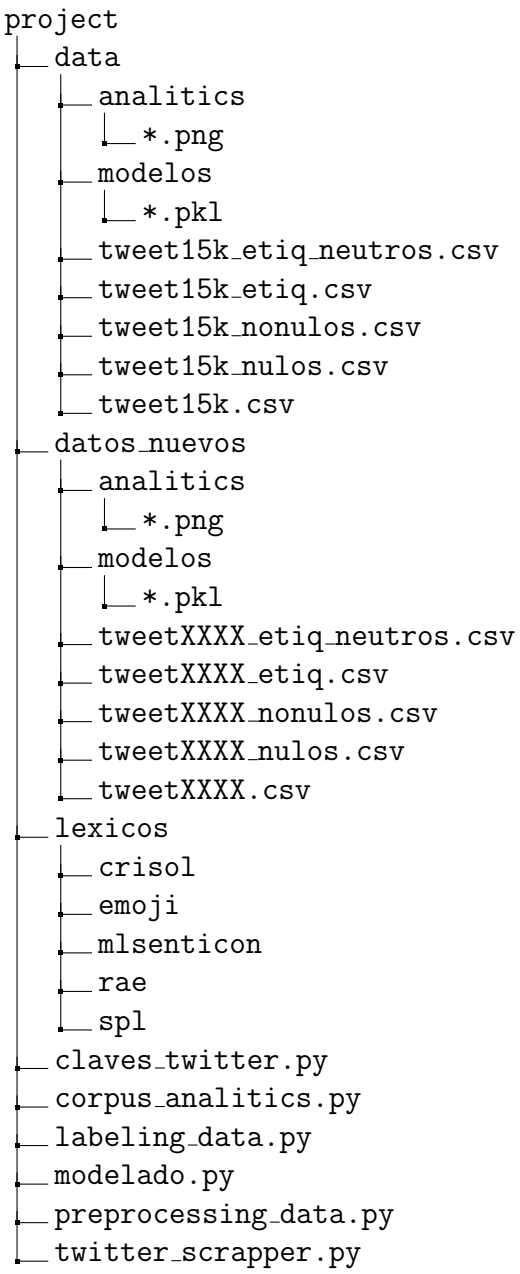


Figura C.1: *Estructura de directorios de los archivos del proyecto.*

C.C.3 Extracción de los datos de Twitter

Para extraer nuevos datos basta con ejecutar el archivo **./twitter_scrapper.py** con una serie de anotaciones anteriores. Para realizar una extracción se debe haber creado una cuenta de desarrollador de **Twitter** como hemos comentado ya. Con dicha cuenta, extraeremos las claves necesarias para acceder a la API de **Twitter** y estas se deben situar correctamente en un archivo que se llame **claves_twitter.py** que irá situado en el directorio principal como se indica en la estructura. Dicho archivo debe tener el formato de la imagen C.2 para poder acceder correctamente a la API.

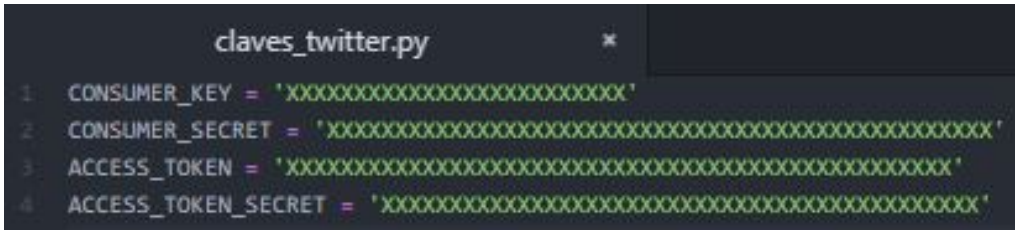


Figura C.2: *Formato del archivo claves_twitter.py*

Podemos ejecutar el programa de dos formas diferentes: se le puede pasar un argumento por línea de comandos que se corresponde con el número de *tweets* que se pretenden extraer o si no se le pasa el argumento, se extraerán por defecto 15000 *tweets*. Los siguientes comandos harían la tarea deseada.

```
$ python twitter_scrapper.py
$ python twitter_scrapper.py 4000
```

Este programa generará un único archivo que se almacenará en el directorio **./datos_nuevos** con el nombre de **./datos_nuevos/tweetsXXXX.csv** siendo XXXX el número de *tweets* indicados a extraer. El archivo cuenta con una entrada por cada *tweet* extraído, donde se muestra el contenido textual del mismo como única información.

C.C.4 Preprocesamiento de los *tweets*

Las técnicas de preprocesamiento de los *tweets* tanto para términos como para emojis, se encuentran en el archivo **./preprocessing_data.py**, que será invocado por primera vez en el siguiente paso que corresponde con el sistema de etiquetado de los *tweets*. Las técnicas utilizadas han sido ya explicadas en la sección 2.4.2.

