Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Visualización y seguimiento de acontecimientos en Twitter

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: José María Mares Giner

Tutor: Lluís Felip Hurtado Oliver

Ferran Pla Santamaría

2016-2017

Resumen

En este trabajo nos hemos centrado principalmente en la parte de seguimiento y análisis más que en la de visualización.

Se ha desarrollado un sistema de clasificación de tuits en categorías utilizando diferentes algoritmos de aprendizaje supervisado, que se puede acceder a través de una aplicación web desarrollada con el micro-*framework* Flask.

Por otra parte, se ha etiquetado manualmente un conjunto de tuits para entrenar los algoritmos de clasificación y a partir de estos datos se ha realizado una serie de experimentos para encontrar el algoritmo que consigue la mayor precisión.

Palabras clave: Twitter, clasificación, SVM, aprendizaje supervisado

Abstract

In this work we have mainly focused on the part of monitoring and analysis rather than the visualization.

We have developed a classification system of tweets in categories using different supervised learning algorithms, which can be accessed through a web application developed with the Flask micro-framework.

On the other hand, we have manually tagged a set of tweets to train the classification algorithms and from this data we have performed a series of experiments to find the algorithm that achieves the highest precision.

Keywords: Twitter, clasification, SVM, supervised learning

Índice general

1. Introducción 11
1.1 Objetivos 11
1.2 Estructura de la memoria 12
2. Tecnologías utilizadas 13
2.1 Python 13
2.1.1 Librerías para Python 13
2.2 MongoDB 14
2.3 JSON 14
2.4 Flask 14
2.5 Jinja2 15
2.6 HTML 15
2.7 CSS 15
2.8 Javascript 15
3. Algoritmos de clasificación 17
3.1 Máquinas de vectores de soporte 17
3.2 Árbol de decisión 18
3.3 K-Vecinos más cercanos 19
3.4 Naive Bayes 20
3.4.1 Bernoulli Naive Bayes 21
3.4.2 Multinomial Naive Bayes 21
4. Funcionamiento de la aplicación 23
4.1 Clasificar tuits 23
4.2 Lista de clasificaciones 24
4.3 Resultado de la clasificación 24
4.4 Ampliar datos de entrenamiento 25
4.5 Precisión de los clasificadores 26
5. Entrenamiento del clasificador 27
5.1 Creación de la matriz de pesos 28
5.2 Ponderación y normalización de la matriz de pesos 28
6. Evaluación de los algoritmos de clasificación 31
6.1 Resultados de los clasificadores de Naive Bayes 31
6.2 Resultados del clasificador de árbol de decisión 33

6.3 Resultados del clasificador K-vecinos más cercanos 34
6.4 Resultados de los clasificadores SVM 35
7. Conclusiones 41
7.1 Trabajo futuro 41
A. Apéndice 43
Bibliografía 53

Índice de figuras

Figura 3.1. Vista bidimensional de un hiperplano óptimo separando datos y vectores de soporte 17
Figura 3.2. Un árbol de decisión simple es utilizado para clasificar la planta de Iris. Las clases posibles son: Iris-Setosa, Iris-Versicolor e Iris-Virginica. [13] 18
Figura 3.3. Ejemplo de k-NN 20
Figura 4.1. Página principal de la aplicación 23
Figura 4.2. Página con los resultados de la colección de prueba Goya2016 24
Figura 4.3. Página de entrenamiento 25

Índice de tablas

Tabla 6.1: Precisión del clasificador BernoulliNB 32
Tabla 6.2: Precisión del clasificador MultinomialNB 33
Tabla 6.3: Precisión del árbol de decisión 33
Tabla 6.4: Precisión del clasificador k-NN 34
Tabla 6.5: Precisión del clasificador LinearSVC 35
Tabla 6.7: Precisión del clasificador SVC con kernel lineal 37
Tabla 6.8: Precisión del clsificador SVC con kernel polinómico 38
Tabla 6.9: Precisión del clasificador SVC con kernel sigmoid 39
Tabla completa de resultados: Precisión de DecisionTreeClassifier 45
Tabla completa de resultados: Precisión de KNeighborsClassifier 44
Tabla completa de resultados: Precisión de LinearSVC 46
Tabla completa de resultados: Precisión de SVC con *kernel* lineal 46
Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 1 48
Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 2. 48
Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 3. 49
Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 4. 50
Tabla completa de resultados: Precisión de SVC con *kernel rbf* 47
Tabla completa de resultados: Precisión de SVC con *kernel* sigmoid 51
Tabla completa de resultados: Precisión del clasificador BernoulliNB 43
Tabla completa de resultados: Precisión del clasificador MultinomialNB 43

1. Introducción

Estos últimos años, la cantidad de información que se genera por los servicios de internet ha ido aumentando. El CEO de Google, Eric Schmidt afirmo que cada dos días se está creando tanta información como se creó la civilización hasta el año 2003 y el ritmo sigue aumentando [21].

Las redes sociales, como Facebook o Twitter, se han vuelto muy populares estos últimos años.

Twitter es una herramienta de *microblogging* que sirve para publicar brevemente un mensaje de hasta 140 letras. Twitter es la segunda mayor red social, justo detrás de Facebook, que genera alrededor de 350.000 de tuits cada minuto. Los usuarios suelen utilizar esta plataforma para expresar sus ideas y opiniones sobre diferentes temas. En el año 2016, Twitter cuenta con más de 328 millones de usuarios activos mensuales [1].

La minería de datos juega un papel importante en las redes sociales debido a la gran cantidad de datos que se genera diariamente. Las posibilidades que ofrece la minería de datos para analizar la información son muy diversas, entre ellas está la clasificación.

Teniendo en cuenta lo anterior, se ha decidido crear una herramienta que permita clasificar los tuits en categorías aplicando algoritmos de aprendizaje supervisado, filtrando así ciertos temas que no se deseen utilizar su información.

1.1 Objetivos

El principal objetivo de este proyecto es el desarrollo de una aplicación web que permita al usuario realizar búsquedas en Twitter y filtrar los resultados por categoría (cine, política, celebridades, deporte, etc.). Concretamente, este objetivo se puede dividir en tres:

- Preparar un sistema de clasificación utilizando algoritmos de aprendizaje que clasifique los tuits en una categoría.
- Construir un corpus de tuits etiquetados manualmente.
- Desarrollar una página web que permita realizar consultas, clasificar los resultados de la consulta utilizando el sistema anterior y mostrar sus resultados.

1.2 Estructura de la memoria

La memoria se estructura de la siguiente manera:

- En el Capítulo 2 se detalla las tecnologías que se han empleado en el desarrollo de la aplicación.
- En el Capítulo 3 se explica los diferentes algoritmos de clasificación que se han utilizado.
- En el Capítulo 4 se describe el funcionamiento y el proceso de la aplicación.
- En el Capítulo 5 se detalla cómo se prepara el clasificador de tuits
- En el Capítulo 6 se describen los experimentos realizados y se muestran en tablas los resultados obtenidos.
- En el Capítulo 7 contiene las conclusiones obtenidas en este proyecto y propuestas para trabajos futuros.

2. Tecnologías utilizadas

En este capítulo se describen las tecnologías y herramientas que se han empleado para el desarrollo del proyecto.

2.1 Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Se ha elegido Python como lenguaje de programación para todo el proyecto por la simplicidad, versatilidad y rapidez de desarrollo. Además, contiene una gran variedad de librerías que facilitan el desarrollo de este trabajo.

2.1.1 Librerías para Python

- Scikit-learn es una librería de código abierto para Python dedicada al aprendizaje automático. Cuenta con varios algoritmos de clasificación, regresión y *clustering*, incluyendo máquinas de vectores de soporte o SVM (*Support Vector Machines*), Naive Bayes, etc. Scikit-learn implementa dos librerías basadas en LIBSVM y LIBLINEAR, dos librerías de aprendizaje automático empleadas en este proyecto.

  La librería se ha utilizado para facilitar la creación de matrices de pesos y para el clasificador. Se ha utilizado la versión 0.18.2.

- Tweepy es una librería de código abierto escrito en Python que ofrece todas las funcionalidades de la API de Twitter. Esta librería permite acceder a la *REST API* de Twitter que provee acceso a lectura y escritura de la información de Twitter, como realizar búsquedas o acceder a la información de un usuario en concreto [2]. También ofrece acceso a la *Streaming API* que permite procesar tuits en tiempo real [3]. Para poder acceder a Twitter, todas las conexiones deben estar certificadas con OAuth y las respuestas de las funciones están en formato JSON. Se ha empleado la versión 3.5.0 de esta librería.

- PyMongo (versión 3.4.0), una distribución de Python que ofrece las herramientas necesarias para trabajar con la base de datos MongoDB, como conectarse a una base de datos de Mongo, realizar consultas, inserciones o eliminar documentos [4].

- El kit de herramientas de lenguaje natural, o más comúnmente NLTK, es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural simbólico y estadísticos para el lenguaje de programación Python. NLTK incluye demostraciones gráficas y datos de muestra [5].

  Se ha utilizado la versión 3.2.4 de esta librería para acceder a una lista de stopwords y utilizarla en la creación de las matrices de peso.

2.2 MongoDB

MongoDB es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto [6].

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos en documentos similares a JSON con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida, como es el caso de este proyecto, ya que todos los datos devueltos por la API de Twitter se encuentran en formato JSON.

Para este proyecto se ha utilizado la versión 2.6.12 para el servidor MongoDB.

2.3 JSON

JSON (*JavaScript Object Notation*) es un formato de archivo estándar abierto que utiliza texto para transmitir los datos del objeto que consiste en pares de atributo-valor y datos de tipo *array* (o cualquier otro valor serializable). Es un formato muy común utilizado para la comunicación asincrónica entre el navegador web y el servidor.

En este proyecto se ha utilizado JSON para almacenar los tuits en la base de datos y para transmitir los resultados de la clasificación entre el servidor y el navegador.

2.4 Flask

Flask [7] es un micro *framework* escrito en Python y basado en Werkzeug y *templates* Jinja2. Se le llama micro *framework* porque no requiere herramientas particulares o librerías. No tiene capa de abstracción para la base de datos, validación de formularios, o cualquier otro componente en el que las bibliotecas preexistentes proporcionan funciones comunes. Sin embargo, Flask acepta extensiones que pueden añadir funciones a la aplicación como si fueran implementadas en Flask. Existen extensiones para el mapeado de objetos relacionales, validación de formulario, controlador de subidas, varias tecnologías de autenticación abierta y varias herramientas relacionadas con el *framework*. Para este proyecto se ha utilizado la versión 0.12.

2.5 Jinja2

Jinja2 es un motor de plantillas o *templates* para Python y motor por defecto de Flask. Una *template* es simplemente un archivo de texto que contiene variables o expresiones, que se reemplazan por valores cuando se procesa, y etiquetas, que controlan la lógica de la plantilla. Jinja puede generar cualquier formato basado en texto (HTML, XML, CSV, LaTeX, etc). La sintaxis de la plantilla está inspirada en las plantillas de Django.

Se ha utilizado Jinja2 para generar archivos HTML a partir de los resultados

2.6 HTML

HTML, sigla en inglés de *HyperText Markup Language* (lenguaje de marcas de hipertexto) [8], hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la *World Wide Web* (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

HTML5 es la última versión de HTML [9], En esta versión se introducen características nuevas, como incluir y manipular contenido gráfico y multimedia. Gracias a estas nuevas características, se ha utilizado el elemento <*canvas*> junto a Javascript para generar una gráfica con los resultados de las clasificaciones.

2.7 CSS

Hojas de estilo en cascada (CSS, del inglés *Cascading Stylesheets*) [8] es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de las páginas web, e interfaces de usuarios escritas en HTML. Se ha empleado la última versión, CSS3.

2.8 Javascript

Javascript es un lenguaje de programación interpretado utilizado normalmente en las páginas web permitiendo mejoras en la interfaz de usuario y páginas webs dinámicas.

