Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

ÁREA DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

ORDENACIÓN Y SELECCIÓN DE TUITS RELACIONADOS CON UN CONCEPTO MEDIANTE INTERFAZ EN LENGUAJE NATURAL Y PREFERENCIAS

D. CASTAÑO DORADO, Pablo
TUTOR: D. QUEVEDO PÉREZ, José Ramón

FECHA: Junio de 2020

Índice

Listado de figuras

FIGURAS

TABLAS

FÓRMULAS

1. Introducción: Ámbito y Estado del arte.

Cuando hablamos de Procesamiento del Lenguaje Natural (NLP) [1]–[4] nos referimos a la rama de la inteligencia artificial que permite a un computador entender e interpretar el lenguaje humano. Los humanos podemos comunicarnos de diferentes formas y en diferentes lenguajes y generamos una gran cantidad de información desestructurada. El objetivo es descifrar, comprender y dar sentido al lenguaje utilizado por los seres humanos de una forma que pueda producir valor.

La comunicación humana es muy compleja ya que existen multitud de variables que pueden dar un significado u otro a lo que estemos expresando, como puede ser el uso de gestos o la entonación. Este proyecto partirá de la clasificación de lenguaje escrito, que si bien pierde la información extra que nos da por ejemplo un gesto debido a que se centra solamente en el mensaje que aparece escrito, es suficiente para desarrollar el tema del proyecto, la ordenación y selección de tuits relacionados con un concepto.

El lenguaje escrito se usa de forma masiva a nivel mundial a través del uso de redes sociales, foros, comentarios, e-mail, etc. Existen diversas técnicas o algoritmos para la realización de tareas de NLP sobre un texto. Estas técnicas podemos agruparlas en aquellas técnicas utilizadas para tratar el texto y en las utilizadas para crear atributos a partir de texto. Algunas de estas técnicas son las siguientes:

- Técnicas para el tratamiento del texto:
    - Tokenización **[5]**: Consiste en separar las palabras que forman un texto en entidades básicas llamadas *tokens*. No es tan sencillo como determinar que cada palabra de un texto es un *token,* sino que requiere un estudio previo para determinar qué se considera *token* y que no. Algunos ejemplos es considerar si

separamos mayúsculas o minúsculas, si tenemos en cuenta signos de puntuación, caracteres numéricos, etc.

- Eliminación de *stopwords*: Consiste en eliminar determinadas palabras que no aportan información al texto y que su uso es muy común. No existe una lista establecida de *stopwords*, pero en el lenguaje castellano podemos considerar preposiciones, pronombres, artículos… como *stopwords*.
- Lematización **[6]**: Proceso consistente en transformar una palabra a su raíz, de forma que palabras diferentes con un significado similar quedarán transformadas en la misma palabra. Un ejemplo sería la lematización de las palabras "estudiar", "estudiante" y "estudio" que ambas quedarían transformadas en su raíz "estudi".
  A esta técnica le podemos dar una mayor complejidad evitando las ambigüedades entre palabras logrando que dos palabras iguales tengan significados distintos según su contexto. Esto logrará mejores resultados, pero también requiere un poder computacional mucho mayor.

- Técnicas para la creación de atributos a partir de texto:
  - Modelo de bolsa de palabras **[7]**: Nos permite contar las palabras que aparecen en un texto y representar una matriz con las ocurrencias de dichas palabras.
  - Words2vec [8]: Conjunto de técnicas que permiten leer un texto, descomponerlo en tokens y transformarlos en vectores de forma que palabras similares tendrán vectores similares.

Algunos ejemplos de uso de estas técnicas o combinaciones de ellas en la vida real pueden ser las siguientes:

- Análisis de sentimiento: Puede ayudar a determinar si por ejemplo un cliente está satisfecho o no con su pedido.
- Traducciones automáticas de un lenguaje a otro.
- Predicción de enfermedades mediante el procesamiento de registros o reportes de los pacientes.

- Descubrimiento y modelado de temas.
- Identificación de *fake news* y detección de spam en los correos.

Como hemos podido observar el NLP puede utilizarse en diversos ámbitos y mediante numerosas técnicas. A continuación, se presentará el objeto o ámbito de este proyecto en particular. Además, se presentará un ejemplo de aplicación real bastante similar al de este proyecto que ha sido implementado recientemente y que puede ser de interés de cara a entender este proyecto.

Más adelante, a lo largo de los distintos apartados podremos ver cómo se ha realizado el diseño y estudio del problema, las distintas fases de experimentación realizadas y se presentarán las soluciones finales implementadas.

Finalmente, se mencionarán las conclusiones obtenidas y posibles ampliaciones futuras que podrán ser interesantes de cara a incrementar las funcionalidades de la aplicación desarrollada.

1.1.- ÁMBITO DEL PROYECTO Y OBJETIVOS.

Anteriormente se ha mencionado que el NLP puede utilizarse en diversos ámbitos. En este proyecto nos centraremos en uno de ellos, la clasificación de texto escrito en una serie de categorías preestablecidas.

Si hablamos de clasificación de texto, el concepto sigue siendo muy amplio ya que el texto escrito está presente en multitud de lugares y formatos diferentes, como puede en libros, periódicos, páginas web, redes sociales… Como en la actualidad prácticamente todo el mundo tiene acceso a Internet y es usuario de diferentes redes sociales, nos hemos centrado en una de las

principales redes sociales en la cual la gente interactúa constantemente comentando acerca de diferentes temas. Esta red social es Twitter [9], [10].

El siguiente paso era plantearse qué hacer con toda la información disponible en Twitter, cómo clasificarla y qué lograr implementar que sea atractivo para los usuarios. Tras una fase de planteamiento del problema se decidió definir una serie de temas que pudiesen descomponerse en un conjunto de posibles subcategorías referentes a cada tema y clasificar futuros tuits o consultas de usuarios en dichos temas además de mostrarles otros tuits de otros usuarios que tengan la misma temática y que le puedan resultar interesantes al usuario que escribe la consulta.

Para la solución propuesta, se plantearon una serie de objetivos a cumplir que serán mencionados a continuación:

- Definición de los principales temas a tratar.
- Definición de subcategorías pertenecientes a cada tema. Como el proyecto se ha centrado en la red social *Twitter*, cada una de las subcategorías será un *hashtag* [11] con información referente a un determinado tema.
- Definición del *corpus*, extracción de tuits de cada una de las subcategorías.
- Procesamiento del Lenguaje Natural e implementación de un algoritmo clasificador.
- Clasificación de texto en Lenguaje Natural en los temas preestablecidos.
- Elección de tuits del *corpus* que se correspondan con los temas predichos en el clasificador.

A lo largo de esta memoria se irán explicando detalladamente las diversas formas de resolver los distintos objetivos y cómo se ha resulto cada uno de ellos finalmente.

1.2.- ESTADO DEL ARTE.

A lo largo de la historia el Procesamiento de Lenguaje Natural ha ido evolucionando progresivamente desde el primer software capaz de traducir automáticamente 60 expresiones del ruso al inglés documentadas en 1954[12], [13], los primeros *Chatbots* en 1972, el desarrollo de los análisis de sentimientos, la introducción de algoritmos de *Machine Learning* en los años 80 y más recientemente la introducción de las redes neuronales en el siglo XXI.

El Procesamiento del Lenguaje Natural es un campo que evoluciona continuamente. Uno de los ejemplos más interesantes en la actualidad que se asemeja a la finalidad de este proyecto es el de los Temas de Twitter. [14], [15]

"Temas" es una nueva finalidad lanzada por Twitter el 13 de noviembre de 2019 que permite en lugar de seguir a usuarios, seguir determinados temas predefinidos. Estos temas están divididos por categorías y navegando sobre ellos te permite seguir un tema en particular. Por ejemplo, si buscamos la categoría "*Deporte"*, dentro engloba multitud de temas como pueden ser "*Fútbol"* o "*Baloncesto*". Un usuario puede seguir al tema general "*Fútbol"* pero también puede seguir otros temas dentro de la categoría "*Deportes -> Fútbol*" como es el caso del tema "*Real Sporting*".

De esta forma mediante NLP, Twitter puede determinar la categoría o categorías de cada tuit que escribe un usuario y mostrártelos si sigues a ese determinado tema.

Este ejemplo es particularmente interesante ya que corresponde a una funcionalidad muy similar a la de este proyecto, en nuestro caso el usuario indicará un tuit sobre un determinado tema y el programa predecirá los temas de los que habla el tuit y le mostrará un conjunto de tuits relativos a dichos temas.

2. Diseño e implementación del problema.

En este apartado se describirá de forma general el problema planteado junto a todas sus fases para tener una visión global del mismo, así posteriormente centrarnos en describir en qué consisten y cómo se han resuelto cada una de las fases en particular.

2.1.- VISIÓN GENERAL DEL PROBLEMA.

El problema por resolver consiste en la clasificación de tuits en unos temas descritos como la unión de categorías definidas por *hashtags* y en la selección y ordenación de tuits referidos a los temas predichos.

Un ejemplo sería un usuario que escribe la frase o tuit "Que alegría pasear por la montaña y disfrutar de un maravilloso paisaje y unas hermosas vistas." El problema consistiría en primero clasificar automáticamente el tuit en uno de los temas previamente definidos y formados por la unión de *hashtags*, en este caso el tema "Naturaleza" formado por los *hashtags* "medioambiente", "ecológico", "sostenible" entre otros, para posteriormente mostrarle un conjunto de tuits acerca del tema o temas referidos.

Para ello se partirá de la definición de los temas a clasificar, de las distintas categorías que forman cada tema y de la extracción de un conjunto de tuits que forman parte de cada una de las categorías. A partir de ese punto, se realizará un proceso de análisis léxico, sintáctico y semántico al texto en lenguaje natural mediante el cual se obtendrán varios hitos. Por una parte, el conjunto de tuits procesados y el tema o temas a los que pertenecen y, por otra parte, el conjunto de palabras o *tokens* que forman los tuits representadas mediante vectores normalizados.

Finalmente, con todo esto y mediante un algoritmo de *Machine Learning* entrenaremos un modelo capaz de predecir a qué tema o temas pertenece cada uno de los tuits extraídos

anteriormente y los nuevos tuits introducidos por el usuario para así mostrar al usuario un conjunto de tuits de los temas deseados.

Todo este proceso previo a la interacción con el usuario desde obtener la base de datos de tuits hasta el entrenamiento de los modelos y posterior análisis de calidad puede ser representado de forma gráfica mediante el esquema de la figura 1 cuyas partes serán explicadas en los siguientes apartados:

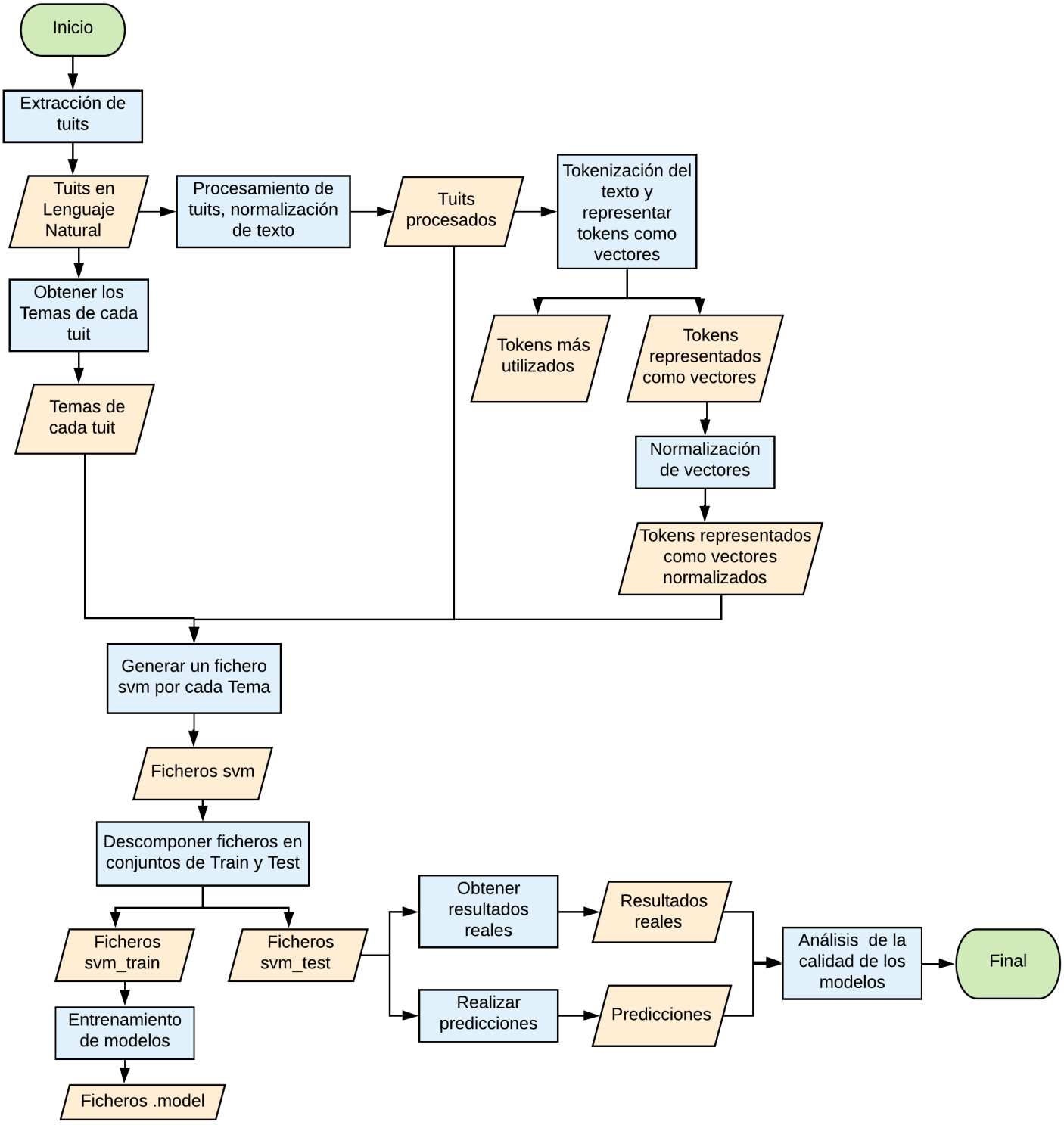

Fig 1. Esquema representativo de todas las fases del problema desde que los tuits son extraídos hasta que el modelo es evaluado.

El esquema anterior se puede descomponer en tres grandes bloques que vamos a explicar a continuación, estos bloques son:

- Extracción de tuits y procesamiento del texto.
- Representación de los tuits como vectores de similitud
- Clasificación de los tuits: modelos y predicciones.

2.2.- BLOQUE 1: EXTRACCIÓN DE TUITS Y PROCESAMIENTO DEL TEXTO.

Este primer bloque contiene todo el proceso desde que los tuits son extraídos en lenguaje natural, son transformados en un conjunto de *tokens* que aporten la mayor cantidad de información posible y finalmente representados como vectores normalizados.

Partiendo del esquema inicial, este bloque puede ser representado mediante la figura 2:

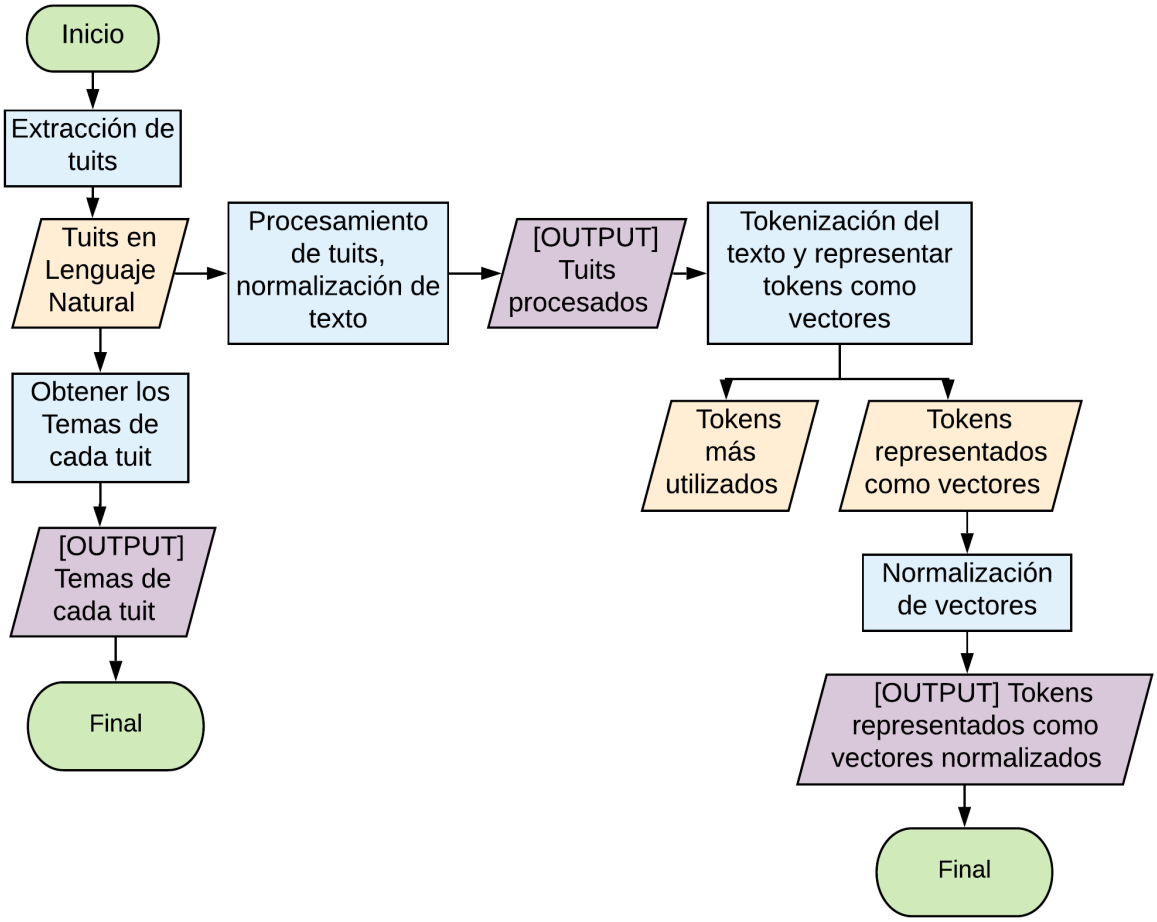

Fig 2. Esquema representativo de la extracción y procesamiento de tuits.

Como podemos ver en la figura 2, este bloque consiste en cinco principales operaciones que generarán una serie de resultados.

En primer lugar, tenemos la extracción de los tuits que generará un conjunto de tuits en lenguaje natural. A partir de estos tuits obtenemos por una parte las categorías o temas a los que pertenece cada tuit y, por otra parte, realizando un proceso de normalización léxica y sintáctica obtenemos el conjunto de tuits procesados.

Finalmente, partiendo de los tuits procesados los separaremos en unidades mínimas de información denominadas *tokens* y los más utilizados serán representados como vectores que posteriormente serán normalizados.

2.2.1 Extracción de tuits.

El primer paso para la resolución del problema fue decidir varios conjuntos de categorías o temas a las cuales puede pertenecer un tuit y cómo obtener tuits pertenecientes a esos temas. Para ello se plantearon una serie de tópicos frecuentemente utilizados en redes sociales como son Naturaleza, Educación, Política, Tecnología, Deporte, Música, Empleo, Cultura, Redes Sociales y Televisión que a partir de ahora llamaremos como *Temas*.

Para obtener tuits relativos a cada uno de estos *Temas* y así definir nuestro conjunto de datos de partida, se realizó una investigación acerca de los *tags* más utilizados para cada uno de estos *Temas*, siempre y cuando el tuit sea escrito en español.

Los resultados obtenidos fueron los siguientes:

Tabla 1: Temas y tags elegidos para la extracción de tuits.

Temas	**Tags**
Naturaleza	#medioambiente, #naturaleza, #fotografia, #montaña, #sunset, #sostenibilidad, #ecologia, #reciclaje, #sostenible, #reciclar, #ecologico, #playa, #verde, #cambioclimatico, #ecofriendly, #zerowaste, #eco, #nature, #contaminacion, #planeta, #sustentable, #reutiliza, #ambiente, #calentamientoglobal, #fauna, #vida, #flora, #landscape, #paisaje, #travel
Educación	#educacion, #salud, #universidad, #exito, #escuela, #valores, #aprendizaje, #motivacion, #colegio, #emprendedor, #educacioninfantil, #aprender, #profesor, #respeto, #finanzas, #esfuerzo, #emprendedores, #maestros, #futuro, #infancia, #formacion
Política	#politica, #corrupcion, #españa, #cataluña, #politicos, #democracia, #periodismo, #unidadespañola, #catalunya, #gobierno, #actualidad, #pp, #psoe, #vox, #podemos, #ciudadanos, #disputa, #elecciones, #debate, #pacto
Tecnología	#tecnologia, #informatica, #smartphone, #iphone, #samsung, #internet, #telefono, #apple, #android, #innovacion, #seguridad, #empresas, #startup, #pc, #ordenador, #web, #digital, #tech, #blockchain, #wifi
Deporte	#deporte, #sport, #fitness, #gym, #salud, #entrenamiento, #vidasana, #nutricion, #motivacion, #futbol, #ejercicio, #gimnasio, #sports, #workout, #healthy, #motivation, #ciclismo, #futbol, #run, #crossfit, #fit, #saludable, #tenis, #esports
Música	#musica, #music, #concierto, #live, #fiesta, #musically, #rap, #rock, #trap, #dj, #baile, #arte, #directo, #singer, #pop, #musicaenvivo, #livemusic, #show, #cover, #cantante, #cultura, #festival, #guitarra
Empleo	#empleo, #trabajo, #trabajar, #rrhh, #formacion, #paro, #cursos, #work, #emprender, #empresa, #cv, #entrevista, #job, #coaching, #profesionales, #marketing, #socialmedia, #oportunidad, #motivacion, #oferta, #ofertadeempleo, #ofertadetrabajo, #bolsadetrabajo, #laboral, #vacante

Temas	Tags
Cultura	#cultura, #arte, #culture, #historia, #turismo, #photography, #arquitectura, #museo, #teatro, #photo, #history, #paint, #pintura, #musica, #literatura, #cine, #cinema, #festival, #music, #pelicula, #fotografia
Redes Sociales	#social, #media, #marketing, #instagram, #twitter, #youtube, #socialmedia, #marketingdigital, #bullying, #estrategia, #facebook, #emprendedor, #communitymanager, #digital, #rrss, #diseño, #web
Televisión	#peliculas, #cine, #movie, #netflix, #hbo, #series, #libros, #estrenos, #actor, #terror, #drama, #comedia, #actriz, #saga, #director, #tv, #art, #television, #disney, #tvshow, #season, #temporada, #hollywood, #anime

Una vez definidos los *Temas* y sus *tags* más comunes se realizó una extracción de un conjunto de tuits de forma que cada *Tema* estuviese representado por un número similar de tuits en español. Esta extracción se realizó mediante la librería *rtweet.* [16]

2.2.2.- Normalización léxica y sintáctica

Tras extraer los tuits, estos estaban en lenguaje natural, por tanto, incluían información irrelevante como por ejemplo enlaces, menciones, signos de puntuación, etc. por lo que el siguiente paso era convertir el texto en un conjunto de *tokens* que aporten la información más útil posible de cara a realizar nuestro modelo con *Machine Learning*.

Para realizar esta trasformación se ha utilizado tanto normalización léxica dirigida al tratamiento de los caracteres y a la obtención de *tokens,* como normalización sintáctica para transformar o eliminar *tokens* de forma que solo nos quedemos con la información más relevante.

