



---

**Universidad de Valladolid**

**Grado en Ingeniería Informática**

**Trabajo de Fin de Grado**

**Análisis de emociones y actitudes en  
redes sociales usando ML**

PRESENTADO POR:

*Alberto Calvo Madurga*

TUTELADO POR:

*Valentín Cardenoso Payo*



# Índice general

<b>1. Introducción</b>	<b>5</b>
1.1. Motivación . . . . .	5
<b>2. Marco Teórico</b>	<b>7</b>
2.1. Emociones y comunicación . . . . .	7
2.2. Modelos de emociones . . . . .	8
2.3. Reconocimiento de emociones . . . . .	9
2.3.1. Selección y extracción de datos . . . . .	9
2.3.2. Preprocesamiento: Filtrado y acondicionamiento . . . . .	10
2.3.3. Extracción de características . . . . .	12
2.3.4. Modelado . . . . .	14
2.3.5. Diseño del experimento y Validación . . . . .	15
2.4. Las Redes Sociales y las emociones . . . . .	16
<b>3. Sistema Experimental</b>	<b>19</b>
3.1. Introducción . . . . .	19
3.2. Arquitectura del experimento . . . . .	19
3.3. Selección de datos . . . . .	20
3.4. Preprocesamiento de los datos . . . . .	22
3.5. Etiquetado de los tweets . . . . .	25
3.6. Extracción de características . . . . .	44
3.6.1. Bag of Words . . . . .	44
3.6.2. Word embeddings (Word2Vec) . . . . .	45
3.7. Modelado . . . . .	46
3.8. Selección de parámetros y Validación . . . . .	48
3.9. Discusión y Resultados . . . . .	53
<b>4. Conclusiones</b>	<b>55</b>
<b>5. Trabajo Futuro</b>	<b>57</b>



# Capítulo 1.

# Introducción

## 1.1. Motivacion

En el presente documento se aborda la problemática de la detección y análisis de sentimientos y emociones en textos. Se pretende dar un enfoque de manera que se realice una introducción al problema de la manera más detallada e ilustrativa posible donde se entienda la naturaleza del problema, se ponga en relieve la importancia de esta tarea y la gran amplitud de sectores donde su aplicación nos podría ser útil.

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 de abordamiento del 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. Hasta ahora se han utilizado conjuntos de datos con *tweets* en inglés ya que esta era la opción más sencilla para encontrar técnicas generales e información que se aplique para este lenguaje. Sin embargo 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 la idea general que transmite cada tweet y 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.

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 del construcción del clasificar se intentará incorporar algo que sea trata con clasificadores comunmente utilizados en este campo, como pueden ser SVM, 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 diseño de experimento por tanto ha sido Holdout 80/20 para separar conjunto de entrenamiento y prueba donde para la selección de parámetros dividiremos de nuevo el conjunto de entrenamiento mediante validación cruzada para optimizar su funcionamiento.

Finalmente conseguimos un sistema que clasifique un *tweet* en base a su polaridad, es decir, como *positivo* o *negativo*.

**TODO:** *Falta el comentar las conclusiones, el como se ha realizado el clasificador final y todo el tema de la aplicación*

## Capítulo 2.

## Marco Teórico

### 2.1. Emociones y comunicación

La naturaleza de los seres humanos hace que cada vez que interactuamos entre nosotros, se produzca un intercambio de pensamientos y actitudes que puede ser efectuado de forma intencionada o no. Este intercambio es una característica inherente en nuestra naturaleza y da pie a un amplio abanico de posibilidades de estudio.

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 existen múltiples formas de expresarnos vía textual, desde la realización de críticas de alguna temática elaboradas por expertos hasta entradas cortas y cargadas de opinión en alguna red social elaboradas por usuarios estándar. 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* 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 permite 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 para entender todo el trabajo, *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** [3]). 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... 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:

- **Categoricos:** Donde existen 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:** Donde 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 como se clasifican las emociones de distintas maneras y basándose en diferentes experimentos o teorías psicológicas. De esta manera se han construido múltiples modelos que se pueden estructurar de manera más o menos compleja como por ejemplo el cubo de las emociones de **Lovheim** [4] (modelo dimensional basado en las interacciones de nuestros neurotransmisores; dopamina, noradrenalina y serotonina), modelo de **Shaver** [5] (modelo categórico que establece una jerarquía en forma de árbol donde encontramos varios niveles de emociones), modelo de **Ekman** [6] (modelo categórico que defiende que el ser



humano reacciona ante los eventos mediante seis emociones básicas) o la rueda de las emociones de **Plutchik** [7] (modelo dimensional que combina ocho emociones básicas y la forma en la que se relacionan mediante la intensidad de cada una). Más modelos han sido contruidos 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. Reconocimiento de emociones

La idea del reconocimiento de emociones pasa por la construcción un sistema que pueda recibir un texto como entrada, y tras procesarlo, pueda generar como salida los sentimientos o emociones que éste pretende transmitir. Ésta 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.

### 2.3.1. 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 ser tan simple como tener una variable que indique 1 si la el sentimiento es positivo, 0 si

es neutral y -1 si es negativo, y puede ser tan complejo como para existir 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) [8] 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) [9] que crean dos corpus bajo la asunción de que cada *tweet* tiene una connotación emocional o **Mohammad M, Bravo-Marquez F** (2017) [10] 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.

### 2.3.2. 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 2.1. 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 sí se tiene en consideración el análisis morfológico de las palabras 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
niñ <b>as</b>	-as	niñ
niñ <b>ez</b>	-ez	niñ

Figura 2.1: *Proceso de Stemming*

Forma	Información morfológica	Lema
niñas	Género femenino, plural del sustantivo <b>niño</b>	niño
niñez	Singular del sustantivo <b>niñez</b>	niñez

Figura 2.2: *Proceso de Lemmatización*

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 los “emojis” o “emoticonos”. Este último caso merece especial detalle dado que admite un análisis específico.

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 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) [11] 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) [12] con este proceso de normalización consigue mejorar su sistema de clasificación de sentimientos por un 4 %.

La tarea de **normalización** consiste en transformar palabras o oraciones en 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 muy importante no saltarnos todos estos términos 2.3. Desafortunadamente no hay una manera única de normalizar los textos ya que la forma suele depender de la tarea específica 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 quizás dependan más del contexto y el experimento específico que se lleva a cabo. Se intenta 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.

### 2.3.3. Extracción de características

Como hemos comentado en la sección 2.3.2, 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. En general podríamos decir que existe una serie de características internas en los textos tales que aplicando técnicas o algoritmos de extracción, nos permitirían comprender cual es la idea que se pretende transmitir. 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**.

Uno de las técnicas más simples para representar numéricamente textos es la **Bolsa de Palabras** (Bag of Words, BOW). 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. 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 particular se crea un vector numérico asociado donde para cada palabra del corpus se marque con un 1 si está presente en el texto y con 0 si no lo está.

Una vez generados los vectores para cada texto individual lo más común es realizar la técnica de **Frecuencia de Términos-Frecuencia Inversa de Documentos** (Term Frequency-Inverse Document Frequency, TF-IDF). Ésta 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 la frecuencia 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 técnica BOW tiene una desventaja que es el ignorar 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 representan de forma similar.