C.C.5 Etiquetado de los datos

Para el etiquetado de los datos, se utiliza el archivo **labeling_data.py**. Deberemos indicarle como argumento el archivo que deseamos etiquetar, en este caso el que acabamos de extraer. Esto se puede conseguir con el comando:

```
$ python labeling_data.py datos_nuevos/tweetsXXXX.csv
```

Al ejecutar el programa se generarán los siguientes archivos:

- **./datos_nuevos/tweetsXXXX_etiq_neutro.csv**: Archivo separado por comas que contiene en cada línea un *tweet* y su etiqueta procedente del sistema de etiquetado (1 para positivo, 0 para neutro y -1 para negativo).
- **./datos_nuevos/tweetsXXXX_etiq.csv**: Archivo separado por comas que contiene en cada línea un *tweet* que no haya sido etiquetado como neutro, y su etiqueta procedente del sistema de etiquetado (1 para positivo y -1 para negativo).
- **./datos_nuevos/tweetsXXXX_nonulos.csv**: Archivo separado por comas que contiene los *tweets* para los que se les ha encontrado mínimo un término o emoji positivo o negativo de alguno de los léxicos utilizados.
- **./datos_nuevos/tweetsXXXX_nulos.csv**: Archivo separado por comas que contiene los *tweets* para los que no se les ha encontrado ningún término o emoji positivo o negativo de alguno de los léxicos utilizados.
- **Gráficos de barras**: En el directorio ./datos_nuevos/analitics, se generarán 8 gráficos de barras correspondientes a las frecuencias de los términos positivos y negativos encontrados para cada uno de los 4 léxicos de palabras utilizados. Un ejemplo de nombre sería **freqpossenticon.png** que se corresponde a las palabras que aparecen como positivas en el léxico senticon.

Además en la salida estándar del programa, se muestra información desglosada sobre los tiempos de ejecución, estadísticas de los términos y resultados del etiquetado que se podrían volcar a un archivo de texto con el siguiente comando:

```
$ python labeling_data.py \\
  datos_nuevos/tweetsXXXX.csv > datos_nuevos/resultados_etiquetado.txt
```

Las razones para hacer este volcado, pasan por pensar que si queremos utilizar un número muy elevado de *tweets*, estaríamos hablando de un tiempo de ejecución muy elevado, y por tanto, solo se debería ejecutar una vez.

C.C.6 Análisis del corpus

El programa **corpus_analitics.py**, realiza un análisis de la naturaleza de los *tweets* como se ha comentado en el documento. Se extrae información sobre la longitud de los *tweets*, los términos utilizados en base al CREA (Corpus de Referencia del Español Actual) y se analiza la distribución de los emojis en los *tweets*; todo esto para buscar explicaciones de por qué hay un porcentaje, pequeño, de *tweets* que se quedan etiquetados como "neutros" que en la mayoría de los casos significa "nulos".

De nuevo basta con ejecutar el programa pasándole como argumento el archivo de *tweets* extraídos.

```
$ python corpus_analitics.py datos_nuevos/tweetsXXXX.csv
```

Este programa genera multitud de gráficos que se situarán en el directorio **./datos_nuevos/analitics/**:

- **boxplot_longitudes.png**: Dos gráficos de cajas que muestran una comparativa de las longitudes de los *tweets* completos, nulos y no nulos con y sin preprocesamiento.
- **freq_emojis_tweet.png y freq_emojis_tweet_lexico.png**: Dos gráficos de barras que muestran la distribución de los *tweets* que contienen un número determinado de emojis. El primer archivo cuenta todos los emojis y el segundo solo los emojis que aparecen en el léxico de emojis utilizado.
- **comparativa_freq_emojis_tweet.png**: Gráfico que compara los dos gráficos de barras anteriores para ver como se reducen los *tweets* con un número elevado de emojis.

De nuevo, en la salida estándar, se muestra toda la información textual deseada que se recomienda volcar a un archivo de texto con el comando:

```
$ python corpus_analitics.py \\
  datos_nuevos/tweetsXXXX.csv > datos_nuevos/resultados_analisis_corpus.txt
```

C.C.7 Clasificación y validación

En el archivo **modelado.py**, se encuentran los dos diseños experimentales para validar nuestro sistema: HoldOut estratificado 80/20 y validación cruzada de 10 folds. Como se ha comentado, se está utilizando Bag of Words como técnica para extraer características y SVM, Random Forest y Regresión Logística como clasificadores junto con un sistema

de voto por mayoría implementado con estos tres algoritmos. El archivo que le tenemos que pasar como argumento ya no es el conjunto de datos inicial, sino el conjunto de datos ya etiquetado y sin los *tweets* "neutros" (**./datos_nuevos/tweetsXXXX_etiq.csv**).

Con un tamaño del conjunto de datos suficientemente grande, la validación cruzada tendrá un tiempo de ejecución muy elevado, por lo que es necesario volcar los resultados a un archivo de texto como en los pasos anteriores.

```
$ python modelado.py \\
  datos_nuevos/tweetsXXXX_etiq.csv > datos_nuevos/resultados_clasificación.txt
```

Además, se pueden extraer los clasificadores entrenados con el conjunto de entrenamiento bajo el diseño de HoldOut en archivos externos para un posible uso posterior ahorrándonos este tiempo de ejecución. Los archivos se encuentrarán en el directorio **./datos_nuevos/modelos**.

C.C.8 Resumen para datos nuevos

A continuación, se muestra el resumen del proceso explicado por el que se podría replicar el trabajo entero con un conjunto de datos nuevo que conste de 4000 *tweets*.

```
$ python twitter_scrapper.py 4000
$ python labeling_data.py \\
  datos_nuevos/tweets4000.csv > datos_nuevos/resultados_etiquetado.txt
$ python corpus_analitics.py \\
  datos_nuevos/tweets4000.csv > datos_nuevos/resultados_analisis_corpus.txt
$ python modelado.py \\
  datos_nuevos/tweets4000_etiq.csv > datos_nuevos/resultados_clasificación.txt
```

C.C.9 Resumen para datos nuevos

Si quisiéramos replicar el trabajo exacto que se ha llevado a cabo, utilizaremos el conjunto de datos **./data/tweet15k.csv** del que se derivará todo el analisis siguiente. Esto lo conseguiremos con las siguientes órdenes,

```
$ python labeling_data.py > data/resultados_etiquetado.txt
$ python corpus_analitics.py > data/resultados_analisis_corpus.txt
$ python modelado.py > data/resultados_clasificación.txt
```