Facultad de Ciencias Sociales y Jurídicas de Elche

Trabajo Fin de Grado

Procesamiento del lenguaje natural: Desarrollo de aplicaciones para Inteligencia Competitiva

Alumno: José Manuel Ortuño Lorente

Tutor: José Luis Sainz-Pardo Auñón

4º ESTADÍSTICA EMPRESARIAL

ÍNDICE

Resumen

La combinación entre el crecimiento que ha experimentado la red durante los últimos años y los avances en la capacidad de computación de los ordenadores han desembocado en una nueva revolución de la información. Cada día en la red se generan volúmenes enormes de datos de toda clase. La mayor parte de ellos son desaprovechados y almacenados en grandes servidores en espera de la aparición de una forma de aprovecharlos y explotarlos. Dentro de estos datos, encontramos que muchos de ellos son textos con información que podría ser valiosa para multitud de usos, esa es la razón por la cual cada día surgen nuevos métodos para analizar y procesarlos.

Al análisis y procesamiento de texto se le conoce como análisis de texto. Una de las tareas que se desarrollan en el análisis de texto es la organización y clasificación de textos.

Existen diversas herramientas y técnicas para abordar esta cuestión como las redes neuronales o los árboles de decisión, entre otros, pero en el presente trabajo nos centraremos en una herramienta conocida como clasificador Naïve Bayes. Esta herramienta destaca por su sencillez y los buenos resultados frente a otras técnicas como las mencionadas anteriormente.

El objetivo de este trabajo es doble: por una parte, nos centraremos en el desarrollo de una aplicación que nos permita clasificar cualquier tipo de texto del que desconozcamos su categoría en otras categorías conocidas mediante la aplicación del clasificador Naïve Bayes; y por otra parte, iremos más allá desarrollando un crawler que, a partir de una serie de enlaces aportados por el usuario, identifique las URLs que contiene las mismas creando copias del texto plano de las sucesivas páginas para posteriormente clasificarlas.

A lo largo del presente trabajo se expondrá todo el marco teórico que envuelve a dichas herramientas con el objetivo de comprender tanto la capacidad que tienen dichas técnicas como las limitaciones que presentan las mismas. Además, exploraremos sus aplicaciones prácticas.

Las aplicaciones desarrolladas serán mostradas en profundidad, con una explicación sobre sus características, estructura y código, para posteriormente

someterla a pruebas con datos reales, clasificando noticias, la detección de Spam en mensajes de SMS y la obtención de un listado de webs que traten una categoría dada. El objeto de estas pruebas no será otro que el de estudiar el rendimiento de las aplicaciones y su capacidad para hacer clasificaciones correctas.

Palabras clave: clasificación de textos, clasificador Naïve Bayes, Machine Learning, Python, procesamiento de lenguaje natural, gestión de documentos, crawler, scraping, web.

Capítulo I. Introducción al Trabajo de Fin de Grado

La tendencia actual en las empresas es intentar obtener la máxima información de sus clientes, tanto *de facto* como potenciales. Para ello se está empezando a utilizar todo tipo de técnicas estadísticas y computacionales que permitan extraer la máxima información de los datos que se generan durante todo el proceso de compra-venta de las empresas. El análisis de los textos contenidos en emails, blogs, tweets, foros, páginas webs, etc. es lo que llamamos análisis de textos dentro del campo del Machine Learning. Esto no es sino un conjunto de estas técnicas estadísticas en este caso enfocadas a la obtención de información de los textos mencionados. El análisis de texto tiene multitud de aplicaciones para todo tipo de empresas, industrias y sectores. Por ejemplo, puedes realizar un análisis de sentimiento a través del cual poder analizar millones de emails rápidamente de tus clientes para extraer la percepción que tienen de tu compañía o producto, otorgándote ventaja frente a la competencia. El análisis de texto también es conocido como minería de texto en el ámbito del Procesamiento del Lenguaje Natural (NLP), uno de los ámbitos principales de la investigación en Inteligencia Artificial. Actualmente el Text Analysis (análisis de texto) está ganando importancia y atención con el desarrollo del Machine Learning y la moda del Big Data.

Según Tom M. Mitchell (1997), uno de los pioneros del Machine Learning y la ciencia de los datos, «El Machine Learning es el estudio de algoritmos informáticos que permiten que los programas informáticos mejoren automáticamente a través de la experiencia».

Una de las ramas del Machine Learning es el aprendizaje supervisado, que consiste en el desarrollo de algoritmos capaces de que, a partir de un sistema de etiquetado de información, el sistema pueda clasificar, tomar decisiones o predecir. En este punto es donde entra en escena el Análisis de Texto y la técnica de clasificación que emplearemos.

En este proyecto nuestro objetivo será doble, en primer lugar, desarrollar una aplicación que mediante un algoritmo sea capaz de clasificar cualquier texto

plano como los presentes en el cuerpo de las páginas webs o los correos electrónicos a partir de un proceso de entrenamiento. En segundo lugar, el desarrollo de un crawler basado en el script de la primera aplicación desarrollada, que tendrá el objetivo de rastrear la web en búsqueda de páginas de una categoría dada. Para ello hemos empleado un clasificador (un algoritmo que asigna a un elemento entrante no etiquetado una categoría concreta conocida) conocido como el Clasificador bayesiano de Naïve Bayes, el cual es muy empleado en estadística y Data Mining. Este clasificador recibe también el nombre de clasificador bayesiano ingenuo debido a que emplea el teorema de Bayes junto con una presuposición de que todas las variables predictoras son independientes entre sí.

Objetivos del Trabajo de Fin de Grado

El objetivo principal de este TFG es, tras realizar un proceso de investigación sobre el análisis de texto, desarrollar dos aplicaciones con multitud de usos que, utilizando el clasificador bayesiano ingenuo, nos permita clasificar todo tipo de textos presentes en la web tanto de manera manual como de forma automatizada. Para ello se requerirá de un entrenamiento previo del algoritmo a partir de textos (o webs para la segunda aplicación) cuya clasificación es conocida para que, una vez realizado esto, el programa sea capaz de estimar la categoría de un texto o web del cual desconocemos su clasificación a partir de probabilidades.

Objetivos personales a desarrollar con el TFG

Los objetivos personales desarrollados durante el proceso de elaboración de este TFG son los siguientes:

- Combinar los conocimientos obtenidos durante el Grado de Estadística Empresarial, junto con otros conocimientos adquiridos de computación y programación durante mis estudios.
- El desarrollo de un proyecto práctico, como es la programación de una aplicación que pueda servir de carta de presentación para el mundo profesional.

Capítulo II. Estado de la cuestión y marco teórico

Aprendizaje bayesiano

El razonamiento bayesiano nos aporta un tipo de inferencia estadística en la que las observaciones se emplean para inferir la probabilidad de que una hipótesis sea cierta o no. Se basa en asumir que podemos tomar la decisión óptima gracias a las probabilidades y los datos observados. Esta clase de aprendizaje es importante dentro del Machine Learning porque nos otorga la capacidad de medir de forma cuantitativa el peso que tienen las variables para aceptar las hipótesis alternativas.

El aprendizaje bayesiano nos aporta todos los instrumentos básicos para los algoritmos de decisión basados en probabilidades como el desarrollado en este trabajo. Además, es uno de los métodos más populares en el análisis de texto debido a que algunos de los algoritmos, como el clasificador Naïve Bayes, nos aportan probabilidades independientes para cada una de las hipótesis, lo cual es bastante importante en ciertos problemas de aprendizaje.

En Michie *et al.* (1994) se realizó un estudio detallado en el que se comparaba el clasificador Naïve Bayes con otra clase de algoritmos de aprendizaje como los árboles de decisión y las redes neuronales, resultando que este clasificador mostraba un rendimiento competitivo con el resto de algoritmos en muchos de los problemas y que, además, en algunos de ellos mostraba rendimientos significativamente superiores como en el caso de la clasificación de textos.

Los métodos de aprendizaje bayesianos presentan una serie de características remarcables:

- Cada observación modifica de forma independiente la probabilidad de que la hipótesis formulada sea correcta o no, lo que nos aporta una mayor flexibilidad en comparación con otros algoritmos que desechan de forma completa las hipótesis que no consideran consistentes.

- Los conocimientos *a priori* pueden ser combinados con observaciones nuevas para determinar las probabilidades de las hipótesis.
- Nos permiten formular hipótesis con capacidad para hacer predicciones probabilísticas (por ejemplo, el paciente tiene una probabilidad del 40 % de recuperarse).
- Los métodos bayesianos son resistentes al ruido causado por ejemplos de entrenamiento con datos incompletos o erróneos.

Del mismo modo, los métodos bayesianos presentan unas dificultades y limitaciones que otros métodos no poseen:

- Necesitan, en la mayoría de los casos, un conocimiento inicial de muchas probabilidades. Esto provoca que en muchos casos estas probabilidades tengan que ser estimadas a través de, por ejemplo, otra clase de información disponible o la asunción de distribuciones.
- Posee una elevada complejidad computacional inicial debido a la cantidad de posibles hipótesis a evaluar. No obstante, hay que añadir en muchos casos que esta complejidad se puede ver significativamente reducida.

A continuación, vamos a ver el fundamento básico del aprendizaje bayesiano, el teorema de Bayes.

El teorema de Bayes

Como hemos mencionado en la introducción, el clasificador bayesiano Naïve Bayes que emplearemos en nuestro proyecto se basa en el teorema de Bayes, por lo que en primer lugar vamos a ver en qué consiste.

A este teorema también se le conoce como el teorema de la probabilidad condicionada. Este nombre nos vislumbra una pista a partir de la cual podemos extraer una idea sobre en qué consiste a rasgos generales.

El teorema de Bayes fue desarrollado por el reverendo Thomas Bayes en el siglo XVII y su objetivo principal fue revisar las probabilidades calculadas *a priori* para luego añadir nueva información.

$$\mathrm{P}\left(\mathrm{h/D}\right) = \frac{P(D/h)\cdot P(h)}{P(D)}$$

De la fórmula podemos extraer que el objetivo del teorema de Bayes es el cálculo de la probabilidad *a posteriori* $P(A_i/B)$. Para ello se multiplica una probabilidad conocida *a priori* $P(A_i)$ con la probabilidad condicional $P(B/A_i)$ y el resultado de este producto es dividido entre la probabilidad total P(B).

Para muchos escenarios además se pueden considerar una serie de hipótesis posibles H y, a partir de esta, encontrar cuál de estas hipótesis es la más probable (h_{MAP}). Esta hipótesis en particular recibe el nombre de *máximum a posteriori* (MAP). Para obtener esta hipótesis se debe calcular la probabilidad *a posteriori* del teorema de Bayes para cada una de las hipótesis h.

$$h_{MAP} \equiv \arg\max_{h\in H} P(h/D) = \arg\max_{h\in H} \frac{P(D/h)\cdot P(h)}{P(D)} = \arg\max_{h\in H} P(D/h)\cdot P(h)$$

Nota: En el último paso se elimina P(D) porque es una constante independiente de h.

Ejemplo

En un país existen 3 periódicos de gran importancia que escriben la mayoría de las noticias del país. Es conocido que el periódico A escribe un 40 % de las noticias del país, el periódico B un 25 % y el periódico C un 35 %. Estos periódicos también tienen una pequeña sección de deportes y sobre esto se conoce que de estos periódicos el periódico A publica el 3 % de las noticias de deportes del país, el B el 1,5 % y el C el 2 %.

¿Cuál es la probabilidad de que una noticia de deportes sea publicada en el periódico B?

Primero calcularemos la probabilidad total que llamaremos P(T):

$$\mathrm{P}\left(D\right) = \left[\ \mathrm{P}\left(\mathrm{A}\right)\ \mathrm{x}\ \mathrm{P}\left(\mathrm{D/A}\right)\ \right] + \left[\ \mathrm{P}\left(\mathrm{B}\right)\ \mathrm{x}\ \mathrm{P}\left(\mathrm{D/B}\right)\ \right] + \left[\ \mathrm{P}\left(\mathrm{C}\right)\ \mathrm{x}\ \mathrm{P}\left(\mathrm{D/C}\right)\ \right] =$$
$$\left[\ 0{,}4\ \mathrm{x}\ 0{,}03\ \right] + \left[\ 0{,}25\ \mathrm{x}\ 0{,}015\ \right] + \left[\ 0{,}35\ \mathrm{x}\ 0{,}02\ \right] = 0.02275$$

De aquí podríamos extraer que la probabilidad de que una noticia de deportes sea publicada es del 2,27 %.

A partir de esta probabilidad calcularemos cuál es la probabilidad de que una noticia de deportes sea publicada en el periódico B con el teorema de Bayes:

$$P(B/D) = \left[P(B)\ x\ P(D/B)\right] / P(D) = \left[0,25\ x\ 0,015\right] / 0,02275 = 0.1648352$$

Podemos observar que esta probabilidad es del 16,4 %.

El clasificador Naïve Bayes

Como comentamos anteriormente en nuestro proyecto, emplearemos el clasificador Naïve Bayes. La elección de este clasificador está motivada por el alto rendimiento que presenta en comparación con otras técnicas más complejas como las redes neuronales o los árboles de decisión.

Este clasificador se basa en el teorema de Bayes, al que se han aplicado una serie de simplificaciones. La más destacable consiste en la presunción de que cada una de las variables predictoras son independientes entre sí, de ahí que se le conozca como el clasificador bayesiano ingenuo.