En este caso se planteó primero que tratamientos realizarle al texto y una vez decidido se planteó el orden de las operaciones. El resultado final es una combinación de normalización léxica y sintáctica formada por los siguientes pasos:

1. Convertir todas las letras a minúsculas.
2. Eliminar saltos de línea ya que como requisito necesitábamos que cada tuit estuviese en una línea y reemplazarlos por " ".
3. Eliminar tildes y caracteres no UTF-8
4. Eliminar menciones y reemplazarlas por el *token* @mencion
5. Eliminar links y reemplazarlos por el *token* @link
6. Eliminar signos de puntuación y reemplazarlos por el carácter " ". Exceptuamos el símbolo "#" al comienzo de palabra al tratarse de un *hashtag* y del símbolo @ que forma nuestros *tokens.*
7. Eliminar números.
8. Eliminar espacios en blanco múltiples.
9. Eliminar palabras sin valor semántico conocidas como *stopwords.*
10. Quedarnos con la raíz de la palabra (*stemming*), es decir, eliminar sufijos, tiempos verbales, plural, etc. Este proceso se ha realizado mediante la librería *SnowballC* [17], [18]

Tras realizar este proceso y plantearse la utilidad de los *tokens* @link y @mencion y el símbolo "#" en los *hashtags*, se decidió eliminar dichos *tokens* y símbolos en el paso 6, "eliminar signos de puntuación", ya que no son relevantes para la clasificación de un determinado tuit.

Un ejemplo de este proceso completo es el siguiente:

Tabla 2: Ejemplo de normalización léxica y sintáctica.

Original: Día 312 año 2019 "No defendimos lo suficiente nuestro ser. Y ahora estamos a merced de los vientos." https://t.co/3AIXJ4m4Fs #Montaña #PuraVida #Asturias https://t.co/NJco4sPWRR
1: día 312 año 2019 "no defendimos lo suficiente nuestro ser. y ahora estamos a merced de los vientos." https://t.co/3aixj4m4fs #montaña #puravida #asturias https://t.co/njco4spwrr

2: día 312 año 2019 “no defendimos lo suficiente nuestro ser. y ahora estamos a merced de los vientos.” https://t.co/3aixj4m4fs #montaña #puravida #asturias https://t.co/njco4spwrr
3: dia 312 año 2019 “no defendimos lo suficiente nuestro ser. y ahora estamos a merced de los vientos.” https://t.co/3aixj4m4fs #montaña #puravida #asturias https://t.co/njco4spwrr
4: dia 312 año 2019 “no defendimos lo suficiente nuestro ser. y ahora estamos a merced de los vientos.” https://t.co/3aixj4m4fs #montaña #puravida #asturias https://t.co/njco4spwrr
5: dia 312 año 2019 “no defendimos lo suficiente nuestro ser. y ahora estamos a merced de los vientos.” @link #montaña #puravida #asturias @link
6: dia 312 año 2019 no defendimos lo suficiente nuestro ser y ahora estamos a merced de los vientos @link #montaña #puravida #asturias @link
6 versión final: dia 312 año 2019 no defendimos lo suficiente nuestro ser y ahora estamos a merced de los vientos montaña puravida asturias
7: dia año no defendimos lo suficiente nuestro ser y ahora estamos a merced de los vientos montaña puravida asturias
8: dia año no defendimos lo suficiente nuestro ser y ahora estamos a merced de los vientos montaña puravida asturias
9: dia año defendimos suficiente ser ahora merced vientos montaña puravida asturias
10: dia año defend suficient ser ahor merc vient montañ purav asturi

2.2.3.- Normalización semántica.

Una vez tenemos nuestros tuits normalizados léxica y sintácticamente, es decir, que se encuentren formados por un conjunto de *tokens*, el siguiente paso es parametrizar el conjunto de tuits indicando para cada uno de los *tokens* el número de veces que se repite en todo el conjunto. De esta forma nos quedamos solamente con los *tokens* más repetidos, es decir, los más importantes del conjunto de tuits.

A continuación, mediante el algoritmo de Words2Vec GloVe *(Global Vectors for Word Representation*) [19] cada uno de los *tokens* es representado como un vector de la misma longitud. A este conjunto de *tokens* representados como vectores lo llamaremos *TokVec.*

Por último, ya que nos interesa clasificar el texto en función de la similitud de las palabras utilizadas y para facilitar el cálculo de similitudes mediante cosenos de vectores, necesitamos que nuestro conjunto de *tokens* en forma de vectores se encuentre normalizado. Por tanto, este conjunto de vectores lo hemos transformado en un conjunto de vectores normalizados.

2.3.- BLOQUE 2: REPRESENTACIÓN DE LOS TUITS COMO VECTORES DE SIMILITUD

Una vez tenemos los tuits procesados y los *tokens* más utilizados están representados mediante vectores normalizados, el siguiente paso es transformar el conjunto de tuits en un fichero donde representemos cada tuit mediante un vector que indique la similitud del conjunto de *tokens* que forman el tuit con el conjunto global de *tokens.*

Este proceso es representado con el esquema de la figura 3:

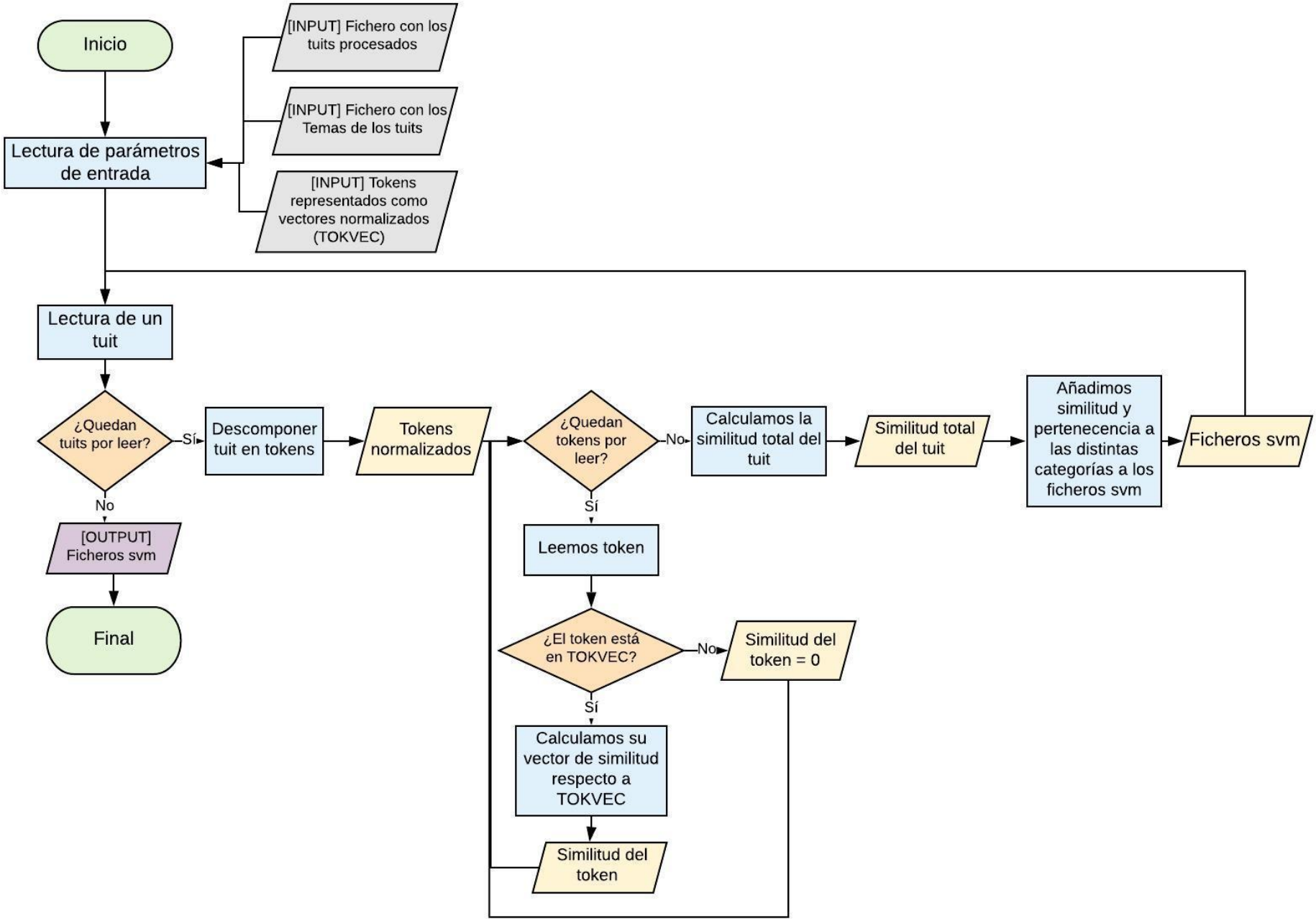

Fig 3. Esquema representativo de la obtención del fichero de similitud de los tuits.

Para este proceso se ha realizado un programa que parte de las salidas del bloque anterior, es decir, el fichero con los tuits procesados, el fichero con los *Temas* a las que pertenece cada tuit y el conjunto de *tokens* representado como vectores normalizados o *TokVec.*

Una vez leídos los distintos ficheros de entrada, el programa leerá tuit a tuit cada uno de los tuits del conjunto global. Cada tuit se descompone según sus *tokens* o palabras, estos serán procesados uno a uno y serán tratados de la siguiente forma hasta que no queden más *tokens* del tuit leído:

1. Observamos el fichero *TokVec* para determinar si el *token* leído se encuentra o no en él.

2. Calcularemos el valor de similitud del *token* respecto al conjunto de *tokens.*
   2.1. Si el *token* no se encuentra *TokVec*, su valor de similitud es 0.
   2.2. Si el *token* se encuentra en *TokVec*, su valor de similitud se calculará mediante una función de similitud "S" que recibirá como parámetros por una parte el vector del *token* y por otra parte el conjunto de vectores de *TokVec* y calculará la similitud de cosenos entre el *token* y los distintos *Tokens* de *TokVec* mediante la siguiente fórmula.[20], [21]

$$\boldsymbol{S}(\vec{\boldsymbol{x}}, \boldsymbol{TokVec}\{\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}}, \dots, \vec{\boldsymbol{z}}\}) = [\cos(\vec{\boldsymbol{x}}, \vec{\boldsymbol{a}}), \cos(\vec{\boldsymbol{x}}, \vec{\boldsymbol{b}}), \dots, \cos(\vec{\boldsymbol{x}}, \vec{\boldsymbol{z}})]$$
$$= \left(\frac{\vec{\boldsymbol{x}} * \vec{\boldsymbol{a}}}{\|\vec{\boldsymbol{x}}\| * \|\vec{\boldsymbol{a}}\|}, \frac{\vec{\boldsymbol{x}} * \vec{\boldsymbol{b}}}{\|\vec{\boldsymbol{x}}\| * \|\vec{\boldsymbol{b}}\|}, \dots, \frac{\vec{\boldsymbol{x}} * \vec{\boldsymbol{z}}}{\|\vec{\boldsymbol{x}}\| * \|\vec{\boldsymbol{z}}\|}\right) \quad (\mathbf{1})$$

Ya que en este caso los vectores han sido previamente normalizados, por tanto, su módulo es 1, la fórmula quedaría simplificada de la siguiente forma:

$$\boldsymbol{S}(\vec{\boldsymbol{x}}, \boldsymbol{TokVec}\{\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}}, \dots, \vec{\boldsymbol{z}}\}) = (\vec{\boldsymbol{x}} * \vec{\boldsymbol{a}}, \vec{\boldsymbol{x}} * \vec{\boldsymbol{b}}, \dots, \vec{\boldsymbol{x}} * \vec{\boldsymbol{z}})$$
$$= (x_1 a_1 + x_2 a_2 + \cdots + x_n a_n, x_1 b_1 + x_2 b_2 + \cdots + x_n b_n, \dots,$$
$$x_1 z_1 + x_2 z_2 + \cdots + x_n z_n, \quad (\mathbf{2})$$

Una vez tenemos calculada la similitud de cada *token* del tuit, la similitud total del tuit será calculada como el sumatorio de las similitudes de sus *tokens*. Por lo tanto, si tenemos un tuit "T" formado por los *tokens* "x", "y" y "z" su similitud total respecto a *TokVec* es:

$$\begin{aligned} &\boldsymbol{S}\left(\boldsymbol{T}\{\vec{\boldsymbol{x}},\vec{\boldsymbol{y}},\vec{\boldsymbol{z}}\},\boldsymbol{TokVec}\{\vec{\boldsymbol{a}},\vec{\boldsymbol{b}},\dots,\vec{\boldsymbol{z}}\}\right) \\ &\qquad = \boldsymbol{S}\left(\vec{\boldsymbol{x}},\boldsymbol{TokVec}\{\vec{\boldsymbol{a}},\vec{\boldsymbol{b}},\dots,\vec{\boldsymbol{z}}\}\right) + \boldsymbol{S}\left(\vec{\boldsymbol{y}},\boldsymbol{TokVec}\{\vec{\boldsymbol{a}},\vec{\boldsymbol{b}},\dots,\vec{\boldsymbol{z}}\}\right) \\ &\qquad + \boldsymbol{S}\left(\vec{\boldsymbol{z}},\boldsymbol{TokVec}\{\vec{\boldsymbol{a}},\vec{\boldsymbol{b}},\dots,\vec{\boldsymbol{z}}\}\right) \end{aligned} \tag{3}$$

Por último, para cada uno de los *Temas* comprobaremos si el tuit pertenece o no a ellos y escribiremos en cada uno de los ficheros SVM, uno por cada *Tema*, el valor "0" en caso de que no pertenezca al *Tema* o "1" si pertenece. Además, se añade en ambos casos el vector con el valor de similitud del tuit.