En esta aplicación, se ha utilizado Javascript para ocultar elementos, como botones y texto, validar los formularios antes de enviar la solicitud al servidor, realizar peticiones asíncronas con el servidor y, como se ha mencionado en el apartado 2.6, crear dinámicamente las gráficas de los resultados de las clasificaciones. Para esto último se ha utilizado el elemento "*canvas*" de HTML5 y la librería Chart.js.

3. Algoritmos de clasificación

En este capítulo se explican los distintos algoritmos de aprendizaje utilizados en este proyecto.

3.1 Máquinas de vectores de soporte

Las máquinas de vectores de soporte (SVM o *Support Vector Machine*) [10] son un conjunto de métodos de aprendizaje supervisado utilizado para la clasificación binaria y regresión. Dado un conjunto de muestras de entrenamiento, cada una perteneciente a una de las dos clases, un algoritmo de entrenamiento SVM construye un modelo que decide a qué clase pertenece la nueva muestra.

En un modelo SVM, un punto se ve como un vector n-dimensional, en un espacio n-dimensional donde los puntos se pueden separar con un hiperplano (n-1)-dimensional (plano canónico). A esto se le conoce como clasificador lineal. Existen infinitos planos que puedan clasificar los datos, pero la mejor opción es aquel hiperplano que represente la mayor separación, o margen, entre las dos clases. Si existe tal plano, se le conoce momo hiperplano de margen máximo y el clasificador lineal se define como un clasificador de margen máximo.

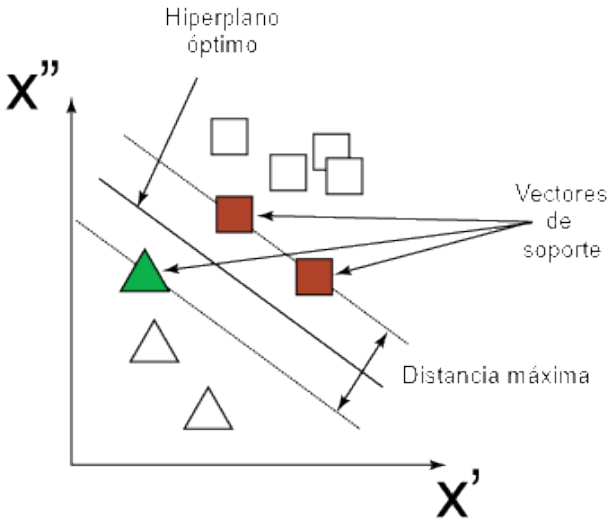

Figura 3.1. Vista bidimensional de un hiperplano óptimo separando datos y vectores de soporte. [13]

En la práctica, los datos reales están desordenados y no pueden separarse perfectamente con un hiperplano. La restricción de maximizar el margen de la línea que separa las clases debe ser más flexible, a esto se le llama margen blando (*soft margin*) [12]. Este cambio permite que algunos puntos en los datos de entrenamiento no violen la línea de separación. Se introduce un parámetro de ajuste *C* que determina el margen de error permitido.

Además de la clasificación lineal, las SVM pueden realizar clasificación no lineal utilizando funciones núcleo o *kernel*, que transforman el espacio de atributos de

entrada en un espacio de características de mayor dimensionalidad donde es posible separar linealmente las muestras. Existen diferentes tipos de *kernel*, entre ellos [10]:

- Linear: $\langle x, x' \rangle$
- Polinomial: $(\gamma \langle x, x' \rangle + r)^d$
- RBF: $\exp(-\gamma \|x - x'\|^2)$
- Sigmoid: $\tanh(\gamma \langle x, x' \rangle + r)$

Las SVM son clasificadores binarios y para poder realizar clasificaciones cuando existe más de dos clases se emplea un clasificador "uno contra el resto" (*one-vs-the-rest*) o "uno contra uno" (*one-vs-one*).

El clasificador *one-vs-the-rest* consiste en aplicar un clasificador por clase. Para cada clasificador, la clase se ajusta contra todas las demás.

El clasificador *one-vs-one* construye un clasificador por cada par de clases. A la hora de predecir, se selecciona la clase que reciba la mayoría de los votos. Este método es más lento que *one-versus-the-rest*, ya que requiere entrenar $C \cdot (C - 1)/2$ clasificadores.

3.2 Árbol de decisión

El aprendizaje por árbol de decisión es un método comúnmente utilizado en la minería de datos. Su propósito es crear un modelo que prediga el valor de la variable del objetivo aprendiendo simples reglas de decisión deducidas a partir de los datos de entrada.

Una de las variables de entrada se selecciona en cada nodo interno (o nodo interno, que no es un nodo terminal) del árbol de acuerdo con un método que depende del algoritmo. Cada nodo interior (nodo no terminal) corresponde a una de las variables de entrada; de manera que el conjunto de aristas para los hijos cubra todos los valores posibles de la variable de entrada. Cada hoja representa una variable objetivo dado los valores de las variables de entrada representados por el camino desde la raíz hasta la hoja.

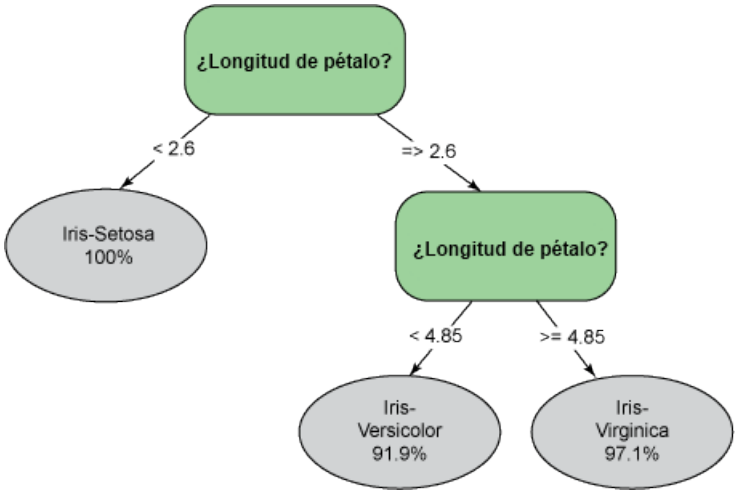

Figura 0.1. Un árbol de decisión simple es utilizado para clasificar la planta de Iris. Las clases posibles son: Iris-Setosa, Iris-Versicolor e Iris-Virginica. [13]

Entre los posibles clasificadores basados en árboles se encuentran ID3, C4.5, C5.0 y CART. Para este proyecto se utilizará CART, puesto que la librería *scikit-learn* implementa una versión optimizada de este algoritmo [14].

Los arboles de clasificación y regresión (*Classification And Regression Trees* o CART) es una técnica de aprendizaje de árbol de decisión no paramétrica que produce arboles binarios usando la característica y el umbral que producen la mayor ganancia de información en cada nodo.

Los arboles de decisión se forman a partir de una colección de reglas basadas en variables en el modelado del conjunto de datos:

- Se seleccionan las reglas basadas en los valores de las variables para conseguir la mejor partición.
- Una vez seleccionada una regla y se divide un nodo en dos, se aplica el mismo proceso a cada hijo.
- El particionado se detiene cuando CART detecta que no se puede realizar más ganancias o se cumplen algunas reglas de parada preestablecidas. Como alternativa, los datos se dividen tanto como sea posible y a continuación se poda el árbol.

3.3 K-Vecinos más cercanos

La clasificación de del vecino [15] más cercano *(k-nearest neighbors* o k-NN) es un método no paramétrico para estimar las funciones de densidad de probabilidad o probabilidad a posteriori de que un elemento x pertenezca a la clase C_j. El algoritmo knn es un metodo de clasificacion, en el que se realiza una asignación de clase teniendo en cuenta sus k-vecinos más cercanos. La parte del aprendizaje consiste simplemente en almacenar los ejemplos de entrenamiento.

La clasificación de un objeto $x \in \mathbb{R}^n$ (a menudo descrito por un vector de características) se lleva a cabo, en caso más simple, por decisión mayoritaria. En esta decisión participan los k-objetos clasificados más cercanos a x. Existen diferentes mediciones de distancia (distancia euclidiana, distancia Minkowski, etc.). A x se le asigna la clase que tenga el mayor número de objetos de estos k-vecinos.

Para una pequeña k existe un riesgo de que el ruido en los datos de entrenamiento empeore los resultados de la clasificación. Y para valores altos de k, suprime los efectos de ruido, pero hace que los límites de clasificación sean menos distintos.

La clasificación del vecino más cercano usa pesos uniformes, el valor asignado a un punto de consulta se calcula a partir de un voto de mayoría simple de los vecinos más cercanos. En algunas circunstancias, es mejor ponderar a los vecinos de manera que los votos de los vecinos más cercanos tengan mayor valor.

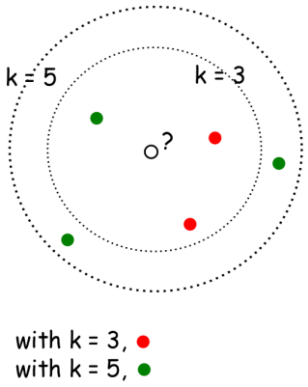

Figura 0.1. Ejemplo de k-NN

3.4 Naive Bayes

Los clasificadores de Naive Bayes [16][17] son un conjunto de algoritmos de aprendizaje supervisado basado en la aplicación del teorema de Bayes con la ingenua suposición de independencia entre cada par de características. Dada una clase c y un vector de característica dependiente $\vec{x}= \langle x_1, \dots, x_m \rangle$, el teorema de Bayes establece la siguiente relación:

$$P(c \mid \vec{x}) = \frac{P(c)P(\vec{x} \mid c)}{P(\vec{x})}$$

Utilizando la ingenua suposición de la independencia de que

$$P(x_i \mid c, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m) = P(x_i \mid c)$$

Para todo i, esta relación se simplifica a

$$P(c \mid \vec{x}) = \frac{P(c) \prod_{i=0}^{m} P(x_i|c)}{P(\vec{x})}$$

Dado que $P(x_1, \dots, x_n)$ es constante dada la entrada, se puede usar la siguiente regla de clasificación:

$$P(c|\vec{x}) \propto P(c) \prod_{i=1}^{m} P(x_i|c)$$

$$\hat{c} = \arg\max_{c} P(c) \prod_{i=1}^{m} P(x_i|c)$$

Y se puede usar una estimación máxima a posteriori para estimar $P(c)$ y $P(x_i|c)$. La probabilidad a priori $P(c)$ es la frecuencia relativa de la clase c en el conjunto de

entrenamiento. Las probabilidades $P(\vec{x}|c)$ se de forma diferente en cada versión de naive Bayes.

A pesar de sus supuestos aparentemente simplificados, los clasificadores naive Bayes han funcionado bastante bien en muchas situaciones reales, la famosa clasificación de documentos y el filtrado de spam. Requieren una pequeña cantidad de datos de entrenamiento para estimar los parámetros necesarios.

3.4.1 Bernoulli Naive Bayes

Sea $F = \{t_1, \dots, t_m\}$ el conjunto de *tokens* que corresponden a los m atributos después de la selección de atributos. Bernoulli naive Bayes trata cada muestra d como un conjunto de *tokens*, conteniendo, solo una vez, cada t_i que ocurra en d. Por lo tanto, d se puede representar como un vector binario $\vec{x} = \langle x_1, \dots, x_m \rangle$, donde cada x_i muestra si t_i ocurre en d. La probabilidad de que t_i ocurra en d es $P(t_i|c)$. Bernoulli naive Bayes hace la suposición adicional de que los resultados de las pruebas son independientes dada la categoría. La regla de decisión para Bernoulli naive Bayes se basa en:

$$P(\vec{x}|c) = \prod_{i=1}^{m} P(t_i|c)^{x_i}\big(1 - P(t_i|c)\big)^{1-x_i}$$

3.4.2 Multinomial Naive Bayes

El multinomial naive Bayes trata cada mensaje d como un conjunto de *tokens*, conteniendo cada t_i tantas veces como ocurra en d. Por lo tanto, d se puede representar cono un vector $\vec{x} = \langle x_1, \dots, x_m \rangle$, donde cada x_i es ahora el número de ocurrencias de t_i en d. Además cada muestra d de categoría c se ve como el resultado de escoger independientemente $|d|$ *tokens* de Fcomo reemplazo, con la probabilidad $P(t_i|c)$ para cada t_i. Entonces, $P(\vec{x}|c)$ es la distribución multinomial:

$$P(\vec{x}|c) = P(|d|) \cdot |d|! \cdot \prod_{i=1}^{m} \frac{P(t_i|c)^{x_i}}{x_{i!}}$$

Se ha seguido la hipótesis común de que $|d|$ no depende de la categoría c.