De nuevo se pretende que mediante algún tipo de algoritmo, las palabras se vinculen a vectores numéricos. Mediante comparaciones de los mismos 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 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 cuales están asociadas a ella o son similares. Existen dos variantes distintas: continuous bag-of-words (CBOW) y continuous skip-gram. Según las notas del autor CBOW produce resultados más rápidos pero skip-gram se adapta mejor a las palabras no frecuentes.

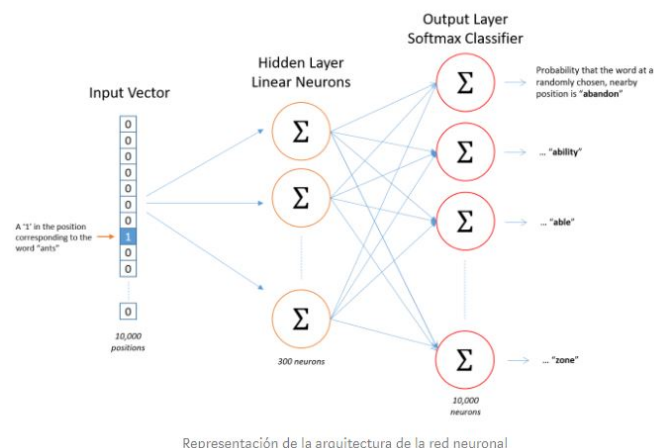


Figura 2.4: Red Neuronal word2vec

Otra técnica Word Embedding utilizada es **GloVe** (Global Vectors). Es un algoritmo de aprendizaje no supervisado desarrollado en Stanford donde el entrenamiento se realiza sobre estadísticas globales de coocurrencia de palabras de un corpus. Por coocurrencia entendemos aquellas palabras que se utilizan comúnmente juntas. Las representaciones resultantes exponen subestructuras lineales en los vectores de palabras en el espacio.

En resumen, Words embeddings 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 obtenemos un mayor entendimiento del texto.

Existen muchas técnicas o algoritmos que se pueden aplicar en esta fase dependiendo del objetivo que se persigue. 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.

## 2.3.4. Modelado

En la tarea de modelado se han utilizado diversas técnicas de ML 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 utilizó 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. El clasificador por tanto tiene que empezar utilizando palabras semilla asociadas a cada emoción para ir clasificando los textos. De esta forma

se entrena el clasificador que entonces se utiliza para etiquetar los datos de prueba. Los ejemplos más utilizados son las redes neuronales y los métodos de clustering (k-medias, k-vecinos etc).

Los métodos no supervisados son más generales en la mayoría de los casos y los supervisados suelen conseguir una mayor precisión. Ya podemos observar que sería interesante combinarlos de alguna forma para obtener lo mejor de ambos. 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 etc.

### 2.3.5. Diseño del experimento y Validación

Una vez tenemos nuestro modelo construido, necesitamos establecer la metodología experimental para estimar la tasa de error que tiene. La metodología suele estar altamente influenciada por el tamaño del conjunto de datos que tenemos, ya que con un número insuficiente habría que ajustar los pasos con más cuidado. En general no vamos a tener este problema y tendremos suficientes datos como para separar tres conjuntos diferenciados: sea  $T$  el conjunto de entrenamiento,  $V$  el conjunto de validación y  $P$  el conjunto de prueba. Cada uno tiene que ser suficientemente grande y estar seleccionado de forma aleatoria.

Con  $T$  y  $V$  construiremos el clasificador (modelo) y con  $P$  evaluaremos el funcionamiento del mismo como podría ser 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 23 para entrenamiento y 13 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 necesaria tomar una decisión que consiga un compromiso entre funcionamiento y tiempo de ejecución.

Con respecto a la forma de validar el 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. Este se calcula como el porcentaje de predicciones correctas sobre el

total de predicciones realizadas. Esta medida 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.

Con todo esto sabido podríamos utilizar otras métricas de validación de modelos más elaboradas y específicas. De las más comunes utilizadas son *precision*, *recall* y *F1-Measure* (ver ecuación 2.1). Ambos 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 de *precision* 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 práctico al hacer más fácil la comparación del rendimiento combinado. El valor *F1* asume que la importancia 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 [9], [13], [14] o [15]

$$precision = \frac{TP}{TP + FN}; \quad recall = \frac{TP}{TP + FN}; \quad F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (2.1)$$

Finalmente la validación es una tarea que proporciona criterios objetivos para comprobar el funcionamiento de nuestros modelos, por tanto da pie a posibles refinamientos que mejoren la calidad del experimento.

## 2.4. 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, de esta forma tenemos una estructura de red.

Las redes sociales sirven como instrumentos 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 críticas de películas de cine redactadas por expertos. 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.



# Capítulo 3. Sistema Experimental

## 3.1. Introducción

Esta sección 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.

## 3.2. Arquitectura del experimento

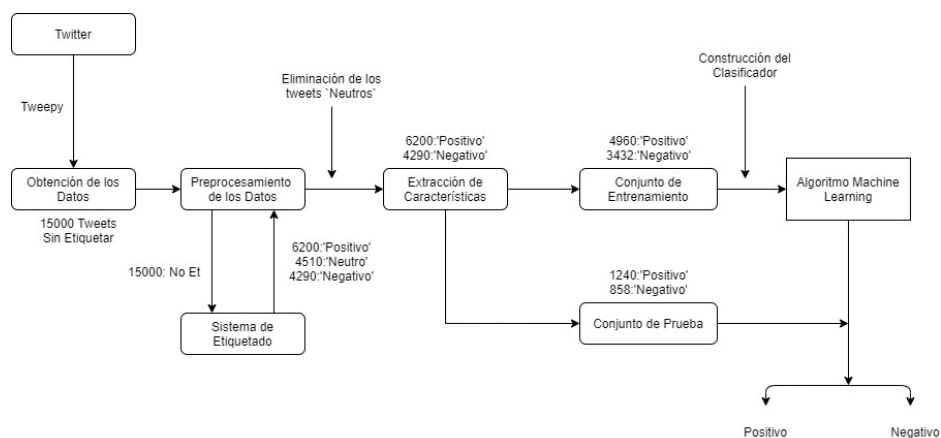


Figura 3.1: *Arquitectura del Sistema de clasificación de tweets*

### 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 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 acceso amplio 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 hacer 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 coinciden 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 caracteres 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`

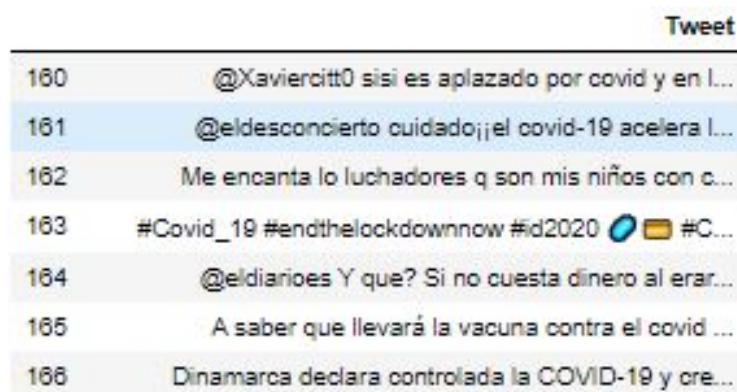
que activándolo, permite esperar automáticamente en cada ventana para poder hacer las peticiones.

