Universidad Rey Juan Carlos (URJC)

Máster en Data Science (MDS)

Asignatura: Text Mining

Práctica de evaluación final:

“*Análisis de emociones en twitter*”

21 de marzo de 2022

○ Autores

Luisa Cristina Yánez Gutiérrez
Miguel García Sánchez

○ Profesor

Soto Montalvo Herranz

ÍNDICE

1. Introducción

El text mining y el NLP (“Natural Language Processing”) son campos del aprendizaje automático relacionados con el entendimiento, análisis y comprensión del lenguaje humano a través de datos principalmente en formato de texto.

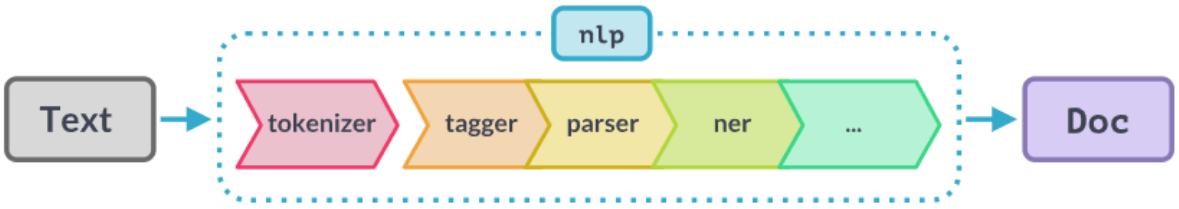
Es por ello que es de gran relevancia conocer siempre de forma exhaustiva el tipo y características del texto que vamos a analizar, las diferentes palabras que más se usan en él y las diferentes estructuras que lo componen, para poder de esa forma pasar a un correcto procesado de texto antes de aplicar algoritmos que nos puedan ayudar en las tareas de predicción y clasificación sobre los mismos.

Por lo tanto, con el presente trabajo trataremos, por un lado, de entender precisamente cuales son las características básicas que definen y distinguen a los tweets como tipo de texto en concreto, analizaremos en profundidad los datos que tenemos a nuestra disposición y aplicaremos las técnicas de limpieza y preprocesado de textos que pueden llegar a encajar mejor con nuestros datos. Y por otro lado, una vez consolidada esta etapa previa de análisis, comprensión y procesado de tweets, iremos un paso más allá identificando qué algoritmos de aprendizaje supervisado (con datos etiquetados de forma previa) trabajan mejor en la tarea de clasificación de tweets en función de las emociones que se quieran transmitir en ellos y que configuración en los parámetros de vectorización y pesado de palabras es la más adecuada para mejorar la precisión en la predicción de las diferentes emociones de los usuarios de la red social.

2. Herramienta de análisis y procesado de texto

Para el correcto desarrollo del trabajo hemos utilizado como base para el procesamiento de texto la librería Spacy (versión 3.2.2) para Python (versión 3.9) con el modelo de lenguaje en español (modelo "es_core_news_sm"), la cual es una herramienta potente, de código abierto y ampliamente utilizada en diferentes trabajos de NLP, text mining y clasificación de texto en la actualidad.

Dicha herramienta permite pasar por todas las fases existentes en el preprocesado de texto así como realizar una limpieza de datos exhaustiva ajustándose siempre a las necesidades y características del texto a analizar. De forma nativa dentro de su procesado de datos tiene diferentes pipelines que permiten de forma ya integrada en la libraría realizar distintas funciones de limpieza y estructuración de textos. Complementariamente, permite añadir manualmente nuevas fases al proceso del pipeline según quiera y considere necesario el usuario, y en función de las características del texto y los objetivos que se quieran llegar a obtener.



Algunas de las principales funciones o métodos de las que dispone Spacy y que usaremos nosotros en los posteriores apartado han sido:

- *Tokenizer* (Tokenización de palabras - *Token.*)
- *NER* (Reconocimiento de Entidades Nombradas - *Doc.ents*)
- *POS* (Part Of Speech: análisis de todas las tipologías de palabras - *Token.pos_*)
- *Stopwords* (Análisis de palabras vacías - *Token.is_stop*)
- *Signos de puntuación* (Análisis de los signos de puntuación - *Token.is_punct*)
- *Lemmatizer* (Lematización de palabras - *Token.lemma_*)
- *Texto en minúsculas* (armonizar el texto a minúsculas - *Token.text.lower()*)

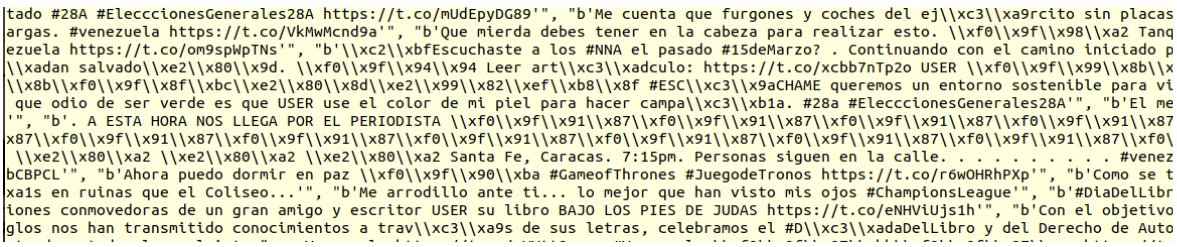
Las referencias consultadas para este apartado han sido las siguientes: [1], [2].

3. Propuestas con experimentos y resultados

3.1. Análisis del texto

Antes de realizar cualquier intento de limpieza y preprocesado, debemos entender cuales son las principales características de los datos que tenemos, como están estructurados los datos en cuestión y qué tipología de palabras y recursos lingüísticos vamos a procesar, para así realizar un análisis más preciso del texto. Por lo tanto, realizaremos un EDA o un breve "Análisis Exploratorio de Datos" inicialmente para tener cierto conocimiento de los datos de los que disponemos de cara a fases posteriores del trabajo.

Para llevar a cabo este análisis preliminar hemos añadido al código base una función llamada "*analisis_tweets*" de donde podremos obtener las entidades nombradas, así como las palabras más frecuentes y el tipo de palabras más frecuentes presentes en nuestros datos. Lo primero que hacemos en esta función es obtener cada uno de los tweets que están sin limpiar y codificados en tipo byte como se puede ver en la siguiente imagen:



Los tweets en formato byte no los podemos usar para crear un "doc" y obtener las entidades nombradas ya que los "doc" de la librería Spacy solo admiten tipo doc o tipo string. Por esta razón usamos la función ".decode()" y transformamos cada tweet a tipo UTF-8 que se leerá como un string. Después de decodificar los tweets vamos a normalizarlos pasando a minúsculas el texto con ayuda de la función ".lower()". Ahora que ya tenemos cada tweet como tipo string y además normalizado, ya podemos ver las entidades nombradas presentes en los datos y el resto de análisis mencionado.

3.1.1. Características de los tweets como texto

De cara a iniciar nuestro análisis de datos sobre los tweets que se han puesto a nuestra disposición debemos de entender el tipo de texto que es un tweet y sus principales características.