4. Funcionamiento de la aplicación

En este capítulo se detalla el funcionamiento de la aplicación.

4.1 Clasificar tuits

Desde el apartado clasificar, el usuario puede realizar búsquedas sobre Twitter o clasificar la colección de pruebas. Por defecto, esta seleccionado la opción de utilizar el mejor clasificador, pero si el usuario decide no usar esta opción, le aparecerá un formulario para elegir el clasificador y sus parámetros.

Cuando el usuario finaliza el formulario, se realizará una petición POST al servidor a la dirección `/classify`, que contendrá la búsqueda, el clasificador y sus parámetros.

Una vez realizada la petición, el servidor comprobará que se han introducido datos correctamente, en caso de que no sea así se redireccionará al usuario a la página anterior.

El servidor creará un clasificador a partir de los datos del formulario y se entrenará utilizando los datos de entrenamiento almacenados en la colección `train` de la base de datos MongoDB. Cada documento de esta colección tiene un campo `text` que contiene el texto del tuit y el campo `category` que indica a qué categoría pertenece. Con estos datos, se crea una matriz de pesos para entrenar el clasificador. Este proceso se verá con más detalle en el capítulo 5.

Una vez preparado el clasificador, se crea un hilo que empezará a clasificar 1000 tuits descargados a partir de la búsqueda o los nuevos tuits que se cuelgan en Twitter utilizando la *Streaming* API de Twitter, o la colección de prueba si se ha seleccionado esta opción.

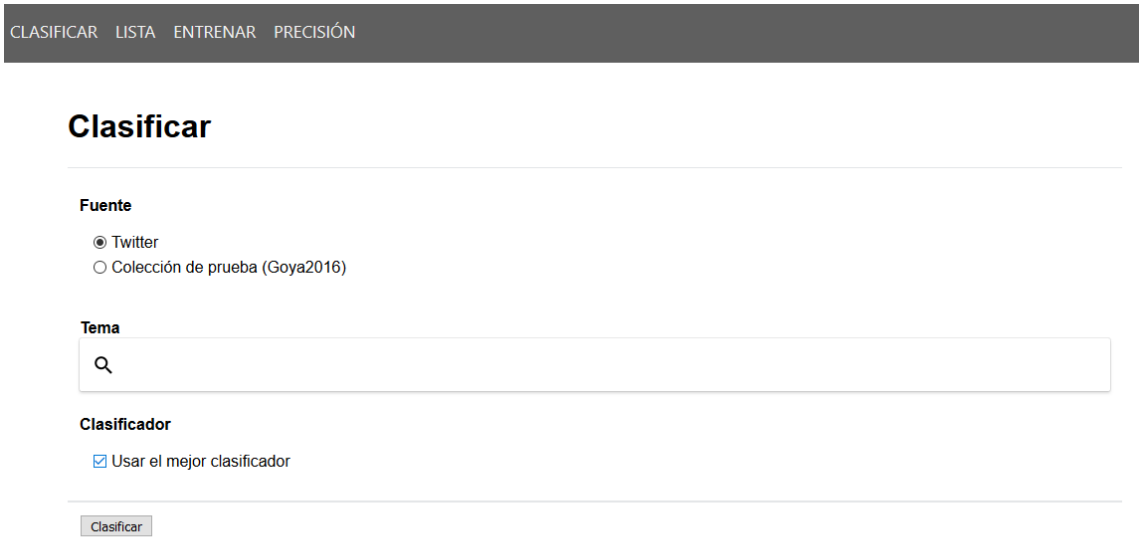

Figura 4.1. Página principal de la aplicación

4.2 Lista de clasificaciones

Después de crear el clasificador, la aplicación redireccionará al usuario a la página `/classify/list`. Esta página contiene una lista de los temas que se están clasificando o ya han sido clasificados. A partir de esta página el usuario puede decidir si ver los resultados de una clasificación, detenerla o eliminarla. Según la opción elegida se realizará una petición a diferentes direcciones. Si se ha elegido ver los resultados, se hará una petición a la dirección `/result/<query>/` donde `<query>` es la búsqueda realizada.

4.3 Resultado de la clasificación

La página del resultado contiene un gráfico que se dibuja en el navegador del usuario a partir de los resultados del clasificador utilizando la librería Chart.js y la etiqueta `<canvas>` de HTML5. Este grafico muestra el número de *tuits* y el porcentaje que se han clasificado en una categoría.

En la parte inferior de la página se muestra una lista con 15 *tuits* clasificados y unos botones para mostrar solo los *tuits* de una categoría y otros botones para mostrar los 15 siguientes o los 15 anteriores. Todos estos botones realizan una petición asíncrona con el servidor en segundo plano a la pagina `/result/request/<query>/<category>/<page>/` donde `<query>` es la búsqueda realizada, `<category>` la

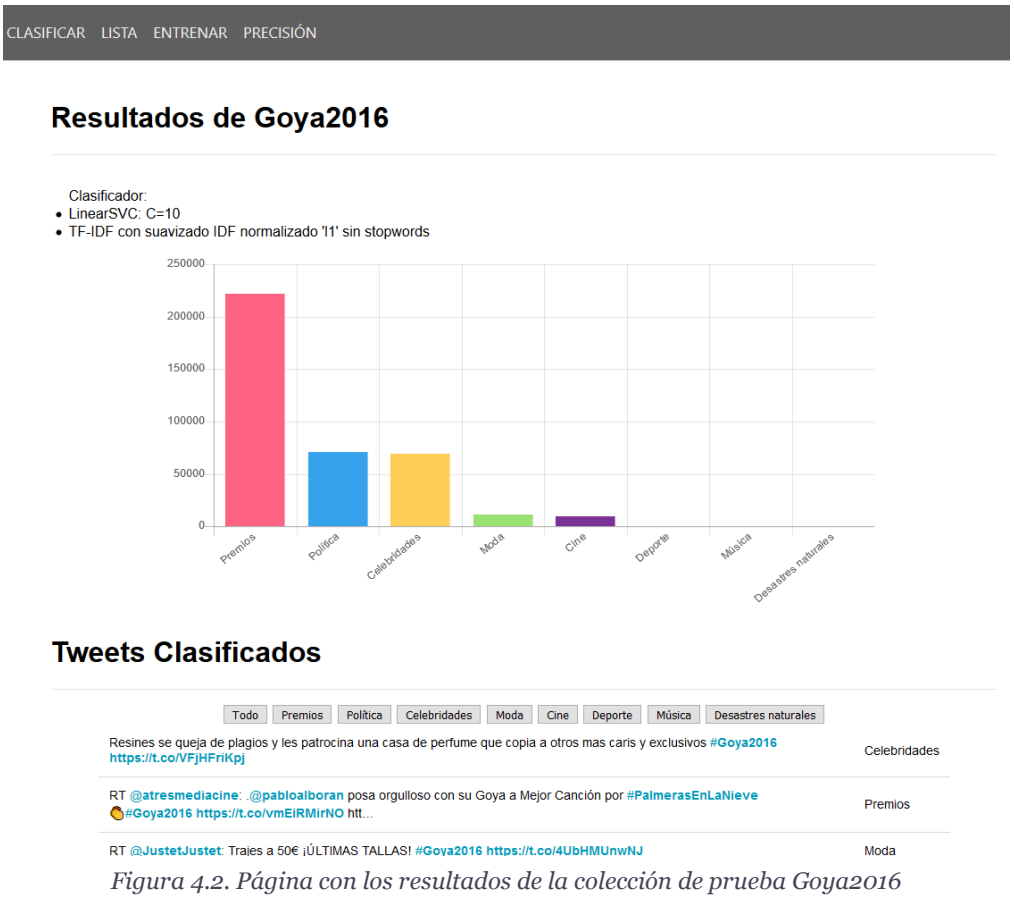

Figura 4.2. Página con los resultados de la colección de prueba Goya2016

categoría que se desea filtrar y `<page>` el número de página que se quiere ver. Esta petición devuelve un documento JSON con 15 *tuits* con su correspondiente categoría que se utilizará para actualizar la lista de *tuits*.

4.4 Ampliar datos de entrenamiento

El apartado "Entrenar" se utiliza para ampliar los datos de entrenamiento del clasificador. Esta página contiene un cuadro de búsqueda para encontrar tuits que contengan un texto introducido por el usuario. Al buscar un texto, la página realiza una petición a la dirección `/training/request/<query>/<page>/` donde `<query>` es el texto que se desea buscar y <page> es el número de página, cada página contiene 50 tuits. Utilizando estos datos, el servidor obtiene una lista de tuits usando la REST API de Twitter que posteriormente se le devuelve al usuario para que los clasifique manualmente. Una vez hecho esto, se le envía al servidor un archivo JSON con los tuits y sus respectivas categorías para insertarlos en la colección de entrenamiento.

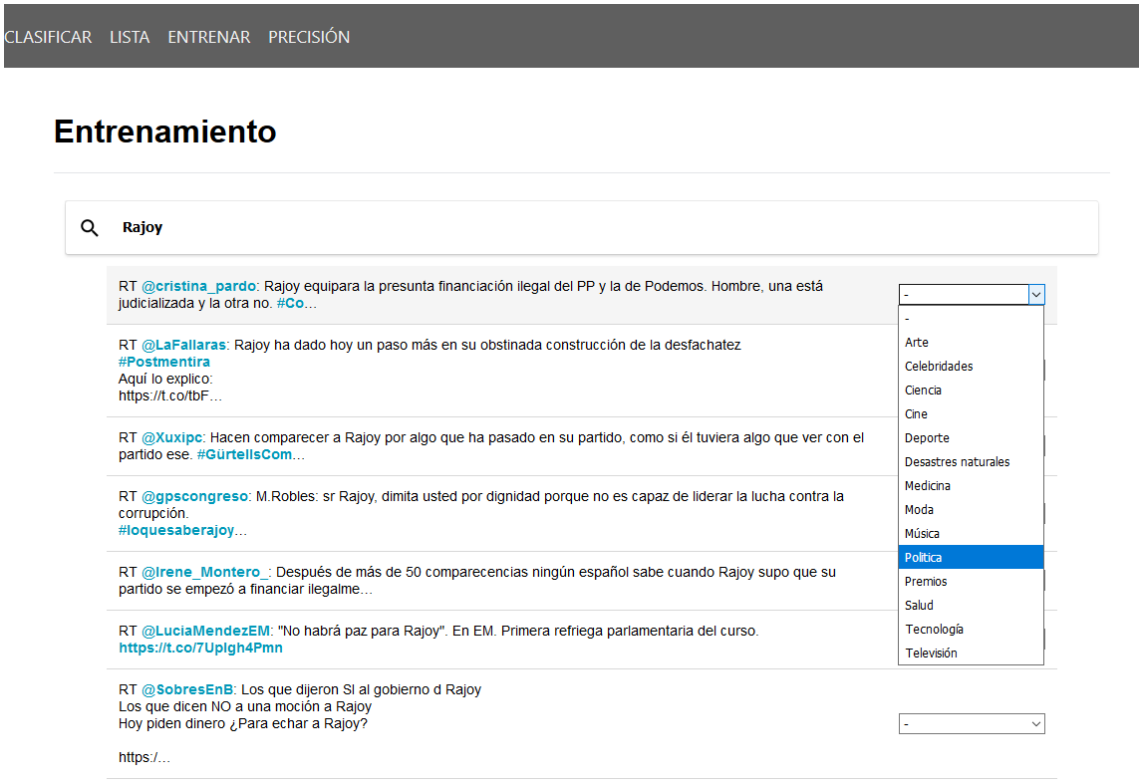

Figura 4.3. Página de entrenamiento

4.5 Precisión de los clasificadores

Esta página calcula y muestra la precisión de todos los clasificadores ordenados de mayor a menor. Para obtener estos resultados, el servidor crea un hilo que evalúa, usando un procedimiento llamado validación cruzada, los clasificadores con distintos parámetros utilizando la colección de entrenamiento. Este proceso puede durar horas en completarse.