Como cualquier clasificador, este se basa en lo siguiente: dado un conjunto de datos, dividido a su vez en dos conjuntos: entrenamiento y test, representado en pares <atributo, valor>, el problema consiste en encontrar una función f(x) que clasifique los datos. La ventaja que presenta el clasificador de Naïve Bayes frente a otros clasificadores es, sin duda, poder contar con una probabilidad *a posteriori* para cada una de las hipótesis, con la posibilidad de escoger la más probable.

$$v_{MAP} = \arg\max_{vj \in V} P(v_j \mid a_1, a_2, ..., a_n)$$

Esta expresión se puede reescribir de acuerdo al teorema de Bayes de la siguiente forma:

$$v_{MAP} = \frac{\arg\max_{vj \in V} P(v_j \mid a_1, a_2, ..., a_n)}{P(a_1, a_2, ..., a_n)} = \arg\max_{vj \in V} P(v_j \mid a_1, a_2, ..., a_n) \cdot P(v_j)$$

Para estimar $P(v_j)$ se procede a contar las veces que aparece el dato v_j en el conjunto de entrenamiento y se divide por el número total de datos que forman el conjunto.

En cuanto a $P(v_j|a_1,a_2,\ldots, a_n)$, su estimación consiste en contar las veces que aparecen los valores del ejemplo para cada categoría, recorriendo todo el conjunto de entrenamiento.

Debido a la dificultad computacional de esta última probabilidad, la expresión se puede simplificar mediante la hipótesis de independencia condicional. Dicha hipótesis expresa lo siguiente:

«Los valores a_j que describen un atributo de un ejemplo cualquiera x son independientes entre sí conocido el valor de la categoría a la que pertenecen» Tom M. Mitchell (1997)

De esta manera, la probabilidad de observar el conjunto de atributos a_j dada una categoría a la que pertenecen es igual al producto de las probabilidades de cada valor por separado:

$$P(a_1, a_2, ..., a_n \mid v_j) = \prod_i P(a_i \mid v_j)$$

Con lo que finalmente obtenemos el clasificador Naïve Bayes:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

Donde v_{NB} es el valor objetivo que produce el clasificador Naïve Bayes. Podemos destacar que el número de distintos $P(a_i/v_i)$ que tendrán que ser estimados desde los datos de entrenamiento son simplemente el número de las distintas categorías de nuestros datos, lo cual es un número mucho menor de valores que si pretendiésemos estimar $P(a_1,a_2 \ldots a_n|v_j)$ como se contemplaba inicialmente.

En resumen, el método de aprendizaje de Naïve Bayes requiere un cálculo previo en el que $P(v_j)$ y $P(a_i|v_j)$ sean estimados en función de sus frecuencias en los datos de entrenamiento. El conjunto de estas estimaciones forma una función que creemos que es similar a la real que describe de forma fehaciente esta

clasificación, es decir, nuestra hipótesis o modelo. Nuestra hipótesis es empleada posteriormente para clasificar cada nuevo elemento en la función final del clasificador Naïve Bayes que hemos visto anteriormente. Si se asume que la hipótesis de independencia se cumple, entonces el clasificador v_{NB} es igual que la clasificación MAP vista inicialmente.

Inteligencia competitiva

La inteligencia competitiva es un proceso por el cual las empresas y organizaciones recopilan información sobre todo tipo de cuestiones que les atañen como son sus productos, clientes, competidores, etc.

El objetivo principal de la inteligencia competitiva no es otro que, a través de la información obtenida, lograr tener una ventaja frente a los competidores. Hay otra serie de objetivos alcanzables como comprender mejor a los consumidores en general y a los clientes de una empresa específica o simplemente entender mejor cómo funciona el negocio.

Se suelen utilizar las herramientas de inteligencia competitiva para comparar nuestra empresa con las de la competencia de cara a tomar decisiones con la mayor información posible, ya que de esta manera podemos conocer cuáles son las mayores fuerzas y debilidades tanto de nuestra empresa como las de la competencia.

En definitiva, la inteligencia competitiva permite a las empresas realizar un seguimiento de los movimientos de otras empresas competidoras, conocer el mercado en el que se mueven y así poder adelantarse a los movimientos de la competencia.

A continuación, vamos a estudiar qué es el web scraping y los crawlers, unas herramientas que utilizan muchas empresas para obtener información a través de los contenidos publicados en la red y que forman parte de la inteligencia competitiva.

Crawler o araña web

Un crawler es un software que inspecciona las páginas que se encuentran en la World Wide Web automáticamente. Esta clase de softwares suelen ser bots. Los

bots son programas informáticos que efectúan tareas repetitivas en internet de forma automatizada.

El funcionamiento de los crawlers suele seguir una serie de pasos: en primer lugar, se le aporta una lista con urls de partida que recibe el nombre de semilla; en segundo lugar, el crawler accede a las páginas webs indicadas en la semilla y las descarga y analiza (web scraping); y en tercer lugar, el crawler, al acceder a estas páginas webs, identifica los enlaces presentes en las mismas y los añade a la lista de webs de las que partía inicialmente siguiendo un conjunto de normas o reglas para proceder a volver a descargar y analizar sus enlaces, y así sucesivamente hasta que se pare su funcionamiento de forma manual o a través de alguna clase de criterio.

Uno de los principales usos está estrechamente relacionado con la inteligencia competitiva, pues estos robots nos permiten poder estudiar la competencia de forma sencilla y automatizada, pudiendo detectar la aparición de nuevas empresas en nuestro sector o el lanzamiento de nuevos productos…

Otro uso podría ser detectar la reputación de nuestra empresa mediante el rastreo de foros o redes sociales detectando la información que los clientes suben sobre nuestra marca o producto, pudiendo comparar la información obtenida con la de la competencia.

Web Scraping

El web scraping consiste en extraer información de páginas webs a través de programas informáticos. Este proceso lo llevan a cabo bots que extraen datos o contenidos de los sitios webs. Estos bots simulan que son un ser humano navegando por la red a través del protocolo HTTP. Esta técnica incluye el proceso de transformar los datos de las páginas webs (el código en formato HTML) en datos estructurados y almacenables para su análisis posterior.

El web scraping es un campo con desarrollo activo debido a la dificultad de simular hasta los comportamientos humanos más sencillos en una web como copiar o pegar textos por la gran complejidad de los códigos empleados actualmente en las páginas webs.

Uno de los problemas del web scraping es que puede ser usado de forma maliciosa para obtener información de las bases de datos de las páginas webs, las cuales pueden contener información sensible o simplemente privada. Es por esto que muchas webs prohíben en sus términos de uso emplear web scraping en ellas. Además, muchas páginas webs actualmente usan técnicas y herramientas que impiden el acceso a sus páginas webs a softwares que empleen esta técnica. Estas técnicas incluyen desde los bloqueos a IPs que generen un exceso de tráfico, el uso de sistemas de verificación manual como captchas o añadir entradas al fichero robots.txt que presentan las webs, aunque esto solo detiene a los robots más sencillos.

En relación con los crawlers se podría considerar que el web scraping es una de las funciones que un crawler puede realizar. Estos dos elementos son, pues, parte de las herramientas que las empresas utilizan en sus procesos de inteligencia competitiva.

Capítulo III. Desarrollo de los Scripts

En este apartado veremos el lenguaje de programación utilizado para el desarrollo de los scripts y sus fundamentos, junto con el entorno de desarrollo que hemos empleado, Spyder, el cual ha sido empleado durante el desarrollo del Grado en Estadística Empresarial. Para esto hemos usado el lenguaje de programación Python, un lenguaje relativamente moderno y en alza en el ámbito de la ciencia de los datos. Analizaremos la teoría vista en el capítulo anterior con vistas a interpretarla e introducirla para la computación y explicaremos las dificultades que nos hemos encontrado para trasladarla a la práctica. Para ello nos hemos basado en el algoritmo presentado en el libro *Machine Learning* de Tom Mitchell.

El lenguaje de programación: PYTHON

Hemos empleado PYTHON como lenguaje de programación para desarrollar nuestros Scripts. Python es un lenguaje de programación interpretado, lo que quiere decir que no requiere una compilación previa a la hora de ejecutar las instrucciones, por lo que requiere un programa que haga de intérprete que ejecute el código directamente y traslade cada sentencia a una serie de subrutinas previamente compiladas.

Como puntos positivos de este lenguaje podemos destacar:

- que tiene capacidad para ser usado con distintos fines, es decir, para todo tipo de programación (orientada a objetos, programación web, programación imperativa…),
- además de que es de código abierto y posee una sintaxis sencilla que permite un código con gran legibilidad y limpieza.

En contra podemos señalar el siguiente punto:

- Al ser un lenguaje interpretado, el uso de la memoria está menos optimizado y requiere un esfuerzo computacional mayor.

Spyder ha sido el entorno de desarrollo integrado (IDE) empleado para trabajar sobre PYTHON. Este IDE ha sido empleado en varias ocasiones durante el Grado en Estadística Empresarial. Es el IDE más utilizado para trabajar con

Python, ya que es el que trae por defecto la distribución de Python más común, Anaconda. Además, es bastante utilizado entre estadísticos y científicos de datos debido a que su desarrollo estuvo enfocado en esta clase de usuarios.

El algoritmo empleado

El algoritmo en el que nos basaremos para el desarrollo de nuestras aplicaciones es el presentado por Tom Mitchell en su libro *Machine Learning* (1997). Se desarrolla de forma general un procedimiento a partir del cual se puede adaptar el clasificador de Naïve Bayes a la computación.

En primer lugar, hemos visto que la fórmula del clasificador Naïve-Bayes es la siguiente:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

Pero dada su bondad computacional, emplearemos la siguiente formula equivalente:

$$v_{NB} = \text{argmax}_{v_j \hat{I} v} \log(P(v_{vj})) + \sum_i \log(P(a_i | v_j))$$

Definida la fórmula con la que vamos a trabajar, procedemos con el algoritmo que nos permitirá el aprendizaje y la clasificación de textos:

Algoritmo Naïve-Bayes de aprendizaje

Siendo *ejemplos* un conjunto de documentos de texto cuyas categorías son conocidas. V es un conjunto de todas las posibles categorías. La función $P(W_k|v_j)$ es la probabilidad de que una vez elegida al azar una palabra de un documento que pertenezca a la categoría conocida V_j, esta sea la palabra del vocabulario w_k. Y, por último, $P(v_j)$ representa la probabilidad de que un documento pertenezca a la clase v_j.

1- Formar el conjunto Vocabulario, el conjunto de todas las palabras distintas presentes en el conjunto de documentos *Ejemplos*.
2- Calcular las probabilidades $P(v_j)$ y $P(W_k|v_j)$.
    a. Para cada categoría v_j en V hacer:
        i. $Docs_j$ – El subconjunto de documentos de Ejemplos que pertenecen a la categoría v_j.
        ii. $P(v_j) = \frac{|docs_j|}{|ejemplos|}$
        iii. $Text_j$ – Un solo documento creado concatenando todos los documentos de $docs_j$.
        iv. N – El número total de diferentes posiciones existentes en $Text_j$.
        v. Para cada palabra w_k en Vocabulario.
            1. n_k – Número de veces que la palabra w_k aparece en $Text_j$.
            2. $P(w_k|v_j) = \frac{n_k + 1}{n + |vocabulario|}$

Algoritmo Naïve-Bayes de clasificación

Devuelve la categoría estimada del documento Doc. a_i representa la palabra encontrada en la posición i dentro de Doc.

- Positions – Todas las posiciones de palabra en Doc que contengan palabras existentes en Vocabulario.
- Devolver v_{NB}, donde:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in positions} P\left(a_i \mid v_j\right)$$

Capítulo IV. Desarrollo de la aplicación de clasificación

Se han desarrollados dos scripts, uno destinado al entrenamiento del sistema de clasificación y otro destinado a realizar las predicciones de clasificación. Estos scripts han sido elaborados bajo el lenguaje de programación Python como hemos mencionado anteriormente. A diferencia de otros lenguajes de programación o herramientas destinadas al desarrollo de aplicaciones, para poder ser ejecutados se debe tener una instalación en el ordenador de Python 3, eso sí, sin la necesidad de ninguna clase de módulo ni paquete para el mismo, ya que se ha procurado usar los elementos ya incluidos en la instalación básica de Python.

El Script de entrenamiento

El objetivo de este script no es otro que el de poder introducir documentos cuya categoría es conocida con el objetivo de poder entrenar al sistema clasificador. Este script permite al usuario introducir a través de la consola de comandos de Python o a través de la consola de comandos de Windows (indicando que se quiere ejecutar el archivo con Python con la sentencia “python” frente al nombre del archivo), dos argumentos que son la ruta del archivo de texto donde se contiene el texto a entrenar y su categoría de clasificación.

Las sentencias para ejecutar el script serian del tipo:

“python entrenamiento.py *ruta categoría*”

Al ejecutar la instrucción, el script memoriza la información en dos bases de datos, por una parte, se memoriza un recuento con las categorías existentes y el número de documentos con dicha categoría que se han registrado y, por otra parte, se registran las palabras del documento junto con su frecuencia y su categoría. Esta información se guarda como hemos indicado anteriormente en dos bases de datos en formato .csv. La creación de estas dos bases de datos está contemplada en el script en el caso de que no se encuentren presentes, por lo que este script puede ser ejecutado por cualquier usuario en su ordenador y

crear de cero la información para la clasificación. También los usuarios pueden trabajar con datos elaborados previamente, por lo que esta herramienta se vuelve polivalente. Otra ventaja consiste en que el usuario puede añadir más textos de entrenamiento en cualquier ocasión si es necesario. Como desventaja se podría destacar la necesidad de ir añadiendo los textos de uno en uno.

En este script se obtienen todos los datos necesarios para calcular las probabilidades $P(v_j)$ y $P(W_k|v_j)$ necesarias para calcular la probabilidad de que el texto pertenezca a una u otra categoría. En concreto, se obtienen los siguientes datos del algoritmo generalizado presentado por Tom Mitchell:

- $Docs_j$, subconjunto de documentos de *Ejemplos* que pertenecen a la categoría v_j.