Gracias a este parámetro podemos automatizar la extracción de los datos que sin embargo lleva entre dos y tres horas en conseguir completar su 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 #d2020 🏡 🇪🇸 #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.

Ésta 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 patron 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 patron 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 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 plantea 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 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 y 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)
```

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 “aborrecíerais” 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 como se detalla a continuación.

Vamos a utilizar la librería **spacy** que es una biblioteca para procesamiento avanzado de lenguaje natural. Incluye modelos preentrenados para predecir etiquetas PoS, dependencias sintácticas o entidades propias. Esta librería proporciona modelos para muchos lenguajes, de entre ellos el castellano que es el que vamos a utilizar *es\_core\_news\_sm*. Este modelo es una Red Neuronal Convolutiva 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.

Terminos	Completo
Tweet Original	En @elconfidencial: Islas contra Península: la guerra del verano covid entre los principales destinos turísticos <a href="https://t.co/4XUQzBmsfX">https://t.co/4XUQzBmsfX</a> ,
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.*

A continuación 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 tiempo 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, 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):** [?] 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 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**.

Terminos	Tamaño
Positivos	1555
Negativos	2720
Total	4275

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

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

- **Emoji Sentiment Ranking v1.0** [?] Léxico de emojis elaborado a partir del etiquetado por parte de 83 personas de 1.6 millones de tweets en 14 lenguajes europeos como positivo, neutral o negativo. 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 fichero al que nos referiremos en nuestro proyecto como `../data/emoji/Emoji Sentiment Ranking 1.0`.

- **Léxico ML-SentiCon:** [?] Conjunto de lexicones de polaridades semánticas a nivel de lemas para inglés, castellano, catalán, gallego y euskera. Los lexicones 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 cada 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**, 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 lexicon a nuestras necesidades. Además existen lemas repetidos, de los que nos quedaremos el primero que aparece en la lista, ya que corresponde a las capas más elevadas donde 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 `../data/mlsenticon/senticon-limpio`.

- **iSOL** [?] Lexicon generado a partir de una traducción automática del inglés al español del lexicon creado por Bing Liu [?] generando así el recurso SOL (*Spanish Opinion Lexicon*). Se realiza una corrección 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 `../data/crisol/isol`.

- **CRiSOL** [?] Léxico generado combinando información de iSOL y SentiWordNet. Se añaden a iSOL las puntuaciones de polaridad de SentiWordNet. iSOL está formado por formas 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.

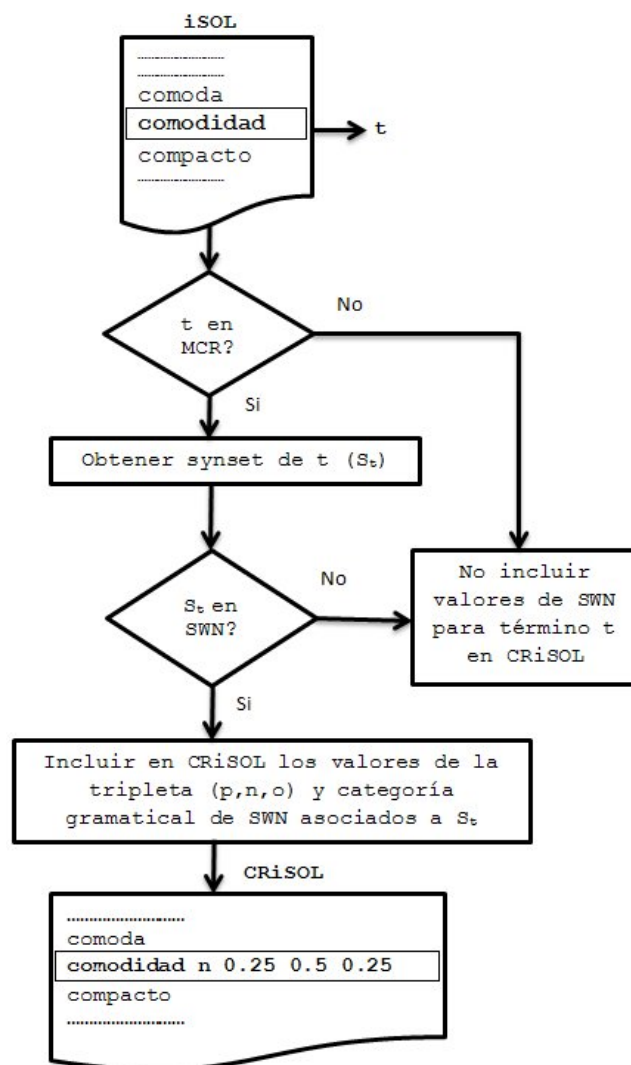


Figura 3.4: *Conjunto de datos inicial*

Terminos	Completo	Reducido
Total	4434	4430

Cuadro 3.7: *Distribución de clases en el léxico CRiSol adaptado.*

En la práctica contamos con un ficheros al que nos referiremos en nuestro proyecto como `../data/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 individual 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 asociados 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 + Emoji S.Ranking.**

El léxico Sentiment Polarity Lexicons (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 datos se calcula si predomina el sentimiento positivo o el negativo. En el algoritmo de etiquetado 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, etiquetaremos como “neutro” momentáneamente.:

---

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

---

**Entrada:** Tweet preprocesado

**if**  $(terminos\_positivos + emojis\_positivos) > (terminos\_negativos + emojis\_negativos)$  **then**  
| etiqueta\_tweet[i] = “positivo”

**else if**  $(terminos\_positivos + emojis\_positivos) < (terminos\_negativos + emojis\_negativos)$   
**then**

| etiqueta\_tweet[i] = “negativo”

**else**

| etiqueta\_tweet[i] = “neutro”

**end**

**Salida:** Etiqueta del subsistema 1

---

Los resultados que arroja este subsistema con los 15000 tweets son los siguientes:

Clasificación	Tamaño	
	Sin Lematización	Con Lematización
Positivos	5102	6536
Neutro	5286	3630
Negativo	4612	4834

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	18660
Total de términos	22494	39172

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

Esta tabla 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 **39172**.

A continuación se muestran 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

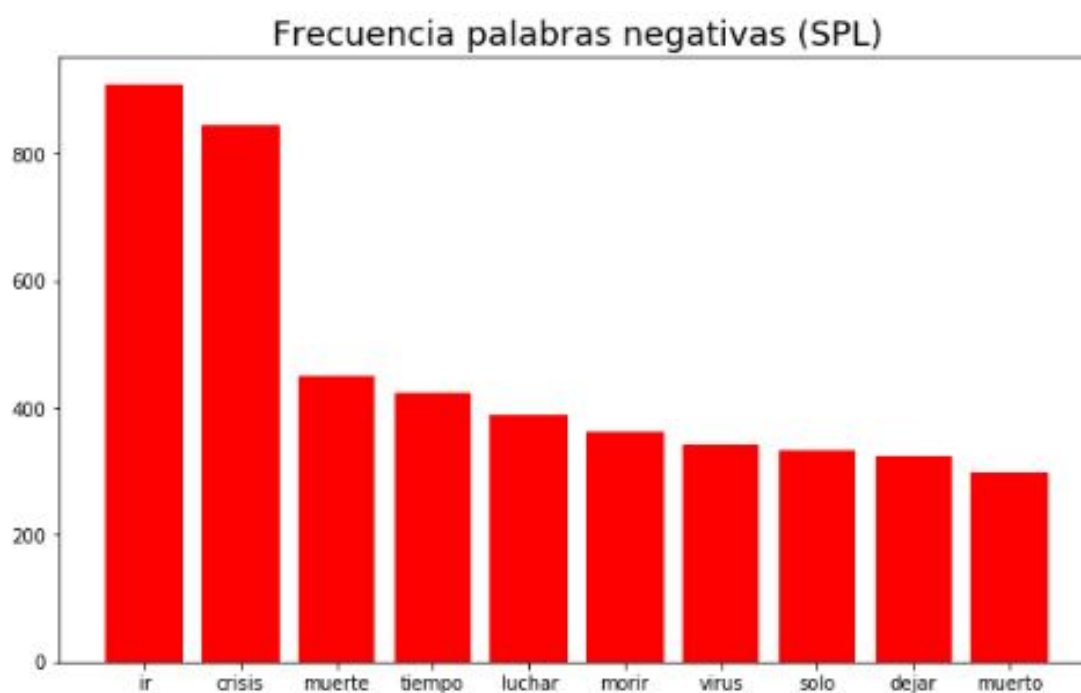


Figura 3.6: 10 términos negativos más utilizados

■ **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 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\_termino\_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 *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 negativa 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”.

---

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

---

**Entrada:** Tweet preprocesado

