**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**

****

**Grado en Ingeniería Informática**

**TRABAJO FIN DE GRADO**

**Detección de Tuits de Odio.**

**Juan Carlos Pereira Kohatsu**

**Tutor: Álvaro Ortigosa Juárez**

**JUNIO 2017**

Tabla de contenido

[Índice de Figuras 2](#_Toc483383217)

[Índice deTablas 2](#_Toc483383218)

[Resumen. 3](#_Toc483383219)

[Palabras clave 4](#_Toc483383220)

[Abstract. 4](#_Toc483383221)

[Keywords 5](#_Toc483383222)

[1 Introducción. 6](#_Toc483383223)

[2 Estructura. 7](#_Toc483383224)

[3 Problemas de clasificación y desequilibrio de clases. 7](#_Toc483383225)

[3.1 Conjuntos de datos. 7](#_Toc483383226)

[3.2 Medida del rendimiento de un clasificador. 8](#_Toc483383227)

[3.3 El ciclo de Atención a un Tema y el vocabulario. 11](#_Toc483383228)

[3.4 Desequilibrio de clases,etiquetado y clasificación. 11](#_Toc483383229)

[3.5 Clasificación supervisada o no supervisada. 13](#_Toc483383230)

[4 Estado del arte 13](#_Toc483383231)

[4.1 Selección de instancias. 14](#_Toc483383232)

[4.2 Clases con probabilidades *a priori* no equilibradas. 15](#_Toc483383233)

[4.3 Etiquetado de instancias. 16](#_Toc483383234)

[4.4 Selección de atributos. 16](#_Toc483383235)

[4.5 Lematización. 17](#_Toc483383236)

[4.6 Clasificadores discretos y por puntuación (‘*scoring*’). 18](#_Toc483383237)

[4.7 Herramientas informáticas disponibles. 20](#_Toc483383238)

[4.7.1 Plataformas de código abierto. 20](#_Toc483383239)

[4.7.2 Plataformas comerciales. 20](#_Toc483383240)

[5 Descripción del proyecto. 20](#_Toc483383241)

[5.1 Especificaciones. 20](#_Toc483383242)

[5.2 Etapas. 20](#_Toc483383243)

[5.2.1 Identidad del tuit, 21](#_Toc483383244)

[5.2.2 Texto del *tuit* (‘documento’) 21](#_Toc483383245)

[6 Diseño. 21](#_Toc483383246)

[7 Desarrollo. 21](#_Toc483383247)

[8 Pruebas. 21](#_Toc483383248)

[9 Resultados. 21](#_Toc483383249)

[10 Utilización. 21](#_Toc483383250)

[Bibliografía 22](#_Toc483383251)

[Glosario 24](#_Toc483383252)

# Índice de Figuras

[Figura. 3‑2: ROC 10](#_Toc485508230)

[Figura 3‑3: Ciclo de Atención a un Tema 11](#_Toc485508231)

[Figura 3‑4: Errores de Clasificación en Muestras Desequilibradas 12](#_Toc485508232)

[Figura 3‑5: Filtrado del Conjunto Inicial 12](#_Toc485508233)

[Figura 3‑6: Clasificación tras Filtrado 13](#_Toc485508234)

[Figura 4‑1: Selección de Instancias 14](#_Toc485508235)

[Figura 4‑2: Procedimientos de Selección de Instancias 15](#_Toc485508236)

[Figura 4‑3: Método de Etiquetado no Supervisado(12. Automatic Text Categorization by Unsupervised Learning) 16](#_Toc485508237)

[Figura 4‑4: Selección de Atributos 17](#_Toc485508238)

[Figura 4‑5: Resultados de la prueba de V. Hollink et al. 18](#_Toc485508239)

[Figura 4‑6: Matrices de Confusión para Umbrales Distintos 18](#_Toc485508240)

[Figura 4‑7: ROC y Curva de Costo 19](#_Toc485508241)

[Figura 6‑2: Uso de conjunto de soporte al odio 23](#_Toc485508242)

[Figura 6‑3: Ejemplo de tabla de atributos 24](#_Toc485508243)

# Índice deTablas

[Tabla 3‑1: Matriz de Confusión 8](#_Toc485507802)

[Tabla 3‑2: Matriz de Confusión con Desequilibrio de Clases 9](#_Toc485507803)

[Tabla 3‑3: Métrica ROC frente a Métrica Convencional 10](#_Toc485507804)

[Tabla 3‑4: Matriz de Confusión del Conjunto Inicial 13](#_Toc485507805)

[Tabla 4‑1: Comparación de Umbrales 19](#_Toc485507806)

[Tabla 7‑1 Correspondecia de semática y símbolo 28](#_Toc485507807)

[Tabla 7‑2 Relación de sufijos y clases de palabras 29](#_Toc485507808)

[Tabla 8‑1: Comparativa de error de precisión frente profundida 34](#_Toc485507809)

[Tabla 8‑2: Comparación de exactitud con bigramas respecto a la profundidad 35](#_Toc485507810)

[Tabla 8‑3: Error de exactitud de KNN por vecinos 39](#_Toc485507811)

[Tabla 8‑4: Error de exactitud de KNN (distancia coseno) 40](#_Toc485507812)

# Resumen.

Las llamadas *redes sociales* constituidas por plataformas tales como **Facebook**™**, Twitter**™ que operan sobre Internet constituyen el soporte de los *medios (de comunicación) sociales* que facilitan el intercambio y la discusión de información, experiencias y opiniones entre individuos de manera rápida y masiva, nunca antes vista en la historia de la humanidad.

El abanico de medios sociales abiertos al uso público es variadísimo y creciente y sus usos son múltiples:

* artículos en *wikis,*
* opiniones sobre la calidad de hoteles, restaurantes ([tripadvisor](http://www.tripadvisor.es), [yelp](https://www.yelp.es)),
* contactos sociales y profesionales ([facebook](https://es-es.facebook.com/), [linkedin](https://es.linkedin.com/)),
* *blogs* o bitácoras *web* ([wordpress](https://es.wordpress.com)).

Ciertamente, como todo lo nuevo, la explosión de los medios sociales ha tenido consecuencias que han sido valoradas tanto positiva como negativamente para el conjunto de la sociedad.

Entre los efectos generalmente considerados como negativos, los medios sociales han hecho persistentemente *‘visibles’* algunas actitudes de ciertos grupos sociales que, hasta la fecha, solo se mostraban de una manera velada y/o esporádica. Entre ellas destacan las que se traducen en ataques a personas o colectivos en razón de su pertenencia a un determinados grupos definidos por características de nacionalidad, preferencias sexuales, raza, religión…

Este fenómeno junto con un cambio de actitud frente a ciertas conductas o grupos sociales ha motivado que, en muchos países, surja una nueva categoría delictiva: los llamados *delitos de odio* que, en España han sido regulados en 2015 mediante modificación del Código Penal, [1]

Desde el trabajo de Gary Becker [2] sobre crimen y castigo sabemos que los resultados que se derivan de los modelos indican que un *incremento en la probabilidad de sanción o arresto*, sin importar la disposición al riesgo del infractor, *tiene un efecto negativo sobre la oferta de delito*. Sin embargo, *el efecto de un incremento de la pena es indeterminado* o ambiguo sin más suposiciones y las suposiciones que se haga sobre la posición frente al riego son la clave del efecto de una mayor severidad sobre el crimen. Este es incierto para los amantes al riego, mientras para los adversos al riesgo, un incremento en la severidad de la pena reduce el delito.