Un tweet es un tipo de texto corto (con estructuras lingüísticas de 280 caracteres como máximo) donde cada usuario trata de plasmar una idea, un sentimiento, una emoción o una opinión sobre un tema específico. Los elementos más frecuentes que lo componen, a parte de las diferentes palabras pertenecientes al vocabulario de cada idioma (nombres,

nombres propios, verbos, adjetivos, adverbios, determinantes, pronombres, determinantes, etc), son:

- Menciones a diferentes usuario de la plataforma (con la estructura “@usuario”),
- Hashtags sobre la idea o tema de referencia sobre la que trata el tweet (con la estructura “#hashtag”)
- Emojis (representaciones gráficas en forma de icono de las diferentes emociones, objetos o conceptos que se quieren expresar)
- URLs (links URL acortados para su uso en un contexto de texto con estructura muy limitada)
- Números
- Abreviaturas
- Signos de puntuación
- Caracteres especiales (por ejemplo: $ % & ' ( ) * + - / [\ ] ^ _ `{ | } ~)

Es destacable que gran parte de los estudios de sentimiento se realizan en textos largos donde se puede expresar una idea con un contexto suficiente y una gran extensión de vocabulario, donde se pueden sacar sus diferentes características y detectan patrones diferenciados de una forma más sencilla y precisa a nivel computacional. Respecto al estudio en textos muy cortos, como es el caso de los ejemplos que nos podemos encontrar en twitter, llegar a establecer el contexto y el significado concreto de cada palabra es un verdadero desafío.

Las referencias consultadas para este apartado han sido las siguientes: [3], [4].

3.1.2. Reconocimiento de Entidades Nombradas (NER)

El reconocimiento de entidades (NER) es un problema estándar de NLP que implica la detección de entidades con nombre (personas, lugares, organizaciones, expresiones de tiempo, objetos, entidad geográfica, cantidades, ubicaciones, etc.) de un fragmento de texto y clasificarlas en un conjunto predefinido de categorías.

Los sistemas NER que clasifican a las entidades se desarrollan en base a varios enfoques lingüísticos, métodos de carácter estadístico y probabilístico, y técnica de aprendizaje automático. El modelo primero identifica las diferentes entidades presentes en el texto que le pasamos, y después las categoriza en la clase más adecuada.

Algunos de los tipos de Entidades Nombradas existentes más comunes son:

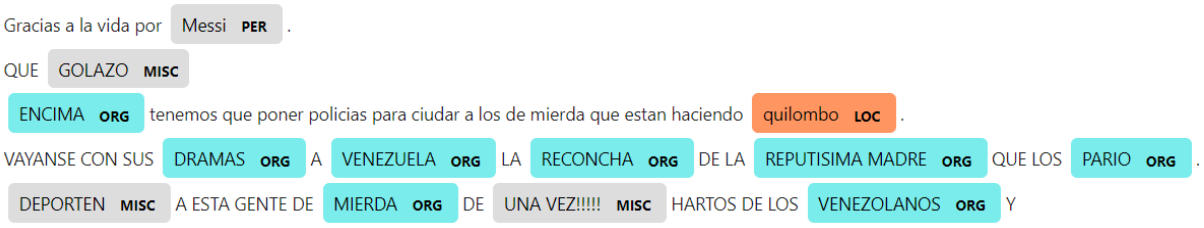
- ORG (empresas, agencias, instituciones, etc)
- LOC (localización, ubicaciones, etc)
- GPE (países, ciudades, Estados, etc)
- PERSON (relacionado con la gente)
- DATE (fechas y periodo de tiempo)
- TIME (periodos de tiempo inferiores al día)
- MISC (miscelánea o cajón de sastre).

Es interesante realizar este tipo de análisis en nuestro caso con datos de tipo tweet para tratar de ver cuales son las que aparecen de forma más recurrente en nuestro texto y poder tener un idea previa sobre cómo se estructura el mismo antes de empezar a preprocesarlo. Para ello utilizaremos la librería Spacy la cual tiene métodos integrados para el reconocimiento de entidades nombradas y permite a su vez crear patrones que cubran necesidades específicas del texto a analizar. En nuestro código hemos creado una función llamada "entidades_nombradas" donde vamos a extraer del doc las entidades con "doc.ents" y las etiquetas de cada entidad con ".label_" y por último imprimir la frecuencia de cada entidad nombrada presente en nuestros tweets.

Procediendo con esta parte del análisis hemos obtenido que en general las Entidades Nombradas que aparecen de forma más recurrente en nuestros tweets son las que se pueden ver en la siguiente imagen, ya que los temas más recurrentes se centran en emociones y opiniones del ámbito deportivo y del entretenimiento, y de ideas políticas y socioeconómicas.

```
Entity categories:  Counter({'MISC': 454, 'LOC': 413, 'PER': 318, 'ORG': 152})
```

Un ejemplo de ello extraído de nuestros tweets sería:



Vemos además como quizá el tema de categorizar Entidades Nombradas no está tan pulido en nuestro idioma como si que puede estarlo en inglés (idioma más global hoy en día y donde más se han centrado los esfuerzos de desarrollo y avances de todos estos campos), siendo poco preciso en algunos casos y categorizando de forma incorrecta ciertas palabras.

Dada esta posible problemática, se podría llegar a plantear la creación manual de reglas o patrones donde se le indica a Spacy o a la herramienta que se utilice en cada caso, cómo clasificar ciertas entidades. Sin embargo, al tener una gran cantidad de tweets y de temas muy diferentes y variados (no se centran solo en una única temática, sino en varias de diferentes ámbitos), es complejo crear un sistema de patrones que llegué a cubrir y arreglar todos los problema de clasificación presentes en el texto a este respecto.

Las referencias consultadas para este apartado han sido las siguientes: [5], [6], [7], [8], [9], [10].

3.1.3. Categorización de los tipos de palabras (POS)

A la hora de entender qué características tiene el texto a analizar, es de interés saber qué tipos de palabras emplea, con qué frecuencia y con qué significado.

En un primer análisis podemos ver cuales son las palabras que más se repiten o más frecuentes entre nuestros Tweets, para así sacar la temática general que se trata en todos ellos.

Para ello en la función "*analisis_tweets*" se ha tokenizado los tweets, se han cogido los que cumplan la condición de no ser stopwords ni signos de puntuación y se ha realizado un conteo de frecuencia de palabras sacando la temática más relevante en el texto:

```
*********************************
Palabras mas frecuentes:  ['|', '⚽', 'diadellibro', 'venezuela', 'barcelona', 'gretathunberg', 'gracias', 'gen
te', 'user', 'díadellibro', 'campeón', 'elecccionesgenerales28a', 'liverpool', 'años', 'mundo', 'messi', 'champ
ionsleague', 'a', '🏆', 'o', 'libros', 'incendio', 'laliga', 'gameofthrones', '', 'juegodetronos', 'libro', 'es
paña', 'notredame', 'historia', 'y', 'capítulo']
*********************************
```

Los temas más recurrentes son:

- *Entretenimiento*: diadellibro, juegodetronos, libro, gameofthrones.
- *Deporte*: laliga, liverpool, messi, barcelona, championsleague.
- *Política*: venezuela, mundo, eleccionesgenerales28a.
- *Socioeconómico*: gretathunberg, notredame, incendio.