- V*ocabulario,* conjunto de todas las palabras distintas presentes en el conjunto de documentos *Ejemplos*.

- N, número total de diferentes posiciones existentes.

- n_k, número de veces que la palabra w_k aparece.

Finalmente, el script que hemos desarrollado es el siguiente:

```
#Librerias
import unicodedata
import pandas as pd
from sys import argv


script, ruta_txt, categoria = argv


# Palabras vacias
empy = open('empy.txt','r')
vacias = empy.readlines()
palabras_vacias = list()
for palabras in vacias:
    listapalabras = palabras.split()
    for palabra in listapalabras:
        palabras_vacias.append(palabra)

#Recuento de categorias de los archivos de entrenamiento

try:
    rd = pd.read_csv('recuento.csv', sep = ';')

except:
    rd = pd.DataFrame(columns=('Categoria', 'Frecuencia' ))

if categoria in rd.Categoria.values:
    rd.loc[(rd.Categoria == categoria), "Frecuencia"] += 1
else:
    rd.loc[len(rd)] = [categoria,1]

rd.to_csv('recuento.csv', index = False, sep = ';')
```

```
#Leemos vocabulario anterior o creamos uno nuevo
try:
    df = pd.read_csv('vocabulario.csv', sep = ';')

except:
    df = pd.DataFrame(columns=('Palabra', 'Frecuencia', 'Categoria' ))

try:
    file=open(ruta_txt,'r', encoding="utf-8")
    palabras = list()
    data=file.readlines()

except:
    file=open(ruta_txt,'r', encoding="ISO-8859-1")
    palabras = list()
    data=file.readlines()
finally:
    for renglon in data:
        renglon = unicodedata.normalize("NFKD",
renglon).encode("ascii","ignore").decode("ascii")
        listapalabras = renglon.replace(',',' ').replace('\"',' ').replace('.',' ')\
                               .replace('?',' ').replace('¿',' ').replace('¡',' ')\
                               .replace('!',' ').replace("'",' ').replace(`',' ')\
                               .replace(';',' ').replace('¿',' ').replace('"',' ')\
                               .replace(':',' ').replace('"',' ').replace('"',' ')\
                               .replace('"',' ').replace("'",' ').replace('(',' ')\
                               .replace(')',' ').lower().split()

        for palabra in listapalabras:
            if len(palabra) <= 2:
                pass
            elif palabra in palabras_vacias:
                pass
            else:
                palabras.append(palabra)
    for palabra in palabras:
        if palabra not in df.Palabra[df.Categoria == categoria].values:
            df.loc[len(df)] = [palabra,1,categoria]
        else:
            df.loc[(df.Palabra == palabra) & (df.Categoria == categoria),
"Frecuencia"] += 1

    file.close()

df.to_csv('vocabulario.csv', index = False, sep = ';')
```

A continuación, vamos a describir el funcionamiento del script dividiéndolo en varias partes.

Librerías

```
#Librerias

import unicodedata

import pandas as pd

from sys import argv


script, ruta_txt, categoria = argv
```

Las librerías que hemos utilizado son 3 y todas ellas vienen incluidas en la distribución básica de python, por lo que su instalación no es necesaria:

- Unicodedata: esta librería nos permite acceder a la “Unicode Character Database”, que proporciona las propiedades para cada uno de los caracteres por defecto. Usamos esta librería para normalizar los caracteres de los textos empleados en el entrenamiento.
- Pandas: es la principal librería de python para el análisis de datos, que nos permite trabajar con series y DataFrames.
- Sys: esta librería nos permite el acceso a funciones y objetos mantenidos por el intérprete. En concreto, importamos la función sys.argv() con la que se obtiene la lista de argumentos al invocar al intérprete en la consola de comandos. De esta forma, requerimos al usuario la ruta del archivo que se quiere añadir al entrenamiento y su categoría como vemos en la última línea del cuadro de texto anterior.

Detección de palabras vacías

```
# Palabras vacias
empy = open('empy.txt','r')
vacias = empy.readlines()
palabras_vacias = list()
for palabras in vacias:
    listapalabras = palabras.split()
    for palabra in listapalabras:
        palabras_vacias.append(palabra)
```

Hemos introducido un sistema de palabras vacías a partir del cual el programa extrae una serie de palabras que omite al hacer el conteo posterior, ya que se considera que estas palabras no aportan valor en la predicción. Algunos ejemplos de estas palabras pueden ser determinantes como los artículos, cifras, preposiciones, algunos verbos, etc. Estas palabras son leídas de un archivo de texto plano que las contiene separadas por espacios y son elegidas a criterio personal.

Recuento de las categorías de los archivos de entrenamiento

```
#Recuento de categorias de los archivos de entrenamiento


try:
    rd = pd.read_csv('recuento.csv', sep = ';')


except:
    rd = pd.DataFrame(columns=('Categoria', 'Frecuencia' ))


if categoria in rd.Categoria.values:
    rd.loc[(rd.Categoria == categoria), "Frecuencia"] += 1
else:
    rd.loc[len(rd)] = [categoria,1]


rd.to_csv('recuento.csv', index = False, sep = ';')
```

En este apartado se produce la modificación o creación de la base de datos que guarda el recuento de la categoría de los archivos de entrenamiento.

En primer lugar, el código trata de leer esta base de datos para ver si ha sido creada con anterioridad. En caso de que esto no se produzca, el código la crea para poder trabajar con ella. Si la categoría del documento que estamos leyendo está presente en la base de datos, el código sumará uno al valor de la frecuencia de la misma. En caso contrario, la registrará como una nueva categoría junto con la frecuencia igual a uno. Y en último lugar, se guardan las modificaciones producidas.

Lectura, modificación o creación del vocabulario

```
#Leemos vocabulario anterior o creamos uno nuevo
try:
    df = pd.read_csv('vocabulario.csv', sep = ';')

except:
    df = pd.DataFrame(columns=('Palabra', 'Frecuencia', 'Categoria' ))

try:
    file=open(ruta_txt,'r', encoding="utf-8")
    palabras = list()
    data=file.readlines()

except:
    file=open(ruta_txt,'r', encoding="ISO-8859-1")
    palabras = list()
    data=file.readlines()
```

```
finally:
    for renglon in data:
        renglon = unicodedata.normalize("NFKD",
renglon).encode("ascii","ignore").decode("ascii")
        listapalabras = renglon.replace(',',' ').replace('\"',' ').replace('.',' ')\
                                .replace('?',' ').replace('¿',' ').replace('¡',' ')\
                                .replace('!',' ').replace("",' ').replace('`',' ')\
                                .replace(';',' ').replace('¿',' ').replace("",' ')\
                                .replace(':',' ').replace("",' ').replace("",' ')\
                                .replace("'",' ').replace("",' ').replace('(',' ')\
                                .replace(')',' ').lower().split()

        for palabra in listapalabras:
            if len(palabra) <= 2:
                pass
            elif palabra in palabras_vacias:
                pass
            else:
                palabras.append(palabra)
    for palabra in palabras:
        if palabra not in df.Palabra[df.Categoria == categoria].values:
            df.loc[len(df)] = [palabra,1,categoria]
        else:
            df.loc[(df.Palabra == palabra) & (df.Categoria == categoria),
"Frecuencia"] += 1

    file.close()

df.to_csv('vocabulario.csv', index = False, sep = ';')
```

Este es el último apartado que compone el script. En primer lugar, se procede a leer o crear la base de datos que contendrá el vocabulario con todas las palabras, junto con su frecuencia y categoría. El procedimiento es similar al del recuento, primero, se intenta leer la base de datos para comprobar si esta ha sido creada con anterioridad y, si es el caso, leer su información. En caso contrario, se crea la base de datos para poder trabajar sobre ella.

El siguiente paso es leer el archivo de texto que contiene el texto con las palabras que se quieren añadir al diccionario. Se han establecido dos procedimientos para esto. En primer lugar, se intenta leer dicho texto con la codificación utf-8, una de las más habituales y estándares para web. Si esto produce algún error, entonces se procede a leer el texto con codificación ISO-8859-1. Esta codificación es necesaria para leer documentos con caracteres latinos como la ñ (fuente de errores al intentar leer archivos con la codificación utf-8).

A continuación, y una vez leído el documento de texto, se divide el texto en renglones, estos renglones son normalizados para aumentar su compatibilidad por el método NFKD, para posteriormente ser codificado en ASCII y decodificado de nuevo en ASCII. Con este procedimiento eliminamos tanto las tildes como

toda clase de caracteres especiales que pudieran alterar el funcionamiento del vocabulario.

Creamos ahora una lista de palabras mediante la división de los renglones a partir de los espacios encontrados. Además, con el fin de que esto funcione correctamente, reemplazamos los símbolos de puntuación por espacios. De esta manera, la única separación posible entre palabras son los espacios. Tras esto, el programa recorre todos los renglones en los que se ha dividido el texto y guarda las palabras en una variable de forma temporal. Después, recorre todas las palabras, pero omite tanto las palabras que tienen menos de dos caracteres como las palabras que se encuentran en el listado de palabras vacías que cargamos con anterioridad. El resto de palabras se guardan en una variable con todas las palabras que definitivamente se añadirán al diccionario. Si la palabra no se encuentra presente en el listado de vocabulario, se crea un registro nuevo con la misma y, en caso contrario, se suma uno al valor que presenta la frecuencia donde la palabra y la categoría coinciden en el registro de la base de datos.

Finalmente, los cambios de la base de datos se guardan y se termina la ejecución del script.

El Script de clasificación

El objetivo de este script es leer y procesar la información contenida en las bases de datos que se generan a través del script de entrenamiento con el objetivo de aplicar el clasificador Naïve Bayes y poder realizar una predicción sobre la pertenencia de un documento con clasificación desconocida a una categoría conocida.

El procedimiento para ejecutar este script es similar al usado para el de entrenamiento. Se debe ejecutar una sentencia desde la consola de comandos de Python o desde la propia consola de comandos de Windows indicando que se debe ejecutar a través de Python. En este caso se debe especificar únicamente la ruta del archivo de texto que queremos clasificar.

“python clasificacion.py *ruta*”

Al ejecutar la sentencia, el programa nos devolvería en la misma consola de comandos la clasificación que ha realizado, es decir, la categoría elegida y su puntuación. Se podría modificar la forma en la que se muestran los resultados para que aparezca cada categoría con su puntuación. Esto supone una ventaja para esta clase de clasificadores frente a otras técnicas de Machine Learning como los árboles de decisión o las redes neuronales, ya que nos aporta más información que estas al poder conocer, por ejemplo, si otras categorías han obtenido puntuaciones similares o distantes.

Finalmente, presentamos el algoritmo y procedemos a su explicación e interpretación:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  3 16:44:02 2019

@author: Chema
"""

import unicodedata
import pandas as pd
from sys import argv
import math
import numpy as np

script, ruta_txt = argv


# Clasificador Naïve-Bayes
# Primero cargamos el vocabulario
try:
    df = pd.read_csv('vocabulario.csv', sep = ';')

except:
    print("Debe entrenar el programa primero")
# Y leemos las categorias y su frecuencia
try:
    rd = pd.read_csv('recuento.csv', sep = ';')

except:
    print("Debe entrenar el programa primero")

#Segundo abrimos el archivo a clasificar y aplicamos el algoritmo de Naïve bayes
try:
    file=open(ruta_txt,'r', encoding="utf-8")
    palabras = list()
    data=file.readlines()

except:
    file=open(ruta_txt,'r', encoding="ISO-8859-1")
    palabras = list()
    data=file.readlines()
finally:
    for renglon in data:
        renglon = unicodedata.normalize("NFKD",
renglon).encode("ascii","ignore").decode("ascii")
        listapalabras = renglon.replace(',',' ').replace('\"',' ').replace('.',' ')\
                               .replace('?',' ').replace('¿',' ').replace('¡',' ')\
                               .replace('!',' ').replace('"',' ').replace('`',' ')\
                               .replace(';',' ').replace('¿',' ').replace('"',' ')\
                               .replace(':',' ').replace('"',' ').replace("'",' ')\
                               .replace('"',' ').replace("'",' ').replace('(',' ')\
                               .replace(')',' ').lower().split()

nvoc = len(df.Palabra.unique())
probfinal = pd.Series(np.repeat(0,
len(rd.Categoria.unique())),index=rd.Categoria).to_dict()
for categoria in df.Categoria.unique():
    n = len(df[df.Categoria == categoria])
    pvj = math.log(rd.Frecuencia[rd.Categoria ==
categoria]/rd.Frecuencia.sum())
    pav2 = 0
    for palabra in listapalabras:
        if palabra in df.Palabra[df.Categoria == categoria].values:
            nk = int(df.Frecuencia[df.Categoria == categoria][df.Palabra ==
str(palabra)])
            pav = ((nk+1)/(n+nvoc))
            pav2 = pav2 + math.log(pav)
        else:
            pass
    probfinal[categoria] = pvj + pav2

probfinal2 = dict((k, v) for k, v in probfinal.items() if v ==
min(probfinal.values()))
print(probfinal2)
```

Librerías del script de clasificación

```
import unicodedata
import pandas as pd
from sys import argv
import math
import numpy as np


script, ruta_txt = argv
```

Para este script hemos utilizado las tres mismas librerías que en el script de entrenamiento, además de dos librerías de carácter matemático. Las tres primeras librerías, unicodedata, pandas y sys ya han sido explicadas anteriormente, por lo que vamos a pasar directamente a explicar las segundas:

- Math: es un paquete que simplemente nos da acceso a una serie de funciones matemáticas. En nuestra aplicación la usaremos para poder realizar los logaritmos para el clasificador de Naïve Bayes.
- Numpy: es un paquete de Python muy conocido debido a que es básico para llevar a cabo análisis de datos con Python, pues aporta una serie de herramientas como los arrays. En nuestra aplicación únicamente lo hemos utilizado para hacer uso de su función numpy.repeat(), que permite crear un vector con un número repetido n veces.

Lectura de vocabulario y recuento de categorías