5. Entrenamiento del clasificador

En este capítulo se describe como se entrena un clasificador para que sea capaz de predecir a qué clase pertenecen los tuits.

En primer lugar, se ha creado un corpus de entrenamiento, compuesto por 821 tuits etiquetados manualmente, a partir de un corpus de los Goya del año 2016 y de otros tuits para ampliar el número de categorías.

Se ha estudiado el corpus de los Goya y se ha observado que la mayoría de los tuits se podían etiquetar en cinco clases distintas, por ejemplo:

1. Premios (267 muestras):

- RT @rtve: 'Truman', la gran triunfadora de los Goya con cinco premios. #Goya2016
- RT @PremiosGoya: El ganador del #Goya2016 a la Mejor Canción Original es para @pabloalboran y @LucasVidalmusic.
- RT @aicortometraje: Enhorabuena a #Alike por ese Premio Goya a Mejor Cortometraje de Animación.Y, por supuesto, al resto de nominados.#G...,

2. Política (171 muestras):

- RT @juanrallo: Si aumentar el IVA al cine destruye la industria del cine,¿q efectos tendrá aumentar impuestos al resto d trabajadores y emp...
- Llevan 4 años diciendo que en España se recortan derechos,...caras de preocupación.LA RICA IZQUIERDA #Goya2016
- RT @Gotaku: "El partido Popular es implacable con la corrupción" goya a la mejor película de ciencia ficción #Goya2016

3. Celebridades (127 muestras):

- RT @elpais_cultura: Hoy Pablo Alborán ha sido el más seguido en los #Goya2016 (han roto hasta una mampara). Así es s...
- RT @ana_morgade: LA CARA DE TIM ROBBINS. #Goya2016
- RT @divinity_es: Isabel y Mario o cómo ser un adolescente enamorado llegando de la mano a los #Goya2016

4. Moda (76 muestras):

- Rovira estuvo muy bien, pero el cuello del esmoquin le sentaba de pena. #goya2016 dani rovira
- Top trend: #Goya2016 Los vestidos maravillos en tonos serenity blue, uno de los colores protagonistas de esta...,
- Una de las mejor vestidas para la noche de los #Goya2016 es Manuela Vellés

5. Cine (75 muestras):

- RT @JosPastr: Resumen reivindicación de la gala de los #Goya2016:Si ver un truño de película española te cuesta 6 €, no vas a verla. Si c...
- RT @ristomejide: Mucha pereza. Lo siento pero, bromas aparte, sigo pensando que nuestro cine y nuestra cultura merecen algo mejor. Buenas n...
- RT @clarialonso_: El CINE es CULTURA #Goya2016 !!!
- RT @Lentejitas: ... pero es que 96 películas españolas (el 63,2% del total) tuvieron menos de 10.000 espectadores. Exitazo total del cine e...

Además, se ha ampliado el corpus de entrenamiento con 51 tuits clasificados en deporte; 42, en desastres naturales; y 12, en música.

Sobre estos tuits, se construye una matriz de pesos que se utilizará para entrenar un clasificador.

En los siguientes apartados, se detalla cómo se crea esta matriz.

5.1 Creación de la matriz de pesos

La primera parte consiste en crear una matriz donde cada fila representa un documento cada columna sea un *token* que ocurra en los documentos. En este caso, las filas serán *tuits* individuales y las columnas serán palabras o n-gramas utilizados en los *tuits*. Los valores de la matriz indican la frecuencia de termino. La frecuencia de termino $tf_{t,d}$ es la frecuencia del termino t en el documento d, es decir, el número de veces que el termino t ocurre en el documento d.

Antes de empezar a crear la matriz de pesos, hay que preprocesar el texto de los *tuits* que consiste en eliminar todos los símbolos de puntuación, convertir el texto a minúscula y la eliminación de las *stopwords*, palabras que por su frecuencia y semántica no poseen valor discriminatorio alguno como artículos, pronombres, preposiciones, etc.

Realizar todo este proceso se encarga la clase `CountVectorizer` de le librería *scikit-learn*.

5.2 Ponderación y normalización de la matriz de pesos

Una vez creada la matriz de pesos, hay que ponderarla y normalizarla. Para ello se utiliza la clase `TfidfTransformer`.

Para ponderar la matriz se le aplica una transformación tf-idf. La frecuencia de termino -frecuencia inversa de documento o tf-idf (del inglés *Term frequency – inverse document frequency*), es una medida numérica que expresa cuán relevante es una palabra para un documento en una colección.

La frecuencia inversa del documento (idf) es una medida que indica si el término es común o no en la colección de documentos. Se define el idf [18] como:

$$\text{idf}_t = \log \frac{N}{\text{df}_t}$$

donde N es el número de documentos en una colección y df_t es el número de documentos en una colección que contienen el termino t. Así pues, el idf de un término raro será elevado, mientras que el idf de un término frecuente será bajo.

En la librería *scikit-learn* el idf_t se calcula como [19]:

$$\text{idf}_t = \log \frac{N}{\text{df}_t} + 1$$

El efecto de añadir un "1" a la fórmula del idf_t es que términos con cero idf, es decir, términos que ocurren en todos los documentos del conjunto de entrenamiento, no se ignore completamente.

El idf suavizado, según la librería *scikit-learn*, consiste en sumar "1" al numerador y al denominador del idf como si un documento extra contuviera cada término de la colección exactamente una vez evitando así las divisiones entre cero.

$$\text{idf}_t = \log \frac{N+1}{\text{df}_t + 1} + 1$$

El tf-idf asigna al termino t un peso en el documento d dado por

$$\text{tfidf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

El $\text{tfidf}_{t,d}$ asigna al termino t un peso en un documento d de modo que cuando t ocurre muchas veces dentro de un pequeño grupo de documentos se asignará un peso alto; cuando t ocurre pocas veces en un documento, un valor bajo; o cuando el termino ocurre en casi todos los documentos, un peso muy bajo.

Las SVM intentan maximizar la distancia entre el plano de separación y los vectores de soporte. Si una característica tiene valores muy grandes, dominará sobre las otras características cuando se calcule la distancia. Para solucionar este problema, se puede aplicar una normalización a la matriz con el fin de reescalar todas las características para que todas tengan la misma influencia en la métrica de la distancia. En este proyecto se ha optado utilizar la norma L1 y la norma L2 [20].

La norma L1 de un vector $\vec{v}$ se indica como $\|\vec{v}\|_1$ y se define como la suma de los valores absolutos de sus componentes:

$$\|\vec{v}\|_1 = \sum_{i=1}^{n} |v_i|$$

$$v_{L1} = \frac{v}{\|\vec{v}\|_1} = \frac{v}{\sum_{i=1}^{n} |v_i|}$$

La norma L2 o norma euclidea de un vector $\vec{v}$ se indica como $\|\vec{v}\|_2$ y se define como la raíz cuadrada de la suma de los cuadrados del valor absoluto de sus componentes:

$$\|\vec{v}\|_2 = \sqrt{\sum_{i=1}^{n} |v_i|^2}$$

$$v_{L2} = \frac{v}{\|\vec{v}\|_2} = \frac{v}{\sqrt{\sum_{i=1}^{n} |v_i|^2}}$$

Una vez normalizada la matriz, se utilizará esta matriz junto a una lista con las clases de los tuits para entrenar un clasificador usando los algoritmos descritos en el capítulo 3. En el siguiente capítulo se mostrará qué algoritmo funciona mejor y con qué parámetros.

6. Evaluación de los algoritmos de clasificación

Existen diferentes clasificadores y métodos para crear matrices de pesos. En este capítulo se realizará una serie de experimentos para encontrar el clasificador con mejores resultados.

Para la realización de estos experimentos se ha preparado un conjunto de 821 tuits clasificados a mano como datos de entrenamiento.

Se ha empleado la clase *GridSearchCV*, en la librería *scikit-learn*, que permite probar un subconjunto de parámetros en el clasificador y en la matriz de pesos e indicar la precisión a partir de un método de validación cruzada.

En concreto, se ha utilizado la validación cruzada de 10 iteraciones, que consiste en dividir los datos de muestra en 10 subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y el resto como datos de entrenamiento. Este proceso es repetido para cada subconjunto y finalmente se hace la media aritmética de los resultados para obtener un único resultado.

Las pruebas se han realizado con distintos clasificadores y parámetros con distintas matrices de peso. Estas matrices se han creado utilizando cualquier combinación posible de las siguientes opciones:

- Eliminar las stopwords o mantenerlas
- Utilizar n-gramas (bigramas)
- Transformación tf, tf-idf o tf-idf suavizado
- Aplicar la norma L1 o L2

A continuación, se mostrarán los resultados obtenidos en esta prueba. En el Apéndice 1 se pueden ver las tablas completas.

6.1 Resultados de los clasificadores de Naive Bayes

Existen distintos clasificadores de naive Bayes, pero únicamente se han realizado las pruebas con el clasificador de Bernoulli y Multinomial, las dos variantes más utilizadas para la clasificación de texto.

En ambos clasificadores se han realizado pruebas cambiando los valores de los parámetros `alpha` y `fit_prior`. El primero asigna un valor a α para el suavizado de Laplace ($\alpha=1$) o Lidstone ($\alpha<1$), y el otro parámetro se indica si aprende las probabilidades a priori o no, en caso de que no, se utilizará una probabilidad a priori uniforme.

El clasificador de Bernoulli utiliza la clase `BernoulliNB`, los parámetros utilizados en el `GridSearchCV` son los siguientes:

- *alpha* ∈ {0.001, 0.01, 0.1, 1}
- *fit_prior* ∈ {True, False}

Como el clasificador Bernoulli "binariza" los datos de entrada, a los valores de entrada mayores que cero se le asigna un uno mientras que a los valores con valor cero siguen manteniendo su valor, por lo tanto, no hace falta aplicarles el idf o el idf suavizado a la matriz de peso ni normalizarla. Tan solo afectará a la matriz la eliminación de las stopwords y el uso de n-gramas. Se puede comprobar en la tabla completa en el anexo 1.

	alpha = 0.001		*alpha* = 0.01		*alpha* = 0.1		*alpha* = 1	
fit_prior	True	False	True	False	True	False	True	False
TF	0.667	0.680	0.686	0.685	0.686	0.678	0.370	0.374
TF stopwords	0.699	0.691	**0.711**	**0.705**	**0.702**	**0.710**	0.373	0.375
TF n-gramas	0.671	0.672	0.677	0.682	0.635	0.644	0.346	0.346
TF n-gramas y stopwords	0.685	0.678	0.689	0.693	0.648	0.660	0.339	0.339

Tabla 6.1: Precisión del clasificador BernoulliNB

Este clasificador en general ofrece buenos resultados, pero los mejores se consiguen cuando se eliminan las *stopwords* de la matriz de peso llegando a obtener valores por encima de 0.7.