Tras realizar el mismo proceso para cada uno de los tuits, tendremos calculados nuestros ficheros SVM de entrenamiento.

A continuación, se mostrará un ejemplo de cómo calcular el valor de similitud de un *token* respecto a un fichero *TokVec*.

En este caso queremos calcular el vector de similitud del *token* "casa" y tenemos un conjunto *TokVec* con los siguientes vectores normalizados:

Tabla 3: *TokVec* de ejemplo para el cálculo de similitud

Token	Vector			
nuev	0.528621	-0.719036	0.109629	0.437640
casa	0.708636	-0.440517	0.141126	-0.532788
verde	0.481990	-0.151745	-0.042779	-0.861875

En primer lugar, buscaremos si el *token* "casa" se encuentra en *TokVec* y como en este caso sí se encuentra en el fichero nos quedaremos con su vector [0.708636, -0.440517, 0.141126, -0.532788]

A continuación, tendremos que calcular el vector de similitud de "casa" respecto a los *tokens* que forman *TokVec*. Para ello nuestra función de similitud "S" recibirá como parámetros por una parte el vector de "casa" y por otra parte el conjunto de vectores de *TokVec*. Entonces al tratarse de similitud de cosenos y los vectores estar normalizados, la similitud de "casa" se calcula como el producto escalar del vector "casa" respecto a cada uno de los *tokens* del fichero *TokVec*.

En este ejemplo el vector de similitud de "casa" es:

$$\boldsymbol{similitud}[\mathbf{0}] = \boldsymbol{casa}[\mathbf{0}] * \boldsymbol{nuev}[\mathbf{0}] + \boldsymbol{casa}[\mathbf{1}] * \boldsymbol{nuev}[\mathbf{1}] + \boldsymbol{casa}[\mathbf{2}] * \boldsymbol{nuev}[\mathbf{2}] + \boldsymbol{casa}[\mathbf{3}] * \boldsymbol{nuev}[\mathbf{3}] = \mathbf{0.4737}$$
$$\boldsymbol{similitud}[\mathbf{1}] = \boldsymbol{casa}[\mathbf{0}] * \boldsymbol{casa}[\mathbf{0}] + \boldsymbol{casa}[\mathbf{1}] * \boldsymbol{casa}[\mathbf{1}] + \boldsymbol{casa}[\mathbf{2}] * \boldsymbol{casa}[\mathbf{2}] + \boldsymbol{casa}[\mathbf{3}] * \boldsymbol{casa}[\mathbf{3}] = \mathbf{1}$$
$$\boldsymbol{similitud}[\mathbf{2}] = \boldsymbol{casa}[\mathbf{0}] * \boldsymbol{verde}[\mathbf{0}] + \boldsymbol{casa}[\mathbf{1}] * \boldsymbol{verde}[\mathbf{1}] + \boldsymbol{casa}[\mathbf{2}] * \boldsymbol{verde}[\mathbf{2}] + \boldsymbol{casa}[\mathbf{3}] * \boldsymbol{verde}[\mathbf{3}] = \mathbf{0.8616}$$

El vector de similitud de cada *token* tendrá tantas posiciones como *tokens* tenga *TokVec*, en este caso 3 posiciones. Por tanto, el vector de similitud de "casa" es [0.4737, 1, 0.8616]

2.4.- CLASIFICACIÓN DE LOS TUITS: MODELOS Y PREDICCIONES

Finalmente, el último paso es generar un modelo que pueda predecir el *Tema* o *Temas* a las que pertenece cada tuit. Para ello hemos utilizado la librería *LIBSVM.* **[22]**

LIBSVM es una librería *open source* de Machine Learning que permite resolver problemas de regresión y de clasificación mediante distintos algoritmos.

Para nuestro problema en particular, necesitamos clasificar texto en diferentes clases. Además, la clasificación multiclase no es suficiente ya que un texto puede pertenecer a una de las múltiples clases o *temas,* pero también puede pertenecer a más de una. Por ello, nuestro problema

al ser un problema multiclase y multietiqueta lo descomponemos en varios problemas de una clase, es decir, tantos problemas simples como temas tenemos. Esta es la razón por la que en lugar de un único fichero SVM hemos generado un fichero por cada uno de los *Temas*. Cada uno de estos ficheros será una entrada del algoritmo de *Machine Learning*, y, por lo tanto, serán generados tantos modelos como *Temas*.

Una vez los modelos son generados utilizando el modelo perteneciente a un *Tema* podemos predecir si un tuit transformado en un vector SVM pertenece o no a ese *Tema*.

En nuestro caso queremos experimentar con diferentes transformaciones del texto, por eso para cada una de ellas calcularemos la efectividad de los modelos. Para ello se ha decidido dividir el fichero SVM en un conjunto de entrenamiento y otro conjunto de prueba, de forma que los modelos son entrenados con el conjunto de entrenamiento y las predicciones son realizadas con el fichero de prueba. Este proceso de separación de conjuntos de entrenamiento y prueba junto a la evaluación de calidad de los modelos será explicado en mayor detalle en el apartado siguiente.

El proceso de clasificación de texto y generación de los modelos puede ser representado con el esquema de la figura 4:

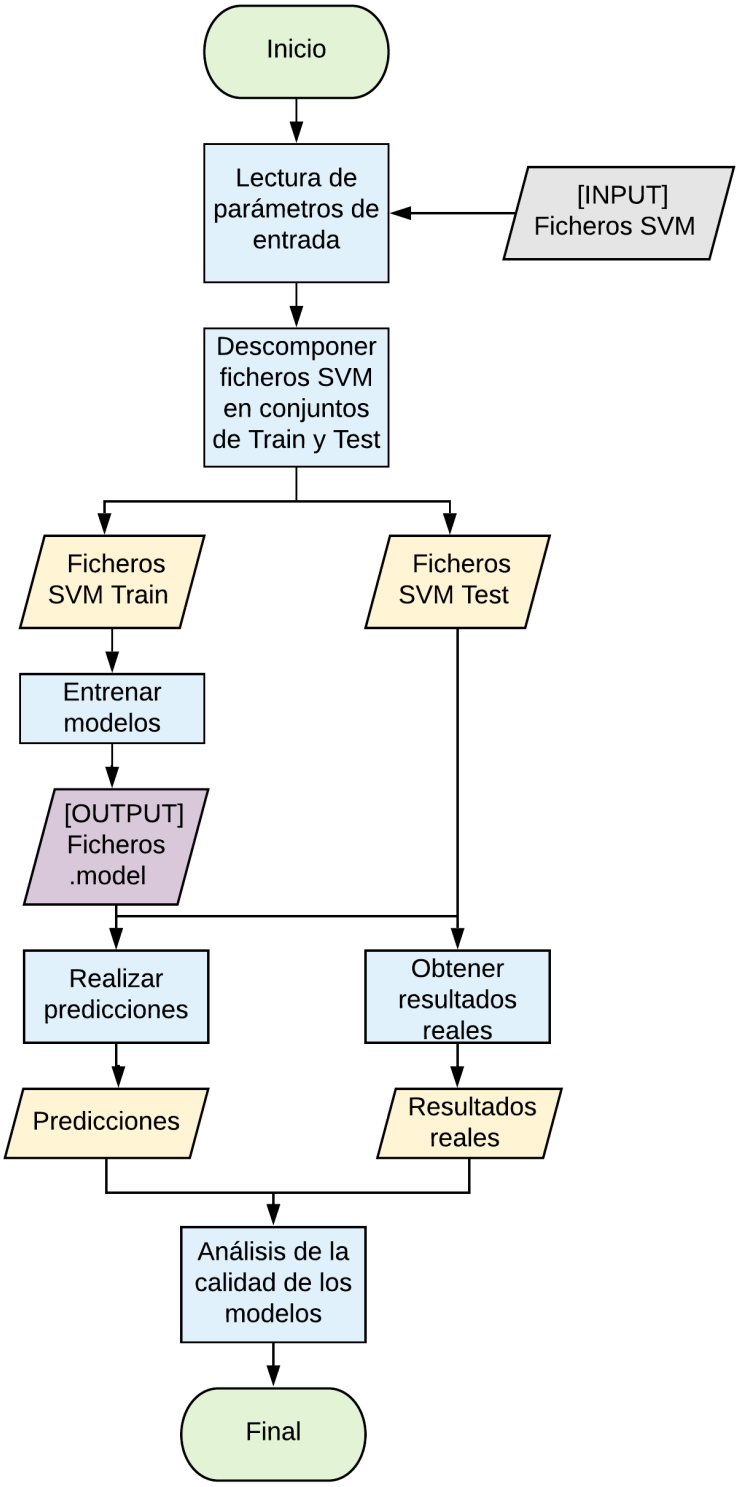

Fig 4. Esquema representativo de la clasificación de texto y entrenamiento de los modelos.

3. Estudio y análisis de resultados.

Una vez explicadas las distintas transformaciones realizadas sobre el texto desde que extraemos la información hasta el resultado final, para conseguir la mejor resolución posible del problema se decidió experimentar con distintos algoritmos de *Machine Learning* y con diferentes tratamientos del texto, incrementando progresivamente la complejidad de estos de forma que pudiésemos decidir cuál es la transformación del texto y el algoritmo mediante los cuales se obtienen mejores predicciones.

En este apartado se describirán las cuatro transformaciones diferentes realizadas, las diferentes experimentaciones realizadas a lo largo del proyecto, cómo se ha planteado el entrenamiento y prueba de los modelos, aquellas métricas relevantes para comparar los resultados y por último se mostrarán los resultados obtenidos y una decisión final sobre la transformación y algoritmo a utilizar.

3.1.- PLANTEAMIENTO DE LAS DIFERENTES TRANSFORMACIONES.

Para la resolución del problema se plantearon cuatro diferentes transformaciones. A continuación, pasaremos a comentarlas de menor a mayor complejidad.

3.1.1.- Texto sin transformar.

El primer planteamiento fue preguntarnos qué resultados podemos obtener sin procesar la información y cómo mejorarán los resultados según la vamos modificando. Por tanto, la primera experimentación consistió en extraer los tuits en lenguaje natural y utilizarlos como entrada de nuestro problema. Las únicas transformaciones realizadas en esta experimentación fueron representar cada tuit como una línea de texto y la eliminación de caracteres no UTF-8 que pudieran causar algún problema en la ejecución de la aplicación.

Un ejemplo de uno de los tuits sin procesar es el siguiente:

#LaRioja | Hallan y resguardan a un cachorro de puma "concolor". El felino deambulaba perdido en cercanias a una vivienda. Fue trasladado por los #gendarmes a un centro veterinario para evaluar su estado de salud. #MedioAmbiente #proteccion https://t.co/pdUMOdgDr5 https://t.co/cz0wSWJQSx

3.1.2.- Transformación completa a excepción de Porter Stemmer.

El segundo tratamiento consistió en realizar una transformación léxica y sintáctica al texto incluyendo la eliminación de las "*stopwords"*, en este caso en castellano, ya que son palabras que apenas nos aportan información. La única excepción es que no se ha utilizado el algoritmo de Porter Stemmer para la lematización de las palabras.

Si aplicamos al mismo ejemplo utilizado anteriormente este tratamiento el resultado sería el siguiente:

larioja hallan resguardan cachorro puma concolor felino deambulaba perdido cercanias vivienda trasladado gendarmes centro veterinario evaluar salud medioambiente proteccion

3.1.3.- Transformación completa con restricciones.