```
# Primero cargamos el vocabulario
try:
    df = pd.read_csv('vocabulario.csv', sep = ';')

except:
    print("Debe entrenar el programa primero")
# Y leemos las categorias y su frecuencia
try:
    rd = pd.read_csv('recuento.csv', sep = ';')

except:
    print("Debe entrenar el programa primero")
```

En este apartado procedemos a cargar el vocabulario y el recuento de las categorías que hemos creado con el script de entrenamiento. Hemos usado las declaraciones try y except para poder considerar el caso de que no se haya entrenado el programa a través del algoritmo de entrenamiento, si esto sucede, el script de clasificación lo único que devolverá será un mensaje informativo de que debe entrenarse el programa para poder clasificar.

Lectura del archivo de texto a clasificar y preprocesamiento del mismo

```
try:
    file=open(ruta_txt,'r', encoding="utf-8")
    palabras = list()
    data=file.readlines()

except:
    file=open(ruta_txt,'r', encoding="ISO-8859-1")
    palabras = list()
    data=file.readlines()
finally:
    for renglon in data:
        renglon = unicodedata.normalize("NFKD",
renglon).encode("ascii","ignore").decode("ascii")
        listapalabras = renglon.replace(',',' ').replace('\"',' ').replace('.',' ')\
                                .replace('?',' ').replace('¿',' ').replace('¡',' ')\
                                .replace('!',' ').replace("",' ').replace('`',' ')\
                                .replace(';',' ').replace('¿',' ').replace("",' ')\
                                .replace(':',' ').replace("""," ').replace("“",' ')\
                                .replace("'",' ').replace('"',' ').replace('(',' ')\
                                .replace(')',' ').lower().split()
```

El procedimiento es idéntico al utilizado en el script de entrenamiento. Se establecen dos métodos para abrir los archivos en función de la codificación que pueda ser necesaria (utf-8, ISO-8859-1). Ambos procedimientos leen el texto del archivo indicado y lo guardan en una variable dividiéndolo en renglones.

Estos renglones son recorridos por un bucle que, como ya explicamos anteriormente, normaliza y estandariza el texto en primer lugar, para posteriormente dividir estos renglones en palabras que son guardadas dentro de una variable para su posterior procesamiento.

Cálculos y clasificación con Naïve Bayes

```
nvoc = len(df.Palabra.unique())
probfinal = pd.Series(np.repeat(0, len(rd.Categoria.unique())),index=rd.Categoria).to_dict()
for categoria in df.Categoria.unique():
    n = len(df[df.Categoria == categoria])
    pvj = math.log(rd.Frecuencia[rd.Categoria == categoria]/rd.Frecuencia.sum())
    pav2 = 0
    for palabra in listapalabras:
        if palabra in df.Palabra[df.Categoria == categoria].values:
            nk = int(df.Frecuencia[df.Categoria == categoria][df.Palabra == str(palabra)])
            pav = ((nk+1)/(n+nvoc))
            pav2 = pav2 + math.log(pav)
        else:
            pass
    probfinal[categoria] = pvj + pav2


probfinal2 = dict((k, v) for k, v in probfinal.items() if v == min(probfinal.values()))
print(probfinal2)
```

En este último apartado vamos a ver el proceso desarrollado para hacer todos los cálculos necesarios para conseguir las probabilidades $P(v_j)$ y $P(W_k|v_j)$, indispensables para el modelo del clasificador Naïve Bayes y finalmente obtener las puntuaciones para cada una de las categorías.

En primer lugar, se calcula el conjunto vocabulario o el número de todas las palabras distintas presentes en todos los documentos de entrenamiento, para ello se utiliza una función que cuenta el número de palabras únicas de la base de datos de vocabulario.

En la segunda línea de código se procede a crear una variable del tipo diccionario <atributo, valor> que almacenará más adelante las puntuaciones que obtendremos al aplicar el clasificador Naïve Bayes para cada una de las categorías. Esta variable se crea en blanco usando la función repeat de numpy para rellenarla con tantos ceros como categorías haya y poniendo de índice el nombre de estas categorías.

El siguiente paso consiste en realizar los cálculos a través de un bucle introducido en otro bucle. El primer bucle recorrerá las distintas categorías, en primer lugar, calcula n, que es el número de palabras en los documentos de cada categoría, y, en segundo lugar, se calcula $P(v_j)$ dividiendo el número de documentos de entrenamiento presentes de cada categoría entre el número total de documentos utilizados en el entrenamiento. A este valor se le aplica ya el logaritmo utilizado en el clasificador.

Antes del siguiente bucle se crea la variable pav2 en vacío que almacenará $P(W_k|v_j)$.

A continuación, se realiza un segundo bucle que, en esta ocasión, recorrerá la lista de palabras obtenidas al procesar el texto a clasificar. En primer lugar, se establecerá una condición que comprueba si la palabra de la lista se encuentra en la base de datos de vocabulario, en caso contrario, esta palabra es omitida por el programa. Una vez comprobada que la palabra está incluida en el vocabulario, se continua el proceso. A continuación, se calcula nk, el número de veces que la palabra aparece en los documentos de entrenamiento de cada categoría. Para ello obtenemos la frecuencia de la base de datos de vocabulario filtrando por la categoría y la palabra en cuestión.

El siguiente paso del bucle es calcular $P(W_k|v_j)$, para ello se aplica la siguiente fórmula con los elementos que hemos calculado anteriormente:

$$P(w_k|v_j) = \frac{n_k + 1}{n + |vocabulario|}$$

La puntuación de esta probabilidad se va actualizando a cada palabra que se recorre en el bucle en la siguiente línea y al mismo tiempo calcula su logaritmo. Estos logaritmos se calculan con la función log del paquete math de Python.

Salimos ya del bucle que recorre las palabras y volvemos al que recorre las distintas categorías para calcular las puntuaciones del clasificador Naïve Bayes para cada una de las categorías. Estas puntuaciones se guardan en la variable del tipo diccionario que creamos con anterioridad vacía.

Finalmente, se escoge la categoría que muestra la puntuación más alta y se ordena su impresión en la pantalla de la consola de comandos.

Capítulo V. Rendimiento y resultados de la aplicación de clasificación

Para comprobar el funcionamiento de nuestra aplicación se han realizado una serie de pruebas entrenándolo y comprobando su eficacia para varios tipos de textos y casos.

Entrenando el programa para clasificar noticias

Para este caso se ha realizado un entrenamiento para una muestra de noticias obtenidas de diversos periódicos digitales a día 28/08/2019, en concreto, de *El Mundo*, *ABC*, *El país* y *La Razón*. La muestra para este caso ha sido de 80 noticias de periódicos de 4 categorías diferentes siguiendo la distribución que muestra la siguiente tabla:

Categoría	Nº de muestras
Cultura	16
Deportes	16
Economía	16
Política	32

Tabla 1: Distribución de las muestras de noticias. Fuente: Elaboración propia.

Se han recopilado 16 muestras de cada categoría, a excepción de política que se ha empleado una muestra mayor para albergar noticias de política tanto nacional como internacional.

A continuación, una vez realizado el entrenamiento, comprobamos el rendimiento del clasificador suministrando cuatro noticias de cada categoría. Estas noticias han sido escogidas del periódico digital *La Vanguardia*, distinto a los usados en la fase de entrenamiento. Los resultados obtenidos se muestran en la siguiente tabla.

Categoría	Aciertos	Falso Negativo	Falso Positivo
Cultura	4	0	0
Deportes	4	0	0
Economía	3	1	0
Política	4	0	1

Tabla 2: Resultados de la clasificación de noticias. Fuente: Elaboración propia.

En vista de los resultados, podemos observar que el sistema comete un único error. Este error se ha producido en la categoría de economía, puesto que el algoritmo lo ha clasificado como política por una puntuación bastante ajustada. Cabe señalar que ambas categorías están estrechamente ligadas. Esto nos da una tasa de acierto para esta prueba del 93,75 %. Quizá con un tamaño muestral mayor incluso este error podría haberse evitado. A pesar de esto y teniendo en cuenta que el tamaño muestral es de un tamaño relativamente pequeño, nos demuestra la potencia de esta clase de clasificadores.

Al final del documento se encuentran dos primeros anexos que contienen la relación de noticias usada tanto para el entrenamiento del programa como las noticias clasificadas.

Entrenando el programa para clasificar Spam

En este experimento vamos a usar el programa para clasificar mensajes de texto SMS de spam. Esta clasificación es más sencilla que la anterior, puesto que solo consideraremos dos categorías, “Spam” o “No Spam”. Para entrenar el programa en esta ocasión, hemos obtenido un DataSet en formato csv que contiene mensajes de correos electrónicos previamente clasificados como Spam o no. Contamos con un DataSet obtenido de la web “Kaggle.com” en inglés, formado por 5572 correos electrónicos clasificados, como hemos dicho anteriormente, en Spam o no Spam, de forma que tenemos 747 mensajes clasificados como Spam y 4825 mensajes clasificados como no spam. Un enlace a dicho archivo podrá ser encontrado en el Anexo III al final de este trabajo.

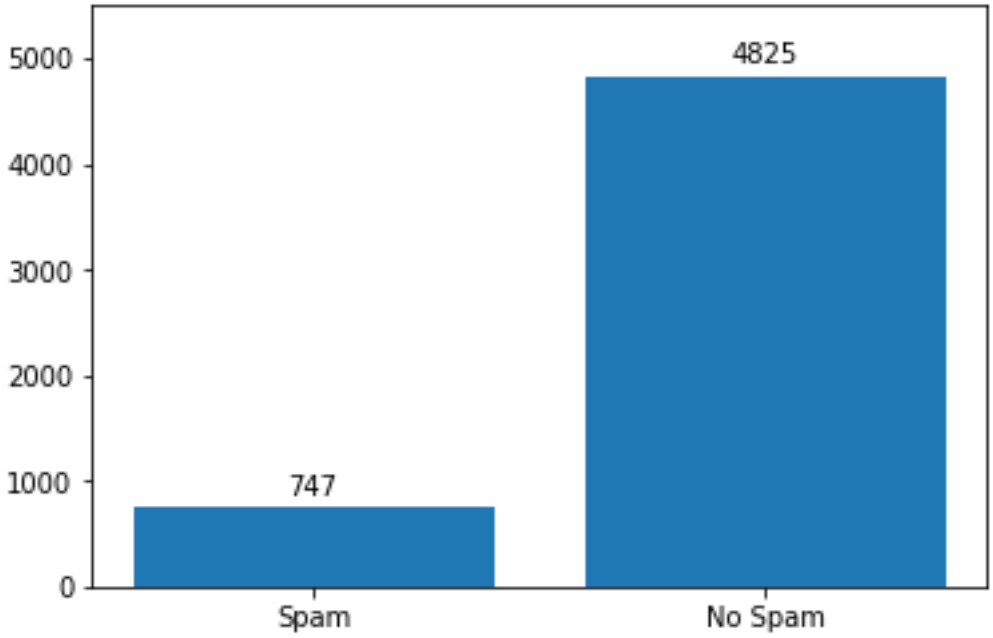

Figura 1: Distribución de los mensajes SMS. Fuente: Elaboración propia.

Hemos tomado una muestra de 1000 de estos mensajes distribuidos de forma que tenemos 250 mensajes de spam y 750 mensajes normales. Estos mensajes han sido seleccionados al azar y transformados en archivos de texto a través de un sencillo código que hemos realizado también en Python. Este código simplemente lee el archivo csv y selecciona una muestra aleatoria de mensajes para después generar archivos de texto con ellos.

```
import pandas as pd


df = pd.read_csv(r'D:\OneDrive\UMH\Estadistica Empresarial\4 curso\TFG 2\Documentos\spam.csv', sep = ',')
dfsample = list(df.Message[df.Category == "spam"].sample(250))
for i in range(250):
    try:
        archivo = open("spam"+str(i)+".txt","w")
        archivo.write(str(dfsample[i]))
        archivo.close
    except:
        pass
dfsample = list(df.Message[df.Category == "ham"].sample(750))
for i in range(750):
    try:
        archivo = open("nospam"+str(i)+".txt","w")
        archivo.write(str(dfsample[i]))
        archivo.close
    except:
        pass
```

Posteriormente, estos mensajes en formato de archivo de texto han sido procesados a través de nuestro script de entrenamiento, al que hemos hecho una pequeña modificación con el fin de automatizar este proceso de entrenamiento para muchos archivos a la vez.

```
for i in range(250):
    try:
        file=open(r'D:\OneDrive\UMH\Estadistica Empresarial\4 curso\TFG 2\Documentos\spam'+ str(i) + '.txt','r', encoding="utf-8")
        palabras = list()
        data=file.readlines()


    except:
        file=open(r'D:\OneDrive\UMH\Estadistica Empresarial\4 curso\TFG 2\Documentos\spam'+ str(i) + '.txt','r', encoding="ISO-8859-1")
        palabras = list()
        data=file.readlines()
```

Hemos modificado el apartado de lectura de archivos del script presentado con anterioridad por el que se muestra en el código superior de tal forma que, en vez de requerir la ruta de un archivo por la consola, hemos creado un bucle que lee todos los archivos de texto que creamos anteriormente y que contiene los mensajes clasificados y los procesa añadiéndolos al vocabulario y al recuento de categorías. También hay que destacar el hecho de que, como estos mensajes están en inglés, hemos cargado un listado de palabras vacías en inglés.

Una vez realizado dicho entrenamiento, hemos procedido a clasificar 100 mensajes también elegidos al azar para comprobar el rendimiento del clasificador bayesiano. Este proceso también ha requerido una pequeña modificación del script para obtener estos mensajes del archivo csv con todos los mensajes de forma automatizada.