La clase `MultinomialNB` es la utilizada para el clasificador Multinomial. Se han empleado los mismos parámetros que en el clasificador de Bermoulli y estos son los resultados.

	alpha = 0.001		*alpha* = 0.01		*alpha* = 0.1		*alpha* = 1	
fit_prior	True	False	True	False	True	False	True	False
TF l1	0.571	0.670	0.463	0.669	0.331	0.516	0.325	0.337
TF l1 sw	0.652	0.697	0.607	**0.706**	0.392	0.683	0.325	0.430
TF l1 n-gram	0.585	0.687	0.423	0.669	0.325	0.457	0.325	0.333
TF l1 n-gram sw	0.655	0.693	0.554	**0.714**	0.348	0.620	0.325	0.406
TF l2	0.654	0.659	0.653	0.666	0.558	0.672	0.353	0.415
TF l2 sw	0.687	0.685	0.693	0.688	0.665	**0.713**	0.391	0.590
TF l2 n-gram	0.669	0.674	0.669	0.686	0.546	0.633	0.351	0.402
TF l2 n-gram sw	0.683	0.687	0.686	0.692	0.629	0.683	0.384	0.544
TF-IDF l1	0.613	0.654	0.571	0.669	0.370	0.654	0.325	0.458
TF-IDF l1 sw	0.652	0.667	0.646	0.697	0.434	**0.721**	0.325	0.636
TF-IDF l1 n-gram	0.627	0.660	0.524	0.682	0.329	0.594	0.325	0.449
TF-IDF l1 n-gram sw	0.652	0.680	0.610	0.699	0.372	0.695	0.325	0.598
TF-IDF suav. l1	0.611	0.657	0.566	0.669	0.369	0.655	0.325	0.454
TF-IDF suav. l1 sw	0.657	0.672	0.647	0.694	0.431	**0.724**	0.325	0.630
TF-IDF suav. l1 n-gram	0.624	0.663	0.518	0.685	0.329	0.590	0.325	0.442

TF-IDF suav. l1 n-gram sw	0.649	0.680	0.610	**0.704**	0.372	0.688	0.325	0.592
TF-IDF l2	0.635	0.638	0.654	0.665	0.633	0.666	0.396	0.563
TF-IDF l2 sw	0.661	0.658	0.683	0.672	0.685	**0.704**	0.462	0.688
TF-IDF l2 n-gram	0.658	0.653	0.667	0.661	0.638	0.681	0.395	0.559
TF-IDF l2 n-gram sw	0.667	0.667	0.669	0.681	0.665	**0.703**	0.428	0.671
TF-IDF suav. l2	0.632	0.643	0.652	0.658	0.632	0.671	0.395	0.554
TF-IDF suav. l2 sw	0.663	0.663	0.685	0.682	0.683	**0.706**	0.463	0.687
TF-IDF suav. l2 n-gram	0.658	0.649	0.674	0.664	0.637	0.678	0.397	0.557
TF-IDF suav. l2 n-gram sw	0.670	0.669	0.669	0.680	0.670	**0.704**	0.431	0.667

Tabla 6.2: Precisión del clasificador MultinomialNB

Como se puede comprobar en la tabla 6.2, los mejores resultados se obtienen al aplicar una probabilidad a priori uniforme y al eliminar las *stopwords*.

6.2 Resultados del clasificador de árbol de decisión

La siguiente tabla muestra los resultados de la clase `DecisionTreeClassifier` con distintas funciones para medir la calidad de una división.

	criterion = gini	criterion = entropy
TF l1	0.559	0.535
TF l1 sw	0.591	0.582
TF l1 n-gram	0.582	0.552
TF l1 n-gram sw	0.604	0.590
TF l2	0.572	0.570
TF l2 sw	0.619	0.570
TF l2 n-gram	0.586	0.548
TF l2 n-gram sw	0.610	0.562
TF-IDF l1	0.562	0.543
TF-IDF l1 sw	0.596	0.574
TF-IDF l1 n-gram	0.575	0.547
TF-IDF l1 n-gram sw	0.568	0.559
TF-IDF suav. l1	0.587	0.525
TF-IDF suav. l1 sw	0.598	0.587
TF-IDF suav. l1 n-gram	0.585	0.531
TF-IDF suav. l1 n-gram sw	0.574	0.558
TF-IDF l2	0.568	0.534
TF-IDF l2 sw	0.581	0.569
TF-IDF l2 n-gram	0.585	0.543
TF-IDF l2 n-gram sw	0.580	0.563
TF-IDF suav. l2	0.590	0.529
TF-IDF suav. l2 sw	0.591	0.559
TF-IDF suav. l2 n-gram	0.575	0.520
TF-IDF suav. l2 n-gram sw	0.585	0.575

Tabla 6.3: Precisión del árbol de decisión

Este clasificador no ha obtenido muy buenos resultados, pero tampoco se han obtenido resultados pésimos. El clasificador que ha utilizado Gini como medida de impureza ha conseguido las mejores puntuaciones.

6.3 Resultados del clasificador K-vecinos más cercanos

Este clasificador utiliza la clase `KNeighborsClassifier`. Para este clasificador se prueba con diferentes números de vecinos (*n_neighbors*) y diferentes funciones de peso que se aplican a la predicción (*weights*). Se ha utilizado dos funciones de peso distintas:

- *uniform*: Cada vecino tiene el mismo valor.
- *distance*: Los vecinos más cercanos tendrán mayor influencia que los vecinos más alejados.

A continuación, se muestra una tabla con los resultados para 5, 15 y 25 vecinos:

	n_neighbors = 5		*n_neighbors* = 15		*n_neighbors* = 25	
weights	*uniform*	*distance*	*uniform*	*distance*	*uniform*	*distance*
TF l1	0.470	0.503	0.465	0.487	0.453	0.480
TF l1 sw	0.615	0.630	0.594	0.609	0.553	0.566
TF l1 n-gram	0.465	0.498	0.463	0.487	0.475	0.496
TF l1 n-gram sw	0.587	0.618	0.554	0.580	0.521	0.546
TF l2	0.499	0.537	0.446	0.485	0.451	0.477
TF l2 sw	0.614	0.646	0.575	0.599	0.530	0.551
TF l2 n-gram	0.520	0.546	0.491	0.519	0.458	0.475
TF l2 n-gram sw	0.590	0.619	0.563	0.583	0.510	0.534
TF-IDF l1	0.530	0.562	0.531	0.549	0.512	0.536
TF-IDF l1 sw	0.580	0.605	0.565	0.615	0.591	0.621
TF-IDF l1 n-gram	0.493	0.552	0.470	0.527	0.451	0.479
TF-IDF l1 n-gram sw	0.523	0.592	0.503	0.576	0.541	0.580
TF-IDF suav. l1	0.538	0.576	0.547	0.565	0.519	0.546
TF-IDF suav. l1 sw	0.580	0.613	0.579	0.624	0.600	0.631
TF-IDF suav. l1 n-gram	0.505	0.562	0.502	0.537	0.460	0.493
TF-IDF suav. l1 n-gram sw	0.530	0.591	0.518	0.577	0.551	0.582
TF-IDF l2	0.613	0.619	0.619	0.625	0.597	0.608
TF-IDF l2 sw	0.635	0.655	0.657	0.675	0.622	0.635
TF-IDF l2 n-gram	0.593	0.609	0.619	0.633	0.591	0.611
TF-IDF l2 n-gram sw	0.629	0.650	0.647	0.665	0.622	0.632
TF-IDF suav. l2	0.613	0.625	0.613	0.622	0.588	0.609
TF-IDF suav. l2 sw	0.644	0.661	0.648	0.666	0.624	0.633
TF-IDF suav. l2 n-gram	0.598	0.616	0.619	0.632	0.588	0.605
TF-IDF suav. l2 n-gram sw	0.629	0.652	0.641	0.658	0.624	0.632

Tabla 6.4: Precisión del clasificador k-NN

Como es posible observar en la Tabla 6.4, los mejores resultados se obtienen al aplicarle el idf y la normalización L2 a la matriz de pesos. Cuando se aplica la función de peso *distance* en lugar de *uniform*, la precisión del clasificador mejora ligeramente. También mejoran cuando no se incluyen las *stopwords*.

6.4 Resultados de los clasificadores SVM

La librería *scikit-learn* ofrece diferentes implementaciones de clasificadores basados en SVM. `LinearSVC` y `SVC` son los que se han empleado en estas pruebas. El primero está basado en la librería *liblinear* y emplea una estrategia "uno contra el resto" para la clasificación multiclase, mientras que `SVC`, su implementación está basada en *libsvm* sigue una clasificación "uno contra uno".

La clase `LinearSVC` hace uso del parámetro `C` para ajustar los márgenes blandos. Los resultados de este clasificador son los siguientes:

	$C = 1$	$C = 10$	$C = 100$
TF l1	0.554	0.678	0.687
TF l1 sw	0.629	**0.726**	**0.710**
TF l1 n-gram	0.441	0.641	0.678
TF l1 n-gram sw	0.526	0.678	**0.715**
TF l2	0.689	0.683	0.675
TF l2 sw	**0.736**	**0.724**	**0.720**
TF l2 n-gram	0.680	0.686	0.683
TF l2 n-gram sw	**0.706**	**0.732**	**0.727**
TF-IDF l1	0.600	**0.708**	0.697
TF-IDF l1 sw	0.663	**0.738**	**0.719**
TF-IDF l1 n-gram	0.406	0.682	0.698
TF-IDF l1 n-gram sw	0.526	**0.717**	**0.730**
TF-IDF suav. l1	0.603	**0.706**	0.694
TF-IDF suav. l1 sw	0.663	**0.742**	**0.715**
TF-IDF suav. l1 n-gram	0.417	0.687	0.694
TF-IDF suav. l1 n-gram sw	0.536	**0.717**	**0.730**
TF-IDF l2	**0.709**	**0.703**	0.699
TF-IDF l2 sw	**0.738**	**0.731**	**0.726**
TF-IDF l2 n-gram	0.698	0.691	0.692
TF-IDF l2 n-gram sw	**0.721**	**0.730**	**0.730**
TF-IDF suav. l2	**0.704**	**0.700**	0.698
TF-IDF suav. l2 sw	**0.738**	**0.720**	**0.720**
TF-IDF suav. l2 n-gram	0.695	0.694	0.695
TF-IDF suav. l2 n-gram sw	**0.724**	**0.730**	**0.728**

Tabla 6.5: Precisión del clasificador LinearSVC

Como se puede ver en la Tabla 5, el clasificador `LinearSVC` ofrece buenos resultados en la mayoría de las configuraciones, casi todas por encima de 0.7. Además,

a las matrices que se les ha eliminado las *stopwords* han obtenido mejores resultados que las matrices que las mantienen.