```
if (max_term_positivo+max_emoji_positivo) > (max_term_negativo+max_emoji_negativo)
  then
    | etiqueta_tweet[i]=“positivo”
else if (max_term_positivo+max_emoji_positivo) < (max_term_negativo+max_emoji_negativo)
  then
    | etiqueta_tweet[i]=“negativo”
else
    | etiqueta_tweet[i]=“neutro”
end
```

**Salida:** Etiqueta del subsistema 2

---

Clasificación	Tamaño	
	Sin Lematización	Con Lematización
Positivos	7235	8567
Neutro	5078	3580
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 %)
NºTérminos positivos	8789	11835
NºTérminos negativos	4107	5348
Total de términos	12896	17813

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

Este sistema observamos claramente como 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 **17813**. Es un número mucho más reducido ya que únicamente se contabiliza el término máximo encontrado por cada tweet.

A continuación se muestran un listado de los 10 términos que más se han utilizado para clasificar como “positivo” o “negativo” utilizando este subsistema.

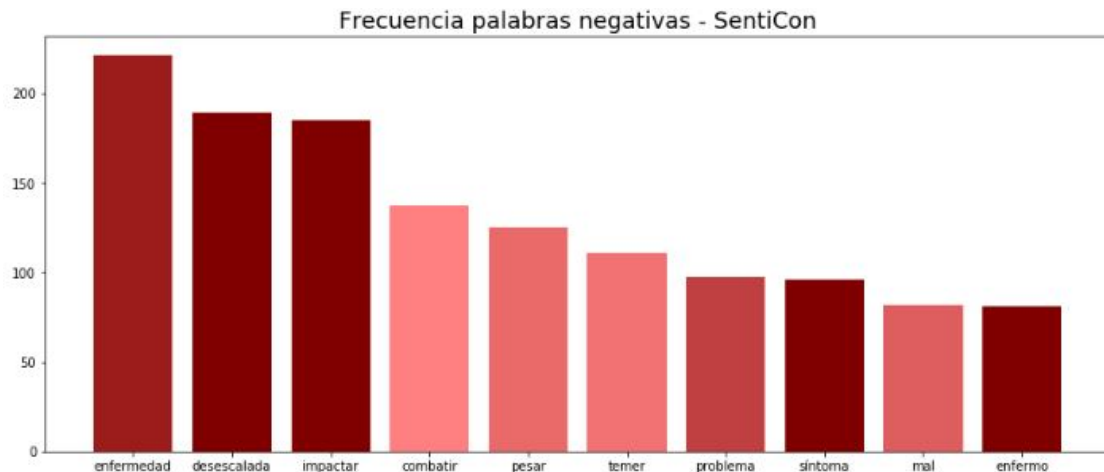


Figura 3.8: 10 términos negativos más utilizados

#### ■ 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

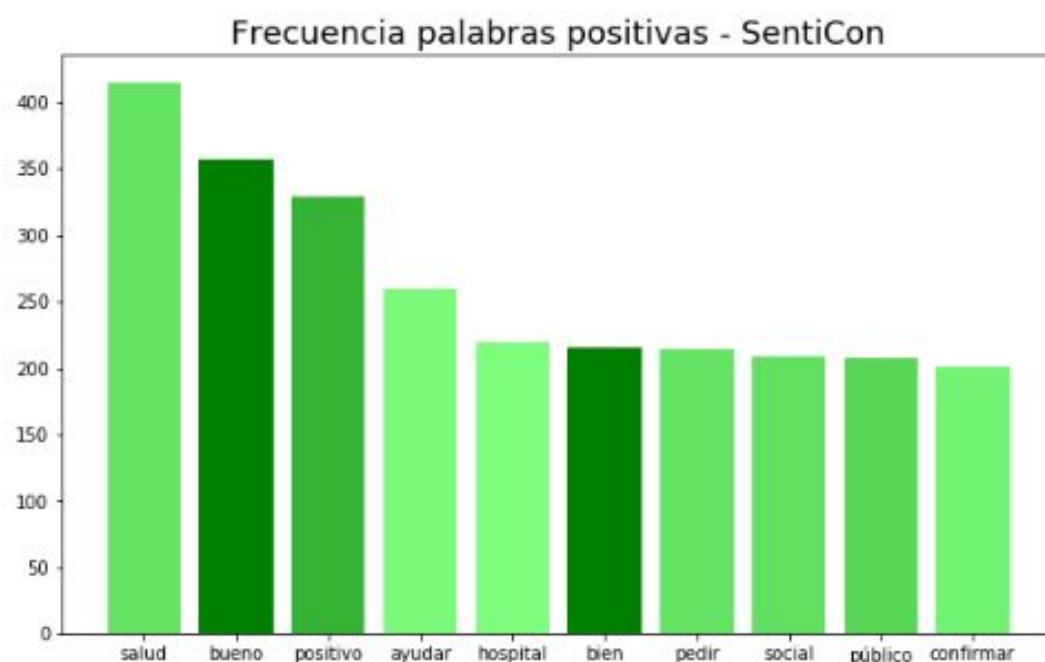


Figura 3.7: 10 términos positivos más utilizados

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 “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 datos se calcula si predomina el sentimiento positivo o el negativo. En el algoritmo de etiquetado 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, etiquetaremos como “neutro” momentáneamente.:

---

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

---

**Entrada:** Tweet preprocesado