```
ds = pd.read_csv(r'D:\OneDrive\UMH\Estadistica Empresarial\4 curso\TFG 2\Documentos\spam.csv', sep = ',')
dfsample = ds.sample(100)


spamspam = 0
spamnospam = 0
nospamspam = 0
nospamnospam = 0
for renglon in dfsample.Message:
    renglon1 = unicodedata.normalize("NFKD", renglon).encode("ascii","ignore").decode("ascii")
    listapalabras = renglon1.replace(',',' ').replace('\"',' ').replace('.',' ')\
```

Hemos cambiado la introducción de la ruta por consola por la lectura de la base de datos en formato csv con los mensajes clasificados. De esta base se han seleccionado 300 mensajes con la función “sample()”, que realiza muestreos.

A continuación, hemos creado unas variables que guardarán la clasificación acertada o no del programa frente a su clasificación real para construir una matriz de confusión en la que presentar el rendimiento.

En la figura también vemos cómo estos mensajes del archivo csv son procesados como texto de la misma forma que hemos visto antes.

Obtenida la clasificación del texto, hemos hecho una modificación en el bucle para que añada valor a las variables que nos servirán para medir el rendimiento que hemos mencionado anteriormente de la siguiente forma:

```
for k in probfinal2.keys():
        if dfsample.Category[dfsample.Message == renglon].item() == "spam":
            if k == "spam":
                spamspam += 1
            else:
                spamnospam += 1
        else:
            if k == "no spam":
                nospamnospam += 1
            else:
                nospamspam += 1
```

El código lee la clasificación que el programa ha dado al texto y lo comprueba con la clasificación real contenida en la base de datos, añadiendo valor a las variables siguiendo una sencilla serie de sentencias if … else.

Finalmente, presentamos los resultados obtenidos en la prueba en la siguiente matriz de confusión:

	Spam Clasificación	No Spam Clasificación
Spam Real	19	17
No Spam Real	15	249

Tabla 3: Matriz de confusión Spam. Fuente: Elaboración propia.

En vista de estos resultados, podemos afirmar que, para este nivel de entrenamiento (con 1000 archivos de entrenamiento), se da una tasa del 89,33 % de acierto global. Para los mensajes que realmente son de Spam nos encontramos con un acierto del 52,77 %, mientras que para los mensajes reales de No Spam la tasa de acierto del 94,31 %. Analizando la clasificación, el porcentaje de acierto de Spam es del 55,88 % y, finalmente, para los mensajes de No Spam, la tasa de acierto es del 93,6 %.

Capítulo VI. Desarrollo del crawler

Para el desarrollo de esta aplicación lo hemos estructurado en dos scripts distintos al igual que la aplicación de clasificación, uno destinado a la descarga de las webs que nos servirán de entrenamiento y otro destinado a las funciones de entrenamiento y clasificación junto con los procesos del crawler. Pese a que lo hemos estructurado de manera similar que la anterior aplicación, vamos a ver cómo tienen poco que ver más allá de la reutilización del algoritmo Naïve Bayes.

El Script de descarga del entrenamiento

El objetivo del Script de descarga del entrenamiento no es otro que el de poder introducir una serie de páginas web previamente clasificadas para poder entrenar al sistema clasificador. Para ejecutar este script, en primer lugar, el usuario tiene que rellenar un fichero CSV con enlaces de páginas webs cuya clasificación es conocida. Este fichero será leído por el script, recorriendo todas las páginas webs para descargar su información, transformarla en texto plano y, por último, guardar el texto plano en archivos de texto.

```
#Librerias
import requests
from bs4 import BeautifulSoup
import pandas as pd
import unicodedata


df = pd.read_csv('entrenamiento.csv', sep = ',')
for categoria in df:
    i = 0
    listna = [x for x in df[str(categoria)] if str(x) != 'nan']
    for link in listna:
        print(link)
        code = requests.get(link)
        plain = code.text
        soup = BeautifulSoup(plain, 'html.parser')
        ##### OBTENER EL TEXTO PLANO#
        # eliminar todos los elementos referidos a estilo y los script del html
        for script in soup(["script", "style"]):
            script.extract()    # rip it out
        # obtener el texto
        text = soup.get_text()
        # dividir en lineas y eliminar el espacio inicial y final de cada
        lines = (line.strip() for line in text.splitlines())
        # separar encabezados multiples en una linea cada uno
        chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
        # eliminar lineas en blanco
        text = '\n'.join(chunk for chunk in chunks if chunk)
        text = text.replace('\n',' ')
        #codificamos el texto
        text = unicodedata.normalize("NFKD", text).encode("ascii","ignore").decode("ascii")
        archivo = open(str(categoria)+str(i)+".txt","w")
        archivo.write(str(text))
        archivo.close
        i += 1
```

A continuación, al igual que los scripts anteriores, vamos a describir el funcionamiento del Script dividiéndolo en varias partes.

Librerías

```
#Librerias
import requests
from bs4 import BeautifulSoup
import pandas as pd
import unicodedata
```

Hemos utilizado 4 librerías, aunque solo dos de ellas no hemos visto anteriormente en este Trabajo de Fin de Grado, BeautifulSoup y requests. De nuevo, estas librerías vienen incluidas en la distribución de Python de Anaconda, por lo que su instalación no es necesaria:

- BeautifulSoup: esta librería sirve para analizar documentos HTML. Funciona de forma que crea un árbol con los elementos del documento HTML del cual se puede extraer información. Es muy utilizada para realizar web scraping.
- Requests: es una librería para realizar todo tipo de peticiones HTTP de manera sencilla en Python.

Obtención y archivado del texto plano de las páginas webs

```
df = pd.read_csv('entrenamiento.csv', sep = ',')

for categoria in df:

    i = 0

    listna = [x for x in df[str(categoria)] if str(x) != 'nan']

    for link in listna:

        print(link)

        code = requests.get(link)

        plain = code.text

        soup = BeautifulSoup(plain, 'html.parser')

       ##### OBTENER EL TEXTO PLANO#

        # eliminar todos los elementos referidos a estilo y los script del html

        for script in soup(["script", "style"]):

            script.extract()    # rip it out


        text = soup.get_text()
```

```
        # obtener el texto

        text = soup.get_text()

        # dividir en lineas y eliminar el espacio inicial y final de cada

        lines = (line.strip() for line in text.splitlines())

        # separar encabezados multiples en una linea cada uno

        chunks = (phrase.strip() for line in lines for phrase in line.split("  "))

        # eliminar lineas en blanco

        text = '\n'.join(chunk for chunk in chunks if chunk)

        text = text.replace('\n',' ')

        #codificamos el texto

        text = unicodedata.normalize("NFKD", text).encode("ascii","ignore").decode("ascii")

        archivo = open(str(categoria)+str(i)+".txt","w")

        archivo.write(str(text))

        archivo.close

        i += 1
```

En este apartado, en primer lugar, se produce la lectura del archivo CSV donde hemos introducido los enlaces cuyo contenido queremos descargar y transformar en texto plano. En segundo lugar, el script recorrerá cada uno de estos enlaces y los procesará para transformarlos en texto plano y luego guardarlos. Cada enlace es descargado a través de una petición HTTP gracias a la librería requests y con un comando de esta misma librería obtenemos solo las etiquetas HTML que contengan texto plano. El siguiente paso será, pues, eliminar todas las etiquetas de HTML con ayuda del paquete BeautifulSoup.

Finalmente, este texto plano se procesa para eliminar los saltos de línea, espacios en blanco y el resto de los elementos de formato, quedando un listado de palabras separadas por un espacio, que guardaremos en un archivo de texto común.

Estos archivos servirán para poder entrenar el clasificador Naïve Bayes como veremos en el apartado siguiente.

El crawler

Este será el Script final encargado de entrenar el clasificador Naïve Bayes para posteriormente poder clasificar las Webs de manera automática. En él se incluyen las diferentes fases necesarias para mostrar los resultados finales, la lectura del listado de palabras vacías, el entrenamiento, la lectura del vocabulario, la obtención de los enlaces de las páginas webs a modo de Crawler y la descarga, transformación en texto plano y procesamiento de los mismos para finalmente aplicar el algoritmo de Naïve Bayes.

Debido a la extensión del Script desarrollado para el crawler, este se podrá encontrar completo en el Anexo IV al final de este Trabajo Fin de Grado.

La fase de entrenamiento desarrollada para este crawler es igual que la usada anteriormente en la aplicación de clasificación por lo que los detalles de esta pueden ser consultados en el Capítulo IV de este trabajo. De igual forma sucede con el algoritmo de clasificación que hemos empleado anteriormente en la aplicación de clasificación.

Vamos a ver a continuación los apartados del Script que pertenecen propiamente al diseño de un crawler y el web scraping.

Web crawling

```
links = ['https://alicantedirectorio.com/informatica/']
n = 0
ident = [links[0].replace("www.","").split(".")[0].split(":")[1]]
while n < 100:
    try:
        code = requests.get(links[n])
        plain = code.text
        soup = BeautifulSoup(plain, 'html.parser')
        for link in soup.find_all(attrs={'href': re.compile("http")}):
            if link.get('href').startswith("https") or link.get('href').startswith("http"):
                linkn = link.get('href').replace("www.","").split(".")[0].split(":")[1]
                if linkn in ident:
                    pass
                else:
                    links.append(link.get('href').replace("www.",""))
                    ident.append(str(linkn.split(".")[0]))
    except:
        pass
    n += 1
```

En este apartado vamos a ver el proceso de web crawling, la obtención de un listado de n enlaces de páginas web que posteriormente se someterán a análisis a partir de uno que emplearemos como semilla.

En primer lugar, vemos cómo introducimos un enlace como semilla. Esta variable nos sirve también para ir guardando los enlaces que vayamos obteniendo posteriormente. Hemos establecido dos criterios en el proceso de web crawling, la obtención de hasta n enlaces y que estos tienen que ser de webs distintas. Para esta última restricción, obtenemos de los enlaces el texto que se encuentra entre “www.” y el siguiente punto para poder identificar las webs. Estas identificaciones se irán comprobando para evitar que se introduzcan dos enlaces

de la misma web. Posteriormente, el script hace una petición a las webs con el paquete requests para descargar su código HTML y, una vez obtenido este código HTML, se extraen del mismo los enlaces filtrando por la etiqueta *href* con la ayuda del paquete BeautifulSoup. Estos enlaces se añaden a la lista que creamos inicialmente con la semilla y se van recorriendo para añadir más enlaces a la misma recursivamente.

Web scraping

```
webs = pd.DataFrame(columns=('Web', 'Texto'))
for enlace in links:
    try:
        code = requests.get(enlace)
        plain = code.text
        soup = BeautifulSoup(plain, 'html.parser')
    # eliminar todos los elementos referidos a estilo y los script del html
        for script in soup(["script", "style"]):
            script.extract()    # rip it out
    # obtenemos el texto
        text = soup.get_text()