Tras ello, pasamos a analizar el POS ("Part Of Speech") para ver cuales son los tipos de palabras más utilizados en el conjunto de tweets, lo cual nos puede llegar a ser útil para la realización de un preprocesado de datos eficiente y más preciso.

Para lograrlo en nuestro código en la función "*analisis_tweets*" se ha tokenizado los tweets, se han cogido los que cumplan la condición de no ser stopwords ni signos de puntuación, y se ha realizado un conteo de frecuencia de tipos de palabras sacando las más frecuentes e influyentes en el texto:

```
*********************************
Tipo de palabras mas frecuentes:  ['ADJ', 'CCONJ', 'PRON', 'NUM', 'DET', 'ADP', 'VERB', 'AUX', 'PROPN', 'NOUN',
'SYM', 'ADV']
*********************************
```

Los tipo de palabras más frecuentes que aparecen en el texto y que pueden llegar a ser más relevantes para nuestro análisis son:

- Adjetivos (ADJ)
- Verbos (VERB)
- Nombres (NOUN)
- Nombres propios (PROPN)
- Adverbios (ADV)

También podemos observar como existen una gran cantidad de:

- Números (NUM) en los tweets (principalmente referentes a fechas y cantidades)
- Auxiliares (AUX)
- Preposiciones (ADP)
- Determinantes (DET)
- Símbolos (SYM)

Los cuales deberemos analizar y considerar en mayor profundidad para valorar si nos puede interesar no tenerlos en cuenta, de cara a reducir el ruido en el texto y simplificarlo para un mejor posterior procesado y clasificación.

Las referencias consultadas para este apartado han sido las siguientes: [11], [12], [13].

3.1.4. Desambiguación de Emojis

Uno de los temas que se deben tener en cuenta al analizar un tipo de texto como es un tweet, muy vinculado a las opiniones y emociones de los diferentes usuarios, son los emojis.

Los emojis son un tipo de simbología referida a un concepto, una palabra, un objeto, un sentimiento o una emoción, que se expresa en el texto de forma gráfica complementando al resto de la frase.

Pueden suponer:

1. Un mero apoyo o refuerzo a la idea o ideas que se tratan de transmitir.
2. Por el contrario tratar de ser un contrapunto de la idea o ideas en un contexto de ironía donde se quiera transmitir justo lo contrario a lo escrito explícitamente con palabras.

Dependiendo del conjunto de datos de los que se disponga puede ser óptimo optar por:

1. Obviar los emojis y quitarlos del texto en la fase de preprocesado.
2. Por el contrario no quitarlos, transformarlos traduciéndose a texto y analizarlos junto al resto de palabras que forman el tweet.

Las referencias consultadas para este apartado han sido las siguientes: [14], [15], [16], [17].

3.2. Procesado de datos

Una vez realizado un análisis previo donde se han visto y analizado los tipos de datos a los que nos vamos a enfrentar así como las principales características de los texto en

formato tweet, es momento de pasar a una fase de preprocesado de los datos donde podremos empezar a pensar cuáles son los recursos lingüísticos y estructuras que conviene tener en consideración y cuáles no. De esta forma más adelante le pasaremos a nuestro algoritmo de clasificación un texto ya procesado para que este haga la mejor predicción posible y clasifique de forma óptima los tweets según el sentimiento que estén queriendo transmitir los diferentes usuarios de la red social.

Para llevar a cabo esta fase de procesado de texto hemos utilizado la función disponible en el código base llamada "*basic_processing*", añadiendo sobre la misma las diferentes fases del preprocesado que hemos ido implementando paso a paso en este apartado.

3.2.1. Limpieza inicial y Tokenización

En la fase de preprocesado lo primero que se ha realizado sobre los tweet ha sido una limpieza inicial usando expresiones regulares y la tokenización de los tweets, para dividirlo y tratar de obtener las diferentes palabras, signos, símbolos y recursos lingüísticos que los componen (palabras, signos de puntuación, números, símbolos, etc).

Con la tarea de tokenización buscamos dividir grandes cadenas de texto solo y exclusivamente en palabras, para así poder pasar a valorar cada una de forma individual y cribar según lo que le indiquemos en las fases de normalización del texto.

```
[['rechazo', 'a', 'opción', 'radical', 'grande', 'ɛ', 's', 'ElecccionesGenerales28A', '28a'],
strucción', 'vestir', 'imagen', 'y', 'video', '😠']]
```

En la función "analisis_tweets" de nuestro código hicimos la decodificación a tipo string de los tweets, por lo que ya podemos pasar a la limpieza de los tweets. Para ello en la función "basic_processing" hemos realizado:

1. **Eliminación de menciones, hashtags y URLs:**

   Decidimos eliminar las menciones, hashtags y URLs de los tweets ya que en la mayoría de fuentes consultadas donde se realizan análisis de sentimientos en textos de tipo tweet, recomiendan la eliminación de estos elementos al añadir ruido y caracteres que no aportan nada de información al análisis lingüístico.

   Para detectar tanto hashtags como menciones hemos utilizado en nuestro código expresiones regulares (regex). En el caso de las URLs hemos tokenizado el texto y utilizado el método *.like_url* para detectar este tipo de estructura y luego hemos procedido a su eliminación.

   Como resultado obtenido tras la eliminación de estos elementos, podemos ver que la precisión del algoritmo de clasificación ha mejorado desde un accuracy base del 50.19% a uno del 51.71%:

```
🔵 🔴', ' Solo hace falta comerme otro spoiler de   que entonces ya me da algo . Luego ve
a usar las RRSS con normalidad 😌 😌 😌 😌 😌', ' Perdiendo la fe en la humanidad , otra v
campeón Barcelona y se acordaron de un equipo blanco . 🤣  ', ' A seis días sin luz , Vene

accuracy 0.5171102661596958
```

Las referencias consultadas para este apartado han sido las siguientes: [18], [19], [20], [21].

2. **Eliminación de emojis**:

   Como continuación de la tarea de preprocesado y debido a las dificultades técnicas devengadas del hecho de tener los tweets en español, hemos optado por la eliminación de los emojis.
   Para su eliminación hemos creado una función llamada "*give_emoji_free_text*" donde se ignoran los emojis por no tener una codificación tipo ASCII. Una vez eliminados de los textos vemos si la capacidad de predicción del clasificador mejora.

   Como resultado obtenido tras la eliminación de los emojis, podemos ver que la precisión del algoritmo de clasificación se mantiene en un accuracy del 51.71%. Esto puede ser debido a que, al no estar traducidos a texto, al algoritmo le da igual el hecho de mantenerlos o no ya que no llega a tenerlos en cuenta bien y no los clasifica.

```
' Mano grosera del central de USER   USER', ' Campeooooones campeooooones ol ol ol ! ! ! aLevante
o spoiler de   que entonces ya me da algo . Luego veo sin falta el captulo y as puedo volver a usar l
endo la fe en la humanidad , otra vez ms ...     ', ' Los memes saludaron al campen Barcelona y se ac
', ' A seis das sin luz , Venezuela anuncia restablecimiento total     ', ' No esperaba para nada ver

accuracy 0.5171102661596958
```

   En caso de haber optado por la alternativa de mantenerlos y traducirlos a texto, habría implicado en una primera fase traducirlos a texto en inglés, para posteriormente volverlos a traducir del inglés al español, ya que no hemos sido capaces de encontrar un diccionario que contenga de forma directa la traducción de los emojis en formato gráfico al castellano.