Para el tercer tratamiento partimos del tratamiento anterior, pero añadimos un mayor punto de complejidad. En este caso se ha aplicado el algoritmo de Porter Stemmer a cada una de las palabras que forman un tuit a excepción de los hashtags. Esto es debido a que los hashtags son palabras con un comportamiento especial ya que pueden incluir palabras diferentes tratadas como una palabra compuesta, palabras en otros idiomas, etc.

En este caso el resultado de la transformación sería el siguiente:

larioja hall resguard cachorr pum concolor felin deambul perd cercan viviend traslad gendarmes centr veterinari evalu salud medioambiente proteccion

3.1.4.- Transformación completa.

Para el cuarto y último tratamiento se decidió además de realizar una transformación léxica y sintáctica completa, aplicar Porter Stemmer a todas y cada una de las palabras que forman un tuit incluidos los hashtags.

El resultado del tuit sería el siguiente:

larioj hall resguard cachorr pum concolor felin deambul perd cercani viviend traslad gendarm centr veterinari evalu salud medioambient proteccion

3.2.- DISEÑO DE LA EXPERIMENTACIÓN.

Aunque la mayor parte de las experimentaciones realizadas hayan sido acerca de buscar la transformación del texto mediante la cual se obtienen mejores resultados en cuanto a la predicción de *Temas*, también es necesario indicar todos los programas y algoritmos utilizados en todas las etapas del proyecto junto a sus parámetros.

3.2.1.- Extracción de tuits.

Para la extracción de tuits, como se mencionó anteriormente se utilizó la librería *rtweet.* Como cada uno de los *Temas* tiene un número diferente de *tags* que los representan, se decidió agrupar el conjunto de *tags* de cada *Tema* y realizar una extracción de unos 1000 tuits por *Tema*. Las diferentes extracciones fueron realizadas para tuits en español e incluyendo sólo tuits escritos por un usuario y no *retweets* [23] *(*tuits formados al compartir el tuit de otro usuario*)* ya

que sólo causarían duplicados. Tras la eliminación de tuits duplicados quedó un conjunto final de 8000 tuits formados por aproximadamente el mismo número de tuits por cada *Tema.*

En un principio se planteó la extracción de un conjunto mayor de tuits (100000 tuits por *Tema*), pero tras ver la velocidad de procesamiento y el conjunto de información almacenada a la hora de realizar los modelos, este planteamiento fue descartado.

3.2.2 Algoritmo de Words2Vectors.

De cara al algoritmo a utilizar para representar el conjunto de *tokens* como vectores, se decidió utilizar GloVe ya que había sido el mismo algoritmo utilizado en las prácticas de NLP en la asignatura de Interfaces Multimodales. En la ejecución de GloVe se utilizaron los siguientes parámetros:

- Mínimo de ocurrencias de una palabra para ser considerada como token: 5. Este número fue elegido en función al conjunto de tuits de entrada.
- Vectores de dimensión 50 y 15 iteraciones del algoritmo (parámetros por defecto)

3.2.3 Algoritmos en LIBSVM.

De entre todos los algoritmos que nos permitía utilizar LIBSVM, al tratarse de un problema de clasificación en el que el objetivo era predecir si pertenece o no a una determinada clase, se decidió comparar los resultados obtenidos mediante un algoritmo SVC (*Support Vector Classification*) [24], algoritmo por defecto de LIBSVM, y los resultados obtenidos mediante una Regresión Logística [25].

Tras comparar los resultados, como veremos en el apartado 4.4 de Análisis de Resultados, hemos decidido utilizar la Regresión Logística.

Además, vimos necesario conocer con qué probabilidad clasifica o no a cada tuit con sus *Temas* y el algoritmo de Regresión Logística era el único mediante el cual LIBSVM nos devolvía dichas probabilidades.

3.2.4 Transformaciones del texto.

De cara a comprobar la efectividad de las diferentes transformaciones del texto, se decidió separar el conjunto de tuits en dos partes, una de entrenamiento y otra de prueba. De esta forma los modelos son entrenados con el conjunto de tuits de entrenamiento y las predicciones son realizadas con los tuits de prueba. Una vez realizadas las predicciones se comprobó la efectividad de estas comparando los resultados reales contra los resultados predichos.

A la hora de realizar las experimentaciones se tuvieron en cuenta las siguientes consideraciones:

- Separación de los conjuntos de entrenamiento y prueba en un 70% de tuits para entrenamiento y un 30% para prueba.
- La separación deberá ser un conjunto de tuits aleatorios. De esta forma, ya que el conjunto está formado por una cantidad similar de tuits por cada *Tema,* se asegura tener tuits de todas los *Temas* en ambos conjuntos.
- Realizar varias experimentaciones para cada uno de los casos, en este caso debido al condicionante de memoria y tiempo de ejecución se han realizado cinco experimentaciones para cada caso.
- Cada conjunto de entrenamiento y prueba deberá estar formado por los mismos conjuntos de tuits en cada una de las experimentaciones y en cada uno de los tratamientos. Es decir, se utilizarán las mismas semillas para la predicción de cada uno de los *Temas*.

Por ello se ha decidido emplear una de las librerías más utilizadas para Machine Learning con Python, *scikit-learn* [26] y en particular la función *train_test_split* [27]

3.3.- EVALUACIÓN DE LA CALIDAD.

Una vez realizadas las predicciones para las distintas transformaciones de texto era necesario compararlas para determinar cuál de ellas obtenía los mejores resultados. Para ello se comprobó la efectividad de estas comparando los resultados reales contra los resultados predichos.

La comparación entre los resultados reales y los predichos se consigue mediante la siguiente tabla de la verdad:

Tabla 4. Tabla de la verdad comparando valores reales con predichos.

	Predicho	
Real	Sí	No
Sí	TP (True Positive)	FN (False Negative)
No	FP (False Positive)	TN (True Negative)

En cuanto a la evaluación de los resultados obtenidos, se han utilizado las siguientes métricas para compararlos: [28]

- Precision: Mide de todos los resultados predichos como positivo, cuáles de ellos son realmente positivos. Su fórmula es la siguiente:

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

- Recall: Mide de todos los resultados que deberían ser positivos, cuántos han sido predichos como positivos. En el caso de que el modelo predijera todo como "Sí" esta métrica sería perfecta, pero su precisión sería muy mala. Lo obtenemos con la siguiente fórmula:

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

- F1: Combina los resultados de Precision y Recall realizando su media armónica. Toma valores entre 0 y 1, siendo 1 el mejor de los casos y no se podrá evaluar en el caso de que los valores TP y FN sean 0.

$$F1 = \begin{cases} \frac{2 * Precision * Recall}{Precision + Recall} & si\ TP > 0 \\ 0 & si\ TP = 0\ and\ FN > 0 \end{cases} \tag{6}$$

  Existen tres formas distintas para calcular el valor de F1:
  - F1 por ejemplo: Evaluar cada ejemplo de experimentación y promediarlo.
  - F1 Macro: Evaluar cada *Tema* y promediarlo.
  - F1 Micro: Evaluar toda la predicción en su conjunto.

- Hamming Loss: Es el porcentaje de valores de resultados que difieren entre los reales y los predichos. Cuanto menor sea mejor, es este valor. Sin embargo, puede llevarnos a equivocaciones ya que, si el modelo predijese siempre que no, el valor de Hamming Loss sería muy bueno. Se calcula mediante la siguiente fórmula:

$$Hamming\ Loss = \frac{FN + FP}{TP + FN + FP + TN} \tag{7}$$

3.4.- ANÁLISIS DE RESULTADOS.

Para cada una de las diferentes transformaciones, se han calculado los valores de Precision y Recall como la media de los valores obtenidos para el conjunto de pruebas en el caso de F1 por ejemplo, y como la media de los valores obtenidos para el conjunto de *Temas* en el caso de F1 Macro. Estos resultados se han calculado tanto para el algoritmo SVC por defecto de LIBSVM como para una Regresión Logística.

Las diferentes trasformaciones para las cuales hemos comparado los resultados obtenidos han sido las siguientes:

1. Texto sin transformar
2. Transformación completa a excepción de Porter Stemmer
3. Transformación completa con restricciones
4. Transformación completa

Tabla 5: Comparativa de resultados de las distintas métricas para cada una de las transformaciones utilizando un algoritmo SVC.

Transformación	F1 por ejemplo	F1 Macro	F1 Micro	Hamming Loss
1	0.2396	0.2404	0.2379	0.2581
2	0.3235	0.3262	0.3238	0.3820
3	0.3198	0.3223	0.3192	0.3597
4	0.3525	0.3560	0.3516	0.3186

Tabla 6: Comparativa de resultados de las distintas métricas para cada una de las transformaciones utilizando una Regresión Logística.

Transformación	F1 por ejemplo	F1 Macro	F1 Micro	Hamming Loss
1	0.3141	0.3069	0.3140	0.1680
2	0.4337	0.4349	0.4337	0.2697
3	0.4425	0.4422	0.4425	0.2417
4	0.4727	0.4722	0.4727	0.2193

3.5.- ELECCIÓN DE LA MEJOR SOLUCIÓN.

Podemos observar rápidamente que, en ambos casos mediante la primera transformación, texto sin transformar, obtenemos los peores valores de F1 tanto por ejemplo, como macro y micro. Gracias a esto podemos deducir que esta aproximación es la peor calculando valores positivos, es decir, cuando un tuit pertenece realmente a un *Tema*. Respecto al H.L. obtenemos el valor más bajo, esto se debe a que el modelo predice en la gran mayoría de los casos como que un tuit no pertenece al *Tema*.

Observando el resto de las transformaciones, observamos que si no utilizamos Porter Stemmer o si lo utilizamos parcialmente obtenemos unos resultados similares en cuanto a valor de F1, pero, sin embargo, la tasa de error o H.L es mejor en el caso de que lo utilicemos parcialmente.

Finalmente, realizando una transformación completa obtenemos los mejores resultados de F1 en todos los casos y una mejor tasa de error que en las transformaciones anteriores. Esto se debe a que al aplicar Porter Stemmer a todas las palabras, el resultado final será un conjunto menor de palabras diferentes, por esto las palabras aparecerán con mayor frecuencia en los tuits logrando que un conjunto mayor de palabras en lenguaje natural estén contenidas en el conjunto *TokVec* como *tokens* normalizados y por tanto la clasificación será más efectiva.

Debido a esto, podemos concluir con que la mejor transformación es la transformación completa y por tanto será la utilizada en los modelos finales.

Respecto al algoritmo a utilizar, en todos los casos los valores de F1 por ejemplo, Macro, Micro y *Hamming Loss* son bastante mejores utilizando una Regresión Logística que un algoritmo SVC, por lo tanto será el algoritmo a utilizar.

4. Modelo y resolución del problema.

Tras el proceso de extracción del texto, definición de los *Temas*, elección y generación de distintos modelos y la evaluación para decidir cuál de los diferentes procesos de transformación de texto es el óptimo, el siguiente paso es resolver el problema.

Para ello se ha planteado un proceso incremental mediante el cual se realizará una solución que cumpla una serie de funcionalidades básicas y posteriormente partir de dicha solución para realizar otra solución más compleja, es decir, con funcionalidades añadidas.

En este apartado se describirán en detalle las diferentes soluciones implementadas.

4.1.- SOLUCIÓN 1: CLASIFICACIÓN DE TUITS EN TEMAS

En la primera solución el objetivo es predecir el *Tema* o *Temas* de una serie de tuits introducidos por el usuario y mostrarle un conjunto de tuits de dichos *Temas*.

Esta solución además de utilizar unos modelos previamente entrenados irá aprendiendo progresivamente según el usuario introduzca nuevos tuits y confirme los *Temas* a los que pertenecen. La información introducida por el usuario servirá para reentrenar los modelos.

La solución desarrollada consiste en el proceso representado en el esquema de la figura 5 y que será descrito a continuación.