De manera que los medios sociales:

1. si somos capaces de analizar masiva y automáticamente mensajes y detectar aquellos que puedan constituir delito de odio;
2. facilitarán enormemente la identificación de los infractores de las leyes y la obtención de pruebas.

Por lo tanto, como corolario [2] aumentará la probabilidad de sanción al infractor y, por consiguiente, disminuirá la frecuencia de este tipo de delitos.

Cabe, incluso, ir más allá y plantearse a futuro la aplicación de la *justicia maquinal* mediante la cual, el infractor es incluso *juzgado* por un sistema informático que utiliza las herramientas desarrolladas en el campo de la inteligencia artificial para determinar su inocencia o culpabilidad, al menos como fase previa a la iniciación de un proceso legal convencional.

Este problema de clasificación de tuits presenta un claro desequilibrio de clases, ya que la constituida por los tuits sospechosos de delito de odio, son muchos menos que los neutros [3]. Este tipo de problemas se denominan ‘*la aguja en el pajar*’.



Figura 0‑1: La Aguja en el Pajar

Palabras clave*:* redes sociales, medios sociales, grupos sociales, colectivos, etnia, raza, nacionalidad, religión, orientación sexual, odio, discriminación, violencia, delitos de odio, desequilibrio entre clases.

# Abstract.

The so-called “social networks” built-up by platforms such as **Facebook**™**, Twitter**™ which operate on the Internet underpin the *social media* that facilitate quick and mass exchange and discussion of information, experiences and opinions between individuals in a way never before seen in human history.

The range of social media available to the public is varied and growing and can be used for multiple purposes:

* articles in *wikis,*
* opinions about quality in hotels, restaurants ([tripadvisor](http://www.tripadvisor.es), [yelp](https://www.yelp.es)),
* social and professional contacts ([facebook](https://es-es.facebook.com/), [linkedin](https://es.linkedin.com/)),
* Web blogs ([wordpress](https://es.wordpress.com)).

Like anything new, the explosion of social media has had consequences that have been valued both positively and negatively for society as a whole.

Among the effects generally considered as negative, social media have persistently made 'visible' some attitudes of certain social groups that, to date were present only in a veiled and/or sporadic way. Prominent amongst them are those, which result in attacks on individuals or groups because of their affiliation to certain groups defined, by characteristics of nationality, sexual preferences, race, religion...

This phenomenon, coupled with a change of attitude towards certain social behaviors or groups, has led to the emergence of a new criminal category in many countries: the so-called *hate crimes.*

Since Gary Becker’s ‘*Crime and Punishment*‘ [2] we know that the punishment of criminals is probabilistic. The offender may escape detection or apprehension, or be apprehended but not convicted. His economic theory of crime states that some criminal justice variables are much more effective than others. Increasing *arrest rates*, followed by increasing the likelihood of *being convicted* have the largest impact. On the contrary, increasing the penalties beyond current levels has an uncertain effect on the crime rate. From this theory, it is clear that public authorities should focus more attention on strategies that increase the risk of arrest and less on strategies that increase the severity of punishment.

So, the social media

1. If we are capable of analyzing automatically mass messages in social media in so that we can detect those that can constitute a hate crime will enormously facilitate the identification of offenders and the collection of evidence.
2. Therefore, as a corollary, both the likelihood of being arrested and that of being convicted will raise for the infringer and, consequentially, the frequency of hate crimes will be reduced.

It is even possible to go further and consider in the future the application of *machine justice* by which the offender is tried by a computer system that uses tools developed in the field of artificial intelligence to determine their innocence or guilt, at least as a phase prior to the start of a conventional legal process.

This classification problem for tweets presents a clear imbalance of classes, since the one formed by messages suspected of hate crime, are much less than its complementary class [3].These types of problems are often called ‘the *needle in a haystack'.*

Keywords*:* Social networks, social media, ethnic groups, race, nationality, religion, sexual orientation, hate crime, discrimination, violence, imbalance of classes.

1

# Introducción.

Los *delitos de odio* son un tipo de infracción de la ley cuyo motivo principal es la existencia de *prejuicios* respecto a la víctima del mismo y tienen lugar cuando el perpetrador del delito elige a su víctima en base a su pertenencia a un cierto *grupo*.

Los atributos principales que definen el grupo de pertenencia de la víctima suelen ser el sexo, la etnicidad o raza, la nacionalidad, el idioma, la orientación sexual, la religión, la discapacidad, la apariencia física o la identidad de género, entre otros.

Existen evidencias de que tales delitos de odio están influidos por eventos *singulares* de *amplia difusión* [4] (atentados terroristas, migración incontrolada, manifestaciones, revueltas, …). Este tipo de sucesos suelen actuar como detonadores de manera que la frecuencia de este tipo de ilícitos aumenta espectacularmente tras ellos. Por ello, parece razonable dotar a los responsables de la seguridad pública de herramientas que permitan evaluar la probabilidad de tales delitos y, si es posible, su localización geográfica y temporal.

Los medios sociales de comunicación juegan un importante papel en la comisión de estos delitos en tanto en cuanto las redes se llenan de mensajes de individuos afines a los perpetradores que incitan a castigar al grupo elegido como diana que, recogidos a lo largo de un periodo temporal posterior al incidente detonante, pueden servir para analizar la evolución de la amenaza: escalada, estabilización, duración y descenso.

Tal es la importancia de estos medios que, en muchos países, se han tipificado recientemente[[1]](#footnote-1) también como pertenecientes a la categoría de delito de odio aquellas *manifestaciones públicas* que puedan considerarse una incitación al odio hacia ciertos colectivos.

Uno de los servicios más utilizados para realizar manifestaciones abiertas mediante la publicación de *microblogs* es Twitter™, motivo por el cual este servicio se ha seleccionado como fuente básica de datos para el desarrollo de un *Sistema para la Detección de Indicios de Delitos de Odio*.

Como en todo proyecto relacionado con la *Ciencia de los Datos*, es evidente que, antes de trabajar con datos es preciso capturarlos, lo que se hará mediante la utilización de la API de Twitter.

A continuación, se realiza un análisis exploratorio de datos que servirá de base para la depuración de los mismos, su formateo y modelización.

A partir de los datos depurados, se procederá a su análisis mediante técnicas de *Procesado de Lenguaje Natural* (NLP) para extraer patrones y atributos de los textos para, finalmente, clasificar los mensajes mediante técnicas de *Inteligencia Artificial*(AI) como positivos (que son indicio de una mayor *oferta* de delitos[[2]](#footnote-2)) o negativos/neutros que no aportan pistas al respecto.

El elemento clave para tal clasificación es el *contenido del mensaje* en e*l* que el redactor del mismo[[3]](#footnote-3) –en este caso del *tuit*- manifiesta su sentimiento u opinión respecto a una *entidad* o aspecto de la misma.

El caso que nos ocupa, se enmarca dentro de un grupo de problemas de clasificación binaria - dos clases: contenido de odio o neutro - caracterizado por un ***desequilibrio*** muy pronunciado entre el número de instancias en cada clase[[4]](#footnote-4), cuya proporción puede alcanzar valores próximos a 1:1000.

Este tipo de situaciones es de importancia en el mundo real en situaciones en que el coste de una clasificación errónea de las instancias de la clase *minoritaria* es muy elevado. Como ejemplos citaremos el diagnóstico de enfermedades o la detección de fraudes en tarjetas de crédito. En el primer caso, unos pocos píxeles del conjunto que constituye una imagen son la base del diagnóstico y en el segundo la proporción de fraudes sobre el total puede ser inferior al 1%. En el primer caso, un falso negativo puede incluso llevar a la muerte del paciente.