   Las referencias consultadas para este apartado han sido las siguientes: [14], [15], [16], [17].

3.2.2. Stopwords

Partiendo de la base de haber eliminado ya en la fase previa las menciones, emojis, hashtags y URLs, el siguiente paso realizado en el preprocesado de los tweets en añadido a ello, ha sido la eliminación de las stopword.

Las stopwords son palabras vacías ampliamente utilizadas pero que no suelen aportar demasiada información al texto que se quiere analizar ni valor al significado del mismo. Suelen ser palabras de tipo artículos, pronombres y preposiciones. Por ello, hemos optado por detectarlas con el método *.is_stop* y después eliminarlas y ver cómo eso afecta en la precisión del algoritmo de clasificación

Como resultado obtenido tras la eliminación de las stopwords, podemos ver que la precisión del algoritmo de clasificación se ha reducido levemente de un accuracy anterior del 51.71% al 51.33%. Esto puede ser debido a que, por la corta extensión de texto que tiene un tweet por definición, se pierde demasiada información con esta funcionalidad, por lo que deberemos analizar si realmente es una buena decisión el quitarlas o por el contrario las volveremos a incluir más adelante.

```
' falta comerme spoiler    . veo falta captulo y as volver a RRSS normalidad', ' Perdiendo f
ron campen Barcelona y acordaron equipo blanco .    ', ' A das luz , Venezuela anuncia restal
ado    ido ms ilusionada a votar esperaba . queda  ', '    USER Infeliz cundo aferra , hora q
ueblo y ponen a llorar    . mamar .', ' Presidente mexicano USER reitera llamado a " solucin
   ? iglesia corrupcin y pederasta catlica .', ' captulo     oscuro probado a hoja blanco a
u visitar ? | 3 escenarios    ', ' Cmo gente burla est pasando ? A falta demasiada cultura

  accuracy 0.5133079847908745
```

Las referencias consultadas para este apartado han sido las siguientes: [22], [23].

3.2.3. Signos de puntuación

Habiendo eliminado hasta el momento menciones, emojis, hashtags, URLs y stopwords, añadimos al procesado el análisis de los signos de puntuación y su aportación al texto, ya que es otra de las fases más utilizadas con frecuencia en NLP.
En textos de gran tamaño si se suelen quitar este tipo de signos para aportar claridad al texto, pero en el caso de textos de menor tamaño como los tweets, en ocasiones es preferible optar por mantenerlos ya que pueden aportan información sobre la emoción que se quiere transmitir y dar énfasis en la frase.

Utilizando el método *is_punct* para detectar los signos de puntuación, hemos optado por eliminarlos para así ver como afecta en nuestro caso. El resultado obtenido tras la eliminación de los signos de puntuación ha supuesto la no variación de la precisión del algoritmo de clasificación al mantenerse en un accuracy del 51.33% respecto a la fase anterior del análisis. Vistos estos resultados deberemos analizar si realmente es una buena decisión el quitarlos o por el contrario los volveremos a incluir más adelante.

```
Messigrado aLFC', ' Pablo Iglesias salv culo PSOE debate televisivo Rivera daba lados injusticia tremend
al poltico favorito votante izquierdas piensa PSOE reformas laborales a frenar fascismo', ' VIDEO As tie
coronarse Campen   2018-2019', ' guerra ea matanza PUTA MADRE A LLORAR  ', ' Cosa ms leer lindo libro
ms asqueroso malditas Oriana a Gahona rpido paso luto   y Sofa clsicas respuestas 2 neuronas ms cerebro'
' triste imgenes   llamas  ', ' captulo debi llamar follada salvar  ', ' pasaron y Rayo gan Real      '

  accuracy 0.5133079847908745
```

Las referencias consultadas para este apartado han sido las siguientes: [18], [19], [20], [21].

3.2.4. Símbolos, Números y otros caracteres

Finalmente con los resultados obtenidos hasta el momento, hemos decidido continuar eliminando los hashtag, menciones, emojis y URLs, pero a partir de este momento volveremos a incluir las stop words y los signos de puntuación por su baja efectividad en las fases donde los hemos eliminado.

Tras ello, hemos considerado razonable tratar de eliminar los posibles símbolos (SYM - $, %, §, ©, +, −, ×, ÷, =, :), números (NUM - 1, 2017, one, seventy-seven, IV, MMXIV) y otros caracteres raros que pueden aparecen en los tweets (X - sfpksdpsxmsa), que realmente no aportan demasiada información al texto, y sin embargo sí pueden añadir cierto ruido de cara a que el algoritmo realiza una predicción precisa. Para detectar y eliminar estos elementos hemos utilizado el método *.pos_*

Como resultado obtenido tras la eliminación de estos elementos (e incluyendo de nuevo las stopwords y los signos de puntuación), podemos ver que la precisión del algoritmo de clasificación se ha incrementado levemente de un accuracy anterior del 51.33% al 52.85%.

```
  tengo aos y vengo a votar con una ilusin brbara entraable no , lo siguiente', ' gracias a puto icl
    edd el penas , el lord comandante , deja su guardia y descansa con sus otros hermanos juramentad
ɨ ya no vieron la luz del da .', ' aprecio el esfuerzo de   por defender el medio ambiente me inquie
ıacin a la hora de convertirla en icono user', ' lo positivo para m es que triunfa la diversidad y l
· tenga un plan serio para dar un golpe contundente al usurpador en las prximas horas , sobre todo m

accuracy 0.5285171102661597
```

Las referencias consultadas para este apartado han sido las siguientes: [13].

3.2.5. Lematización

Partiendo de la base de un texto limpio de hashtag, menciones, emojis, URLs, símbolos, números y otros caracteres, pasamos a analizar cómo puede llegar a influir la lematización sobre los tweets.

Por lo tanto, pasamos a realizar una lematización del texto mediante el método *.lemma*, el cual consiste en transformar las palabras eliminando las terminaciones flexivas y devolviendo la forma base de la palabra, la cual se conoce como lema. Este proceso puede ser útil para realizar una normalización de las diferentes construcciones que aparecen en el texto sobre las palabras con una estructura básica idéntica. Puede valer para simplificarlo aunque en ocasiones supone una reducción demasiado grande de información.