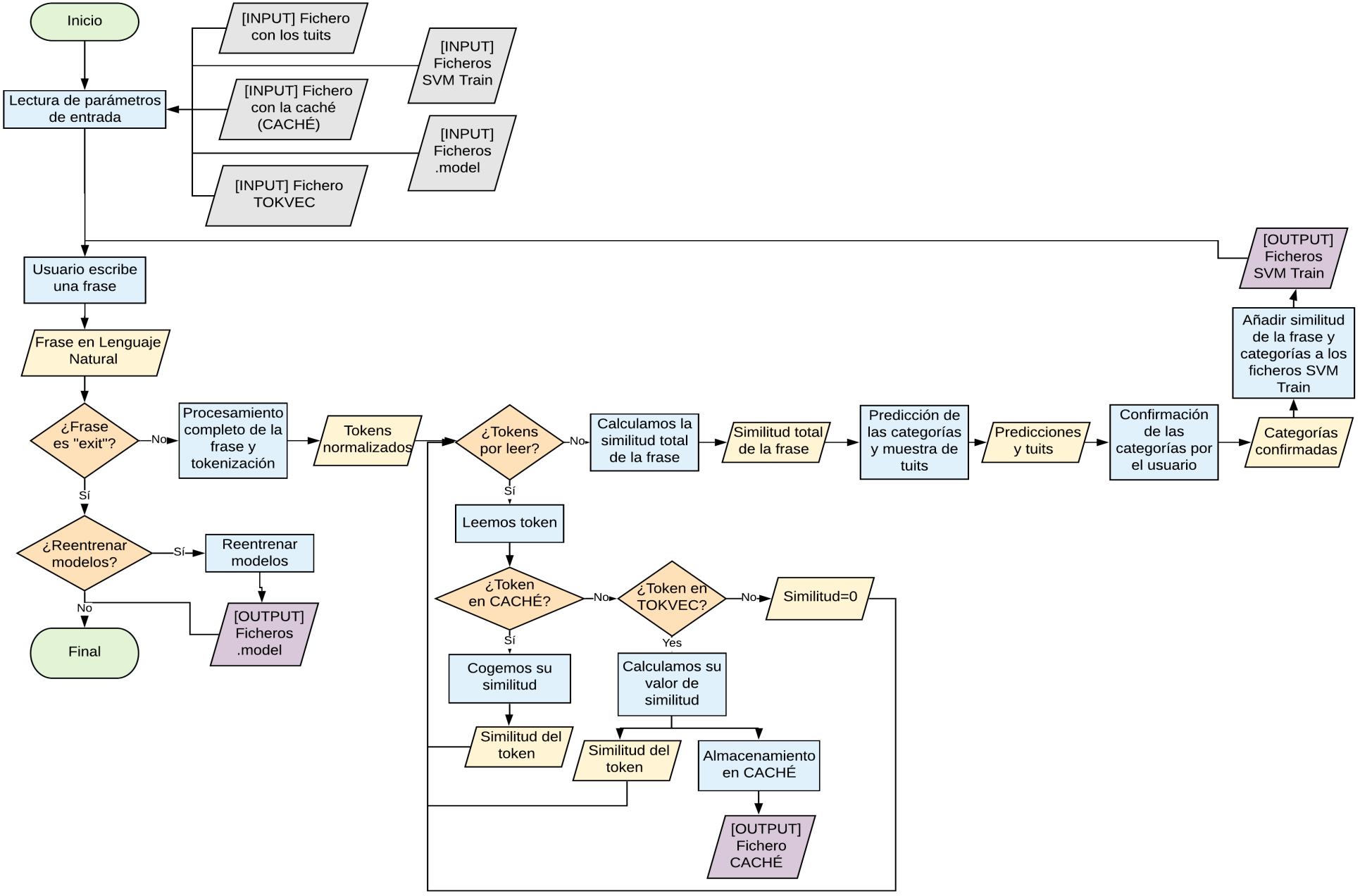

Fig 5. Esquema representativo de la clasificación de tuits en Temas.

Para iniciar el programa, el usuario deberá introducir una serie de parámetros que son:

- Un fichero con el conjunto *TokVec*, es decir, formado por un conjunto de tokens representados como vectores normalizados.
- Un fichero caché con la similitud de palabras previamente calculadas respecto al fichero *TokVec* utilizado. La finalidad de este fichero es agilizar la ejecución del programa.
- El conjunto de ficheros SVM utilizados para el entrenamiento de los modelos. Estos ficheros serán necesarios ya que según el usuario introduzca tuits y sus *Temas* sean predichos y confirmados, almacenarán más información útil para el reentrenamiento de los modelos.
- Modelos necesarios para la predicción de los *Temas*.
- Un corpus con el conjunto de tuits extraídos y sus *Temas*.

Además, el usuario podrá indicar si desea utilizar el programa en castellano “es” o en inglés “en”, de forma que el procesamiento de los tuits introducidos sea realizado de acuerdo con el idioma introducido.

Una vez iniciado el programa con los parámetros de entrada, el programa leerá el conjunto de ficheros y le pedirá al usuario que introduzca una frase o que escriba la palabra “exit” si desea salir del programa.

En el caso de escribir la palabra “exit” se le preguntará al usuario si desea salir directamente del programa o reentrenar los modelos. Si desea reentrenarlos con la nueva información, el programa cogerá el conjunto de ficheros SVM y los utilizará para reentrenar los modelos. Tras esto el programa se cerrará.

En el caso de que el usuario introduzca cualquier otra frase el programa realizará una serie de pasos:

1. La frase, introducida en lenguaje natural, será procesada mediante la transformación completa explicada en apartados anteriores. En función de que el usuario haya

introducido lenguaje “es” o “en”, la eliminación de *stopwords* y lematización del texto aplicando Porter Stemmer se hará para el lenguaje indicado.

2. Una vez que la frase esté formada por un conjunto de tokens normalizados, será recorrida *token* a *token*. En el caso de que todavía no hayan sido leídos todos los *tokens*, el programa leerá un *token*.
    a. Se comprobará si el *token* se encuentra o no en el fichero con la caché. En el caso de que se encuentre en la caché, la similitud del *token* respecto a los *tokens* del conjunto *TokVec* será el vector de similitud representado en la caché. Dicho valor será almacenado y se procederá a leer el siguiente *token.*
    b. En el caso de que el *token* no se encuentre en la caché, se procederá a comprobar si dicho *token* se encuentra o no en *TokVec*.
        i. Si no forma parte de *TokVec*, su vector de similitud será un vector cuyos elementos son 0. Se almacenará el valor y se procederá a leer el siguiente *token*.
        ii. Si forma parte de *TokVec*, se calculará su similitud mediante la función de similitud del *token* respecto a los distintos *tokens* de *TokVec* explicada en el apartado 2.3

            El vector de similitud será almacenado en la caché para no tener que calcularlo cada vez que aparezca el mismo *token*. Se procederá a leer el siguiente *token*.

3. Una vez que todos los *tokens* que forman la frase han sido leídos se calculará la similitud total de la frase como el sumatorio de los vectores de similitud de los distintos *tokens* que la forman.

4. Utilizando la similitud de la frase y los modelos entrenados, para cada uno de los posibles *Temas* se realizará una predicción de si la frase pertenece o no al mismo y en función de los *Temas* que el programa determine como correctos mostrará un conjunto de tuits formado por un conjunto del mismo tamaño de los *Temas* predichos hasta un

máximo de 10 tuits. Por ejemplo, si el programa predice que el usuario habla de un *Tema* el programa mostrará 10 tuits de ese *Tema*, pero si en cambio predice que habla de 2 *Temas* mostrará 5 de uno y 5 de la otro.

5. Se le presentará al usuario una solución con el conjunto de *Temas* predichos por el programa. El usuario podrá confirmar si el programa ha acertado o en caso contrario indicar a qué *Temas* pertenece realmente.

6. Gracias a la confirmación por parte del usuario, el programa almacenará en cada uno de los ficheros SVM un elemento más con el vector de similitud de la frase y la pertenencia o no al *Temas*.

7. El usuario podrá introducir una nueva frase o finalizar la ejecución del programa que se cerrará tras el reentrenamiento de los modelos en el caso de que el usuario lo desee.

4.2.- SOLUCIÓN 2: CLASIFICACIÓN Y MUESTRA DE TUITS EN FUNCIÓN DE LA PROBABILIDAD

Esta solución se presenta como una ampliación de la anterior. En el caso anterior el programa simplemente predecía si la frase escrita por el usuario pertenece o no a un *Tema*, pero en este caso mostrará la probabilidad de pertenecer o no a cada uno de los *Temas* y en función de la probabilidad mostrará un conjunto de tuits de los distintos *Temas* según su porcentaje. Además, al igual que en el caso anterior se le comunicará al usuario el principal o principales *Temas* predichos para que así confirme si el programa ha acertado o no y continuar con el aprendizaje.

El funcionamiento de esta solución es el siguiente:

1. Una vez calculado el vector de similitud total de la frase, el programa utilizando los distintos modelos realizará una predicción de si la frase pertenece a cada uno de los *Temas* almacenando internamente la probabilidad de pertenencia obtenido tras aplicar la Regresión Logística.

2. Ya que la probabilidad será un valor entre 0 y 1, para darle mayor relevancia a las categorías más probables variaremos sus valores aplicando el concepto de elitismo. Esto consiste en alterar la probabilidad elevando su valor a un exponente, de forma que si elevamos a 0 obtendremos el valor original de probabilidad y a medida que aumentamos el valor, la diferencia entre probabilidades aumenta.

   En este caso se han elevado las distintas probabilidades con el exponente 2 de forma que las mayores cobren una mayor importancia.

3. Ahora se asignará cada uno de los tuits a mostrar a un *Tema* siguiendo el sistema D'Hondt [29], es decir, una vez hemos aplicado elitismo a las probabilidades se calcularán cocientes sucesivos para cada una de ellas aplicando la siguiente fórmula donde T es el número de tuits ya asignados al *Tema*:

   $$cociente = \frac{Probabilidad}{T+1} \quad (8)$$

   A continuación, se asignará el tuit al *Tema* que presente un mayor cociente y se pasará a la siguiente iteración hasta que todos los tuits estén repartidos.

4. Al igual que en la solución anterior se le indicarán al usuario los principales *Temas* predichos y el usuario indicará si la solución es correcta o en caso de error cuales deberían ser los correctos.

5. Gracias a la confirmación por parte del usuario, el programa almacenará en cada uno de los ficheros SVM un elemento más con el vector de similitud de la frase y la pertenencia o no al *Tema*.

6. El usuario podrá introducir una nueva frase o finalizar la ejecución del programa que se cerrará tras el reentrenamiento de los modelos en el caso de que el usuario lo desee.

La asignación de tuits a categorías mediante el método D'Hondt se puede observar mejor con el siguiente ejemplo:

Tabla 7: Clasificación de los tuits en *Temas* siguiendo el método de D'Hondt.

	Tema 1	**Tema 2**	**Tema 3**	**Tema 4**	**Tema 5**	**Tema 6**	**Tema 7**	**Tema 8**	**Tema 9**	**Tema 10**
Prob	0.1567	0.3390	0.4565	0.2643	0.7817	0.1722	0.3358	0.1159	0.2432	0.1740
Prob²	0.0246	0.1149	0.2084	0.0699	0.6111	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 1	0.0246	0.1149	0.2084	0.0699	0.6111	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 2	0.0246	0.1149	0.2084	0.0699	0.3055	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 3	0.0246	0.1149	0.2084	0.0699	0.2037	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 4	0.0246	0.1149	0.1042	0.0699	0.2037	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 5	0.0246	0.1149	0.1042	0.0699	0.1528	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 6	0.0246	0.1149	0.1042	0.0699	0.1222	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 7	0.0246	0.1149	0.1042	0.0699	0.1018	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 8	0.0246	0.0575	0.1042	0.0699	0.1018	0.0297	0.1128	0.0134	0.0591	0.0303
Tuit 9	0.0246	0.0575	0.1042	0.0699	0.1018	0.0297	0.0564	0.0134	0.0591	0.0303
Tuit 10	0.0246	0.0575	0.0695	0.0699	0.1018	0.0297	0.0564	0.0134	0.0591	0.0303

En primer lugar, partimos de las probabilidades devueltas por el programa las cuales aplicamos elitismo elevando su valor al cuadrado. Aplicamos la fórmula de cociente a cada una de las probabilidades, pero como aún no se ha repartido ningún tuit el valor será el mismo. Repartimos el primer tuit al *Tema* con el mayor cociente, en este caso el *Tema* 5.

A continuación, como no hemos repartido todos los tuits aplicaremos de nuevo la fórmula del cociente a todos los *Temas*. En este caso tras haber repartido un tuit a la 5, su valor cambiará y repartiremos el segundo tuit al *Tema* con mayor cociente. Seguiremos el mismo proceso hasta haber repartido todos los tuits.