```

if (terminos_positivos+emojis_positivos) > (terminos_negativos+emojis_negativos) then
  | etiqueta_tweet[i]="positivo"
else if (terminos_positivos+emojis_positivos) < (terminos_negativos+emojis_negativos)
  then
  | etiqueta_tweet[i]="negativo"
else
  | etiqueta_tweet[i]="neutro"
end

```

**Salida:** Etiqueta del subsistema 3

---

Los resultados que arroja este subsistema con los 15000 tweets son los siguientes:

Clasificación	Tamaño	
	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 %)
NºTérminos positivos	8890	9783
NºTérminos negativos	8895	10664
Total de términos	17845	20447

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

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.

A continuación se muestran un listado de los 10 términos que más se han utilizado para clasificar como “positivo” o “negativo” utilizando este subsistema.

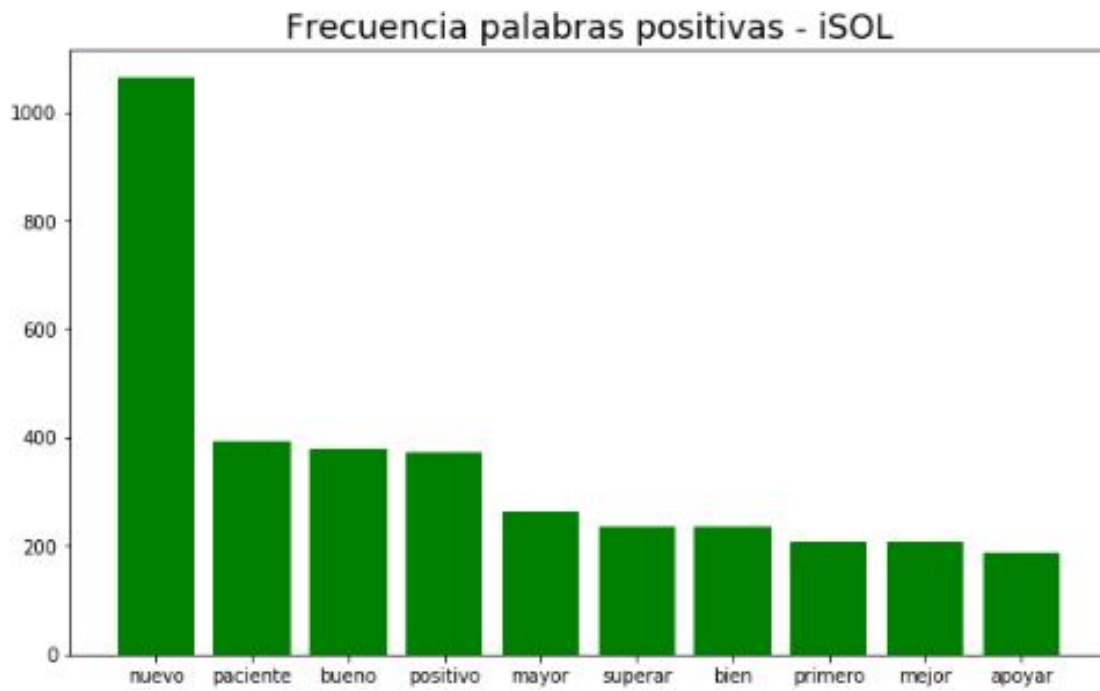


Figura 3.9: 10 términos positivos más utilizados

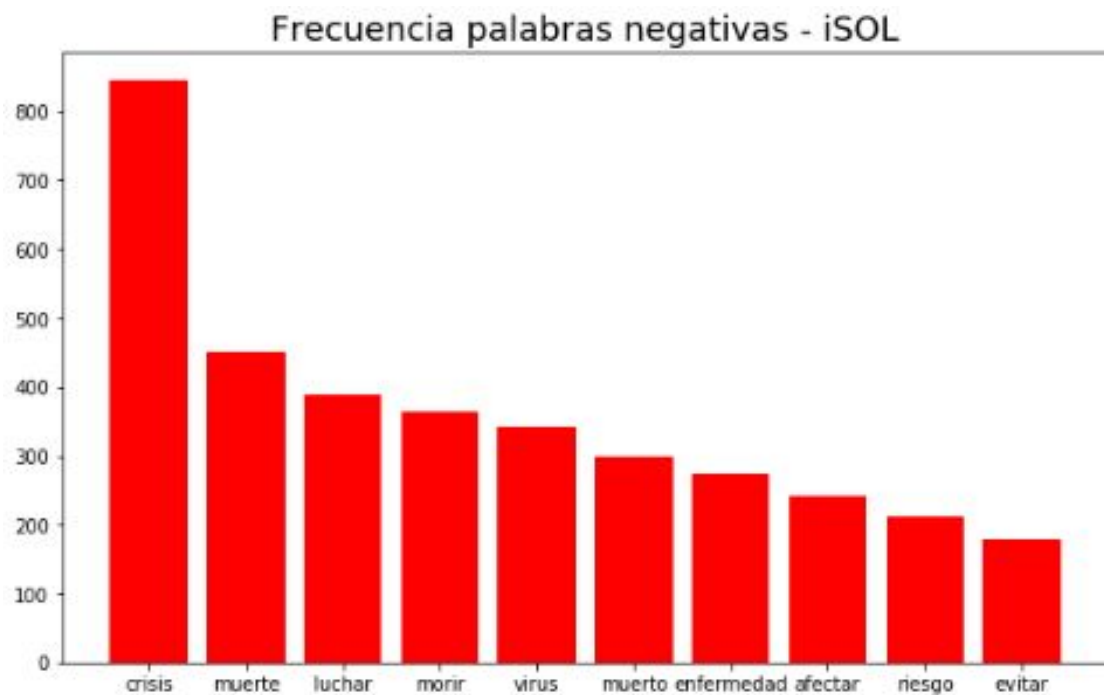


Figura 3.10: 10 términos negativos más utilizados

#### ■ Subsistema 4: CRiSOL + Emoji S.Ranking

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” y 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\_termino\_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 *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 negativa tenga en base al diccionario de emojis. En este caso pueden existir términos del léxico que presenten valores superiores a cero dado a que su pertenencia a una clase u a otra sea 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