        # separamos en lineas
        lines = (line.strip() for line in text.splitlines())
        # eliminamos saltos de linea
        chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
        # eliminamos lineas en blanco
        text = '\n'.join(chunk for chunk in chunks if chunk)
        text = text.replace('\n',' ')
        #codificamos el text
        text = unicodedata.normalize("NFKD", text).encode("ascii","ignore").decode("ascii")
          webs.loc[len(webs)] = [enlace,text]
    except:
        pass
```

Este apartado es muy parecido al visto en el script de descarga de entrenamiento. Durante el mismo la aplicación va procesando el código HTML

para obtener el texto plano de las páginas webs obtenidas en el apartado anterior de web crawling, el texto de estas páginas webs es almacenado en un DataFrame junto con su enlace. Esta variable es la que finalmente pasaremos al algoritmo de clasificación para que la procese.

En primer lugar, se crea el DataFrame para almacenar la información. En segundo lugar, se recorren los enlaces obtenidos por el apartado de web crawling, descargándose el código HTML de cada uno de ellos para finalmente ser procesados como vimos en el apartado de descarga del entrenamiento para obtener, finalmente, el texto plano de las páginas webs preparado para su análisis separando las palabras por un único espacio.

Resumen del funcionamiento del crawler

En este apartado vamos a procurar hacer un resumen esquematizado y global para entender el funcionamiento del crawler en su conjunto.

1) Cargamos todas las librerías necesarias para el script.
2) Leemos el diccionario de palabras vacías.
3) Leemos el vocabulario o creamos uno nuevo si no está creado previamente.
4) Entrenamos la aplicación con las webs de entrenamiento que hemos descargado y procesado anteriormente.
5) Obtenemos n enlaces a partir de uno que empleamos de semilla.
6) Descargamos el código HTML de las webs y lo procesamos para obtener el texto plano.
7) Aplicamos el algoritmo de clasificación.
8) Guardamos la clasificación en un archivo CSV para su posterior consulta.

Complicaciones encontradas durante el desarrollo

Creemos que podría ser interesante comentar la experiencia de programación del crawler y comentar algunos de los problemas que nos hemos encontrado.

La programación de la aplicación no fue realmente complicada en cuanto a su funcionamiento base, sino que el verdadero reto estaba en procurar obtener una buena clasificación. Nos encontramos, en primer lugar, que había distintas webs que bloqueaban esta clase de herramientas solicitando credenciales a la hora de realizar peticiones de descargas, con estas páginas webs simplemente decidimos que las omitiese con el fin de respetar sus términos de uso. Seguidamente nos dimos cuenta de que había muchas webs que la aplicación clasificaba erróneamente cuyo contenido distaba mucho del tema entrenado, no tardamos en percatarnos de que estas webs simplemente mostraban mensajes de error comunes en internet como páginas no encontradas o que estaban caídas. Para evitar estas webs se incluyó en el listado de palabras vacías las palabras que contenían los mensajes de errores más comunes de las webs.

Por último, nos encontramos con que las webs sin texto eran clasificadas erróneamente debido a que a la aplicación daba mayor peso a una de las categorías. Para subsanar este error se decidió aplicar una comprobación para ver si la puntuación de la categoría con la puntuación base más alta es igual que la puntuación que le ha dado el sistema de clasificación a la página web, clasificando estas de forma negativa directamente.

Capítulo VII. Rendimiento y resultados de la aplicación crawler

Para ver en funcionamiento el comportamiento de la aplicación, se ha realizado una prueba que presentaremos a continuación.

Clasificando webs en búsqueda de páginas sobre venta de material informático

Como ejemplo del uso de la aplicación como una herramienta de inteligencia competitiva, se ha simulado el caso de que somos una empresa de informática de la provincia de Alicante dedicada a la venta de material informático y queremos encontrar y detectar webs de empresas que pueden ser competencia directa nuestra.

Para esto hemos realizado una clasificación previa de unas webs que emplearemos como entrenamiento en dos categorías, “Si”, “No”; para la primera categoría se han seleccionado webs y foros con temática de informática y hardware, y para la segunda categoría se ha introducido webs con todo tipo de temáticas distintas a la informática. La selección de estas páginas web se encontrará en el Anexo V al final de este documento.

Tras entrenar la aplicación, procedemos a ejecutar el crawler partiendo con la web “https://alicantedirectorio.com/informatica/” como semilla a partir de la cual buscaremos las páginas webs de la competencia. Hemos limitado al programa a poder extraer los enlaces de 100 páginas webs distintas. Una vez ejecutado, obtenemos un listado de 339 enlaces de páginas webs (esta cifra puede variar en función de las modificaciones que se produzcan en estas páginas). Finalmente, para estos enlaces, la aplicación descarga su texto plano y las clasifica según el clasificador de Naïve Bayes.

Hemos obtenido finalmente un archivo CSV con un listado de 260 webs y su clasificación. Este número es más pequeño que el anterior debido al proceso de

depurado que hemos explicado en el apartado anterior en el que se omiten las webs con mensajes de error o que piden credenciales para hacer peticiones a su servidor. Los resultados obtenidos son los siguientes:

	Material informático Clasificación Si	Material informático Clasificación No
Material informático Real Si	3	0
Material informático Real No	3	254

Tabla 4: Matriz de confusión Crawler. Fuente: Elaboración propia.

Vemos que la tasa de acierto para las páginas webs que son realmente de material informático es del 100 %. Por otra parte, la tasa de acierto para las páginas web que no son de material informático es del 98,83 %. Analizando la clasificación, la tasa de acierto de las páginas web clasificadas como de material informático es del 50 %, en cambio, para las clasificadas como que no son sobre material informático hemos obtenido una tasa de acierto del 100 %. Finalmente, podemos afirmar que la tasa de acierto global es del 98,84 %.

Observamos que el modelo que hemos construido con las páginas web de entrenamiento ha cometido algunos errores al clasificar webs que no son sobre componentes informáticos en la categoría “Si”. Analizando estas webs manualmente, nos encontramos que son sobre marketing digital y posicionamiento web. Este error puede deberse, por lo tanto, a que ambos tipos de web comparten gran parte del lenguaje. Una mejor selección de webs durante la etapa de entrenamiento podría evitar esta clase de errores.

El listado completo de las webs y su clasificación puede consultarse en el Anexo VI.

Capítulo VIII. Conclusiones y trabajo futuro

En este Trabajo de Fin de Grado hemos llevado a cabo una investigación sobre el aprendizaje automático centrándonos en la aplicación de esta cuestión al análisis de textos a través del clasificador Naïve Bayes. Comenzamos introduciendo qué es el Machine Learning y cómo nuestra investigación sobre el clasificador Naïve Bayes se enmarca en él, siendo esta una posible aplicación práctica directa del mismo al mundo real. Para ello, presentamos y desarrollamos toda la teoría respectiva a este clasificador con el objetivo de sentar las bases y el marco teórico para desarrollar una serie de aplicaciones que nos permitan analizar texto y clasificarlo, objetivo último de este trabajo.

En primer lugar, fue desarrollada una aplicación para clasificar cualquier clase de textos y esta fue presentada en los siguientes apartados de este trabajo, explicando todos los detalles de la misma, desde el lenguaje de programación empleado, el algoritmo utilizado hasta la estructura de la misma. Posteriormente, se detalló y explicó el código de la misma de manera minuciosa para la comprensión total de su funcionamiento por parte del lector.

Para dar sentido a nuestra investigación y al desarrollo de esta aplicación, procedimos a aplicar la misma sobre dos casos prácticos, la clasificación de noticias y la detección de mensajes SMS de Spam. En ambas pruebas el rendimiento de la aplicación y el clasificador fue significativo, destacando la media de 89,3 % de clasificaciones correctas para el caso de los SMS de Spam.

En segundo lugar, se dio un paso más allá, se desarrolló una aplicación del tipo crawler que aplicando web scraping es capaz de analizar páginas web y clasificarlas bajo los mismos preceptos que la aplicación anterior, pero de manera automatizada. Igualmente, se llevó a la práctica el uso de la aplicación desarrollada mediante su aplicación a un supuesto real en el que se obtuvieron unos resultados significativos, los cuales se expusieron en sus respectivos apartados de este Trabajo de Fin de Grado.

La investigación desarrollada en las páginas de este Trabajo de Fin de Grado sienta las bases para futuras investigaciones o el desarrollo de nuevas aplicaciones para el clasificador Naïve Bayes. En cuanto a las aplicaciones

desarrolladas, pueden ser el punto de partida para un proyecto mayor y más complejo con aplicación directa en el mundo real, como podría ser el desarrollo de una interfaz de usuario para la misma con el objetivo de comercializarla. También sería interesante investigar la forma de aumentar el abanico de posibilidades que ofrece el clasificador de Naïve Bayes o su rendimiento para textos complejos o incluso un repertorio más extenso de categorías.

Capítulo IX. Bibliografía y Recursos Web

Bibliografía

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine learning, neural and statistical classification, (edited collection). New York: Ellis Horwood.

Mitchell, T. (1997). Machine Learning. New York: McGraw-Hill.

Ian, H. et al. (2011). Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers.

Alonso, R. et al. (1992). Inteligencia artificial y estadística. Revista Estadística Española. Vol. 34, Núm. 131. Págs. 407 a 430.

Moreno, A. et al (2016). Text Analytics: the convergence of Big Data and Artificial Intelligence. International Journal of Interactive Multimedia and Artificial Intelligence. Vol. 3, Núm. 6. Págs. 57 a 64.

Recursos Web

Principal distribución de Python : https://www.anaconda.com/

Web sobre ciencia de datos y repositorio: https://www.kaggle.com/

Librería para graficar con Python : https://matplotlib.org/

Importante librería para análisis de datos con Python : https://pandas.pydata.org/

Anexos

Anexo I. Noticias para la clasificación

Cultura:

https://www.lavanguardia.com/cine/20190828/464276289131/76-mostra-cine-venecia-hollywood-glamur-estrellas.html

https://www.lavanguardia.com/cultura/20190827/464274981569/muere-leopoldo-pomes-fotografo-publicista-hoy.html

https://www.lavanguardia.com/cultura/20190828/476756119/placido-domingo-regresa-escenarios-ovacion-salzburgo.html

https://www.lavanguardia.com/musica/20190828/464287550218/rosalia-premios-mtv-music-canciones-j-balvin.html

Deportes:

https://www.lavanguardia.com/deportes/fc-barcelona/20190828/472340555/barca-fichaje-neymar.html

https://www.lavanguardia.com/deportes/fc-barcelona/20190828/477021182/ousmane-dembele-neymar-barca-psg.html

https://www.lavanguardia.com/deportes/20190828/479619141/real-madrid-isco-lesiones-zidane.html

https://www.lavanguardia.com/deportes/fc-barcelona/20190828/478643456/leo-messi-soleo-barca.html

Economía:

https://www.lavanguardia.com/economia/20190827/464269172669/horas-extra-empleo-espana-trabajo.html

https://www.lavanguardia.com/economia/20190827/464275746091/philip-morris-altria-tabaco-fusion.html

https://www.lavanguardia.com/economia/20190827/464261687442/vuelta-cole-colegio-cuesta-septiembre-financiar-gasto.html

https://www.lavanguardia.com/economia/20190827/464267314349/gasto-pensiones-agosto-record.html

Política:

https://www.lavanguardia.com/politica/20190828/474369659/gobierno-hacienda-pp-ciudadanos-bloqueo-fondos-autonomias.html

https://www.lavanguardia.com/internacional/20190827/464287204071/italia-m5e-pd-pacto-ministerios-gobierno-mattarella.html

https://www.lavanguardia.com/politica/20190828/464287297287/pedro-sanchez-investidura-congreso-pp-cs-podemos.html

https://www.lavanguardia.com/internacional/20190828/477155786/johnson-pedira-isabel-ii-suspenda-parlamento-proximo-14-octubre.html

Anexo II. Noticias entrenamiento

Política:

Nacional:

https://www.abc.es/espana/abci-fallo-proces-dictara-primera-quincena-octubre-201908272055_noticia.html

https://www.abc.es/espana/abci-artimana-psoe-para-entregarle-bildu-alcaldia-pueblo-navarro-201908272007_noticia.html

https://www.abc.es/espana/abci-gobierno-vuelve-negar-coalicion-interesa-investidura-pero-tambien-gobernabilidad-201908271252_noticia.html

https://www.abc.es/espana/abci-podemos-permite-pedro-sanchez-seguir-esquivando-control-congreso-201908272301_noticia.html

https://www.elmundo.es/espana/2019/08/27/5d652227fdddff923e8b45c4.html

https://www.elmundo.es/espana/2019/08/27/5d650448fc6c8380168b45ef.html

https://www.elmundo.es/espana/2019/08/27/5d651ad9fdddff881a8b45a9.html

https://www.elmundo.es/espana/2019/08/28/5d6579cb21efa0ce0c8b45f6.html

https://elpais.com/politica/2019/08/26/actualidad/1566808559_310413.html

https://elpais.com/politica/2019/08/27/actualidad/1566932064_866331.html

https://elpais.com/politica/2019/08/27/actualidad/1566895287_780721.html

https://elpais.com/politica/2019/08/27/actualidad/1566928910_817292.html

https://www.larazon.es/espana/los-etarras-de-venezuela-exigen-volver-a-espana-sin-rendir-cuentas-ante-la-justicia-FA24715404

https://www.larazon.es/espana/calvo-y-no-sanchez-comparecera-por-el-open-arms-a-instancias-de-podemos-HA24713217

https://www.larazon.es/espana/francia-entregara-a-los-etarras-txeroki-y-ata-para-juzgarles-en-espana-AA24711450

https://www.larazon.es/espana/ciudadanos-y-vox-rechazan-sumarse-a-la-plataforma-del-pp-espana-suma-CA24710354

Internacional:

https://www.abc.es/internacional/abci-brasil-rechaza-ayuda-20-millones-para-combatir-incendios-amazonas-201908271039_noticia.html

https://www.abc.es/internacional/abci-conversaciones-entre-y-para-formar-gobierno-estancan-falta-acuerdo-201908271456_noticia.html

https://www.abc.es/internacional/abci-amazonia-vuelve-asunto-personal-entre-bolsonaro-y-macron-201908280114_noticia.html

https://www.abc.es/internacional/abci-venezolanos-saturan-albergues-ecuador-tras-exigencia-visado-201908280115_noticia.html

https://www.elmundo.es/internacional/2019/08/27/5d64e22c21efa01e238b459b.html

https://www.elmundo.es/internacional/2019/08/27/5d657cdffdddff6ebb8b468f.html

https://www.elmundo.es/internacional/2019/08/27/5d65639afdddfffe0a8b462f.html

https://www.elmundo.es/internacional/2019/08/28/5d657930fdddff65108b45a4.html

https://elpais.com/internacional/2019/08/27/actualidad/1566896828_980599.html

https://elpais.com/internacional/2019/08/27/actualidad/1566889091_205938.html

https://elpais.com/internacional/2019/08/27/actualidad/1566925884_340824.html

https://elpais.com/internacional/2019/08/27/actualidad/1566918480_742608.html

https://www.larazon.es/internacional/la-oposicion-britanica-se-une-contra-el-brexit-caotico-AA24715371

https://www.larazon.es/internacional/bolsonaro-y-macron-se-declaran-la-guerra-diplomatica-EA24712464

https://www.larazon.es/internacional/m5e-y-pd-negocian-la-formacion-de-gobierno-hasta-el-ultimo-minuto-CA24714204

https://www.larazon.es/internacional/macron-y-trump-acuerdan-la-continuidad-de-la-tasa-google-hasta-que-legisle-la-ocde-MB24709948

Economía:

https://www.abc.es/economia/abci-bruselas-investiga-nuevo-google-practica-anticompetitiva-201908271717_noticia.html

https://www.abc.es/economia/abci-eeuu-y-china-rebajan-tension-pero-mantienen-amenaza-mas-aranceles-201908270138_noticia.html

https://www.abc.es/economia/abci-espanoles-realizaron-166-millones-horas-extras-remuneradas-mayor-cifra-10-anos-201908271339_noticia.html

https://www.abc.es/economia/abci-china-abraza-capitalismo-plan-para-fomentar-consumo-201908280114_noticia.html

https://www.elmundo.es/economia/macroeconomia/2019/08/27/5d654ec4fc6c83403b8b462b.html

https://www.elmundo.es/economia/macroeconomia/2019/08/27/5d65448bfc6c8380168b4610.html

https://www.elmundo.es/economia/macroeconomia/2019/08/27/5d6512bdfc6c83f32f8b468f.html

https://www.elmundo.es/economia/macroeconomia/2019/08/27/5d64e27cfc6c83c73c8b4643.html

https://elpais.com/economia/2019/08/27/actualidad/1566912677_989472.html

https://elpais.com/economia/2019/08/27/actualidad/1566927700_300044.html

https://elpais.com/economia/2019/08/27/actualidad/1566934129_964983.html

https://elpais.com/economia/2019/08/27/actualidad/1566896821_209082.html

https://www.larazon.es/economia/la-pension-media-de-los-autonomos-solo-alcanza-el-76-del-salario-minimo-MJ24699984

https://www.larazon.es/economia/el-ibex-35-despierta-plano-009-y-se-mantiene-por-debajo-de-los-8700-enteros-OB24708723

https://www.larazon.es/economia/el-yuan-en-minimos-de-once-anos-PJ24697585

https://www.larazon.es/economia/los-nuevos-jubilados-ganan-89-euros-mas-al-mes-que-los-asalariados-LA24714096

Deporte:

https://www.abc.es/deportes/futbol/abci-negociador-llevo-hijo-precipicio-201908272135_noticia.html

https://www.abc.es/deportes/baloncesto/abci-espana-argentina-mundial-baloncesto-201908271228_directo.html

https://www.abc.es/deportes/real-madrid/abci-madrid-vuelve-decir-no-canje-entre-neymar-y-vinicius-201908261745_noticia.html

https://www.abc.es/deportes/real-madrid/abci-zidane-entredicho-201908260110_noticia.html#vca=mod-lo-mas-p5&vmc=leido&vso=deportes&vli=portadilla.deportes&vtm_loMas=si

https://elpais.com/deportes/2019/08/27/actualidad/1566929803_047849.html

https://elpais.com/deportes/2019/08/27/actualidad/1566919731_143399.html

https://elpais.com/deportes/2019/08/27/actualidad/1566905431_664620.html

https://elpais.com/deportes/2019/08/27/actualidad/1566937392_757635.html

https://www.elmundo.es/deportes/baloncesto/2019/08/27/5d6563b021efa0b4098b463f.html

https://www.elmundo.es/deportes/futbol/primera-division/2019/08/27/5d655d02fdddff5f5c8b45d1.html

https://www.elmundo.es/deportes/tenis/us-open/2019/08/27/5d65752afc6c83396c8b45d9.html

https://www.elmundo.es/deportes/futbol/2019/08/26/5d6411f7fc6c8352648b4630.html

https://www.larazon.es/deportes/nadal-contundente-para-empezar-en-el-abierto-de-estados-unidos-AA24717777

https://www.larazon.es/deportes/la-clave-de-la-seleccion-espanola-de-baloncesto-EA24714716

https://www.larazon.es/deportes/neymar-por-que-es-una-operacion-de-riesgo-para-el-barcelona-NA24714180

https://www.larazon.es/deportes/el-once-del-real-madrid-de-zidane-contra-el-villarreal-CA24713980

Cultura:

https://www.abc.es/cultura/teatros/abci-juan-jose-campanella-cine-esta-representacion-actores-teatro-esta-alma-201908280056_noticia.html

https://www.abc.es/cultura/musica/abci-spotify-revela-cuales-sido-canciones-mas-escuchadas-verano-201908271640_noticia.html

https://www.abc.es/cultura/musica/abci-placido-domingo-regresa-escenarios-concierto-hungria-tras-ovacion-salzburgo-201908281058_noticia.html

https://www.abc.es/cultura/musica/abci-noche-historica-rosalia-doctora-video-music-awards-201908272052_noticia.html

https://www.elmundo.es/cultura/cine/2019/08/27/5d65833cfdddff7c288b45ab.html

https://www.elmundo.es/cultura/2019/08/27/5d654cb521efa0a67b8b458a.html

https://www.elmundo.es/cultura/musica/2019/08/27/5d647e8721efa0b51a8b4594.html

https://www.elmundo.es/cultura/toros/2019/08/27/5d65932e21efa0043f8b45f1.html

https://elpais.com/cultura/2019/08/27/actualidad/1566919676_264971.html

https://elpais.com/cultura/2019/08/27/actualidad/1566919659_260712.html

https://elpais.com/cultura/2019/08/27/actualidad/1566888044_478086.html

https://elpais.com/cultura/2019/08/27/actualidad/1566914950_868315.html

https://www.larazon.es/cultura/hay-quien-le-tosa-a-rosalia-KD24722293

https://www.larazon.es/cultura/muere-el-fotografo-leopoldo-pomes-a-los-87-anos-JA24712658

https://www.larazon.es/cultura/millennium-la-despedida-mas-oscura-de-salander-DB24708568

https://www.larazon.es/toros/puerta-grande-de-urena-y-ventura-en-la-ultima-de-colmenar-AA24716644

Anexo III. Enlace de descarga del dataset de spam

https://www.kaggle.com/uciml/sms-spam-collection-dataset

Anexo IV. Script completo del crawler