De la misma manera que el clasificador anterior, la clase `SVC` también utiliza el parámetro `C` para ajustar los márgenes blandos. Los resultados según el tipo de *kernel* son los siguientes:

	kernel = *rbf*		
	gamma = 0.1		*gamma* = 0.01
	C = 10	*C* = 100	*C* = 100
TF l1	0.342	0.638	0.346
TF l1 sw	0.453	0.725	0.454
TF l1 n-gram	0.325	0.591	0.325
TF l1 n-gram sw	0.331	0.680	0.331
TF l2	0.669	0.669	0.671
TF l2 sw	0.724	0.716	0.726
TF l2 n-gram	0.658	0.661	0.660
TF l2 n-gram sw	0.699	0.704	0.702
TF-IDF l1	0.329	0.676	0.329
TF-IDF l1 sw	0.380	0.706	0.386
TF-IDF l1 n-gram	0.325	0.614	0.325
TF-IDF l1 n-gram sw	0.326	0.695	0.326
TF-IDF suav. l1	0.329	0.678	0.329
TF-IDF suav. l1 sw	0.390	0.705	0.396
TF-IDF suav. l1 n-gram	0.325	0.619	0.325
TF-IDF suav. l1 n-gram sw	0.326	0.698	0.326
TF-IDF l2	0.685	0.685	0.687
TF-IDF l2 sw	0.711	0.699	0.711
TF-IDF l2 n-gram	0.674	0.675	0.676
TF-IDF l2 n-gram sw	0.708	0.706	0.709
TF-IDF suav. l2	0.683	0.677	0.687
TF-IDF suav. l2 sw	0.708	0.703	0.709
TF-IDF suav. l2 n-gram	0.672	0.672	0.677
TF-IDF suav. l2 n-gram sw	0.713	0.711	0.713

Tabla 6.6: Precisión del clasificador SVC con kernel rbf

La tabla anterior contiene los resultados al aplicarle el *kernel* rbf con diferentes valores para `C` y `gamma`. Como se puede comprobar, se obtienen mejores resultados cuando se eliminan las *stopwords* y se aplica una normalización L2, excepto *C*=100 con *gamma*=0.1 que las matrices con normalización L1 también se consigue una buena precisión.

	kernel = *linear*		
	C = 1	*C* = 10	*C* = 100
TF l1	0.325	0.592	0.654
TF l1 sw	0.356	0.686	0.711
TF l1 n-gram	0.325	0.480	0.638
TF l1 n-gram sw	0.325	0.582	0.697

TF l2	0.629	0.660	0.658
TF l2 sw	0.683	0.720	0.715
TF l2 n-gram	0.624	0.667	0.669
TF l2 n-gram sw	0.659	0.711	0.711
TF-IDF l1	0.325	0.652	0.675
TF-IDF l1 sw	0.326	0.699	0.703
TF-IDF l1 n-gram	0.325	0.421	0.681
TF-IDF l1 n-gram sw	0.325	0.600	0.711
TF-IDF suav. l1	0.325	0.652	0.670
TF-IDF suav. l1 sw	0.326	0.704	0.703
TF-IDF suav. l1 n-gram	0.325	0.436	0.682
TF-IDF suav. l1 n-gram sw	0.325	0.608	0.716
TF-IDF l2	0.669	0.678	0.678
TF-IDF l2 sw	0.695	0.698	0.698
TF-IDF l2 n-gram	0.647	0.676	0.676
TF-IDF l2 n-gram sw	0.687	0.709	0.709
TF-IDF suav. l2	0.667	0.678	0.678
TF-IDF suav. l2 sw	0.695	0.694	0.694
TF-IDF suav. l2 n-gram	0.653	0.674	0.674
TF-IDF suav. l2 n-gram sw	0.689	0.710	0.710

Tabla 6.7: Precisión del clasificador SVC con kernel lineal

La tabla 6.7 muestra la precisión del clasificador con *kernel* lineal. Al igual que los otros clasificadores, este también mejora la precisión cuando no se incluyen en la matriz de pesos las *stopwords*.

La siguiente tabla contiene los resultados para el *kernel* polinómico.

	kernel = *poly*			
	degree = 1			*degree* = 2
	gamma = 0.1		*gamma* = 0.01	*gamma* = 0.1
	C = 10	*C* = 100	*C* = 100	*C* = 100
TF l1	0.325	0.592	0.325	0.325
TF l1 sw	0.356	0.686	0.356	0.325
TF l1 n-gram	0.325	0.480	0.325	0.325
TF l1 n-gram sw	0.325	0.582	0.325	0.325
TF l2	0.629	0.660	0.629	0.592
TF l2 sw	0.683	0.720	0.683	0.643
TF l2 n-gram	0.624	0.667	0.624	0.568
TF l2 n-gram sw	0.659	0.711	0.659	0.580
TF-IDF l1	0.325	0.652	0.325	0.325
TF-IDF l1 sw	0.326	0.699	0.326	0.325
TF-IDF l1 n-gram	0.325	0.421	0.325	0.325
TF-IDF l1 n-gram sw	0.325	0.600	0.325	0.325
TF-IDF suav. l1	0.325	0.652	0.325	0.325
TF-IDF suav. l1 sw	0.326	0.704	0.326	0.325

TF-IDF suav. l1 n-gram	0.325	0.436	0.325	0.325
TF-IDF suav. l1 n-gram sw	0.325	0.608	0.325	0.325
TF-IDF l2	0.669	0.678	0.669	0.457
TF-IDF l2 sw	0.695	0.698	0.695	0.457
TF-IDF l2 n-gram	0.647	0.676	0.647	0.414
TF-IDF l2 n-gram sw	0.687	0.709	0.687	0.414
TF-IDF suav. l2	0.667	0.678	0.667	0.475
TF-IDF suav. l2 sw	0.695	0.694	0.695	0.474
TF-IDF suav. l2 n-gram	0.653	0.674	0.653	0.417
TF-IDF suav. l2 n-gram sw	0.689	0.710	0.689	0.419

Tabla 6.8: Precisión del clsificador SVC con kernel polinómico

Con los datos de la tabla anterior, la mayor precisión se obtiene cuando los parámetros del clasificador tienen valor `degree`=1, `gamma`=0.1 y `C`=100. Los mejores resultados se han obtenido al aplicarle a la matriz de peso una normalización L2. Los valores empiezan a empeorar al aumentar el grado (`degree`).

A continuación, se muestran los resultados obtenidos al utilizar un *kernel* sigmoid.

	kernel = sigmoid		
	gamma = 0.1		*gamma* = 0.01
	C = 10	*C* = 100	*C* = 100
TF l1	0.325	0.592	0.325
TF l1 sw	0.356	0.686	0.356
TF l1 n-gram	0.325	0.480	0.325
TF l1 n-gram sw	0.325	0.582	0.325
TF l2	0.627	0.659	0.627
TF l2 sw	0.683	0.720	0.683
TF l2 n-gram	0.624	0.667	0.624
TF l2 n-gram sw	0.658	0.711	0.659
TF-IDF l1	0.325	0.652	0.325
TF-IDF l1 sw	0.326	0.699	0.326
TF-IDF l1 n-gram	0.325	0.421	0.325
TF-IDF l1 n-gram sw	0.325	0.600	0.325
TF-IDF suav. l1	0.325	0.652	0.325
TF-IDF suav. l1 sw	0.326	0.704	0.326
TF-IDF suav. l1 n-gram	0.325	0.436	0.325
TF-IDF suav. l1 n-gram sw	0.325	0.608	0.325
TF-IDF l2	0.669	0.677	0.669
TF-IDF l2 sw	0.695	0.698	0.695
TF-IDF l2 n-gram	0.647	0.675	0.647
TF-IDF l2 n-gram sw	0.687	0.708	0.687
TF-IDF suav. l2	0.667	0.678	0.667
TF-IDF suav. l2 sw	0.695	0.694	0.695
TF-IDF suav. l2 n-gram	0.653	0.674	0.653

TF-IDF suav. l2 n-gram sw	0.689	0.709	0.689

Tabla 6.9: Precisión del clasificador SVC con kernel sigmoid

Este clasificador ofrece unos resultados muy similares al clasificador SVC con *kernel* polinómico con grado 1.

7. Conclusiones

Se ha logrado implementar un sistema de clasificación de tuits utilizando algoritmos de aprendizaje además de la aplicación web donde se realizan las consultas y se muestran los tuits filtrados.

A partir de los resultados de las experimentaciones realizadas en el capítulo anterior, se puede comprobar que eliminar las *stopwords* de las matrices de peso aumenta la precisión de los clasificadores. Adicionalmente, en la mayoría de los casos, utilizar la norma L2 también ayuda a mejorar la precisión de los clasificadores.

En cuanto a los clasificadores, los arboles de decisión y el algoritmo de los k-vecinos más cercanos no han obtenido buenos resultados en la clasificación de tuits, en cambio, los clasificadores de Naive Bayes y las SVM han conseguido una precisión por encima del 70%. De entre ellos, destaca el clasificador LinearSVC que ha conseguido la mejor precisión, un 74.2%, cuando se han ajustado los márgenes blandos (C=10) además de aplicar a la matriz de pesos una transformación TF-IDF suavizado, la norma L1 y eliminando las *stopwords*.

Por otro lado, crear un corpus de entrenamiento ha resultado ser una tarea muy laboriosa, ya que, a la hora de etiquetar manualmente los tuits, se han encontrado muchos que no se pueden clasificar por su escasez de contenido, o eran respuestas a otros tuits y no se podía deducir a que clase pertenecía, o que el contenido importante del tuit fuera una imagen o video. Incluso tuits que se podrían clasificar en dos o más categorías diferentes.

7.1 Trabajo futuro

Dado que existe una gran cantidad de tuits que no se pueden clasificar por los motivos que se han mencionado en el apartado anterior. Se pueden plantear diferentes mejoras en el sistema de clasificación.

- Utilizar un algoritmo de aprendizaje no supervisado para detectar si un nuevo tuit no pertenece a ninguna clase, como por ejemplo *one-class* SVM. Esto haría que algunos tuits no se clasificaran en ninguna clase en lugar de una errónea, por falta de datos de entrenamiento o que todavía no haya ningún dato en la categoría a la que pertenezca.

- Al clasificar los tuits que sean respuestas a otros y no se pueda deducir a que clase pertenece, se podría clasificar en la misma clase que el tuit contestado y mostrar la conversación clasificada en la página de resultados.

- Utilizar algoritmos de clasificación multi-etiqueta en lugar de los que se han empleado para poder clasificar los tuits en una o más categorías.

Además de ampliar la colección de entrenamiento para que se puedan predecir los tuits con mayor precisión.

A. Apéndice

Tabla completa de resultados: Precisión del clasificador BernoulliNB.

	alpha = 0.001		*alpha* = 0.01		*alpha* = 0.1		*alpha* = 1	
fit_prior	True	False	True	False	True	False	True	False
TF l1	0.66748	0.67966	0.68575	0.68453	0.68575	0.67844	0.37028	0.37393
TF l1 sw	0.69915	0.69062	**0.71133**	**0.70524**	**0.70158**	**0.71011**	0.37272	0.37515
TF l1 n-gram	0.67113	0.67235	0.67722	0.6821	0.63459	0.64434	0.34592	0.34592
TF l1 n-gram sw	0.68453	0.67844	0.6894	0.69306	0.64799	0.66017	0.33861	0.33861
TF l2	0.66748	0.67966	0.68575	0.68453	0.68575	0.67844	0.37028	0.37393
TF l2 sw	0.69915	0.69062	**0.71133**	**0.70524**	**0.70158**	**0.71011**	0.37272	0.37515
TF l2 n-gram	0.67113	0.67235	0.67722	0.6821	0.63459	0.64434	0.34592	0.34592
TF l2 n-gram sw	0.68453	0.67844	0.6894	0.69306	0.64799	0.66017	0.33861	0.33861
TF-IDF l1	0.66748	0.67966	0.68575	0.68453	0.68575	0.67844	0.37028	0.37393
TF-IDF l1 sw	0.69915	0.69062	**0.71133**	**0.70524**	**0.70158**	**0.71011**	0.37272	0.37515
TF-IDF l1 n-gram	0.67113	0.67235	0.67722	0.6821	0.63459	0.64434	0.34592	0.34592
TF-IDF l1 n-gram sw	0.68453	0.67844	0.6894	0.69306	0.64799	0.66017	0.33861	0.33861
TF-IDF suav. l1	0.66748	0.67966	0.68575	0.68453	0.68575	0.67844	0.37028	0.37393
TF-IDF suav. l1 sw	0.69915	0.69062	**0.71133**	**0.70524**	**0.70158**	**0.71011**	0.37272	0.37515
TF-IDF suav. l1 n-gram	0.67113	0.67235	0.67722	0.6821	0.63459	0.64434	0.34592	0.34592
TF-IDF suav. l1 n-gram sw	0.68453	0.67844	0.6894	0.69306	0.64799	0.66017	0.33861	0.33861
TF-IDF l2	0.66748	0.67966	0.68575	0.68453	0.68575	0.67844	0.37028	0.37393
TF-IDF l2 sw	0.69915	0.69062	**0.71133**	**0.70524**	**0.70158**	**0.71011**	0.37272	0.37515
TF-IDF l2 n-gram	0.67113	0.67235	0.67722	0.6821	0.63459	0.64434	0.34592	0.34592
TF-IDF l2 n-gram sw	0.68453	0.67844	0.6894	0.69306	0.64799	0.66017	0.33861	0.33861
TF-IDF suav. l2	0.66748	0.67966	0.68575	0.68453	0.68575	0.67844	0.37028	0.37393
TF-IDF suav. l2 sw	0.69915	0.69062	**0.71133**	**0.70524**	**0.70158**	**0.71011**	0.37272	0.37515
TF-IDF suav. l2 n-gram	0.67113	0.67235	0.67722	0.6821	0.63459	0.64434	0.34592	0.34592
TF-IDF suav. l2 n-gram sw	0.68453	0.67844	0.6894	0.69306	0.64799	0.66017	0.33861	0.33861