Esta situación de desequilibrio presenta tres problemas importantes para la clasificación de instancias:

1. Los mensajes de odio sobre un colectivo concreto *varían* a lo largo del tiempo *ligados a ciertos eventos* (atentado yihadista🡪mensajes antiislámicos, casos de corrupción🡪mensajes antipartidistas, premios Goya🡪cine español, …) lo que hace que los atributos relevantes para la clasificación sean variables y deban revisarse continuamente.
2. Se dificulta la aplicación de *métodos de clasificación supervisada* ya que para el etiquetado manual de unos cientos de casos, se requiere el examen de cientos de miles de tuits lo que alarga y encarece el etiquetado.
3. Por otro lado, el desequilibrio entre las clases provoca que el algoritmo que entrenamos sobre un conjunto de con muy pocas instancias de la clase minoritaria sea, con frecuencia, incapaz de generalizar el comportamiento de esta clase y, por tanto, puede tener una escasa capacidad predictiva u una baja exhaustividad.

Estos problemas, entre otros, se tratan en este trabajo.

## Estructura.

La memoria explicativa del proyecto se estructura de la siguiente manera:

Comenzamos revisando algunos *conceptos básicos* referidos a la clasificación en conjuntos con desequilibrio y revisaremos el *estado del arte* al respecto así como las herramientas existentes para manejar el proyecto.

Seguidamente, en el apartado *descripción del proyecto* se detallan aspectos fundamentales del mismo tales como

* + - Especificación del producto,
    - Herramientas utilizadas
    - Metodología de gestión de proyectos utilizada
    - Fuentes de datos,
    - Software y hardware,
    - Control de versiones

A continuación, se expone el *diseño del proyecto* y tras analizar los resultados de la fase de *investigación* se detallan las características del *sistema de explotación* así como las *instrucciones* para su uso disponibles en un *wiki* en Github.

## Problemas de clasificación y desequilibrio de clases.

## Conjuntos de datos.

Para el desarrollo de un clasificador partimos de un conjunto de instancias (*mensajes o textos*) que forman un *corpus*. Lo ideal es recoger *varios conjuntos de datos independientes*, si ello no es posible, debemos conformarnos con un solo conjunto de datos que habremos de dividir en dos o tres subconjuntos.



Figura 1‑1: División del Conjunto de Datos

La estrategia consiste en usar

1. un *conjunto de* ***entrenamiento*** para aprender y estimar los parámetros del modelo;
2. un *conjunto de* ***validación*** para evaluar modelos y seleccionar uno de ellos y
3. un *conjunto de* ***prueba*** o test para valorar la capacidad de predicción de los modelos.

Existen múltiples métodos de división de datos, de los cuales el más simple es el *método de retención* (*holdout*) que consiste en dividir aleatoriamente el conjunto original en dos subconjuntos ( o para entrenamiento y el resto para prueba). Si el conjunto original no es lo bastante grande, el método es ineficiente.

En *aprendizaje estadístico* se utiliza como supuesto básico que *tanto el conjunto de entrenamiento como el de prueba se extraen de una misma distribución subyacente* constituida por la combinación de las distribuciones de la clase mayoritaria y minoritaria. Si se llevan a cabo modificaciones en el conjunto de entrenamiento para tratar de reequilibrar las clases, el *conjunto de entrenamiento y el de prueba tendrán distribuciones diferentes* violando este supuesto.

## Medida del rendimiento de un clasificador.

Para la evaluación de clasificadores es preciso utilizar alguna *métrica* que nos permita estimar su *rendimiento*.

La *matriz de confusión*, elemento básico de evaluación de clasificadores, se expresa en el caso de que solo existan dos clases como:



Tabla 1‑1: Matriz de Confusión

Las medidas más inmediatas de evaluación del modelo que se nos ocurren son:

* *Exactitud*:
* *Tasa de error*:

Sin embargo, cuando *existe un claro desequilibrio entre clases*, como es el caso de los tuits de odio que pueden ser un 2‰ del total, es posible obtener una *exactitud enorme* aun clasificando todos los tuits (erróneamente) como negativos (no de odio), y *equivocándonos*, por tanto, *en todos los positivos*:



Tabla 1‑2: Matriz de Confusión con Desequilibrio de Clases

Lo que nos daría unos valores de exactitud del 99,8%:



Por ello, son más útiles los indicadores:

1. ***Precisión***(*p*recision)
2. ***Exhaustividad***(*r*ecall)

* *Precisión* **p** es el porcentaje de los *tuits* realmente pertenecientes a una clase que se asignan a la misma(aciertos) sobre el total de los asignados a dicha clase por el clasificador:

Es decir, el porcentaje de predicciones que se acierta.

* *Exhaustividad*(o proporción de verdaderos positivos ***tp*** ) **r** es el porcentaje de los *tuits* que han sido clasificados como pertenecientes una clase sobre el total de miembros de dicha clase en el conjunto (porcentaje de instancias de la clase bien clasificadas):

Se utilizan también combinaciones de p y r, tales como la media geométrica:

Y la media armónica:

|  |  |
| --- | --- |
|  | Ec. ( 1‑1) |

Un valor p=1 nos dice que todos los elementos recuperados como relevantes, lo son, pero no nos dice nada acerca de si hemos recuperado todos los documentos relevantes.

Una cuestión importante es cómo *equilibrar los falsos positivos y los falsos negativos* cuestión que, evidentemente, depende del *coste de las consecuencias de cada error*.

Para esto puede utilizarse una herramienta que se está usando cada vez más: el ***ROC[[5]](#footnote-5)*** [5]habitual en Medicina y Biología donde las consecuencias de la detección de falsos positivos y negativos puede tener costes muy diferentes.

## El ciclo de Atención a un Tema y el vocabulario.

Como se ha mencionado anteriormente, los delitos de odio tienden a ser más frecuentes y a crecer en periodos de tiempo posteriores a un suceso antecedente (‘detonador’) [4] y el interés del público sobre un asunto determinado sigue el llamado *Ciclo de Atención a un Tema* que fue descrito inicialmente por Downs [6] y que se muestra en la Figura 1‑2.



Figura 1‑2: Ciclo de Atención a un Tema

Pueden verse las diferentes etapas por las que pasa la relevancia del tema para el público a lo largo de un periodo de tiempo que, además no suele ser muy largo.

Este punto es relevante por cuanto afecta tanto a la recolección de tuits como a los términos – atributos – a utilizar que se encuentran relacionados.

Sobre el primer punto, si se desea el seguimiento de un tema concreto una vez ocurrido un suceso (p.e. aparición de un nuevo caso de corrupción), hemos de recoger los tuits en la cresta de la ola y, además, hemos de tener en cuenta que, dependiendo de los asuntos que sean más *trendy* el vocabulario que se usa para su comentario es diferente.

## Desequilibrio de clases, etiquetado y clasificación.

En aprendizaje supervisado, tenemos un conjunto – muestra – de datos S sobre el cual queremos construir un modelo de clasificación binario. La primera suposición que haremos es que las instancias del conjunto positivas y negativas observadas (S+ y S-) se extraen de dos distribuciones diferentes P y G. Las instancias positivas son las minoritarias. En estas circunstancias es fácil ver que, por una parte,

1. el etiquetado manual de las instancias resulta engorroso puesto que para encontrar un tuit – *la aguja* – de la clase minoritaria se precisa examinar una cantidad enorme de tuits de la otra clase – *la paja*; y, por otra,
2. se explica el motivo por el cual un modelo clasificador sobre el conjunto produce una baja exhaustividad: la distribución positiva está subrepresentada y los valores atípicos de G – de mucha mayor cardinalidad – aunque sean una pequeña fracción, influirán en el clasificador ya que se considerarán por este como pertenecientes a S+ y el clasificador inducido estará sesgado hacia la clase minoritaria, es decir más cercano a los puntos de esta de lo que debiera, produciendo un rendimiento bajo del clasificador (Figura 1‑3 (A)).