Como resultado obtenido tras la lematización de los tweets, podemos ver que la precisión del algoritmo de clasificación se ha reducido levemente de un accuracy anterior del 52.85% al 52.47%. Creemos que al ser los tweets una serie de textos con una densidad de palabras muy reducida, el hecho de lematizar quita demasiada información para la futura predicción, no aportando el efecto positivo que buscábamos.

```
' Francia est herido ! Europa est triste ! el mundo est conmocionado !  ', ' estar de acuerdo que el  él
ıo y pues ya poder ir contndo él el campeonato de USER Pero ser honesto y fuera de su fanatismo responder
ı Messi Podra ser siquiera campar ?', ' si no haber accin , no haber esperanza !  ', ' hoy  recuerdo que
de hacer uno ao , aunque seguro haber pasar mas tiempo .', ' Maneras de ver , uno suceso para el reflexin
ɔdo y hoy él celebramos . Feliz dar del libro !  ', '  él que maas yo alegrar ea que el  ser campeon de
ıo bolsa , desgracia para el uniforme , que tú tener que escribir él que ir a decir , tu ar desde el Imper

accuracy 0.5247148288973384
```

Las referencias consultadas para este apartado han sido las siguientes: [24], [25].

3.2.6. Determinantes y preposiciones

Con los resultados obtenidos con la lematización, hemos decidido continuar con el texto limpio de hashtag, menciones, emojis, URLs, símbolos, números y otros caracteres, pero manteniendo los tweets como texto sin lematizar.

Hemos querido también, ver qué efecto tienen sobre el texto palabras de tipo determinante (DET - la, lo, un, este, esa, aquellos, algún, muchos, etc) y de tipo preposición (ADP - a, ante, entre, hacia, según, sin, hasta , para, por, etc), que aunque están entre las más frecuentes en los tweets, y puede terminar siendo negativo eliminarlas pues se reduce demasiado la información en el texto, es interesante ver si es positivo su eliminación o por el contrario conviene mantenerlas. Para detectar y eliminar del texto estos elementos hemos utilizado el método *.pos_*.

Como resultado obtenido tras la eliminación de determinantes y preposiciones de los tweets, podemos ver que la precisión del algoritmo de clasificación se ha reducido levemente de un accuracy anterior del 52.85% al 52.09%. Finalmente vemos que si tiene una repercusión negativa y se pierde información, por lo que optamos por su no eliminación por el momento, y se deberá valorar en mayor profundidad en futuras fases del procesado.

```
' Lo que mas me alegra todo es ver caida PP', ' catlicos ser prdida que fe haremos reconstruir
no catlicos Notre-Dame Paris ser prdida irreparable . Que nos costar volverla ver como antes . J
', ' Les dur poco show . Capturados y gatos que robaron tanquetas y atropellaron gente . Guiad

accuracy 0.5209125475285171
```

Las referencias consultadas para este apartado han sido las siguientes: [13].

3.2.7. Conjunciones y pronombres

Con los resultados obtenidos en la fase anterior, hemos decidido continuar con el texto limpio de hashtag, menciones, emojis, URLs, símbolos, números y otros caracteres, pero manteniendo los tweets como texto sin quitar los determinantes y las preposiciones.

Otros de los elementos a analizar que como hemos visto anteriormente aparecen con cierta frecuencia en nuestros tweets son las conjunciones (CCONJ, CONJ y SCONJ - y, o, pero, si, durante, qué, etc) y los pronombres (PRON - ella, el, tú, yo, etc). Este tipo de palabras, como hemos visto en otros casos, pueden llegar a aportar ruido al texto y poca información, llegando a repercutir negativamente en la capacidad predictiva del algoritmo de clasificación. Para detectar y eliminar del texto estos elementos hemos utilizado el método *.pos_*

Como resultado obtenido tras la eliminación de conjunciones y pronombres de los tweets, podemos ver que la precisión del algoritmo de clasificación se ha reducido levemente de un accuracy anterior del 52.85% al 52.09%. Finalmente vemos que si tiene una repercusión negativa y se pierde información, por lo que optamos por su no eliminación por el momento, y se deberá valorar en mayor profundidad en futuras fases del procesado.

```
 El puto amo Leo Messi   goles  ', ' Espaa une dice no al racismo , a la xenofobia , al machismo , a
do a la involucin . , por primera vez en aos , siento orgullosa .', '    BTS USER mejor ayudar a las
ao !', ' queda poco respuesta preferida a todos los tuits de la gente de izquierdas . Poco para ? ...

accuracy 0.5209125475285171
```

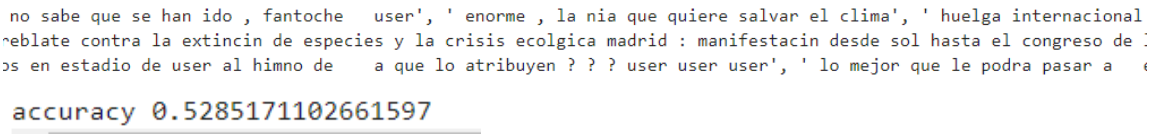
Las referencias consultadas para este apartado han sido las siguientes: [13].

3.2.8. Transformación del texto a minúsculas

Hemos visto que en la mayoría de trabajos de análisis y procesado de texto, una de las técnicas más utilizadas para terminar de realizar una normalización del mismo, es la transformación de todos los caracteres alfabéticos a un estado homogéneo, ya sea todo en mayúsculas o todo en minúsculas.

Con ello y partiendo de la base de tener el texto limpio de hashtag, menciones, emojis, URLs, símbolos, números y otros caracteres, pero manteniendo los tweets como texto sin quitar las conjunciones y los pronombres, hemos decidido normalizar los tweets pasando todos los caracteres a minúsculas. Para tal cometido hemos utilizado el método *.lower()*

Como resultado obtenido tras la transformación a minúsculas de los caracteres de los tweets, podemos ver que la precisión del algoritmo de clasificación se ha mantenido el accuracy anterior del 52.85%.



Las referencias consultadas para este apartado han sido las siguientes: [18], [19], [20], [21], [26].

A pesar de los modestos resultados obtenidos hasta el momento en la mejora de la precisión de nuestro algoritmo de clasificación a través del preprocesado de texto, las conclusiones generales que podemos sacar hasta ahora son:

- Es positivo quitar las menciones (@usuario), los hashtag (#hashtag) y los emojis (en caso de no traducirlos a texto en castellano), ya que no están aportando información al texto y si están generando cierto nivel de ruido.

- La eliminación de stopwords y signos de puntuación, no ha sido satisfactoria en el caso de un texto en forma de tweet, ya que por sus características de corta extensión y pocos caracteres utilizados, creemos que reduce en exceso la información y no termina siendo positivo para la precisión del predictor. En adición a esto, los signos de puntuación pueden dar cierta información de efusividad (¡!), duda (¿?) y otros matices que sí pueden aportar a la emoción que desean transmitir los usuarios a través de los tweets.

- Por el mismo motivo anterior de que los tweets son un tipo de texto muy concreto y de poca extensión, no ha resultado positivo tratar de lematizar las palabras. Se pierde mucha información al aplicar esta técnica.

- En siguientes fases del trabajo tendremos que analizar de forma más detallada el caso de conjunciones, pronombres, determinantes, preposiciones y números, ya que en principio se puede llegar a la conclusión de que no aportan mucha información al texto y que se deben eliminar. Pero nuevamente, al estar ante un texto especial como son los tweets, puede ser que algunos de ellos si sea beneficioso mantenerlos en vez de optar por quitarlos, ya que como además vimos, son un tipo de palabra que aparece de forma recurrente en nuestros tweets.