```
if (max_term_positivo+max_emoji_positivo) > (max_term_negativo+max_emoji_negativo)
  then
    | etiqueta_tweet[i]=“positivo”
else if (max_term_positivo+max_emoji_positivo) < (max_term_negativo+max_emoji_negativo)
  then
    | etiqueta_tweet[i]=“negativo”
else
  | etiqueta_tweet[i]=“neutro”
end
```

**Salida:** Etiqueta del subsistema 2

---

Clasificación	Tamaño	
	Sin Lematización	Con Lematización
Positivos	4535	4844
Neutro	6834	6294
Negativo	3631	3862

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	5126
Total de términos	10641	10387

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

Este último subsistema es el que menos tweets consigue clasificar como “positivo” o “negativo”, podría deberse a que el léxico que 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. De nuevo el aplicar el proceso de lematización consigue clasificar **540** tweets más de una forma balanceada, 309 como positivos y 231 como negativos. El número de términos totales encontramos en este caso se reduce mínimamente (250 términos), esto se puede deber tanto al sistema de lematización como a la construcción de este léxico.

A continuación se muestran un listado de los 10 términos que más se han utilizado para clasificar como “positivo” o “negativo” utilizando este subsistema.

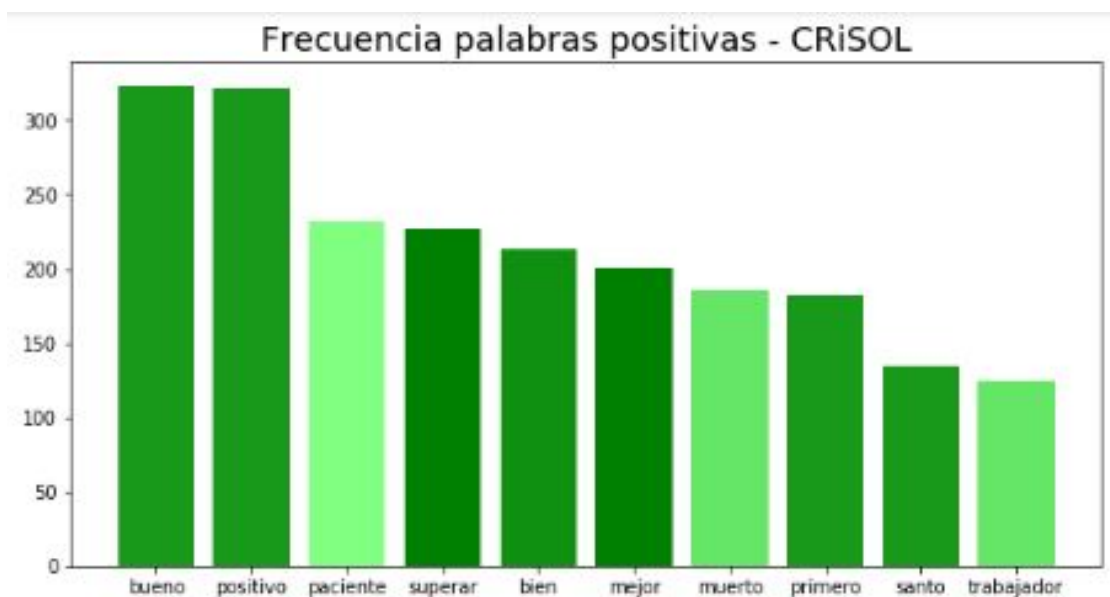


Figura 3.11: *10 términos positivos más utilizados*



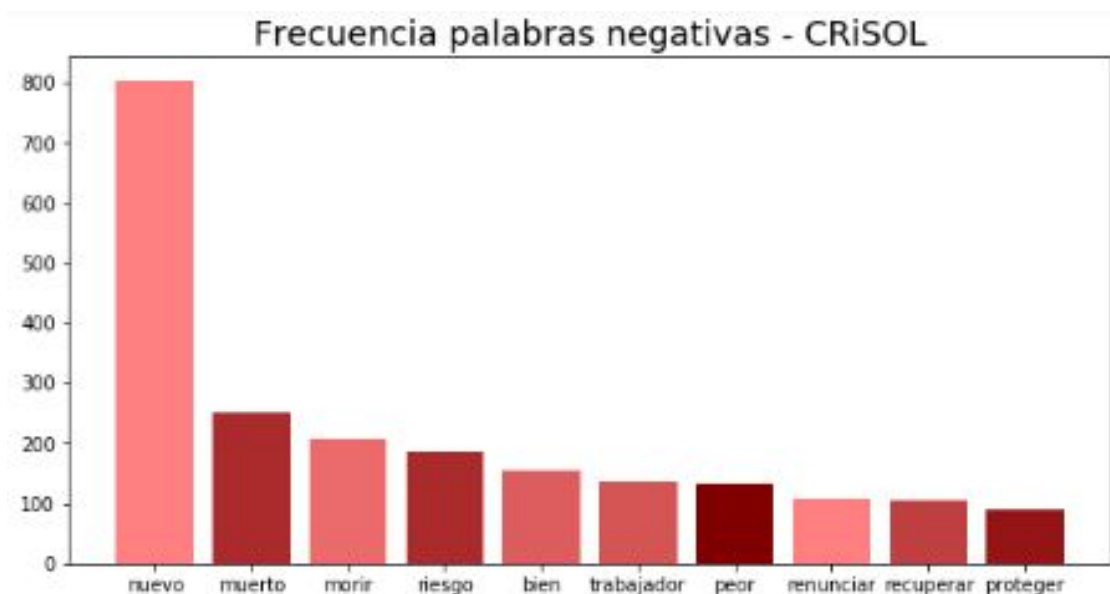


Figura 3.12: 10 términos negativos más utilizados

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 siguiente diagrama se muestra un esquema conceptual del sistema descrito en esta sección.

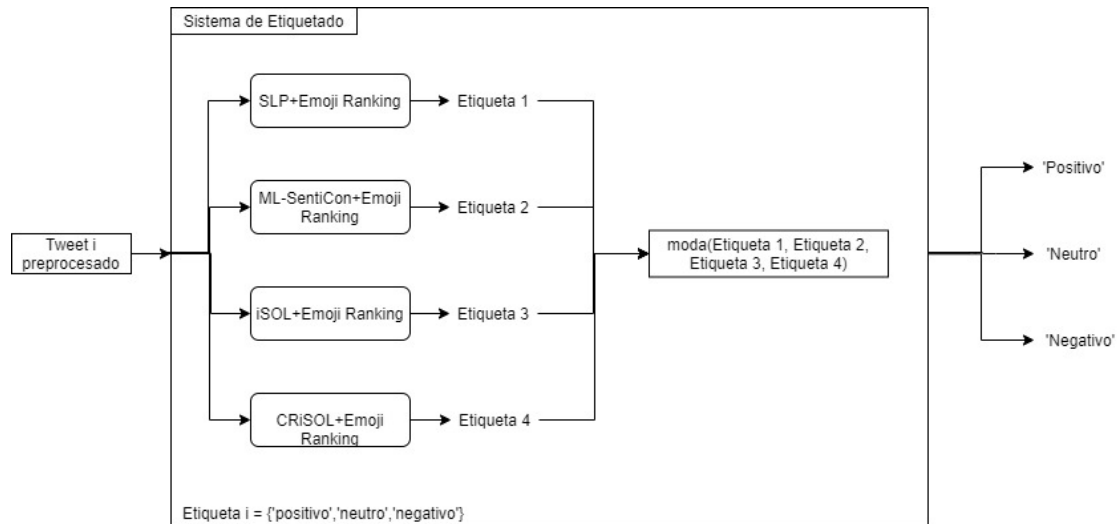


Figura 3.13: *Arquitectura del sistema de etiquetado*

Hemos conseguido etiquetar mediante cuatro sistemas diferentes un mismo tweet cuatro veces como “positivo”, “negativo” o “neutro”. Para obtener la etiqueta final asociada a cada tweet calcularemos la moda de las etiquetas resultantes y esta será 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.

	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	-	322.71 seg	2.01 seg

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

Clasificación	Tamaño	
	Sin Lematización	Con Lematización
Positivos	5588	6494
Neutro	5565	4426
Negativo	3847	4080

NºTérminos	73306	87189
NºEmojis	4963	4963
<b>NºElementos</b>	<b>78269</b>	<b>92152</b>

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

Léxico de Términos	Positivas	Negativas	Total
SPL	20512	18660	39172
ML-SentiCon	11835	5348	17183
iSOL	9783	10664	20447
CRiSOL	5261	5126	10387
<b>Total</b>	<b>47391</b>	<b>39798</b>	<b>87189</b>

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

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 etiquetados por el sistema final como “positivo” o “negativo”. De esta forma conseguimos un conjunto final que consta de **10574** tweets, de los cuales **6494** son considerados “positivos” y **4080** “negativos”.

En las tablas 3.16 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 siguiente resultado.

	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	1776.9 seg	1660.79 seg

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

## 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.

### 3.6.1. Bag of Words

Para realizar BoW con TF-IDF crearemos una instancia **TfidfVectorizer** que encontramos en `sklearn.feature_extraction.text` que 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.

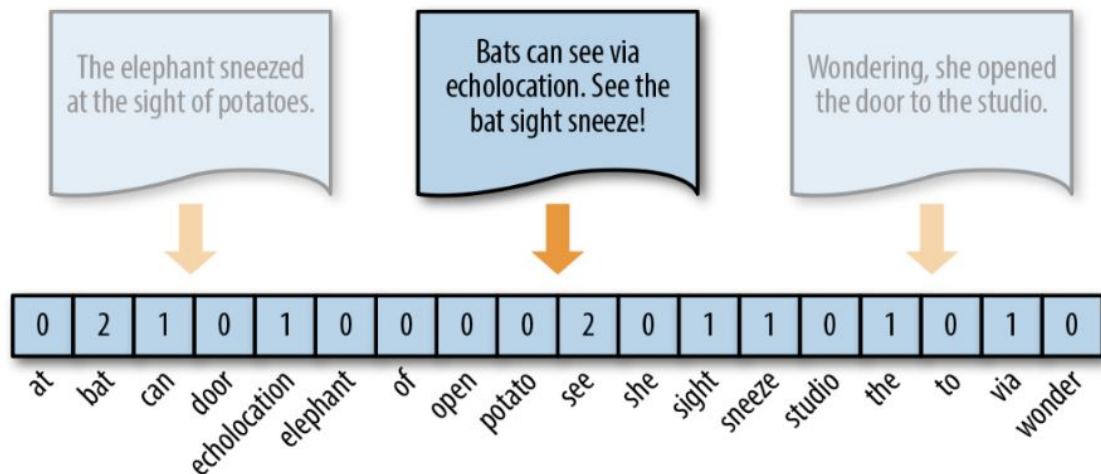


Figura 3.14: *Funcionamiento de Bag of Words*

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 (stop-words=“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 como 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 considerada en la construcción de la bolsa de palabras (min\_df) se decidirán más tarde en una tarea de selección de parámetros conjunta con la elaboración del modelo de clasificación.

### 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, que es la distancia máxima entre la palabra actual y la palabra predicha dentro del mismo tweet (window), 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).

Se toma 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.

Una vez hemos construido el modelo con los vectores de palabras vamos a construir los vectores asociados a cada tweet. Esto lo vamos a hacer de dos formas distintas: la primera forma es calculando la media de los vectores de cada palabra de un tweet individual y la segunda forma es calculando la suma de los vectores de cada palabra de un tweet. De nuevo más adelante se construyen pruebas para definir cual proporciona mejores resultados.