La Figura 1‑3 (B) muestra otro ejemplo cuando se usa 1-NN como clasificador en un conjunto desequilibrado. Lo mismo puede decirse de la clasificación bayesiana.



Figura 1‑3: Errores de Clasificación en Muestras Desequilibradas

Debido a la dificultad de etiquetado manual, hemos utilizado un procedimiento que filtra el conjunto inicial extraído de Twitter mediante el uso de un vocabulario con términos de odio obtenidos de diferentes fuentes y el conjunto filtrado, de una cardinalidad mucho menor, se *etiqueta* y usa como conjunto de *entrenamiento* (Figura 1‑4).



Figura 1‑4: Filtrado del Conjunto Inicial

Seguidamente, el conjunto de entrenamiento se usa para estimar los parámetros del clasificador y, con este, se clasifican los tuits del conjunto de test.

Para la utilización del clasificador a otros conjuntos de tuits, caben dos opciones:

1. *Aplicarlo tras filtrar* el nuevo conjunto, asignando a todos los tuits que no pasan el filtro a *C-.*
2. *Aplicarlo al conjunto completo* con lo cual cabe la posibilidad de que algunos de los tuits que no pasan el filtro, se clasifiquen en *C+* (correcta o incorrectamente).

En el primer caso (Figura 1‑5), la *matriz de confusión* muestra como FN a las instancias que no pasan el filtro que debieran estar etiquetadas con (+), una en nuestro ejemplo.



Tabla 1‑3: Matriz de Confusión del Conjunto Inicial



Figura 1‑5: Clasificación tras Filtrado

Esta clasificación tras filtrado implica necesariamente (

El segundo caso es más complejo y, como veremos más adelante (§4.2), requiere un *recalibrado* de las probabilidades del modelo entrenado.

## Clasificación supervisada o no supervisada.

Hemos dado por sentado que utilizaremos métodos de clasificación supervisados sin haber justificado los motivos para el descarte de los no supervisados.

Podemos plantearnos una clasificación no supervisada basada en análisis de conglomerados (*cluster*) [7] p.e.

En este caso, nos enfrentamos a un problema de clasificación con clases *desequilibradas* y una de ellas debe ser la clase de *tuit de odio.*

Resulta imposible que, utilizando el procedimiento algorítmico k-medias con *selección aleatoria de centroides iniciales* se clasifiquen las instancias en las clases de odio y neutras. Aumentaría esta posibilidad si asignásemos previamente uno de los centroides iniciales con unos valores aproximados al verdadero de la clase de odio, pero esto requeriría un *etiquetado previo* de las instancias que, en definitiva, es lo que hacemos directamente.

En cambio, cabría considerar la utilización del *clustering* tras el filtrado inicial para tratar de descubrir nuevas instancias que, de nuevo, serían etiquetadas a mano para aplicar finalmente un clasificador supervisado, proceso en cierto modo similar al de ampliar vocabulario a partir de los tuits de odio que usamos.

2

# Estado del arte

Se trata de un trabajo que persigue la *clasificación* de una colección de tuits en las categorías:

* De odio y
* Neutra

Mediante procedimientos de clasificación *supervisada* de Aprendizaje Máquina, para lo cual debe procederse a un etiquetado previo de un conjunto de entrenamiento.

El proyecto se mueve, por una parte, en el terreno problemático de los conjuntos no equilibrados, tema sobre el que existe una abundante literatura que se refiere fundamentalmente a:

* cómo simplificar el etiquetado de las instancias y
* cómo paliar los efectos de la asimetría entre clases.

Por otro lado, el proyecto de clasificación de tuits utiliza herramientas del campo del *Procesamiento del Lenguaje Natural*(NLP),rama de la Inteligencia Artificial que tiene sus orígenes en los años 50 del pasado siglo cuando aparecieron las primeras computadoras. Su desarrollo ha venido acompasado a los avances en la capacidad de cómputo (*Ley de Moore*) y en los algoritmos de *aprendizaje estadístico.*

Por último, conviene pasar revista a las *herramientas informáticas* que facilitan el tratamiento de este tipo de problemas.

## Detección de lenguaje insultante (*abusive language*).

Los intentos de aplicación de la inteligencia artificial a la detección de lenguaje de odio, son relativamente recientes y pueden seguirse en [8].

Una interesante conclusión que se repite es relacionar el mensaje con la fuente de manera que se detecten más mensajes de odio. Un medio de hacer esto es a través de la propia denuncia de los usuarios de la red

## Selección de instancias.

Como hemos dicho, el *corpus* de tuits es un conjunto en el que la clase de tuits de odio es muy minoritaria respecto a la de tuits neutros. Este problema – junto con el de selección de instancias y desequilibrio de clases – ha sido estudiado desde muchos puntos de vista y T. Borovicka et al. [8] proporciona un resumen de los procedimientos desarrollados para seleccionar conjuntos de entrenamiento *equilibrado* en los casos en que existen grandes desequilibrios entre las clases de clasificación que básicamente son de tres tipos:

1. A nivel de datos
2. A nivel de algoritmo y
3. Conjuntos

Los primeros se basan en el sobremuestreo o submuestreo de la clase minoritaria o mayoritaria. Los segundos en ponderar de modo diferente la importancia de la instancia en función de la clase a que pertenezca y el tercero utiliza una combinación de métodos.

La fase inicial de *selección de instancias* [9]es un proceso de reducir el conjunto de datos original.

La salida ideal de la selección de instancias es una muestra mínima independiente del modelo que pueda cumplir su objetivo con el menor deterioro posible, es decir, que el rendimiento P de un modelo M sea aproximadamente el mismo sobre la muestra S que sobre la población W.

P(Ms)≈P(Mw)(Ecuación 1)

El método clásico de obtención de muestras se basa en las técnicas de muestreo (aleatorio simple, estratificado, adaptativo,…)

Con frecuencia, la muestra puede reducirse para generar un *conjunto de entrenamiento* más manejable[10].



Figura 4‑1: Selección de Instancias

Esto puede hacerse utilizando algún tipo de algoritmo de *selección* de instancias bien relacionado con

1. el rendimiento de algún algoritmo de clasificación (*wrapper methods*) o con
2. el vector de atributos de la instanciación independencia del algoritmo utilizado (*filter methods*)



Figura 4‑2: Procedimientos de Selección de Instancias

Los enfoques del primer tipo (*wrapper*) subrayan el aspecto de minería de datos del problema ejecutando un algoritmo específico de tal campo para disparar la selección de instancias, p.e. seleccionando un subconjunto inicial, ejecutando un algoritmo sobre este subconjunto inicial, evaluando sus resultados y ampliando incrementalmente el subconjunto inicial hasta que los resultados del algoritmo sean lo bastante buenos.

Los enfoques del tipo filtro, son más simples e independientes del algoritmo de clasificación.

## Clases con probabilidades *a priori* no equilibradas.

Hemos mencionado en 3.3 tanto la dificultad de etiquetar conjuntos con gran desequilibrio de clases como la de entrenar algoritmos en conjuntos de entrenamiento ya que ello conlleva un elevado *nivel de ruido* y un pobre rendimiento del clasificador.