Tabla completa de resultados: Precisión del clasificador MultinomialNB.

	alpha = 0.001		*alpha* = 0.01		*alpha* = 0.1		*alpha* = 1	
fit_prior	True	False	True	False	True	False	True	False
TF l1	0.57125	0.66991	0.46285	0.6687	0.3313	0.51644	0.32521	0.33739
TF l1 sw	0.65164	0.69671	0.60658	**0.70646**	0.3922	0.68331	0.32521	0.42996
TF l1 n-gram	0.58465	0.68697	0.42266	0.6687	0.32521	0.45676	0.32521	0.33252
TF l1 n-gram sw	0.6553	0.69306	0.5542	**0.71376**	0.34836	0.61998	0.32521	0.4056
TF l2	0.65408	0.65895	0.65286	0.66626	0.55786	0.67235	0.35323	0.41535
TF l2 sw	0.68697	0.68453	0.69306	0.68819	0.66504	**0.71255**	0.39099	0.58952

TF l2 n-gram	0.6687	0.67357	0.6687	0.68575	0.54568	0.63337	0.35079	0.40195
TF l2 n-gram sw	0.68331	0.68697	0.68575	0.69184	0.6285	0.68331	0.38368	0.54446
TF-IDF l1	0.61267	0.65408	0.57125	0.6687	0.37028	0.65408	0.32521	0.45798
TF-IDF l1 sw	0.65164	0.66748	0.64555	0.69671	0.43362	**0.72107**	0.32521	0.63581
TF-IDF l1 n-gram	0.62728	0.66017	0.52375	0.6821	0.32887	0.5944	0.32521	0.44945
TF-IDF l1 n-gram sw	0.65164	0.67966	0.61023	0.69915	0.3715	0.69549	0.32521	0.59805
TF-IDF suav. l1	0.61145	0.65652	0.56638	0.6687	0.36906	0.6553	0.32521	0.45432
TF-IDF suav. l1 sw	0.65652	0.67235	0.64677	0.69428	0.43118	**0.72351**	0.32521	0.62972
TF-IDF suav. l1 n-gram	0.62363	0.66261	0.51766	0.68453	0.32887	0.58952	0.32521	0.44214
TF-IDF suav. l1 n-gram sw	0.64921	0.67966	0.61023	**0.70402**	0.3715	0.68819	0.32521	0.59196
TF-IDF l2	0.63459	0.63825	0.65408	0.66504	0.63337	0.66626	0.39586	0.56273
TF-IDF l2 sw	0.66139	0.65773	0.68331	0.67235	0.68453	**0.70402**	0.46163	0.68819
TF-IDF l2 n-gram	0.65773	0.65286	0.66748	0.66139	0.63825	0.68088	0.39464	0.55907
TF-IDF l2 n-gram sw	0.66748	0.66748	0.6687	0.68088	0.66504	**0.7028**	0.42753	0.67113
TF-IDF suav. l2	0.63216	0.64312	0.65164	0.65773	0.63216	0.67113	0.39464	0.5542
TF-IDF suav. l2 sw	0.66261	0.66261	0.68453	0.6821	0.68331	**0.70646**	0.46285	0.68697
TF-IDF suav. l2 n-gram	0.65773	0.64921	0.67357	0.66382	0.63703	0.67844	0.39708	0.55664
TF-IDF suav. l2 n-gram sw	0.66991	0.6687	0.6687	0.67966	0.66991	**0.70402**	0.43118	0.66748

Tabla completa de resultados: Precisión de KNeighborsClassifier.

	n_neighbors =5		*n_neighbors* = 15		*n_neighbors* = 25	
weights	*uniform*	*distance*	*uniform*	*distance*	*uniform*	*distance*
TF l1	0.47016	0.50305	0.46529	0.48721	0.45311	0.4799
TF l1 sw	0.6151	0.62972	0.5944	0.60901	0.55298	0.56638
TF l1 n-gram	0.46529	0.49817	0.46285	0.48721	0.47503	0.49574
TF l1 n-gram sw	0.58709	0.61754	0.5542	0.57978	0.52132	0.54568
TF l2	0.49939	0.53715	0.4458	0.48477	0.45067	0.47747
TF l2 sw	0.61389	0.64555	0.57491	0.59927	0.52984	0.55055
TF l2 n-gram	0.5201	0.54568	0.49086	0.51888	0.45798	0.47503
TF l2 n-gram sw	0.58952	0.61876	0.56273	0.58343	0.51035	0.5335
TF-IDF l1	0.52984	0.56151	0.53106	0.54933	0.51157	0.53593
TF-IDF l1 sw	0.57978	0.60536	0.56516	0.6151	0.59074	0.62119
TF-IDF l1 n-gram	0.4933	0.55177	0.47016	0.52741	0.45067	0.47868
TF-IDF l1 n-gram sw	0.52253	0.59196	0.50305	0.57613	0.5408	0.57978
TF-IDF suav. l1	0.53837	0.57613	0.54689	0.56516	0.51888	0.54568
TF-IDF suav. l1 sw	0.57978	0.61267	0.57856	0.62363	0.60049	0.63094
TF-IDF suav. l1 n-gram	0.50548	0.56151	0.50183	0.53715	0.46041	0.4933
TF-IDF suav. l1 n-gram sw	0.52984	0.59074	0.51766	0.57734	0.55055	0.58222
TF-IDF l2	0.61267	0.61876	0.61876	0.62485	0.59683	0.6078
TF-IDF l2 sw	0.63459	0.6553	0.65652	0.67479	0.62241	0.63459
TF-IDF l2 n-gram	0.59318	0.60901	0.61876	0.63337	0.59074	0.61145
TF-IDF l2 n-gram sw	0.6285	0.65043	0.64677	0.66504	0.62241	0.63216

TF-IDF suav. l2	0.61267	0.62485	0.61267	0.62241	0.58831	0.60901
TF-IDF suav. l2 sw	0.64434	0.66139	0.64799	0.66626	0.62363	0.63337
TF-IDF suav. l2 n-gram	0.59805	0.61632	0.61876	0.63216	0.58831	0.60536
TF-IDF suav. l2 n-gram sw	0.6285	0.65164	0.64068	0.65773	0.62363	0.63216

Tabla completa de resultados: Precisión de DecisionTreeClassifier.

	criterion = gini	*criterion = entropy*
TF l1	0.58465	0.53471
TF l1 sw	0.60292	0.58222
TF l1 n-gram	0.57247	0.55177
TF l1 n-gram sw	0.6078	0.58952
TF l2	0.55177	0.57004
TF l2 sw	0.61632	0.57004
TF l2 n-gram	0.57613	0.54811
TF l2 n-gram sw	0.61023	0.56151
TF-IDF l1	0.58465	0.54324
TF-IDF l1 sw	0.58952	0.57369
TF-IDF l1 n-gram	0.58952	0.54689
TF-IDF l1 n-gram sw	0.58343	0.55907
TF-IDF suav. l1	0.59074	0.52497
TF-IDF suav. l1 sw	0.61023	0.58709
TF-IDF suav. l1 n-gram	0.59683	0.53106
TF-IDF suav. l1 n-gram sw	0.58831	0.55786
TF-IDF l2	0.57369	0.5335
TF-IDF l2 sw	0.58831	0.56882
TF-IDF l2 n-gram	0.58222	0.54324
TF-IDF l2 n-gram sw	0.58222	0.56273
TF-IDF suav. l2	0.58587	0.52862
TF-IDF suav. l2 sw	0.60292	0.55907
TF-IDF suav. l2 n-gram	0.57491	0.5201
TF-IDF suav. l2 n-gram sw	0.57978	0.57491

Tabla completa de resultados: Precisión de LinearSVC.

	C = 1	*C* = 10	*C* = 100
TF l1	0.5542	0.67844	0.68697
TF l1 sw	0.6285	**0.72594**	**0.71011**
TF l1 n-gram	0.44093	0.64068	0.67844
TF l1 n-gram sw	0.52619	0.67844	**0.71498**
TF l2	0.6894	0.68331	0.67479
TF l2 sw	**0.73569**	**0.72351**	**0.71985**
TF l2 n-gram	0.67966	0.68575	0.68331
TF l2 n-gram sw	**0.70646**	**0.73203**	**0.72716**
TF-IDF l1	0.60049	**0.70767**	0.69671
TF-IDF l1 sw	0.66261	**0.73812**	**0.71864**
TF-IDF l1 n-gram	0.4056	0.6821	0.69793
TF-IDF l1 n-gram sw	0.52619	**0.71742**	**0.7296**
TF-IDF suav. l1	0.60292	**0.70646**	0.69428
TF-IDF suav. l1 sw	0.66261	**0.74178**	**0.71498**
TF-IDF suav. l1 n-gram	0.41657	0.68697	0.69428
TF-IDF suav. l1 n-gram sw	0.53593	**0.71742**	**0.7296**
TF-IDF l2	**0.70889**	**0.7028**	0.69915
TF-IDF l2 sw	**0.73812**	**0.73082**	**0.72594**
TF-IDF l2 n-gram	0.69793	0.69062	0.69184
TF-IDF l2 n-gram sw	**0.72107**	**0.7296**	**0.7296**
TF-IDF suav. l2	**0.70402**	**0.70037**	0.69793
TF-IDF suav. l2 sw	**0.73812**	**0.71985**	**0.71985**
TF-IDF suav. l2 n-gram	0.69549	0.69428	0.69549
TF-IDF suav. l2 n-gram sw	**0.72351**	**0.7296**	**0.72838**

Tabla completa de resultados: Precisión de SVC con *kernel* lineal.

	kernel = linear		
	C = 1	*C* = 10	*C* = 100
TF l1	0.32521	0.59196	0.65408
TF l1 sw	0.35566	0.68575	0.71133
TF l1 n-gram	0.32521	0.4799	0.63825
TF l1 n-gram sw	0.32521	0.58222	0.69671
TF l2	0.6285	0.66017	0.65773
TF l2 sw	0.68331	0.71985	0.71498
TF l2 n-gram	0.62363	0.66748	0.6687
TF l2 n-gram sw	0.65895	0.71133	0.71133
TF-IDF l1	0.32521	0.65164	0.67479
TF-IDF l1 sw	0.32643	0.69915	0.7028
TF-IDF l1 n-gram	0.32521	0.42144	0.68088

TF-IDF l1 n-gram sw	0.32521	0.60049	0.71133
TF-IDF suav. l1	0.32521	0.65164	0.66991
TF-IDF suav. l1 sw	0.32643	0.70402	0.7028
TF-IDF suav. l1 n-gram	0.32521	0.43605	0.6821
TF-IDF suav. l1 n-gram sw	0.32521	0.6078	0.7162
TF-IDF l2	0.6687	0.67844	0.67844
TF-IDF l2 sw	0.69549	0.69793	0.69793
TF-IDF l2 n-gram	0.64677	0.676	0.676
TF-IDF l2 n-gram sw	0.68697	0.70889	0.70889
TF-IDF suav. l2	0.66748	0.67844	0.67844
TF-IDF suav. l2 sw	0.69549	0.69428	0.69428
TF-IDF suav. l2 n-gram	0.65286	0.67357	0.67357
TF-IDF suav. l2 n-gram sw	0.6894	0.71011	0.71011

Tabla completa de resultados: Precisión de SVC con *kernel rbf*.