Finalmente, tras repartir todos los tuits el programa devolverá al usuario un conjunto de 10 tuits donde los dos primeros pertenecerán al *Tema* 5, el tercero al *Temas*3, cuarto, quinto y sexto al 5 de nuevo, séptimo al 2, octavo al 7, noveno de nuevo al 3 y finalmente el último tuit perteneciente al *Tema* 5.

5. Posibles ampliaciones.

En base a la solución implementada consistente en mostrar un conjunto aleatorio de tuits correspondiente a las categorías predichas por el programa en función de una entrada del usuario, se pueden plantear varias posibles ampliaciones para el proyecto.

Una de ellas podría ser la implementación de un sistema en tiempo real. Esto consistiría en que el usuario proponga una frase de entrada y el programa le devuelva un conjunto de tuits en tiempo real correspondientes a las categorías predichas, mostrándole los tuits más recientes de esas categorías. Esta solución no es realmente interesante ya que sería prácticamente imitar la funcionalidad íntegra de *Temas* de Tuiter, por lo que no aportaría nada realmente novedoso.

En cambio, existe otra posible ampliación que si pudiera ser realmente interesante. Esta solución consiste en mejorar el conjunto de tuits mostrados basándonos en las preferencias del usuario. De esta forma el programa según las categorías predichas mostraría al usuario un conjunto de tuits de estas como hasta ahora, pero en lugar de mostrar un conjunto aleatorio de tuits, mostraría un conjunto de tuits del gusto del usuario. Tras mostrarlos, el programa interactuaría con el usuario para conocer si el conjunto mostrado es correcto o no y aplicando técnicas de Aprendizaje de Preferencias o *Preference Learning* [30] aprendería cómo mostrar tuits más afines al usuario en futuras interacciones.

Otra posible solución más compleja sería el tratamiento multiusuario. Con esta solución diferentes usuarios podrían utilizar la aplicación y los algoritmos de aprendizaje deberían utilizarse para cada usuario aprendiendo y registrando las preferencias de cada uno de ellos. Ya que las preferencias de dos usuarios no tienen por qué ser las mismas, es más podrían ser completamente diferentes, si se da el caso de que dos usuarios con gustos distintos usan la aplicación con un modelo de gustos entrenado y sin identificar qué usuario la usa en cada caso, el aprendizaje no valdría para nada. Por tanto, en este caso para hacer el programa multiusuario

sería necesario desarrollar un sistema de registro, *login* y almacenamiento de datos de usuarios teniendo en cuenta una serie de políticas de seguridad que habría que definir y llevar a cabo. Además, sería necesario un servicio de hospedaje de aplicaciones, bien sea utilizando servidores físicos o en la nube.

6. Conclusiones.

El Procesamiento de Lenguaje Natural es una disciplina en constante evolución y desarrollo que puede utilizarse en una gran variedad de ámbitos y mediante numerosas técnicas. En este caso hemos podido comprobar cómo utilizarlo en el ámbito de la clasificación de texto mediante un proceso end-to-end que abarca diferentes formas de tratamiento de texto y diferentes algoritmos para realizar las clasificaciones.

No existe una solución que sea óptima en todos los casos o aplicaciones de NLP, sino que para cada caso real se requiere un estudio previo de cuál puede ser la mejor solución y por qué. Además, al tratarse de un campo en constante evolución se irán descubriendo nuevos casos y nuevas formas de resolución de problemas que traerán grandes mejoras en las soluciones.

Los resultados obtenidos han sido muy satisfactorios ya que, a lo largo de la implementación e investigación sobre las posibles soluciones en todas las fases del proyecto se han ido comparando los resultados y determinando cuál es la mejor solución en cada caso para así mejorar progresivamente el comportamiento de la aplicación. El resultado final, aunque se trate actualmente de un prototipo y esté limitado por el conjunto inicial de tuits extraídos, es capaz de clasificar correctamente en la gran mayoría de los casos de que tema o temas está hablando el usuario identificando cuál es el tema principal y aquellos temas secundarios para posteriormente devolver un conjunto de tuits en función de la importancia de los temas obtenidos. Además, gracias a almacenar los resultados de las clasificaciones y al tener en cuenta el *feedback* del usuario a la hora de determinar si las clasificaciones son correctas o no, se permite reentrenar el conjunto de modelos lo cual acaba garantizando una mejora en los resultados según el usuario vaya utilizando la aplicación.

Respecto al ámbito académico, este proyecto ha servido para utilizar y ampliar mis conocimientos acerca de varias de las asignaturas del Máster como Interfaces Multimodales o

Métodos Basados en el Conocimiento Aplicados a la Empresa además de todas aquellas asignaturas orientadas a la programación y a la gestión de proyectos. También me ha servido como un aprendizaje importante de cara al ámbito laboral ya que la utilización de algoritmos de *Machine Learning* y algunas técnicas de NLP son requeridas día a día en numerosos proyectos para muchas empresas actualmente.

7. Referencias.

[1] "Procesamiento del lenguaje natural: Qué es y por qué es importante." https://www.sas.com/es_ar/insights/analytics/what-is-natural-language-processing-nlp.html.
[2] D. M. J. Garbade, "A Simple Introduction to Natural Language Processing," *Medium*, Oct. 15, 2018. https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32 (accessed May 14, 2020).
[3] Na8, "Procesamiento del Lenguaje Natural (NLP)," *Aprende Machine Learning*, Dec. 27, 2018. https://www.aprendemachinelearning.com/procesamiento-del-lenguaje-natural-nlp/ (accessed May 14, 2020).
[4] D. L. Yse, "Your Guide to Natural Language Processing (NLP)," *Medium*, Apr. 30, 2019. https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1 (accessed May 14, 2020).
[5] "Tokenization." https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html (accessed May 15, 2020).
[6] "NLP: Text Processing Via Stemming And Lemmatisation In Data Science Projects." https://medium.com/fintechexplained/nlp-text-processing-via-stemming-and-lemmatisation-in-data-science-projects-ad4d5176060e (accessed May 15, 2020).
[7] "A Gentle Introduction to the Bag-of-Words Model." https://machinelearningmastery.com/gentle-introduction-bag-words-model/ (accessed May 15, 2020).
[8] "Introduction to Word Embedding and Word2Vec - Towards Data Science." https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa (accessed May 15, 2020).
[9] "Inicio / Twitter," *Twitter*. https://twitter.com/home (accessed Jun. 10, 2020).
[10] Webempresa, "¿Qué es twitter? ¿Cómo funciona? ¿Cómo puedo usarlo para mi organización?," *Webempresa*, Mar. 01, 2018. https://www.webempresa.com/blog/que-es-twitter-como-funciona-2.html (accessed Jun. 10, 2020).
[11] "Hashtag (#): ¿Qué es, para qué sirve y cómo usarlo en las redes sociales?" https://rockcontent.com/es/blog/hashtags/ (accessed Jun. 10, 2020).
[12] "Procesamiento del lenguaje natural. Un poco de historia.," Oct. 01, 2019. http://www.ibm.com/developerworks/ssa/library/natural-language-processing-um-pouco-de-historia/index.html (accessed Jun. 10, 2020).
[13] J. Hutchins, "The history of machine translation in a nutshell," p. 5.
[14] "Product Introducing Topics." https://blog.twitter.com/en_us/topics/product/2019/introducing-topics.html.
[15] "Twitter now lets you follow topics, not just accounts | Engadget." https://engt.co/3dQYwlD (accessed May 15, 2020).
[16] "rtweet package | R Documentation." https://www.rdocumentation.org/packages/rtweet/versions/0.4.0 (accessed May 15, 2020).

[17] "Stemming algorithms - Snowball." https://snowballstem.org/algorithms/ (accessed May 15, 2020).
[18] "Spanish stemming algorithm." http://snowball.tartarus.org/algorithms/spanish/stemmer.html (accessed May 15, 2020).
[19] "GloVe: Global Vectors for Word Representation." https://nlp.stanford.edu/projects/glove/ (accessed May 15, 2020).
[20] Levy, Omer, and Yoav Goldberg., "'Linguistic regularities in sparse and explicit word representations.' Proceedings of the eighteenth conference on computational natural language learning. 2014."
[21] Pennington, J., Socher, R., & Manning, C. D., "Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).," Oct. 2014.
[22] "LIBSVM -- A Library for Support Vector Machines." https://www.csie.ntu.edu.tw/~cjlin/libsvm/ (accessed May 15, 2020).
[23] "Cómo retwittear." https://help.twitter.com/es/using-twitter/how-to-retweet (accessed May 25, 2020).
[24] A. Ben-Hur, "Support vector clustering," *Scholarpedia*, vol. 3, no. 6, p. 5187, Jun. 2008, doi: 10.4249/scholarpedia.5187.
[25] "Regresión logística simple y múltiple." https://www.cienciadedatos.net/documentos/27_regresion_logistica_simple_y_multiple (accessed May 25, 2020).
[26] "scikit-learn: machine learning in Python — scikit-learn 0.23.0 documentation." https://scikit-learn.org/stable/ (accessed May 15, 2020).
[27] "sklearn.model_selection.train_test_split — scikit-learn 0.23.0 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (accessed May 15, 2020).
[28] J. R. Q. Pérez, "Procesamiento del Lenguaje Natural," p. 13.
[29] "Sistema D'Hondt," *Wikipedia, la enciclopedia libre*. May 28, 2020, Accessed: Jun. 04, 2020. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Sistema_D%27Hondt&oldid=126453452.
[30] J. Fürnkranz and E. Hüllermeier, "Preference Learning and Ranking by Pairwise Comparison," in *Preference Learning*, J. Fürnkranz and E. Hüllermeier, Eds. Berlin, Heidelberg: Springer, 2011, pp. 65–82.

8. Apéndices.

8.1.- TABLAS CON LOS RESULTADOS COMPLETOS DE LA EXPERIMENTACIÓN.

A continuación, se mostrarán los resultados completos de F1 por ejemplo, F1 Macro, F1 Micro y Hamming Loss para las distintas transformaciones del texto y algoritmos. En el caso de F1 por ejemplo los valores *True Positive, False Negative*, *False Positive* y *True Negative* son calculados para cada una de las pruebas realizadas y en el caso de F1 Macro son calculados respecto a cada uno de los *Temas*. Si realizamos el sumatorio de estos valores en F1 por ejemplo o en F1 Macro tendremos el total para el conjunto de pruebas y *Temas* que son los valores representados a la hora de calcular F1 Micro.

8.1.1.- Texto sin procesar.

Tabla 8. Resultados de F1 por ejemplo para el texto sin procesar utilizando un algoritmo SVC.

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	1256	2726	6105	20163	0.1706	0.3154	0.2215
2	1278	2735	5122	21115	0.1997	0.3184	0.2455
3	1244	2749	5370	20887	0.1881	0.3116	0.2346
4	1270	2947	4049	21984	0.2388	0.3012	0.2664
5	1047	2988	4256	21959	0.1974	0.2595	0.2242
						Media	**0.2384**

Tabla 9. Resultados de F1 por ejemplo para el texto sin procesar utilizando una Regresión Logística.

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	1200	2762	2223	24065	0.35057	0.30288	0.32498
2	1150	2863	2303	23934	0.33304	0.28657	0.30806
3	1169	2824	2250	24007	0.34191	0.29276	0.31543
4	1150	2901	2168	24031	0.34659	0.28388	0.31212
5	1148	2887	2231	23984	0.33975	0.28451	0.30968
						Media	**0.31406**

Tabla 10. Resultados de F1 Macro para el texto sin procesar utilizando un algoritmo SVC.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	519	1291	2095	8140	0.1985	0.2867	0.2346
2	353	1074	1779	8839	0.1656	0.2474	0.1984
3	465	676	2378	8526	0.1636	0.4075	0.2334
4	407	980	1857	8801	0.1798	0.2934	0.2230
5	513	1009	1988	8535	0.2051	0.3371	0.2550
6	645	1078	2290	8032	0.2198	0.3743	0.2769
7	396	1531	1355	8763	0.2262	0.2055	0.2153
8	705	1399	1452	8489	0.3268	0.3350	0.3309
9	309	982	1660	9094	0.1569	0.2393	0.1896
10	513	1196	2879	7457	0.1512	0.3002	0.2011
						Media	**0.2358**

Tabla 11. Resultados de F1 Macro para el texto sin procesar utilizando una Regresión Logística.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	443	1367	823	9412	0.34992	0.24475	0.28804
2	223	1204	896	9722	0.19929	0.15627	0.17518
3	355	786	952	9952	0.27161	0.31113	0.29003
4	320	1067	759	9899	0.29657	0.23071	0.25953
5	527	995	679	9844	0.43698	0.34625	0.38636
6	704	1019	987	9335	0.41632	0.40859	0.41242
7	595	1204	960	9286	0.38264	0.33074	0.35480
8	741	1338	1164	8802	0.38898	0.35642	0.37199
9	247	1044	796	9958	0.23682	0.19132	0.21165
10	457	1252	834	9502	0.35399	0.26741	0.30467
						Media	**0.30547**

Tabla 12. Resultados de F1 Micro para el texto sin procesar utilizando un algoritmo SVC.