P.e. si estamos realizando un experimento para el diagnóstico de alguna enfermedad cuya tasa de prevalencia es del 1 ‰ no elegiremos un conjunto de entrenamiento de 1 enfermo y 999 individuos sanos, sino un conjunto 50%-50%, donde los sanos constituyen el grupo de control y los enfermos el experimental, lo que modifica las probabilidades a priori de cada clase en el conjunto de entrenamiento ye introduce sesgo en el clasificador.

En [8] se hace referencia a los siguientes grupos de métodos para vencer este problema:

1. Métodos de *nivel de datos*. Se utilizan en el preprocesado y se basan en varios tipos de *remuestreo*. Buscan aumentar el número de instancias de la clase minoritaria (sobremuestreo) y/o reducir los de la clase mayoritaria (submuestreo).
2. Métodos de *nivel de algoritmo*. Se basan sobre todo en dar una sobreponderación a la clase mayoritaria.
3. Métodos *conjuntos* que usan una combinación de métodos.

Todos ellos generan conjuntos de entrenamiento equilibrados sobre los que se entrenan algoritmos.

Resulta evidente que al reequilibrar artificialmente el conjunto de entrenamiento, las distribuciones en el conjunto de entrenamiento y en el de prueba son diferentes, de manera que se viola una de las hipótesis básicas de aprendizaje estadístico: que *tanto el conjunto de entrenamiento como el de prueba siguen la misma distribución*.

Esto motiva que la aplicación de un modelo entrenado sobre un conjunto de entrenamiento reequilibrado por el analista a un conjunto donde esto no se ha llevado a cabo (prueba) exige calibrar las probabilidades obtenidas del conjunto de entrenamiento retocado para poder aplicar el modelo recalibrado directamente al conjunto de test (no modificado).

En el artículo [11] se analiza este problema y se desarrolla un método para corregir el sesgo introducido.

## Etiquetado de instancias.

Un serio problema que afecta a los conjuntos desequilibrados, es el del *etiquetado del conjunto* de entrenamiento (3.3). Si bien resulta relativamente fácil recopilar cientos de miles de *tuits no etiquetados*, no es tan fácil proceder a su clasificación manual, especialmente cuando, como en este caso, la clase de interés es muy minoritaria.

Se han ensayado algunos métodos para vencer esta dificultad como el descrito en [12] que, a pesar de ser un método *no supervisado* está relacionado con la selección de instancias y ha servido de guía a nuestro proyecto y cuyo esquema es el siguiente (Figura 10):



Figura 4‑3: Método de Etiquetado no Supervisado(12. Automatic Text Categorization by Unsupervised Learning)

## Selección de atributos.

Una de las etapas más importantes en clasificación es la *selección de atributos*.



Figura 4‑4: Selección de Atributos

Yan y Pedersen [13] hacen un estudio comparativo de diferentes procedimientos para la selección de atributos en clasificación de textos comenzando por el más simple: umbral de frecuencia documental en el que se eliminan aquellos términos (*atributos*) que aparecen muy raras veces en los documentos de una clase. Además, es de fácil comprensión y utilización el método de la *Chi cuadrado* (χ2) que usa tablas de contingencia que muestran la frecuencia con que aparece cada atributo en cada clase etiquetada y contrasta la hipótesis de independencia entre *término* y *clase* utilizando la f.de D. Χ2.

## Lematización.

En los lenguajes donde es frecuente el uso de formas flexionadas de un mismo *lexema*(es decir, en plural, en femenino, conjugada, etc) la *lematización* consiste en hallar el *lema* correspondiente a una palabra, es decir, la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. Hallar el ***lema*** correspondiente es equivalente a una ***entrada de diccionario***.

De esta manera múltiples representaciones y tiempos de una palabra (‘*ataco*’,’*atacante*’,’*atacado*’…) pueden reducirse a un solo *token*- ***atac[[6]](#footnote-6)***- y, por tanto, a un solo atributo.

Una buena lematización requiere un conocimiento del contexto en que se usa la palabra, para lo cual es necesario realizar el *etiquetado gramatical* (*POS Tagging*) del texto. No obstante, es más frecuente, sencillo y rápido la utilización de una forma tosca de lematización llamada *stemming* que prescinde del etiquetado gramatical y que, entre otras aplicaciones, se ha utilizado en Google Search desde 2003 [14].

De los algoritmos de *stemming* de uso libre, el más conocido es el de Porter [15] que se encuentra incluido dentro de *Snowball* [16]que es un software para procesamiento de cadenas de caracteres orientado a la creación de algoritmos de *stemming[[7]](#footnote-7)*.

Se han llevado a cabo diferentes ensayos de evaluación de la eficacia de este tipo de algoritmos en distintos idiomas. Hollink et al. [17] manifiestan que las técnicas básicas pero ciertos métodos auxiliares (palabras vacías, *stemmers*,..) deben adaptarse al idioma que se esté utilizando. Básicamente comparan la eficacia de la recuperación de información sobre textos sin *stemmizar* o *stemmizados* para siete idiomas europeos (alemán, español, finlandés, francés, holandés, inglés y sueco).

La conclusión fue que la mejora en español al usar *stemming* fue significativa (Figura 12).



Figura 4‑5: Resultados de la prueba de V. Hollink et al.

## Clasificadores y costes.

Ya hemos manifestado las razones que hacen imprescindible la utilización de métodos supervisados de clasificación, ahora examinaremos algunas características de estos relacionadas con la *diferencia de costes* en *falsos positivos* y *falsos negativos*.

Un clasificador mapea las *instancias* en *clases.* Ello puede llevarse a cabo con

dos tipos de clasificadores según el *tipo de salida* que proporcionan:

1. *discreta*: proporcionan solo una etiqueta que indica la clase predicha para la instancia (p.e. árboles de decisión) y
2. *continua*: estimación de la *probabilidad* (entre 0 y 1) de pertenecer a una clase o un *puntaje*. La asignación a una u otra clase se hace depender de un ***umbral*** de corte para dicho puntaje. Los métodos *bayesianos* y *redes neuronales* pertenecen a este segundo tipo.

Un clasificador *discreto* proporciona solo *un punto* en el diagrama *ROC* y *una sola matriz de confusión,* mientras que otro que funcione *por puntuación* proporcionará un *punto* y una *matriz de confusión* diferente *para cada* ***umbral***.



Figura 4‑6: Matrices de Confusión para Umbrales Distintos

En efecto, si llamamos tendremos que

|  |  |
| --- | --- |
|  | Ec. ( 2‑1) |
|  | Ec. (2‑2) |

Que son las ecuaciones paramétricas de la curva ROC al variar los valores del umbral.

Entre Ec. (2) y Ec. (3) podemos eliminar θ y obtenemos la ecuación de la *ROC*:

Si planteamos lo anterior en términos de *contraste de hipótesis* en que *contrastamos que una instancia pertenece a la clase* ***C+***, *tp* sería el error tipo I y *fp* el error tipo II.

Se nos plantea el problema de seleccionar el *umbral de clasificación* más adecuado.

1. Si no existen *costes diferentes de una clasificación errónea* para falsos positivos y falsos negativos, puede enfocarse bien maximizando *F* [Ec. (1)] o utilizando el ROC de manera que la *AUC sea lo mayor posible*.
2. Si los costes son diferentes y conocidos, cabe construir una curva de costes a partir de la ROC y seleccionar el umbral de coste mínimo (Figura 14).