	kernel = *sigmoid*											
	gamma = 0.1			*gamma* = 0.01			*gamma* = 0.001			*gamma* = 0.0001		
	C = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100
TF l1	0.32521	0.34227	0.63825	0.32521	0.32521	0.34592	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 sw	0.32521	0.45311	0.72473	0.32521	0.32521	0.45432	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram	0.32521	0.32521	0.59074	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram sw	0.32521	0.3313	0.67966	0.32521	0.32521	0.3313	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2	0.38733	0.6687	0.6687	0.32521	0.40804	0.67113	0.32521	0.32521	0.40804	0.32521	0.32521	0.32521
TF l2 sw	0.42387	0.72351	0.7162	0.32521	0.43727	0.72594	0.32521	0.32521	0.43971	0.32521	0.32521	0.32521
TF l2 n-gram	0.33252	0.65773	0.66139	0.32521	0.34714	0.66017	0.32521	0.32521	0.34836	0.32521	0.32521	0.32521
TF l2 n-gram sw	0.36906	0.69915	0.70402	0.32521	0.39099	0.70158	0.32521	0.32521	0.3922	0.32521	0.32521	0.32521
TF-IDF l1	0.32521	0.32887	0.676	0.32521	0.32521	0.32887	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 sw	0.32521	0.38002	0.70646	0.32521	0.32521	0.38611	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram	0.32521	0.32521	0.61389	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram sw	0.32521	0.32643	0.69549	0.32521	0.32521	0.32643	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1	0.32521	0.32887	0.67844	0.32521	0.32521	0.32887	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 sw	0.32521	0.38977	0.70524	0.32521	0.32521	0.39586	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram	0.32521	0.32521	0.61876	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram sw	0.32521	0.32643	0.69793	0.32521	0.32521	0.32643	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2	0.32643	0.68453	0.68453	0.32521	0.33009	0.68697	0.32521	0.32521	0.3313	0.32521	0.32521	0.32521
TF-IDF l2 sw	0.33009	0.71133	0.69915	0.32521	0.33861	0.71133	0.32521	0.32521	0.34227	0.32521	0.32521	0.32521
TF-IDF l2 n-gram	0.32643	0.67357	0.67479	0.32521	0.33009	0.676	0.32521	0.32521	0.33009	0.32521	0.32521	0.32521
TF-IDF l2 n-gram sw	0.32643	0.70767	0.70646	0.32521	0.32765	0.70889	0.32521	0.32521	0.32765	0.32521	0.32521	0.32521
TF-IDF suav. l2	0.32643	0.68331	0.67722	0.32521	0.33252	0.68697	0.32521	0.32521	0.33252	0.32521	0.32521	0.32521
TF-IDF suav. l2 sw	0.33009	0.70767	0.7028	0.32521	0.3447	0.70889	0.32521	0.32521	0.34592	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram	0.32643	0.67235	0.67235	0.32521	0.3313	0.67722	0.32521	0.32521	0.3313	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram sw	0.32643	0.71255	0.71133	0.32521	0.32765	0.71255	0.32521	0.32521	0.32765	0.32521	0.32521	0.32521

Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 1.

	kernel = *poly*											
	degree = 1											
	gamma = 0.1			*gamma* = 0.01			*gamma* = 0.001			*gamma* = 0.0001		
	C = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 10	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100
TF l1	0.32521	0.32521	0.59196	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 sw	0.32521	0.35566	0.68575	0.32521	0.32521	0.35566	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram	0.32521	0.32521	0.4799	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram sw	0.32521	0.32521	0.58222	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2	0.32521	0.6285	0.66017	0.32521	0.32521	0.6285	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 sw	0.34957	0.68331	0.71985	0.32521	0.34957	0.68331	0.32521	0.32521	0.34957	0.32521	0.32521	0.32521
TF l2 n-gram	0.32521	0.62363	0.66748	0.32521	0.32521	0.62363	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram sw	0.32521	0.65895	0.71133	0.32521	0.32521	0.65895	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1	0.32521	0.32521	0.65164	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 sw	0.32521	0.32643	0.69915	0.32521	0.32521	0.32643	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram	0.32521	0.32521	0.42144	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram sw	0.32521	0.32521	0.60049	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1	0.32521	0.32521	0.65164	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 sw	0.32521	0.32643	0.70402	0.32521	0.32521	0.32643	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram	0.32521	0.32521	0.43605	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram sw	0.32521	0.32521	0.6078	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2	0.32521	0.6687	0.67844	0.32521	0.32521	0.6687	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 sw	0.32521	0.69549	0.69793	0.32521	0.32521	0.69549	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram	0.32521	0.64677	0.676	0.32521	0.32521	0.64677	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram sw	0.32521	0.68697	0.70889	0.32521	0.32521	0.68697	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2	0.32521	0.66748	0.67844	0.32521	0.32521	0.66748	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 sw	0.32521	0.69549	0.69428	0.32521	0.32521	0.69549	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram	0.32521	0.65286	0.67357	0.32521	0.32521	0.65286	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram sw	0.32521	0.6894	0.71011	0.32521	0.32521	0.6894	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521

Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 2.

	kernel = *poly*											
	degree = 2											
	gamma = 0.1			*gamma* = 0.01			*gamma* = 0.001			*gamma* = 0.0001		
	C = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100
TF l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521

TF l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2	0.32521	0.32521	0.59196	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 sw	0.32521	0.32521	0.64312	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram	0.32521	0.32521	0.5676	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram sw	0.32521	0.32521	0.57978	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2	0.32521	0.32521	0.45676	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 sw	0.32521	0.32521	0.45676	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram	0.32521	0.32521	0.41413	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram sw	0.32521	0.32521	0.41413	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2	0.32521	0.32521	0.47503	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 sw	0.32521	0.32521	0.47381	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram	0.32521	0.32521	0.41657	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram sw	0.32521	0.32521	0.419	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521

Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 3.

	kernel = *poly*											
	degree = 3											
	gamma = 0.1			*gamma* = 0.01			*gamma* = 0.001			*gamma* = 0.0001		
	C = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100
TF l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521

TF-IDF suav. l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521

Tabla completa de resultados: Precisión de SVC con *kernel* polinómico grado 4.

	kernel = *poly*											
	degree = 4											
	gamma = 0.1			*gamma* = 0.01			*gamma* = 0.001			*gamma* = 0.0001		
	C = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100
TF l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram sw	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521

Tabla completa de resultados: Precisión de SVC con *kernel* sigmoid.

	kernel = *sigmoid*											
	gamma = 0.1			*gamma* = 0.01			*gamma* = 0.001			*gamma* = 0.0001		
	C = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100	*C* = 1	*C* = 10	*C* = 100
TF l1	0.32521	0.32521	0.59196	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 sw	0.32521	0.35566	0.68575	0.32521	0.32521	0.35566	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram	0.32521	0.32521	0.4799	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l1 n-gram sw	0.32521	0.32521	0.58222	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2	0.32521	0.62728	0.65895	0.32521	0.32521	0.62728	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 sw	0.34957	0.68331	0.71985	0.32521	0.34957	0.68331	0.32521	0.32521	0.34957	0.32521	0.32521	0.32521
TF l2 n-gram	0.32521	0.62363	0.66748	0.32521	0.32521	0.62363	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF l2 n-gram sw	0.32521	0.65773	0.71133	0.32521	0.32521	0.65895	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1	0.32521	0.32521	0.65164	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 sw	0.32521	0.32643	0.69915	0.32521	0.32521	0.32643	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram	0.32521	0.32521	0.42144	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l1 n-gram sw	0.32521	0.32521	0.60049	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1	0.32521	0.32521	0.65164	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 sw	0.32521	0.32643	0.70402	0.32521	0.32521	0.32643	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram	0.32521	0.32521	0.43605	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l1 n-gram sw	0.32521	0.32521	0.6078	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2	0.32521	0.6687	0.67722	0.32521	0.32521	0.6687	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 sw	0.32521	0.69549	0.69793	0.32521	0.32521	0.69549	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram	0.32521	0.64677	0.67479	0.32521	0.32521	0.64677	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF l2 n-gram sw	0.32521	0.68697	0.70767	0.32521	0.32521	0.68697	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2	0.32521	0.66748	0.67844	0.32521	0.32521	0.66748	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 sw	0.32521	0.69549	0.69428	0.32521	0.32521	0.69549	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram	0.32521	0.65286	0.67357	0.32521	0.32521	0.65286	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521
TF-IDF suav. l2 n-gram sw	0.32521	0.6894	0.70889	0.32521	0.32521	0.6894	0.32521	0.32521	0.32521	0.32521	0.32521	0.32521

Bibliografía

[1] Twitter, Inc (2016). "*About*."
<https://about.twitter.com/es/company>

[2] Twitter, Inc (2016). "*REST APIs*"
<https://dev.twitter.com/rest/public>

[3] Twitter, Inc (2016). "*Streaming APIs*"
<https://dev.twitter.com/streaming/overview>

[4] PyMongo. "*PyMongo 3.4.0 Documentation*"
<https://api.mongodb.com/python/current/>

[5] Steven Bird, Ewan Klein, y Edward Loper (2009) "*Natural Language Processing with Python*"
<http://www.nltk.org/book_1ed/ch00.html>

[6] MongoDB, Inc. "*What is MongoDB?*"
<https://www.mongodb.com/what-is-mongodb>

[7] Armin Ronacher (2017) "*User's Guide*"
<http://flask.pocoo.org/docs/0.12/foreword/#what-does-micro-mean>

[8] World Wide Web Consortium (W3C). "*HTML & CSS*"
<https://www.w3.org/standards/webdesign/htmlcss>

[9] World Wide Web Consortium (W3C) (2014). "*HTML5*"
<https://www.w3.org/TR/2014/REC-html5-20141028/introduction.html>
[Consulta:]

[10] D. Cournapeau, M. Brucher, J. Millman y a. et, (2010) "*Support Vector Machines*"
<http://scikit-learn.org/stable/modules/svm.html>

[11] Amarappay Dr. S V Sathyanarayana (2012) "*Data classification using Support vector Machine (SVM), a simplified approach*"
<http://www.ijecse.org/wp-content/uploads/2012/06/Volume-3Number-4PP-435-445x.pdf>

[12] Jason Brownlee (2016) "*Support Vector Machines for Machine Learning*"
<https://machinelearningmastery.com/support-vector-machines-for-machine-learning/>

[13] Alex Guazzelli (2012) "*Técnicas de modelado predictivo*"
< https://www.ibm.com/developerworks/ssa/industry/library/ba-predictive-analytics2/index.html>

[14] D. Cournapeau, M. Brucher, J. Millman y a. et, (2010) "*Decision Trees*"
<http://scikit-learn.org/stable/modules/tree.html>

[15] D. Cournapeau, M. Brucher, J. Millman y a. et, (2010) "*Nearest Neighbors*"
<http://scikit-learn.org/stable/modules/neighbors.html>

[16] V. Metsis, I. Androutsopoulos y G. Paliouras (2006). "*Spam filtering with Naive Bayes – Which Naive Bayes?*"
<http://www2.aueb.gr/users/ion/docs/ceas2006_paper.pdf>

[17] D. Cournapeau, M. Brucher, J. Millman y a. et, (2010) "*Naive Bayes*"
<http://scikit-learn.org/stable/modules/naive_bayes.html>[Consulta:]

[18] Manning, Christopher D, Prabhakar Raghavan, y Hinrich Schütze (2008) "*Introduction to Information Retrieval*"
<https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf>

[19] D. Cournapeau, M. Brucher, J. Millman y a. et, (2010) "*Feature extraction*"
<http://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction>

[20] Mark Cowlishaw, Nathanael Fillmore (2010) "*Introduction to numerical analysis*"
<http://pages.cs.wisc.edu/~amos/412/lecture-notes/lecture14.pdf>

[21] Redacción TICbeat (2011) "Cuánta información se genera al día"
<http://www.ticbeat.com/tecnologias/cuanta-informacion-genera-dia/>