```
######## FASE DE ENTRENAMIENTO ######################

#Librerias

import unicodedata

import pandas as pd

import os

import math

import numpy as np

import requests

import re

from bs4 import BeautifulSoup


# Palabras vacias
```

```
empy = open("vacias.txt",'r')
vacias = empy.readlines()
palabras_vacias = list()
for palabras in vacias:
    listapalabras = palabras.split()
    for palabra in listapalabras:
        palabras_vacias.append(palabra)

#Leemos vocabulario anterior o creamos uno nuevo
try:
    df = pd.read_csv('vocabulario.csv', sep = ';')

except:
    df = pd.DataFrame(columns=('Palabra', 'Frecuencia', 'Categoria' ))

#Leemos el recuento de categorias
try:
    rd = pd.read_csv('recuento.csv', sep = ';')

except:
    rd = pd.DataFrame(columns=('Categoria', 'Frecuencia' ))

categorias = ["Si", "No"]
for categoria in categorias:
    #Entrenamiento
    for i in os.listdir():
        if i.startswith(categoria):
            print(i)
            print(categoria)
            # Añadimos al recuento de categorias
            if categoria in rd.Categoria.values:
                rd.loc[(rd.Categoria == categoria), "Frecuencia"] += 1
            else:
                rd.loc[len(rd)] = [categoria,1]

            #Abrimos los archivos
            try:
                file=open(str(i), encoding="utf-8")
                palabras = list()
```

```
            data=file.readlines()

        except:
            file=open(str(i), encoding="ISO-8859-1")
            palabras = list()
            data=file.readlines()
        finally:
            for renglon in data:
                renglon = unicodedata.normalize("NFKD", renglon).encode("ascii","ignore").decode("ascii")
                listapalabras = renglon.replace(',',' ').replace('\"',' ').replace('.',' ')\
                                        .replace('?',' ').replace('¿',' ').replace('¡',' ')\
                                        .replace('!',' ').replace("",' ').replace('`',' ')\
                                        .replace(';',' ').replace('¿',' ').replace("",' ')\
                                        .replace(':',' ').replace("",' ').replace("",' ')\
                                        .replace("",' ').replace("',' ').replace('(',' ')\
                                        .replace(')',' ').lower().split()

                for palabra in listapalabras:
                    if len(palabra) <= 2:
                        pass
                    elif palabra in palabras_vacias:
                        pass
                    else:
                        palabras.append(palabra)
            for palabra in palabras:
                if palabra not in df.Palabra[df.Categoria == categoria].values:
                    df.loc[len(df)] = [palabra,1,categoria]
                else:
                    df.loc[(df.Palabra == palabra) & (df.Categoria == categoria), "Frecuencia"] += 1

            file.close()

df.to_csv('vocabulario.csv', index = False, sep = ';')
rd.to_csv('recuento.csv', index = False, sep = ';')

######## FASE DE CLASIFICACIÓN + CRAWLER ###############
#### OBTENER TODOS LOS ENLACES DE N PAGINAS WEB Y ALMACENARLAS EN UN LIST ####
links = ['https://alicantedirectorio.com/informatica/']
n = 0
```

```
ident = [links[0].replace("www.","").split(".")[0].split(":")[1]]
while n < 100:
    try:
        code = requests.get(links[n])
        plain = code.text
        soup = BeautifulSoup(plain, 'html.parser')
        for link in soup.find_all(attrs={'href': re.compile("http")}):
            if link.get('href').startswith("https") or link.get('href').startswith("http"):
                linkn = link.get('href').replace("www.","").split(".")[0].split(":")[1]
                if linkn in ident:
                    pass
                else:
                    links.append(link.get('href').replace("www.",""))
                    ident.append(str(linkn.split(".")[0]))
    except:
        pass
    n += 1

##### OBTENER EL TEXTO PLANO#
## Recorrer los enlaces obtenidos por el crawler

webs = pd.DataFrame(columns=('Web', 'Texto'))
for enlace in links:
    try:
        code = requests.get(enlace)
        plain = code.text
        soup = BeautifulSoup(plain, 'html.parser')
    # eliminar todos los elementos referidos a estilo y los script del html
        for script in soup(["script", "style"]):
            script.extract()    # rip it out
    # obtenemos el texto
        text = soup.get_text()

        # separamos en lineas
        lines = (line.strip() for line in text.splitlines())
        # eliminamos saltos de linea
        chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
        # eliminamos lineas en blanco
        text = '\n'.join(chunk for chunk in chunks if chunk)
```

```
        text = text.replace('\n',' ')
        #codificamos el text
        text = unicodedata.normalize("NFKD", text).encode("ascii","ignore").decode("ascii")
        webs.loc[len(webs)] = [enlace,text]
    except:
        pass
# Clasificador Naive-Bayes
# Primero cargamos el vocabulario
try:
    df = pd.read_csv('vocabulario.csv', sep = ';')
except:
    print("Debe entrenar el programa primero")
# Y leemos las categorias y su frecuencia
try:
    rd = pd.read_csv('recuento.csv', sep = ';')

except:
    print("Debe entrenar el programa primero")
#Segundo abrimos el archivo a clasificar y aplicamos el algoritmo de naive bayes
try:
    results = pd.read_csv('resultados.csv', sep = ';')

except:
    results = pd.DataFrame(columns=('Url', 'Clasificacion', 'Puntuacion' ))

for renglon in webs.Texto:
    try:
        renglon1 = unicodedata.normalize("NFKD",  renglon).encode("ascii","ignore").decode("ascii")
        listapalabras = renglon1.replace(',',' ').replace('\"',' ').replace('.',' ')\
                                .replace('?',' ').replace('¿',' ').replace('¡',' ')\
                                .replace('!',' ').replace("",' ').replace('`',' ')\
                                .replace(';',' ').replace('¿',' ').replace('"',' ')\
                                .replace(':',' ').replace("",' ').replace("“",' ')\
                                .replace('“',' ').replace("',' ').replace('(',' ')\
                                .replace(')',' ').lower().split()

        nvoc = len(df.Palabra.unique())
        probfinal = pd.Series(np.repeat(0, len(rd.Categoria.unique())),index=rd.Categoria).to_dict()
        for categoria in df.Categoria.unique():
```

```
            n = len(df[df.Categoria == categoria])
            pvj = math.log(rd.Frecuencia[rd.Categoria == categoria]/rd.Frecuencia.sum())
            pav2 = 0
            for palabra in listapalabras:
                if palabra not in palabras_vacias:
                    if palabra in df.Palabra[df.Categoria == categoria].values:
                        try:
                            numero = int(palabra)
                        except:
                            nk = int(df.Frecuencia[df.Categoria == categoria][df.Palabra == str(palabra)])
                            pav = ((nk+1)/(n+nvoc))
                            pav2 = pav2 + math.log(pav)
#                             print(str(palabra)+"  " + str(pav2)+"  "+str(categoria))
                    else:
                        pass
            probfinal[categoria] = pvj + pav2

        probfinal2 = dict((k, v) for k, v in probfinal.items() if v == min(probfinal.values()))
#         print(probfinal)
#         print(probfinal2)
        if probfinal["Si"] == math.log(rd.Frecuencia[rd.Categoria == "Si"]/rd.Frecuencia.sum()):
            pass
        else:
            try:
                if list(probfinal2.keys())[0] == "No":
                    results.loc[len(results)] = [webs.Web[webs.Texto == renglon].item(),"No",str(min(probfinal.values()))]

                if list(probfinal2.keys())[0] == "Si":
                    results.loc[len(results)] = [webs.Web[webs.Texto == renglon].item(),"Si",str(min(probfinal.values()))]
                else:
                    pass
            except:
                pass
    except:
        pass