Figura 4‑7: ROC y Curva de Costo

Para ello tendremos en cuenta que

No obstante, es difícil establecer una relación de costes explícita y, por ello, una alternativa es examinar las *matrices de confusión de umbrales diferentes* y seleccionar entre ellos aquel que más nos interese.

P.e. entre los resultados de un clasificador con dos umbrales distintos y matrices de confusión mostradas en la figura con sus métricas:



Tabla 2‑1: Comparación de Umbrales

Es probable que si el usuario da más importancia a la exhaustividad (recuperar el máximo de elementos de la clase +) que a la precisión elegirá el umbral *b* con mayor *r*. Como se ve esto provoca un aumento de los falsos positivos que, por consiguiente, tienen un coste menor que los falsos negativos.

## Herramientas informáticas disponibles.

Para la ejecución del proyecto, nos interesan las herramientas para minería de datos disponibles para el analista. Tales herramientas se refieren tanto al campo del *PNL* propiamente dicho como al de los *algoritmos* de clasificación y recuperación de información que son necesarios para clasificar los tuits.

A continuación, mencionamos algunas de las herramientas y plataformas existentes que se encuentran en permanente evolución.

### Plataformas de código abierto.

* Basadas en Python:
  + Natural Language Tool Kit (NLTK) [18]
  + gensim [19]
  + Scikit-learn [20]
  + TensorFlow [21]
  + Pattern
  + Keras
* Basadas en Java:
  + CoreNLP [22]
  + MALLET [23]programas para modelizado temático☹ y clasificación de textos.
  + LingPipe [24]
  + Weka [25]
  + yTextMiner [26] plataforma desarrollada en la universidad de Yonsei, Corea que integra parte de los modelos anteriores con otras librerías tanto para modelización temática como clasificación de textos y análisis de sentimiento.
  + Apache Lucene [27].

### Plataformas comerciales.

Probablemente la más interesante es la que forman los diferentes paquetes del Sistema *SAS™[[8]](#footnote-8)*, más concretamente:

* *SAS™ Enterprise Miner*: paquete de aprendizaje automático y minería de datos enfocado a *business analytics*
* *SAS™Text Miner*: utiliza la técnica SVD (la misma que gensim) para simplificar y acelerar la clasificación de textos y extracción de información.

3

# Diseño.

El diseño del proyecto se basa en dos partes:

Parte I: *Investigación y diseño* *de un clasificador* de tuits de odio.

Parte II: *Desarrollo de un clasificador* de tuits de odio utilizable por el usuario.

## Parte I:

Para llevar acabar el desarrollo de esta primera parte, se ha dividido esta en pequeños módulos que unidos proporcionan la funcionalidad necesaria de esta parte:

* Obtención de tuits.
* Limpieza de tuits.
* Extracción de información de los tuits.
* Selección de tuits.
* Etiquetado de tuits.
* Selección de atributos.
* Clasificación supervisada.
* Validación de los resultados.

### Obtención de tuits

Para realizar este apartado es necesario idear un plan de captura de tuits de tal forma que nuestras peticiones siempre sean aceptadas por los servidores de Twitter. Por ello se propone la creación de varias cuentas de Twitter para no exceder el número de peticiones a la API Rest de Twitter. Como requisito, los tuits a ser descargados tienen que proceder de España y estar en español. Cada tuit se compondrá de: identificador, autor, fecha y texto. A lo largo de las demás etapas se trabaja con un *corpus* de aproximadamente 1.000.000 de tuits obtenidos en la etapa actual.

### Limpieza de tuits

Se deberá de tener en cuenta el formato Unicode del texto contenido en los tuits por lo que se pueden encontrar caracteres extraños en los tuits tales como: *kanjis*, eñes, tildes… Estos caracteres al igual que los *emojis* deben de ser eliminados ya que se considera que la extracción de información de los mismos es complicada. También se lleva a cabo la eliminación de los signos de puntuación: puntos, comas, puntos y comas. Todos los caracteres a ser borrados se consideran *stopwords.* También se realiza la eliminación de más de 3 duplicados de un carácter consecutivo convirtiéndolo en solo 1, al no haber ninguna palabra en español con 3 caracteres consecutivos iguales, se considera que el usuario que ha escrito el tuit muestra énfasis en lo que quiere expresar. Finalmente se transforma todo el tuit a caracteres en minúsculas.

### Extracción de información de los tuits.

Una vez que se han limpiado los tuits se extraerán de los mismos tipos de palabras relevantes en el español, es decir, tipos de palabras que transmitan información acerca del contenido semántico del tuit. Para realizar esto se utiliza un analizador morfológico, también conocido como *POS Tagger*. Este analizador morfológico permite diferenciar entre diferentes tipos de palabras: verbos, nombres, preposiciones, adverbios… En este caso se usan verbos, nombres y adjetivos ya que este tipo de palabras proporcionan la mayor parte de la semántica de un texto en español. También se introducen en los tuits nuevos términos que contienen significado que posteriormente puedan ser de utilidad. Por ejemplo: “@usuario” se sustituye por USER (se menciona a un usuario), “#tema” se sustituye por HASHTAG (indicando un tema de Twitter), “ y “ se sustituyen por TOKCOMI… De alguna forma estos nuevos símbolos permiten obtener información sobre: la mención a un usuario, mención de una temática o la presencia de comillas que pueden mostrar si es una cita textual o si hay cierta ironía en la palabra entrecomillada.



Figura 3‑1:Tuit una vez que se ha extraído la información.

En la Figura 3‑1 se puede observar el resultado de extracción de palabras relevantes de un tuit. Las palabras eliminadas se muestran cruzadas por una línea horizontal y los términos en cajas rojas muestras términos a transformarse en símbolos que contienen significado concreto.

### Selección de tuits.

Como se menciona en la introducción el problema tratado se puede comparar a buscar una aguja en un pajar por lo que no se pueden seleccionar 1.000.000 de tuits con los que trabajar, esto se debe a que es probable que únicamente se encuentren 1.000 tuits de odio entre el millón y además este trabajo solo se podría realizar mediante un etiquetado manual que en tales dimensiones es imposible llevarse a cabo. Por ello la selección de tuits se realiza mediante el uso de un filtro basado en términos que contengan connotación negativa.

Solo pasarán por el filtro aquellos tuits que contengan términos considerados de odio por el filtro. Consideramos este un buen método para equilibrar en cierta medida la proporción de tuits de odio y no odio.

El diseño del filtro está basando en conjuntos de palabras que contengan odio en distintas temáticas: raza, género, religión… Sin embargo, se debe tener en cuenta que ciertas palabras pueden resultas ambiguas en el momento de decir si la palabra contiene odio o no p.e. negro, comunista, facha… Por lo tanto, se necesita aportar una gradualidad de odio a los términos y para ello se considera necesario un conjunto de palabras aparte de “soporte al odio”. Este conjunto contiene adjetivos peyorativos y despectivos genéricos, es decir, pueden acompañar a cualquier palabra de los temas mencionados anteriormente.



Figura 3‑2:Uso de conjunto de soporte al odio

En la Figura 3‑2 se muestra como se utilizaría el conjunto de términos de soporte al odio. Claramente, términos como “feminazi” y “maricona” ambos de la categoría “sexual/genero” se consideran ofensivas por sí mismas por lo que podemos decir que el tuit que contiene dichos términos puede pasar por el filtro. Sin embargo, los términos “moro” y “negro” no siempre contienen odio p.e. “Hoy he tenido un día muy negro” o “Los moros huyeron de la península”. Por lo tanto, en ese caso es necesario el uso de un desambiguador, el conjunto de soporte al odio. Si un término de soporte al odio se encuentra en el mismo tuit que un término ambiguo respecto al odio, entonces consideramos que el tuit puede pasar el filtro.