## 3.7. Modelado

Una vez tenemos los vectores numéricos contruidos 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, de forma que el margen de separación entre las dos clases en los datos se amplíe al máximo. 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.

$$K(x_1, x_2) = x_1^\top x_2 \quad (3.1)$$

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 comprobar 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 calcula la salida final del ensemble como la moda de las salidas de cada uno de los árboles de decisión. Los Random Forest corrigen la querencia a sobreajustar que tienen los árboles de decisión por separado y son una de los algoritmos más utilizados en ML.

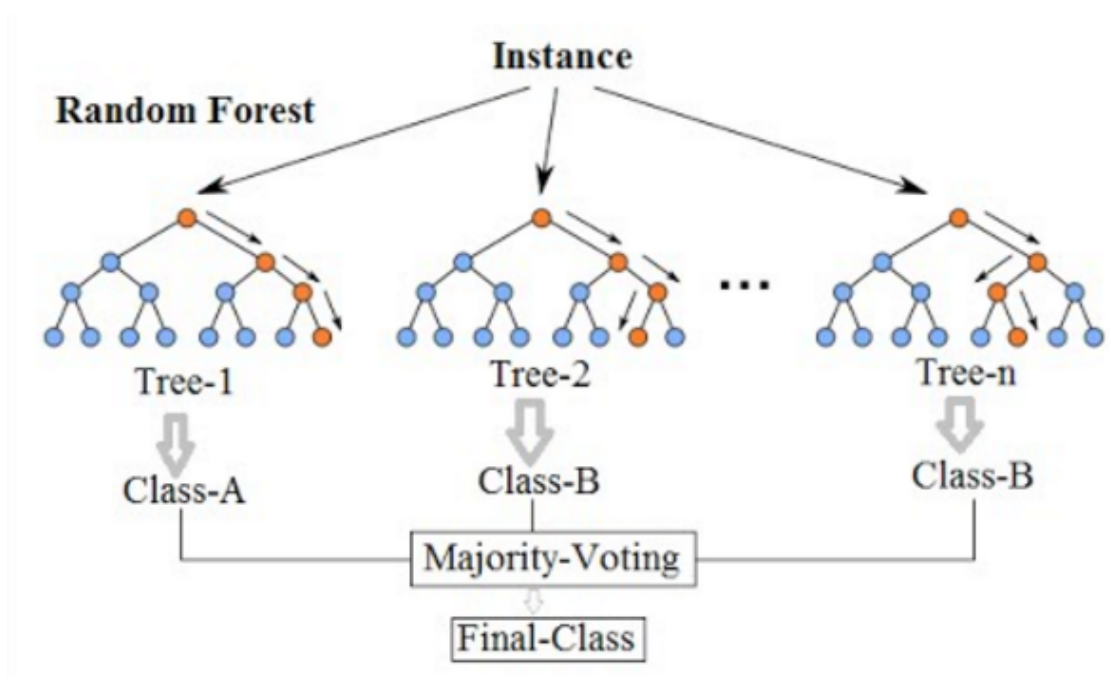


Figura 3.15: *Funcionamiento de Random Forest*

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 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**. Esta función 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.

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \dots + \beta_k x_{k,i} \quad (3.2)$$

La implementación sigue de nuevo el mismo esquema que los algoritmos anteriormente comentados. Tenemos una función `LogisticRegression()` en el módulo `sklearn.linear_model`, que recibe los parámetros de configuración deseados. En nuestro caso igual que en SVM realizaremos una tarea de selección de parámetros para el parámetro de regularización. 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 y Validación

#### Selección de parámetros

Vamos a dividir el conjunto de entrenamiento en entrenamiento y prueba donde este último no se va a utilizar hasta la fase de validación. El conjunto de entrenamiento consta de **8459** tweets y el de prueba de **2115**, estando ambos estratificados por la clase de los mismos.

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

Para realizar esta labor vamos a definir dos experimentos distintos: 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 y como algoritmos de clasificación también los tres que hemos comentado.

**TODO:** *Foto aquí de la estructura de este experimento*

Los posibles parámetros de la fase de extracción de características se introducen ma-



		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	0.7
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	0.87	1000	0.83	0.86	10	0.84	0.86
	3	1	0.84	0.86	300	0.83	0.86	10	0.84	0.87
	5	1	0.84	0.87	1000	0.83	0.86	10	0.84	0.87
	10	1	0.84	0.87	1000	0.83	0.86	1	0.83	0.87

Cuadro 3.17: Selección de parámetros con BoW (TF-IDF) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores

nualmente. Sin embargo para los relacionados con los algoritmos de clasificación existe el módulo **sklearn.model.selection** con una función clave que es **GridSearchCV** al que le pasas un estimador junto con una lista de parámetros de configuración a explorar y esta realiza una búsqueda exhaustiva 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, el método 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. Este parámetro se puede tunear (**scoring=None**).

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 *precision* 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.

A continuación se muestran las tablas 3.17 y 3.18 con los resultados de los estimadores que mejores resultados dan en cada uno de los dos experimentos y las tablas 3.19 y 3.19 (cuando estén) con 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.

Claramente obtenemos unos mejores resultados utilizando BoW como opción para extraer características. Recordemos que la proporción de nuestros datos es de 61.4% de

		Clasificador								
Extr.Caract: W2V		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 3.18: Selección de parámetros con Word Embeddings (Word2Vec) 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 3.19: 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.Logística
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 3.20: *Tiempos de ejecución selección de parámetros con Word Embeddings (Word2Vec) como extracción de características y SVM, Random Forest y Regresión Logística como clasificadores*

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 %. Utilizar Word2Vec para extracción de característica solo consigue elevar en casi 10 puntos esta métrica, siendo Random Forest el clasificador que mejor parece funcionar ya que es el único que consigue valores de 70 % . No se muestra ninguna información que nos pueda indicar que algún parámetro funcione de mejor forma.

Sin embargo utilizando BoW (TF-IDF) como extracción de características, observamos claramente que a mayor tamaño de la bolsa de palabras, mejores valores obtenemos. Recordemos que como métrica de evaluación estamos utilizando la medida f1 . Ya con el menor tamaño probado (250 palabras) obtenemos un valor del 77-78 %, 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.