- Creemos que la normalización del texto a minúsculas puede ser interesante de mantener ya que se homogenizan las diferentes palabras de los tweets y ayuda a simplificar el procesado y la predicción final.

En base a todo este análisis y a las conclusiones obtenidas por el momento, pasaremos a analizar qué tipo de clasificador nos conviene más de cara a mejorar la predicción final de las emociones transmitidas por los usuarios a través de los tweets.

3.3. Algoritmos de clasificación

Una vez hemos entendido las características que tienen un tweet como texto, hemos analizado la estructura y tipo de palabras específicas y características de nuestros datos, y hemos completado la tarea de limpiarlos y procesarlos, queda entender de qué algoritmo de clasificación encaja de la mejor manera en base a las diferentes 7 emociones de las que disponemos (ANGER, DISGUST, FEAR, JOY, SADNESS, SURPRISE y OTHERS) y función del tipo de dato que hemos procesado.

Para llevar a cabo esta fase de clasificación hemos utilizado la función disponible en el código base llamada "*classifier*", modificando sobre la misma los diferentes parámetros de vectorización y transformación que se pueden utilizar e incorporando los diferentes algoritmos de clasificación implementados.

Como punto de partida teníamos una fase de procesado donde se introducían los tweets en bruto sin realizar ningún tipo de limpieza y preprocesado sobre ellos, y una fase de clasificación donde se utiliza el algoritmo de aprendizaje supervisado Naïve-Bayes más básico ("BernoulliNB"). En base a estos los resultados de precisión de los que partíamos eran de un 50.19%:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	31
1	0.00	0.00	0.00	7
2	0.00	0.00	0.00	5
3	0.60	0.05	0.09	59
4	0.50	0.98	0.66	131
5	0.00	0.00	0.00	22
6	0.00	0.00	0.00	8
accuracy			0.50	263
macro avg	0.16	0.15	0.11	263
weighted avg	0.38	0.50	0.35	263

---

accuracy 0.5019011406844106

En nuestro caso finalmente, la configuración elegida para el procesado final del texto, la cual creemos que es la que tiene más sentido en base a todo lo analizado anteriormente, ha sido:

- *Eliminamos*:
  - Emojis (en forma gráfica sin transformar a texto y traducir al castellano), menciones, hashtags y URLs ya que como hemos comentado con anterioridad no aportan nada al texto.
  - Números, símbolos y caracteres especiales ya que añaden complejidad al texto y aportan escasa información de relevancia.
  - Determinantes y pronombres, que a pesar de ser un tipo de palabra abundante en nuestros datos, no tiene quizá el carácter informativo necesario que buscamos y aporta ruido de cara a la precisión de la predicción.

- *Mantenemos*:
    - Verbos, verbos auxiliares, nombres, nombres propios y adjetivos, ya que son los tres grupos principales de palabras en un texto, los que aportan mayor información, y los que son la base del lenguaje.
    - Conjunciones y preposiciones ya que a pesar de poderse entender como palabras que aportan un bajo nivel de información en general, en nuestro caso al estar analizando tweets que son textos de corta extensión, consideramos que lo mejor es mantenerlos.
    - Signos de puntuación al considerarlos relevantes a la hora de expresar sentimientos y emociones en frases o textos cortos como son los tweets.

- Optamos por no lematizar las palabras y mantenerlas como texto, añadiendo la normalización de transformar todas las letras de las diferentes palabras de los tweets a minúsculas.

Utilizando esta configuración mencionada, realizaremos las tareas de clasificación de cara a obtener la mejor predicción posible y entender cuál es el algoritmo de aprendizaje supervisado que mejor se ajusta a nuestros datos. Todo en base a los tipos de algoritmos más utilizados en la tareas de análisis de sentimientos y emociones en textos de tipo tweet, y usando un 80% de los datos como train y un 20% como test.

Los clasificadores elegidos para probarlos en este trabajo han sido seleccionados en base a diferentes trabajos de investigación y artículos consultados. Hemos visto cuáles son los tipos de algoritmos más utilizados para la tarea de clasificación de textos y análisis de sentimientos y emociones en tweets.

Las referencias consultadas para este apartado han sido las siguientes: [27], [28], [29], [30], [31], [32], [33].

3.3.1. Algoritmos de clasificación - Naïve-Bayes

Este tipo de algoritmos de clasificación, basados en los conceptos de conocimiento a priori, probabilidad condicional y el teorema de Bayes, son ampliamente utilizados en tareas de clasificación en el área de aprendizaje automático con textos por su simplicidad y su potencia. En nuestro caso probaremos con BernoulliNB (el usado de base) y con MultinomialNB, sacados de scikitlearn con “sklearn.naive_bayes” y utilizando los métodos “BernoulliNB().fit” y “MultinomialNB().fit” respectivamente.

- **BernoulliNB:**

Es un modelo teórico basado en la distribución binomial donde el posible resultado de los experimentos se base sólo en dos posibles sucesos de éxito o no éxito, y en base a eso realiza la clasificación. Es ampliamente utilizado en tareas básicas de clasificación de textos por su simplicidad y su facilidad de interpretación.

Los resultados obtenidos con este algoritmo han sido positivos mejorando con la tarea de procesado de datos realizada, pasando de un valor de accuracy del 50.19% al 51.71%.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	31
1	0.00	0.00	0.00	7
2	0.00	0.00	0.00	5
3	0.83	0.08	0.15	59
4	0.51	1.00	0.68	131
5	0.00	0.00	0.00	22
6	0.00	0.00	0.00	8
accuracy			0.52	263
macro avg	0.19	0.15	0.12	263
weighted avg	0.44	0.52	0.37	263

```
----------------------------------------
accuracy 0.5171102661596958
```

- **MultinomialNB:**

Basado en una generalización del método de regresión logística, es utilizado en problemas multiclase donde existen más de dos resultados discretos.

Los resultados obtenidos con este algoritmo no han sido positivos, empeorando los resultados anteriores en Bernoulli de un 51.71% a un 50.57%. Aunque sí que siguen siendo mejores que los de base.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	31
1	0.00	0.00	0.00	7
2	0.00	0.00	0.00	5
3	1.00	0.02	0.03	59
4	0.50	1.00	0.67	131
5	1.00	0.05	0.09	22
6	0.00	0.00	0.00	8
accuracy			0.51	263
macro avg	0.36	0.15	0.11	263
weighted avg	0.56	0.51	0.35	263

```
----------------------------------------
accuracy 0.5057034220532319
```

3.3.2. Algoritmos de clasificación - Tree

Este tipo de algoritmos de clasificación, basados en los conceptos de máxima entropía y reducción de la incertidumbre, son también ampliamente utilizados en tareas de clasificación de textos. En nuestro caso probaremos con DecisionTreeClassifier y ExtraTreeClassifier, sacados de scikitlearn con “sklearn.tree” y utilizando los métodos “DecisionTreeClassifier().fit” y “ExtraTreeClassifier().fit” respectivamente.