results.to_csv('resultados.csv', index = False, sep = ';')
```

Anexo V. Entrenamiento del crawler.

Si	No
https://www.adslzone.net/foro/hardware.25/	https://elpais.com/
https://forohardware.com/	https://www.marca.com/
https://www.geeknetic.es/	https://vandal.elespanol.com/
https://www.info-computer.com/ordenadores/	https://www.zara.com/es/
https://www.pc-portatil.com/pub.shop/index.php	https://www.carrefour.es/
https://www.elgrupoinformatico.com/foro/foro-windows-f98.html	http://www.fit21.es/
https://foro.elchapuzasinformatico.com/index.php	https://elderecho.com/
https://elchapuzasinformatico.com/	https://www.umh.es/
https://www.3djuegos.com/	http://www.elche.es/
https://www.3djuegos.com/foro-de/3/0/pc/	https://www.casadellibro.com/libros-recomendados
https://hardzone.es/	https://www.es.catholic.net/
	https://es-es.facebook.com/
	https://aquihayquimica.iqs.edu/la-quimica-y-sus-aplicaciones/
	https://www.lanasa.net/
	https://es.wikipedia.org/wiki/Wikipedia:Portada
	https://www.coches.net/
	https://www.mundojardineria.com/
	http://www.sensacine.com/
	https://www.elle.com/es/belleza/

Anexo VI. Resultados completos crawler

Url	Clasificación	Clasificación real
https://alicantedirectorio.com/informatica/	No	No
https://fonts.googleapis.com/css?family=Open+Sans	No	No
http://lmental.com	No	No
http://masadelante.com	No	No
https://lobocom.es/	No	No
http://cdmap.es	No	No
http://ambigramaec.com	No	No
http://neodym.com.es/	No	No
http://pymeup.com/	No	No
https://seosem.es/	No	No
https://bonnieandclick.com/	No	No
https://telemaco.es/	No	No
https://eclipse-sid.es/	No	No
https://wificom.es/	No	No
https://sollutia.com	No	No
https://lotusinnovation.es/	No	No
http://elchedigital.es/	No	No
https://ilimarketing.com/	No	No
https://devilishgames.com/	No	No
http://gaia-soft.com/	No	No
http://rdx.es/	No	No
http://alebus.com	Si	Si
http://gcssoftware.es	Si	Si
https://asci-ntd.com/	No	No
https://anelis.com/	No	No
https://aplicacionesmovilesalicante.es/	No	No

http://digi-tel.es	No	No
https://dragonet.es/	No	No
https://noumaster.com/	No	No
https://natural.es/	No	No
https://instagram.com/alicanteenfotos/	No	No
https://twitter.com/alicantenfotos	No	No
https://es-la.facebook.com/alicantedirectorio/	No	No
https://youtube.com/channel/UCXDmPlJ9TzLmhR6gqqtmfiA	No	No
http://webpositerlabgroup.com	No	No
https://facebook.com/pages/Lmental/713588842057260?ref=hl	No	No
https://plus.google.com/u/0/b/112856742822844352742/112856742822844352742/about?hl=es&service=PLUS	No	No
http://pymessomostodos.com	No	No
http://webpositer.com/market/	No	No
http://gabilos.com/	No	No
http://ready4social.com/	No	No
http://lenceria-sexy.net/	No	No
http://abcimprenta.com/	No	No
http://mariajoyas.es/	No	No
http://ideasdespedidassoltera.es/	No	No
http://iday.es/	No	No
http://arquesclinic.com/	No	No
http://dacceso.com/	No	No
https://thecooksters.com/	No	No
https://garciaanton.com/	No	No
https://soloesunperro.com/	No	No
http://siemservicios.com/	No	No

http://webconversionmaster.com/master-en-conversion-web.html	No	No
https://ajax.cloudflare.com/	No	No
https://bolsasdevacioonline.com/	No	No
https://sacsousvide.com/fr/	No	No
http://crossfitbluewolf.com	No	No
http://citiusbox.com/	No	No
http://masquelimpio.com/	No	No
https://vivendus.net	No	No
https://manuelduran.es/	No	No
http://gmpg.org/xfn/11	No	No
https://confianzaonline.es/empresas/seosem.htm	No	No
https://statcounter.com/	No	No
https://empresasmantenimientoinformatico.com/	No	No
http://apps.camaltec.es	No	No
http://diarioinformacion.com/empresas-en-alicante/	No	No
https://protecciondedatosmarcablanca.es/	No	No
https://redacciondecontenidosmarcablanca.es/	No	No
https://exclusiveseo.es/	No	No
https://posicionamientowebgarantizado.net/distribuidores-seo/	No	No
http://nirvel.es	No	No
http://aitex.es	No	No
http://avamet.org	No	No
http://alcoi.org	No	No
http://terrvs.com	No	No
http://evolutia.eu	No	No

http://unionalcoyana.com	No	No
http://losdi.com	No	No
http://ccalzamora.es	No	No
http://areagraficadigital.es	No	No
http://ancavico.com	No	No
http://alhambrasl.com	No	No
http://httpd.apache.org/	No	No
http://centos.org/	No	No
http://internic.net/whois.html	No	No
https://es.pinterest.com/lotusinnovation/	No	No
http://wordpress.org/plugins/asesor-cookies-para-la-ley-en-espana/	No	No
https://wame.chat/powered/?site=ilimarketing&url=https%3A%2F%2Filimarketing.com	No	No
https://youtu.be/OY-Ok3cP2yk	No	No
https://t.co/9Dtnrr91pa	No	No
https://store.steampowered.com/app/770410/Path_to_Mnemosyne/	No	No
https://livedealer21.com/	No	No
https://1.bp.blogspot.com/-n01Shfoa4wE/Xamglh_JcJI/AAAAAAAABPA/5OSvEKVmRVsuX9YoiRcNy5lReDLBPUJgwCLcBGAsYHQ/s1600/manoplanta.png	No	No
https://blogger.com/	No	No
http://alumasa.com/	No	No
http://hp.es	Si	Si
http://microsoft.es	No	No
http://pandasecurity.com/	No	No
http://citrix.es	No	No
http://aplicacionesmovilesvalencia.es/	No	No

http://aplicacionesmovilesmadrid.es	No	No
http://desarrollodeaplicaciones.es	No	No
http://fincapp.es/	No	No
http://naturalformacion.es	No	No
https://abs.twimg.com/a/1572376946/css/t1/twitter_core.bundle.css	No	No
https://publish.twitter.com/oembed?url=https://twitter.com/alicantenfotos	No	No
https://mobile.twitter.com/alicantenfotos	No	No
https://help.twitter.com/rules-and-policies/twitter-cookies	No	No
http://support.twitter.com/forums/26810/entries/78525	No	No
https://dev.twitter.com/web/embedded-tweets	No	No
https://m.facebook.com/alicantedirectorio/?locale2=es_LA	No	No
https://chrome.google.com/webstore/detail/gpdjojdkbbmdfjfahjcgigfpmkopogic	No	No
https://yt3.ggpht.com/a/AGF-l7_yACu2UFEC7e1y5HdW8DQvUO_w8jTJKQdYjw=s900-c-k-c0xffffffff-no-rj-mo	No	No
https://accounts.google.com/ServiceLogin?passive=true&hl=es&uilel=3&continue=https%3A%2F%2Fyoutube.com%2Fsignin%3Fnext%3D%252Fchannel%252FUCXDmPlJ9TzLmhR6gqqtmfiA%26action_handle_signin%3Dtrue%26hl%3Des%26app%3Ddesktop%26feature%3Dsign_in_button&service=youtube	No	No
http://alicanteseo.es/	No	No
http://linkbuilded.com/	No	No

http://seomental.com/raiola-manda-y-no-el-panda.html	No	No
https://webpositeracademy.com/seleccion.php?opt=programador&x=1572782022&y=1572522822	No	No
http://ine.es/varipc/index.do	No	No
http://loogic.com/ready4social-herramienta-de-gestion-de-redes-sociales-con-motor-de-seleccion-de-contenidos/	No	No
http://josefacchin.com/2015/12/14/curacion-de-contenidos-y-content-curator/	No	No
http://abc.es/local-comunidad-valenciana/20121120/abci-gosende-articulo-201211201212.html	No	No
http://caracolaregalos.es	No	No
https://review-port.com/es/bentolit-opiniones-criticas-comprar/	No	No
https://likeastore.com/es/bentolit-opiniones-criticas-comprar/	No	No
https://product-club.com/es/drivelan-en-mexico-opiniones-donde-comprar-precio-efectos/	No	No
https://svartd.es/taneral-pro-cinturon-criticas-comprar.htm	No	No
https://tarif-lettre.com/tarif-colissimo	No	No
http://joselab.com/	Si	No
https://campamentoweb.com/	Si	No
https://mjcachon.com/	No	No
https://digital-nature.com/	No	No
https://jordiob.com/	No	No
http://seobox.club/	No	No
https://manipulador-de-alimentos.com/	No	No

https://use.fontawesome.com/releases/v5.7.2/css/all.css	No	No
https://tumblr.com/blog/siemservicios	No	No
https://dash.cloudflare.com	No	No
https://tools.ietf.org/html/rfc6750#section-2.1	No	No
https://en.wikipedia.org/wiki/Zone_file	No	No
http://iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4	No	No
https://tus.io	No	No
https://workers.cloudflare.com/docs/reference/tooling/api/	No	No
http://yelp.com/biz/crossfit-citius-san-vicente-del-raspeig	No	No
http://slideshare.net/vivendus_net	No	No
https://cdn.shortpixel.ai	No	No
https://arpapatrimonio.com/	No	No
https://tcomsl.com/	No	No
http://5ocio.com/	No	No
https://nauticost.com/	No	No
https://flyboat.es/	No	No
http://xixogelat.com/	No	No
http://amplivox.eu/	No	No
https://mvoy.es/	No	No
http://opticatorrellano.es/	No	No
https://pascualehijos.es/	No	No
https://crecimientodigital.es/	No	No
https://farmaciaelaltet.com/	No	No
https://apartamentoenbenidorm.net/	No	No
http://yjcmarketing.com/	Si	No
http://frannavarro.es/	No	No

https://hnosjimenez.com/	No	No
http://fidelizo.es/	No	No
https://drivecars.es/	No	No
http://publigrafic.es/	No	No
https://korearqueologia.com/	No	No
http://tantek.com/	No	No
http://photomatt.net/	No	No
http://meyerweb.com/	No	No
http://w3.org/TR/html401/struct/global.html#h-7.4.4.3	No	No
http://creativecommons.org/licenses/by-nd/2.0/	No	No
https://ecommercetrustmark.eu	No	No
https://zh_CN.statcounter.com/	No	No
https://zh_TW.statcounter.com/	No	No
https://ca.statcounter.com/	No	No
https://fr.statcounter.com/	No	No
https://nb.statcounter.com/	No	No
https://sr.statcounter.com/	No	No
https://pt.statcounter.com/	No	No
https://pt_BR.statcounter.com/	No	No
https://tl.statcounter.com/	No	No
https://af.statcounter.com/	No	No
https://ml.statcounter.com/	No	No
https://cs.statcounter.com/	No	No
https://th.statcounter.com/	No	No
https://de.statcounter.com/	No	No
https://gl.statcounter.com/	No	No
https://az.statcounter.com/	No	No
https://hi.statcounter.com/	No	No

https://fa.statcounter.com/	No	No
http://gs.statcounter.com/	No	No
https://forum.statcounter.com/	No	No
https://itunes.apple.com/ie/app/statcounter-free-real-time/id903409665?mt=8	No	No
https://play.google.com/store/apps/details?id=com.statcounter.statcounterapp	No	No
http://miniapps.camaltec.es/	No	No
https://estaticos.diarioinformacion.com/elementosWeb/gen/css/2019/08/06/seccion20190806155150.min.css	No	No
https://micuenta.diarioinformacion.com/suscripcion/galeria/	No	No
https://ocio.diarioinformacion.com/cine/cartelera/alicante/	No	No
https://tiempo.diarioinformacion.com/	No	No
https://iberempleos.es/ofertas-empleo/alicante/	No	No
https://tucasa.com/compra-venta/viviendas/alicante/?r=&idz=0003	No	No
https://mas.diarioinformacion.com/farmacias/index.php	No	No
https://comunidad.diarioinformacion.com/servicios/Promociones/promociones.jsp	No	No
https://afondo.diarioinformacion.com/	No	No
https://formula1.lne.es/	No	No
https://soiscultura.diarioinformacion.com/	No	No
http://compramejor.es/	No	No
http://iberpisos.es/	No	No
http://ibercoches.es/	No	No

https://laloterianavidad.com	No	No
http://premios-cine.com/	No	No
http://epi.es/politica-de-privacidad.html	No	No
http://becontent.es/	No	No
https://diaridegirona.cat/	No	No
https://diariodeibiza.es/	No	No
http://diariodemallorca.es/	No	No
https://emporda.info/	No	No
https://farodevigo.es/	No	No
https://laopinioncoruna.es/	No	No
https://laopiniondemalaga.es/	No	No
https://laopiniondemurcia.es/	No	No
https://eldia.es/	No	No
https://laopiniondezamora.es/	No	No
https://laprovincia.es/	No	No
https://lne.es/	No	No
https://levante-emv.com/	No	No
https://mallorcazeitung.es/	No	No
https://regio7.cat/	No	No
https://superdeporte.es/	No	No
http://adelaidereview.com.au/	No	No
http://la977.com/	No	No
http://euroresidentes.com	No	No
https://prensaiberica360.es/	No	No
https://neomotor.com	No	No
https://generatepress.com	No	No
http://memoria.aitex.es/2018/	No	No
http://extranet.aitex.net/wps/portal	No	No
http://meteovalencia.avamet.org	No	No

http://electradelmaestrazgo.es/	No	No
https://ficlima.org/	No	No
http://altocumulo.com/	No	No
http://adene.es/	No	No
http://hotelprimusvalencia.com/	No	No
http://meteorage.fr/	No	No
http://cnavenidafrancia.com/	No	No
http://accioecologista-agro.org/	No	No
http://ecologialitoral.com/	No	No
http://pasionporvolar.com/	No	No
http://campingmariola.com/	No	No
http://cortesdepallas.es	No	No
http://villamalur.es/	No	No
http://vistabelladelmaestrat.es/	No	No
http://detoras.es/	No	No
http://castelldecastells.es/	No	No
http://portelldemorella.es/ca	No	No
http://lavalldelaguar.es/	No	No
http://lavalldebo.org/	No	No
http://escuelacosmofisica.com/	No	No
https://fincasanblas.com/	No	No