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 se reducirán. En general SVM y especialmente Random Forest, son con diferencia los 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.

## Clasificador final

Como acabamos de comentar, utilizando BoW con tamaño 1000 en la fase de extracción de características, conseguimos un valor de la métrica f1 de 83-84 % 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:

- 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

## Validación

Se comprueba tanto el funcionamiento de los tres clasificadores por separado como conjuntamente con un sistema de votación de mayoría puro y se obtienen los siguientes resultados con el conjunto de validación que contiene 2115 tweets. Las métricas presentadas corresponden a la media ponderada entre las métricas obtenidas para las dos clases.

	Accuracy	Precision	Recall	F1-Weighted
SVM	84 %	84 %	84 %	84 %
Random Forest	83 %	84 %	83 %	84 %
Reg.Logística	84 %	84 %	84 %	84 %
Voto por Mayoría	85 %	85 %	85 %	85 %

Cuadro 3.21: *Validación del clasificador*

## 3.9. Discusión y Resultados

Desde el primer momento habíamos reservado un conjunto test que nos serviría para comprobar la precisión del sistema que hemos elaborado. Recordemos el proceso que se ha realizado.

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: **6494** tweets “positivos”, **4426** “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 **10574** 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 igual, por lo tanto se propone un clasificar final construido como un sistema de voto por mayoría de los tres clasificadores individuales.

Finalmente utilizamos el conjunto de validación que habíamos reservado para observar el funcionamiento global del sistema construido. Obtenemos los siguientes resultados ...

**TODO:** *Faltan los resultados cuando ya esté todo claro*

<b>Fase</b>	<b>Tiempo</b>
Preprocesamiento	172.91 seg
Etiquetado de los Datos	1493.68 seg (sin lema)
Eliminación de los neutros	0.76 seg
Extracción de Características	
Construcción del clasificador	
Validación	

Cuadro 3.22: *Tiempos de ejecución*

## **Capítulo 4.**

## **Conclusiones**





## Capítulo 5.

## Trabajo Futuro

**TODO:** *Voy anotando ideas*

- Análisis de otras formas de comunicación como audio o video
- Análisis de otras formas de comunicación textual como las citadas: textos de opinión, cartas, libros, blogs...
- Análisis más profundo de trabajos existentes, como por ejemplo ganadores de competiciones.
- Extrapolar el proceso utilizado a otro conjunto de datos, bien procedente de redes sociales (temática específica, procedencia específica) o de otras formas de comunicación textual
- Probar la diferencia de funcionamiento del sistema elaborado con textos en otros lenguajes. Quizás sería interesante realizar una buena traducción de los datos originales para ver si el rendimiento es similar.
- Análisis exhaustivo de las posibilidades de modelado con técnicas de ML



# Bibliografia

- [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] C.D.BROAD (1954) *Emotion and sentiment*. Journal of Aesthetics and Art Criticism 13(2):203-214.
- [4] LÖVHEIM H. (2011). *A new three-dimensional model for emotions and monoamine neurotransmitters*. Medical hypotheses. 78. 341-348.
- [5] 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.
- [6] EKMAN P (1992) *An argument for basic emotions*. Cognition and Emotion 6(3-4):169-200.
- [7] PLUTCHIK R (1980). *Emotion: a psychoevolutionary synthesis*. Harper and Row, New York.
- [8] PAK A, PAROUBEK P (2010). *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*. LREC.
- [9] DINI L, BITTAR A (2016). *Emotion Analysis on Twitter: The Hidden Challenge*. LREC.
- [10] MOHAMMAD S.M, BRAVO-MARQUEZ F (2017). *Emotion intensities in tweets*. Proceedings of the sixth Joint Conference on Lexical and Computational Semantics.
- [11] 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.
- [12] 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.

- [13] 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.
- [14] 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.
- [15] 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.
- [16] CHEN Y, SKIENA S. (2014). *Building Sentiment Lexicons for All Major Languages*. In ACL (2) (pp. 383-389)
- [17] KRALJ NOVAK P, SMAILOVIC J, SLUBAN B, MOZETIC I (2015). *Sentiment of Emojis*, PLoS ONE 10(12).
- [18] 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, n° 13, pp. 5984-5994.
- [19] 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*.
- [20] 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.
- [21] 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.
- [22] GHELANI S. (2019) *From Word Embeddings to Pretrained Language Models - A new Age in NLP - Part 1*, Towards Data Science.