- **DecisionTreeClassifier:**

Los resultados obtenidos con este algoritmo no han sido positivos, empeorando los resultados anteriores en Bernoulli de un 51.71% a un 48.66%. Empeorando incluso los resultados de base.

	precision	recall	f1-score	support
0	0.24	0.19	0.21	31
1	1.00	0.14	0.25	7
2	0.67	0.40	0.50	5
3	0.40	0.42	0.41	59
4	0.56	0.63	0.59	131
5	0.61	0.50	0.55	22
6	0.14	0.12	0.13	8
accuracy			0.49	263
macro avg	0.52	0.34	0.38	263
weighted avg	0.49	0.49	0.48	263

```
----------------------------------------
accuracy 0.4866920152091255
```

- **ExtraTreeClassifier:**

Los resultados obtenidos con este algoritmo no han sido positivos, empeorando los resultados anteriores en Bernoulli de un 51.71% a un 49.04%. Empeorando también los resultados de base.

	precision	recall	f1-score	support
0	0.12	0.03	0.05	31
1	0.33	0.14	0.20	7
2	0.00	0.00	0.00	5
3	0.35	0.29	0.32	59
4	0.56	0.76	0.65	131
5	0.48	0.45	0.47	22
6	0.00	0.00	0.00	8
accuracy			0.49	263
macro avg	0.26	0.24	0.24	263
weighted avg	0.42	0.49	0.44	263

```
----------------------------------------
accuracy 0.49049429657794674
```

3.3.3. Algoritmos de clasificación - Neighbors

Este tipo de algoritmos de clasificación, basados en los conceptos de distancias y vecinos más cercanos, son también ampliamente utilizados en tareas de clasificación. En nuestro caso probaremos con KNeighborsClassifier, sacado de scikitlearn con “sklearn.neighbors” y utilizando el método “KNeighborsClassifier().fit”.

- **KNeighborsClassifier:**

Los resultados obtenidos con este algoritmo han sido positivos, empeorando ligeramente los resultados anteriores en Bernoulli de un 51.71% a un 51.33%, pero mejorando también los resultados de base.

	precision	recall	f1-score	support
0	0.27	0.13	0.17	31
1	0.00	0.00	0.00	7
2	0.60	0.60	0.60	5
3	0.43	0.61	0.50	59
4	0.59	0.65	0.62	131
5	0.50	0.32	0.39	22
6	0.00	0.00	0.00	8
accuracy			0.51	263
macro avg	0.34	0.33	0.33	263
weighted avg	0.48	0.51	0.49	263

```
----------------------------------------
accuracy 0.5133079847908745
```

3.3.4. Algoritmos de clasificación - SVM

Este tipo de algoritmos de clasificación, basados en los conceptos de hiperplanos, vectores de soporte, márgenes óptimos y Kernels, son igualmente usados en tareas de clasificación de texto. En nuestro caso probaremos con LinearSVC, sacado de scikitlearn con “sklearn.svm” y utilizando el método “LinearSVC().fit”.

- **LinearSVC:**

Los resultados obtenidos con este algoritmo han sido muy positivos, mejorando en gran medida los resultados anteriores en Bernoulli de un 51.71% a un 61.21%, y mejorando también de forma muy significativa los resultados de base.

	precision	recall	f1-score	support
0	0.33	0.06	0.11	31
1	0.00	0.00	0.00	7
2	1.00	0.40	0.57	5
3	0.62	0.49	0.55	59
4	0.61	0.89	0.72	131
5	0.79	0.50	0.61	22
6	0.25	0.12	0.17	8
accuracy			0.61	263
macro avg	0.51	0.35	0.39	263
weighted avg	0.57	0.61	0.56	263

```
----------------------------------------
accuracy 0.6121673003802282
```

Atendiendo a los resultados con la configuración final seleccionada para el procesado de los tweets, optamos por quedarnos para la clasificación de las emociones de los usuarios con la opción de LinearSVC como algoritmo supervisado de predicción, ya que es el que nos da el mejor resultado final entre todas las alternativas probadas.

3.4. Vectorización

Por último, tras realizar el procesado de los tweets y decidir el mejor algoritmo de clasificación para nuestros datos, trataremos de realizar diferentes configuraciones de vectorización para analizar si de esa forma conseguimos mejorar en un mayor grados nuestros resultados de precisión.

Esta fase de vectorización la realizamos dentro de la función "*classifier*", modificando sobre la misma los diferentes parámetros que se pueden utilizar con el método *CountVectorizer.*

Hasta el momento, hemos venido aplicando la vectorización TF-IDF implementada en el código base (configuración tomando exclusivamente unigramas). En los siguientes pasos trataremos de probar:

- TF-IDF con configuración en *CountVectorizer()* del parámetro *ngram_range*
- TF con configuración en *CountVectorizer()* del parámetro *ngram_range*
- Binario con configuración en *CountVectorizer()* del parámetro *ngram_range*

Una forma de enriquecer las tareas de clasificación de textos, es utilizar n-gramas donde n > 1. En general, los bi-gramas y tri-gramas pueden capturar mayor información contextual en comparación con los unigramas. Además, para tareas como la extracción de palabras clave, los unigramas por sí solos, aunque son útiles, proporcionan una información algo limitada.

La configuraciones finalmente probadas para el parámetro *ngram_range* ha sido unigramas y bi-gramas (1,2) y unigramas y tri_gramas (1,3):

3.4.1. TF-IDF con parámetro ngram_range (1, 2) y (1, 3)

Teniendo en cuenta que TF analiza la frecuencia de aparición de las diferentes palabras dentro del corpora (cuenta las palabras, siendo las más influyentes las que más aparecen), y considerando que la frecuencia inversa del documento (IDF) añade a TF una penalización en forma de ponderación negativa en el caso de que el término aparezca de forma recurrente y constante en todos los documentos. Los resultados obtenidos con esta vectorización y con las diferentes configuraciones de n-gramas han sido los siguientes:

- **Unigramas y trigramas (1, 3)**

```
accuracy 0.5361216730038023
```

- **Unigramas y bigramas (1, 2)**

```
accuracy 0.5817490494296578
```

Vemos que en ambos casos empeora los resultados de la precisión del accuracy con la configuración de vectorización inicial. Pasando de 61.21% a 53.61% en el primer caso y a 58.17% en el segundo.

3.4.2. TF con parámetro ngram_range (1, 2) y (1, 3)

Teniendo en cuenta TF y con las diferentes configuraciones de n-gramas, los resultados obtenidos han sido los siguientes:

- **Unigramas y trigramas (1, 3)**

```
accuracy 0.5893536121673004
```

- **Unigramas y bigramas (1, 2)**

```
accuracy 0.5627376425855514
```

Vemos que en ambos casos empeora los resultados de la precisión del accuracy con la configuración de vectorización inicial. Pasando de 61.21% a 58.93% en el primer caso y a 56.27% en el segundo.