### Etiquetado de tuits

Durante esta etapa será necesario etiquetar a mano los tuits que han pasado el filtro mencionado anteriormente. Estos tuits etiquetados serán utilizados posteriormente como conjunto de clasificación para realizar las pruebas necesarias con el objetivo de desarrollar un buen clasificador. Es necesario tener en cuenta que todos aquellos tuits que no pasen por el filtro se considerarán que pertenecen a la clase de no odio. Para realizar el etiquetamiento correcto de los tuits es necesario que sea una persona externa al desarrollo del clasificador quien realice esta actividad para que las pruebas y los resultados obtenidos sean objetivos. Otra cosa importante a tener en cuenta es que el etiquetado se verá influenciado por la ideología y el estado del ánimo del etiquetador ya que la interpretación de odio y no odio puede tener cierto grado de subjetividad. Para resolver este tipo de problemas lo ideal es que haya más de un etiquetador y que por votación decidan a que clase pertenece un tuit, por lo general se suele escoger un número impar de etiquetadores para que no haya empates de decisiones.

### Selección de atributos

Hasta este punto del proyecto ya contaríamos con un conjunto limpio de tuits que han pasado el filtro y están correctamente etiquetados. Sin embargo, únicamente tendríamos texto plano y este es difícil de procesar por clasificadores supervisados. Por lo que se propone representar los tuits con el modelo *BOW* (saco de palabras). Este modelo permite obtener características o atributos de los tuits más fácilmente. Dichos atributos aportarán la información necesaria para que el clasificador permita determinar si un tuit contiene odio o no.

Se proponen las siguientes variables como atributos:

* + - Aparición de un término en un tuit (1 si está, 0 si no está).
    - Frecuencia de un término en un tuit.
    - *TF-IDF*



Figura 3‑3: Ejemplo de tabla de atributos

En la Figura 3‑3 se muestra la transformación de un tuit ya limpio a su *patrón* a clasificar usando una tabla de atributos de 5 términos. En el ejemplo se utiliza el modelo 1 o 0 (está o no el término en el tuit).

Las anteriores variables explicadas están enfocadas al uso de un solo término, pero se podrían usar haciendo agrupaciones de n términos consecutivos (*n-gramas*). En este caso se propone realizar pruebas con bigramas o digramas (2-gramas), para poder realizar comparaciones con los unigramas. El uso de bigramas y unigramas no son exclusivos por lo que ambos pueden encontrarse en la misma tabla de atributos.

Una vez se han seleccionado el modelo de atributos es conveniente utilizar un selector de atributos relevantes. Puede que la tabla de atributos que hemos generado anteriormente contenga atributos no relevantes a la hora de decidir si un tuit pertenece a la clase de odio o no odio. Para realizar esta tarea se utiliza el test de *Chi cuadrado* (χ2) que permite determinar la independencia de variables y la clase.

### Clasificación supervisada

Llegados a este apartado, ya se tiene construido el conjunto de clasificación formado por *patrones*, donde cada *patrón* representa un tuit, etiquetados previamente a mano con su clase correspondiente (odio o no odio) y representado por medio de atributos relevantes que aportan información sobre la clase a la que pertenecen. Por lo tanto, se puede iniciar la fase de clasificación supervisada. Para ello es necesario seleccionar el tipo de particionamiento y las características del mismo. Para llevar acabo los experimentos se usa *particionamiento cruzado* y *estratificado* para mantener la proporción de *patrones* con clase de odio y no odio en cada una de las particiones. A continuación, se procedería la selección de clasificadores, los utilizados son:

* + Clasificador *vecinos próximos* o *KNN*
  + Clasificador *Naive Bayes*
  + Clasificador basado en *redes neuronales*
  + Clasificador basado en *árboles de decisión*

Durante la fase de desarrollo habría que tener en cuenta las diferentes variantes que tiene cada clasificador como sus métricas y los parámetros que hay que ajustar para cada uno de ellos. Para verificar lo “bueno” que es un clasificador se tendrán en cuenta la *Tasa de error* y la *desviación estándar* de la misma y se llevará a cabo un análisis de las matrices de confusión generadas. Dentro de esta misma fase se describen las comparaciones de los clasificadores.

### Validación de los resultados.

Para realizar esta base se seleccionan los mejores clasificadores de la fase anterior, ya se ha realizado un análisis de los resultados obtenidos con el conjunto de clasificación. También se fijan las mejores métricas, estructuras y parámetros basándonos en el análisis realizado previamente.

Para evitar el sobreajuste de los clasificadores ya sea por: parámetros, métricas, estructuras o incluso la propia selección de atributos, se utilizarán nuevos tuits para realizar la validación, son tuits que no se encuentran dentro del millón utilizado para desarrollar los experimentos. Por lo tanto, es necesaria la obtención de nuevos tuits, en este caso se utilizarán 100.000 tuits aleatorios, en español y de España, es decir, capturados en las mismas condiciones que el millón utilizado previamente para experimentos. La fase de validación permite probar los clasificadores como si de una situación real se tratara, es decir, en esta fase se determina si un clasificador es útil o no.

## Parte II

En esta parte se lleva acabo el desarrollo del clasificador que será usado directamente por el usuario. Por lo tanto, se deben especificar requisitos que debe de cumplir el clasificador:

### Requisitos funcionales

RF1 Entrenamiento del clasificador.

Un usuario debe entrenar previamente el clasificador antes de su uso.

RF2 Clasificación de tuits.

Un usuario puede clasificar un tuit introduciendo el texto contenido del tuit en el clasificador. El clasificador devuelve 1 si el tuit contiene odio hacia un grupo de personas o una persona en particular. En caso de no haber odio, el clasificador devuelve 0.

RF3 Clasificación de tuits.

Un usuario puede seleccionar usar el prefiltrado, utilizando el mismo filtro que se usaron para obtener el conjunto de clasificación, antes de realizar la clasificación del tuit deseado.

### Requisitos no funcionales

Clase rendimiento

RNF1: Se trata de un clasificador que funciona en modo online (en el aire) por lo que la respuesta del clasificador no puede ser muy lenta.

RNF2:El clasificador debe de tener una *exactitud* superior a la del clasificador a priori, en este caso el que indica que todos los tuits no son de odio.

Clase usabilidad

RNF3: La utilización del clasificador debe de ser sencilla y muy intuitiva para el usuario.

RNF4: El clasificador contendrá un manual de usuario donde se especifica su funcionamiento.

Clase portabilidad

RNF5: El clasificador podrá ser utilizado en sistemas que tengan Python 2.7 instalado.

### Clasificador orientado al usuario

Para satisfacer los requisitos especificados en los dos apartados anteriores es necesario completar la primera parte del proyecto, es decir, la fase de experimentación para decidir que clasificadores se adecúan a los requisitos del usuario. Posteriormente, se realiza el encapsulamiento del clasificador en un módulo de Python que pueda ser importado fácilmente facilitando su uso al usuario. A parte, también se desarrollará un manual para el usuario bastante intuitivo para que este pueda usar el clasificador con la mayor comodidad posible. Este manual se encontrará en Github en formato Wiki, en ella se encuentran las instrucciones correspondientes al uso del clasificador y las librerías de Python necesarias para el funcionamiento del mismo.