TP	FN	FP	TN	Precision	Recall	F1
6095	14145	24902	106108	0.1966	0.3011	**0.2379**

Tabla 13. Resultados de F1 Micro para el texto sin procesar utilizando una Regresión Logística.

TP	FN	FP	TN	Precision	Recall	F1
5817	14237	11175	120021	0.34234	0.29007	**0.31404**

Tabla 14. Resultados de Hamming Loss para el texto sin procesar utilizando un algoritmo SVC.

Hamming Loss
0.2582

Tabla 15. Resultados de Hamming Loss para el texto sin procesar utilizando una Regresión Logística.

Hamming Loss
0.16801

8.1.2.- Transformación completa a excepción de Porter Stemmer.

Tabla 16. Resultados de F1 por ejemplo para la transformación completa sin Porter Stemmer utilizando un algoritmo SVC.

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	2181	1029	8003	12877	0.2142	0.6794	0.3257
2	2137	1081	7873	12999	0.2135	0.6641	0.3231
3	2239	971	8628	12252	0.2060	0.6975	0.3181
4	2299	964	8528	12299	0.2123	0.7046	0.3263
5	2141	1154	7787	13008	0.2157	0.6498	0.3238
						Media	**0.3234**

Tabla 17. Resultados de F1 por ejemplo para la transformación completa sin Porter Stemmer utilizando una regresión Logística

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	2464	746	5680	15200	0.3026	0.7676	0.4340
2	2494	724	5788	15084	0.3011	0.7750	0.4337
3	2459	748	5810	15073	0.2974	0.7668	0.4285
4	2498	765	5738	15089	0.3033	0.7656	0.4345
5	2522	773	5711	15084	0.3063	0.7654	0.4375
						Media	**0.4337**

Tabla 18. Resultados de F1 Macro para la transformación completa sin Porter Stemmer utilizando un algoritmo SVC.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	1187	615	3532	6711	0.2515	0.6587	0.3641
2	1049	503	4563	5930	0.1869	0.6759	0.2929
3	861	448	4504	6232	0.1605	0.6578	0.2580
4	1009	445	3839	6752	0.2081	0.6939	0.3202
5	1087	372	4184	6402	0.2062	0.7450	0.3230
6	1394	325	3927	6399	0.2620	0.8109	0.3960
7	1100	799	3450	6696	0.2418	0.5793	0.3411
8	1305	702	3360	6678	0.2797	0.6502	0.3912
9	810	488	4882	5865	0.1423	0.6240	0.2318
10	1195	502	4578	5770	0.2070	0.7042	0.3199
						Media	**0.3238**

Tabla 19. Resultados de F1 Macro para la transformación completa sin Porter Stemmer utilizando una Regresión Logística.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	1370	432	2632	7611	0.3423	0.7603	0.4721
2	1103	449	3395	7098	0.2452	0.7107	0.3646
3	1044	265	3256	7480	0.2428	0.7976	0.3723
4	1073	381	2635	7956	0.2894	0.7380	0.4157
5	1137	322	2856	7730	0.2847	0.7793	0.4171
6	1472	247	2673	7653	0.3551	0.8563	0.5020
7	1416	483	2706	7440	0.3435	0.7457	0.4704
8	1583	424	2552	7486	0.3828	0.7887	0.5155
9	991	307	3402	7345	0.2256	0.7635	0.3483
10	1248	446	2620	7731	0.3226	0.7367	0.4488
						Media	**0.4327**

Tabla 20. Resultados de F1 Micro para la transformación completa sin Porter Stemmer utilizando un algoritmo SVC.

TP	FN	FP	TN	Precision	Recall	F1
10997	5199	40819	63435	0.2122	0.6790	**0.3234**

Tabla 21. Resultados de F1 Micro para la transformación completa sin Porter Stemmer utilizando una Regresión Logística.

TP	FN	FP	TN	Precision	Recall	F1
12437	3756	28727	75530	0.3021	0.7680	**0.4337**

Tabla 22. Resultados de Hamming Loss para la transformación completa sin Porter Stemmer utilizando un algoritmo SVC.

Hamming Loss
0.3821

Tabla 23. Resultados de Hamming Loss para la transformación completa sin Porter Stemmer utilizando una Regresión Logística.

Hamming Loss
0.2697

8.1.3.- Transformación completa con restricciones.

Tabla 24. Resultados de F1 por ejemplo para la transformación completa con restricciones utilizando un algoritmo SVC.

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	1743	1439	6589	14329	0.2092	0.5478	0.3028
2	2009	1217	7583	13291	0.2094	0.6228	0.3135
3	2292	929	9066	11813	0.2018	0.7116	0.3144
4	2182	1123	7107	13688	0.2349	0.6602	0.3465
5	1936	1363	6923	13878	0.2185	0.5868	0.3185
						Media	**0.3191**

Tabla 25. Resultados de F1 por ejemplo para la transformación completa con restricciones utilizando una Regresión Logística.

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	2338	904	4906	15952	0.3227	0.7212	0.4459
2	2334	930	4871	15965	0.3239	0.7151	0.4459
3	2284	979	4773	16064	0.3237	0.7000	0.4426
4	2285	927	4938	15950	0.3164	0.7114	0.4379
5	2314	967	4928	15891	0.3195	0.7053	0.4398
						Media	**0.4424**

Tabla 26. Resultados de F1 Macro para la transformación completa con restricciones utilizando un algoritmo SVC.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	986	816	3563	6685	0.2168	0.5472	0.3105
2	1141	403	4651	5855	0.1970	0.7390	0.3111
3	722	587	3369	7372	0.1765	0.5516	0.2674
4	863	607	3203	7377	0.2122	0.5871	0.3118
5	975	434	3964	6677	0.1974	0.6920	0.3072
6	1166	555	2446	7883	0.3228	0.6775	0.4373
7	1171	752	4401	5726	0.2102	0.6089	0.3125
8	1167	848	3849	6186	0.2327	0.5792	0.3320
9	875	482	4051	6642	0.1776	0.6448	0.2785
10	1096	587	3771	6596	0.2252	0.6512	0.3347
						Media	**0.3203**

Tabla 27. Resultados de F1 Macro para la transformación completa con restricciones utilizando una Regresión Logística.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	1176	652	2316	7906	0.3368	0.6433	0.4421
2	1038	557	2780	7675	0.2719	0.6508	0.3835
3	869	417	2684	8080	0.2446	0.6757	0.3592
4	1058	460	2453	8079	0.3013	0.6970	0.4208
5	1056	368	2404	8222	0.3052	0.7416	0.4324
6	1376	325	2136	8213	0.3918	0.8089	0.5279
7	1352	584	2414	7700	0.3590	0.6983	0.4742
8	1461	489	2279	7821	0.3906	0.7492	0.5135
9	980	428	2613	8029	0.2728	0.6960	0.3919
10	1189	427	2337	8097	0.3372	0.7358	0.4625
						Media	**0.4408**

Tabla 28. Resultados de F1 Micro para la transformación completa con restricciones utilizando un algoritmo SVC

TP	FN	FP	TN	Precision	Recall	F1
10162	6071	37268	66999	0.2143	0.6260	**0.2682**

Tabla 29. Resultados de F1 Micro para la transformación completa con restricciones utilizando una Regresión Logística.

TP	FN	FP	TN	Precision	Recall	F1
11555	4707	24416	79822	0.3212	0.7106	**0.4424**

Tabla 30. Resultados de Hamming Loss para la transformación completa con restricciones utilizando un algoritmo SVC.

Hamming Loss
0.3597

Tabla 31. Resultados de Hamming Loss para la transformación completa con restricciones utilizando una Regresión Logística.

Hamming Loss
0.2417

8.1.4.- Transformación completa.

Tabla 32. Resultados de F1 por ejemplo para la transformación completa utilizando un algoritmo SVC.

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	2099	1111	6431	14449	0.2461	0.6539	0.3576
2	2314	904	6373	14499	0.2664	0.7191	0.3887
3	1931	1276	5649	15234	0.2547	0.6021	0.3580
4	1925	1338	6429	14398	0.2304	0.5899	0.3314
5	2133	1162	7697	13098	0.2170	0.6473	0.3250
						Media	**0.3522**

Tabla 33. Resultados de F1 por ejemplo para la transformación completa utilizando una Regresión Logística.

Prueba	TP	FN	FP	TN	Precision	Recall	F1
1	2345	884	4422	16439	0.3465	0.7262	0.4692
2	2329	896	4370	16495	0.3477	0.7222	0.4694
3	2385	885	4358	16462	0.3537	0.7294	0.4764
4	2393	901	4415	16381	0.3515	0.7265	0.4738
5	2390	896	4390	16414	0.3525	0.7273	0.4749
						Media	**0.4727**

Tabla 34. Resultados de F1 Macro para la transformación completa utilizando un algoritmo SVC.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	1315	487	2981	7262	0.3061	0.7297	0.4313
2	933	619	3430	7063	0.2138	0.6012	0.3155
3	875	434	4063	6673	0.1772	0.6684	0.2801
4	802	652	2520	8071	0.2414	0.5516	0.3358
5	939	520	3028	7558	0.2367	0.6436	0.3461
6	1115	604	2460	7866	0.3119	0.6486	0.4212
7	1070	829	2937	7209	0.2670	0.5635	0.3623
8	1248	759	3533	6505	0.2610	0.6218	0.3677
9	893	405	4535	6212	0.1645	0.6879	0.2655
10	1212	482	3092	7259	0.2816	0.7155	0.4041
						Media	**0.3530**

Tabla 35. Resultados de F1 Macro para la transformación completa utilizando una Regresión Logística.

Categoría	TP	FN	FP	TN	Precision	Recall	F1
1	1353	444	1893	8355	0.4168	0.7529	0.5366
2	988	575	2446	8036	0.2877	0.6321	0.3954
3	979	344	2441	8281	0.2863	0.7400	0.4128
4	1007	446	2147	8445	0.3193	0.6930	0.4372
5	1112	335	2233	8365	0.3324	0.7685	0.4641
6	1417	329	1917	8382	0.4250	0.8116	0.5579
7	1362	596	2167	7920	0.3859	0.6956	0.4964
8	1524	514	2152	7855	0.4146	0.7478	0.5334
9	880	467	2550	8148	0.2566	0.6533	0.3684
10	1220	412	2009	8404	0.3778	0.7475	0.5020
						Media	**0.4704**

Tabla 36. Resultados de F1 Micro para la transformación completa utilizando un algoritmo SVC.

TP	FN	FP	TN	Precision	Recall	F1
10402	5791	32579	71678	0.2420	0.6424	**0.3516**

Tabla 37. Resultados de F1 Micro para la transformación completa utilizando una Regresión Logística.

TP	FN	FP	TN	Precision	Recall	F1
11842	4462	21955	82191	0.3504	0.7263	**0.4727**

Tabla 38. Resultados de Hamming Loss para la transformación completa utilizando un algoritmo SVC.

Hamming Loss
0.3186

Tabla 39. Resultados de Hamming Loss para la transformación completa utilizando una Regresión Logística.

Hamming Loss
0.2193

8.2.- Anexo con el código de la aplicación

En este enlace se podrá descargar un .zip con los anexos del proyecto para poder utilizar la aplicación. El fichero contiene un archivo README donde se incluye una pequeña descripción de cada uno de los ficheros incluidos, los requisitos necesarios en cuanto a software para poder utilizar la aplicación y un ejemplo de utilización de la aplicación.