3.4.3. Binario con parámetro ngram_range (1, 2) y (1, 3)

Activando el parámetro *binary* en método *CountVectorizer* ya no se tendría en cuenta la frecuencia de la palabra y por tanto se le daría la misma importancia a todas las palabras a la hora de la clasificación independientemente de si aparecen muchas veces o no en el texto. Teniendo en cuenta el pesado binario y con las diferentes configuraciones de n-gramas, los resultados obtenidos han sido los siguientes:

- **Unigramas y trigramas (1, 3)**

```
accuracy 0.5703422053231939
```

- **Unigramas y bigramas (1, 2)**

```
accuracy 0.5741444866920152
```

Vemos que en ambos casos empeora los resultados de la precisión del accuracy con la configuración de vectorización inicial. Pasando de 61.21% a 57.03% en el primer caso y a 57.41% en el segundo.

Después de realizar este análisis de vectorización la configuración que arroja los mejores resultados de predicción en la clasificación es la utilizada en el código base con TF-IDF y unigramas.

Las referencias consultadas para este apartado han sido las siguientes: [34], [35], [36], [37], [38], [39], [40].

4. Conclusiones generales del estudio

- Es de gran relevancia realizar un análisis exploratorio previo de los datos para conocer las características principales del texto que se quiere procesar y clasificar, ya que es la clave que llevará a una óptima limpieza y preprocesado del texto y a una mejora en la precisión de la clasificación del mismo.

- Al procesar tweets debemos tener siempre muy presente sus características y la corta extensión de su texto, ya que es algo que puede limitar y ser decisivo a la hora de elegir qué técnicas de preprocesado aplicar y cuáles no. Un ejemplo de ello, es la aplicación de técnicas como la lematización y el análisis POS, ya que si se realiza una limpieza y preprocesado de los tweets muy agresiva a través de las mismas, quitaremos ruido pero perderemos demasiada información. Como resultado el algoritmo no sabrá buscar las palabras clave que definan y caractericen a cada emoción y la precisión en los resultados bajará considerablemente.

- Los emojis son elementos gráficos que realmente pueden llegar a aportar información a un tweet, tanto reforzando el mensaje o la emoción que se quiere transmitir, como de forma irónica dando un mensaje opuesto a lo transmitido con palabras escritas. Su tratamiento y traducción se puede complicar siendo menos accesibles los diccionarios de emojis disponibles en idioma español.

- En líneas generales las herramientas encontradas para el procesado de texto están mucho más pulidas y mejor desarrolladas para el idioma inglés que para el castellano.

- De cara a mejorar la precisión en la clasificación de los tweets sería conveniente tener una fase de train donde hubiera mayor número de tweets de los que pudieran aprender los algoritmos aplicados. Creemos que con ello se mejoraría en gran medida los resultados obtenidos.

- Otra posibilidad para mejorar la precisión en la clasificación de las emociones sería tratar de utilizar algoritmos y técnicas más complejas y potentes relacionadas con el

deep learning y las redes neuronales, ya que hemos visto otros trabajos desarrollados en este campo que así lo hacen.

5. Referencias y bibliografía

- Libraría Spacy

[1] Language Processing Pipelines · spaCy Usage Documentation
[2] Comenzando con spaCy para procesamiento de lenguaje natural | by Matthew Mayo | Ciencia y Datos | Medium

- Análisis de un tweet como texto:

[3] 2.3. Estructura de Twitter y de un tweet – Redes Sociales en la Enseñanza (upm.es)
[4] ¿Qué es un Tweet? Todo lo que necesitas saber - Ryte Wiki

- Reconocimiento de Entidades Nombradas (NER):

[5] https://spacy.io/usage/spacy-101#annotations-ner
[6] Named Entity Recognition (NER) in Spacy Library - MLK - Machine Learning Knowledge
[7] https://sitiobigdata.com/2018/08/27/guia-de-practicante-para-nlp/#
[8] Reconocimiento de entidades nombradas: Una breve introducción – SoldAI
[9] Reconocimiento de entidades nombradas (Named Entity Recognition-NER), un ejemplo básico | by Miguel Ángel Flores | Medium
[10] Python | Reconocimiento de entidad nombrada(NER) usando spaCy – Acervo Lima

- Categorización de tipos de palabras (POS):

[11] https://spacy.io/usage/linguistic-features
[12] https://programminghistorian.org/es/lecciones/contar-frecuencias
[13] Tutorial on Spacy Part of Speech (POS) Tagging - MLK - Machine Learning Knowledge

- Desambiguación y eliminación de emojis:

[14] https://stackabuse.com/convert-bytes-to-string-in-python/
[15] https://es.acervolima.com/convierte-emoji-en-texto-en-python/
[16] Emoji unicode characters for use on the web (timwhitlock.info)
[17] Remove emojis and @users from a list in Python and punctuation, NLP problem, and my emoji function does not work - Stack Overflow

- Limpieza y preprocesado de texto:

[18] Context-specific Pre-processing for NLP with spaCy: Tweets | by Wei-Ting Yap | Towards Data Science
[19] NLP_with_Python/04_Preprocesamiento_de_textos_Normalizacion.ipynb at master · RicardoMoya/NLP_with_Python (github.com)
[20] spaCy increíble para procesar y limpiar tweets (jairoandres.com)
[21] Primeros pasos en NLP con spaCy, un vistazo general. | by Gerardo Fosado | Medium

- Stopwords:

[22] Tutorial for Stopwords in Spacy Library - MLK - Machine Learning Knowledge
[23] How To Remove Stopwords In Python | Stemming and Lemmatization (analyticsvidhya.com)

- Lematización:

[24] Python: enfoques de lematización con ejemplos – Acervo Lima
[25] ▷ Python para PNL: tokenización, derivación y lematización con la biblioteca SpaCy | Pharos

- Transformación de texto en minúsculas:

[26] Limpieza de datos de texto con Python (ichi.pro)

- Análisis y procesado de texto y algoritmos de clasificación:

[27] spaCy Tutorial - Learn all of spaCy in One Complete Writeup | ML+ (machinelearningplus.com)
[28] Análisis de sentimientos en Twitter (uoc.edu)
[29] Caso de estudio de análisis de sentimientos en Twitter: Tratado de libre comercio de América del Norte (ipn.mx)
[30] https://github.com/taljuk01/political_sentiment_analysis
[31] https://github.com/RicardoMoya/NLP_with_Python
[32] API Reference — scikit-learn 1.0.2 documentation
[33] (PDF) Clasificación de Textos Multi-etiquetados con Modelo Bernoulli Multi-variado y Representación Dependiente de la Etiqueta | Rodrigo Alfaro - Academia.edu

- Vectorización:

[34] https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c
[35] How to Use Tfidftransformer & Tfidfvectorizer - A Short Tutorial - Kavita Ganesan, PhD (kavita-ganesan.com)
[36] sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 1.0.2 documentation
[37] sklearn.feature_extraction.text.TfidfTransformer — scikit-learn 1.0.2 documentation
[38] sklearn.feature_extraction.text.CountVectorizer — scikit-learn 1.0.2 documentation
[39] https://kavita-ganesan.com/how-to-use-countvectorizer/#.Yjbt_H_MKV4
[40] sklearn.feature_extraction.text.CountVectorizer — scikit-learn 1.0.2 documentation