4

# Desarrollo.

El objetivo del apartado de desarrollo es explicar cómo se ha llevado a cabo el desarrollo del proyecto y las decisiones que se han ido tomando con el objetivo de elaborar un modelo de clasificador que funcione correctamente satisfaciendo los requisitos del usuario.

## Parte I:

### Obtención de tuits

Para desarrollar este punto de la primera parte se ha necesitado registrar 4 usuarios en la aplicación de Twitter. A partir de cada uno de los usuarios se crean 3 aplicaciones de usuarios (número máximo permitido). Por lo tanto, se han generado 12 aplicaciones en total con las que tendremos acceso a la API Rest de Twitter. La necesidad de creación de tantas aplicaciones se debe a que la API de Twitter tiene un número de peticiones limitadas cada 15 minutos. En el caso de la búsqueda de tuits el número de peticiones máximo en esta ventana de tiempo es de 180, con cada una la cuales pediremos 100 tuits (cantidad máxima por petición), por lo tanto 18.000 tuits cada 15 minutos. Al exceder esta cantidad de peticiones, la siguiente aplicación se encargará de realizar las peticiones hasta alcanzar el límite, mientras la anterior aplicación queda en espera. Por lo tanto, al realizar las peticiones con la decimoquinta aplicación, la primera ya habrá esperado lo suficiente como para volver a empezar a enviar peticiones. En la consulta de cada petición se especificará que los tuits están en español y que se obtengan los más recientes. Para asegurarnos de que los tuits son de España, hemos tenido que acceder a los metadatos de la petición para extraer el huso horario de la fecha del tuit, de tal forma que este coincida con el de España. De alguna forma forzamos a que el tuit esté en español y que tenga el huso horario de España, con ello conseguimos que sea probable que el tuit sea de España. Finalmente, se consiguió descargar 886.532 tuits con el sistema mencionado anteriormente.

### Limpieza de tuits

Tras haber descargado los 886.532 tuits es necesario llevar a cabo la limpieza de su contenido. El limpiador se ha desarrollado en el módulo “limpiador.py”. La limpieza se realizará mediante el uso de *expresiones regulares* que permitirán: eliminación de espacios innecesarios, eliminación repetición de caracteres o símbolos, detección de enlaces a dominios web, detección de menciones a usuarios, detección de temas de Twitter (hashtags) e imposición de uso exclusivo de caracteres del alfabeto latino. Las tildes se han eliminado durante este proceso y las eñes se convertirán en enes. Finalmente, en este módulo también se llevará a cabo la creación de nuevos símbolos (tokens) dotados de significado.

En la Tabla 4‑1se muestra la correspondencia de la semántica contenido del tuit a cada uno de los nuevos símbolos insertados en el texto limpiado:

|  |  |
| --- | --- |
| **Semántica** | **Símbolo/Token Type** |
| Enlace a un dominio web | TOKENURL |
| Mención a un usuario | USER |
| Temática de Twitter | HASHTAG |
| Interrogación | TOKENQUES |
| Exclamación | TOKENEXC |
| Risas usando lenguaje de redes sociales: XD,xd,XDD | TOKENXD |
| Comillas | TOKENCOMI |
| Risas normales: jajaja, ajaja, jaj,… | TOKENRISAS |
| Incredulidad o sorpresa: WTF | TOKENWTF |

abla 4‑1 Correspondecia de semática y símbolo

### Extracción de información de los tuits

Durante esta etapa se lleva a cabo la extracción de información relevante dentro de un tuit. Se ha considerado que las palabras con más aporte de semántica para este problema de clasificación de tuits son: verbos, nombres y adjetivos. Por lo que se necesita hacer uso de un *pos-tagger* o etiquetador morfológico. El analizador morfológico se implementará en el módulo “POS Tagger.py”. Para desarrollar el analizador morfológico se han utilizado las librerías Pattern y NLTK. También ha sido necesario descargar el Wikicorpus para entrenar el analizador morfológico. Este Wikicorpus está formado por todos los artículos de Wikipedia de 2006 y en total contiene 120 millones de palabras etiquetadas con su clase correspondiente. Para realizar el entrenamiento del *POS Tagger* ha sido necesario:

1. Cargar todos los términos del vocabulario contenido en los artículos del Wikicorpus, almacenando por cada término las diferentes clases que toma dicho término con la frecuencia correspondiente. Únicamente nos quedamos con las 100.000 palabras más frecuentes y por cada palabra su tipo más frecuente. El resultado obtenido se debe almacenar en un fichero (persistencia para su posterior uso).
2. Generar un conjunto de reglas para determinar a qué clase de palabra pertenece un término basándose en su contexto. Para ello se hará uso de la librería NLTK que contiene un módulo de entrenamiento basado en el algoritmo de Brill. Las reglas son muy sencillas ya que se basan en el análisis de la clase de palabras que rodean el término que estamos analizando, es decir, su contexto. Algunas reglas son: verificar la clase de la palabra que se encuentra 1 posición adelante o 2 posiciones adelante, verificar las palabras que se encuentran 1 posición adelante o 2 posiciones adelante, verificar el tipo de palabra que hay 1 posición atrás… El propio módulo de entrenamiento almacenará aquellas reglas que maximicen la precisión de la predicción. Estas reglas deberán almacenarse para su posterior uso.
3. Generar un conjunto de reglas basadas en sufijos que permitan determinar palabras que nunca antes se han visto. Se han utilizado los 5 últimos términos de 2.000.000 de términos del Wikicorpus para obtener una predicción basada en sufijos. En la siguiente tabla se muestran algunas relaciones entre sufijos y clases de palabras:

|  |  |  |  |
| --- | --- | --- | --- |
| Frecuencia | Sufijo | Clase de palabra | Ejemplo |
| 5986 | -ación | 99% nombre común  1% preposición | natación |
| 4875 | -mente | 98% adverbio  1% preposición  1% verbo | tranquilamente |
| 3276 | -iones | 99% nombre común  1% verbo | vacaciones |

Tabla 4‑2 Relación de sufijos y clases de palabras

Tras haber realizado la limpieza de tuits, en esta etapa tendremos 885.758, es decir, 774 tuits menos que en la etapa anterior. Este hecho puede deberse a que tras la limpieza estos tuits no contuvieran término alguno por lo que decidimos descartarlos, otros de ellos se han descartado porque la codificación no permitía la interpretación de los mismos, es posible embeber en los tuits cierto contenido en HTML que no se ha considerado útil en el proyecto, sucede lo mismo con las imágenes, vídeos o gifs que se pueden añadir a los tuits, su contenido no se tiene en cuenta. También es de destacar que tras la limpieza habrá un total de 204.041 términos únicos (*token types*).

### Selección de tuits

Una vez finalizada la limpieza se procede a la selección de tuits a etiquetar ya que el etiquetamiento de 885.758 es inviable, para seleccionar los tuits se desarrollará el filtro mencionado en el apartado de diseño (6.1.4). Todo este proceso está implementado en el archivo “kimchi.py”. Los filtros contienen las siguientes categorías de palabras fomentadoras de odio:

* + *Apariencia* (clases.txt): hace referencia a la apariencia física y forma de ser percibida externamente p.e: marcarras, chonis y canis.
  + *Discapacidad* (discapacidad.txt): hace referencia a las diversas discapacidades que pueden presentar las personas y anormalidades físicas que pueden ser objetivo de burla p.e: mongolo, retrasado y enano.
  + *Género y orientación sexual* (genero.txt): hace referencia a la discriminación o desprecio por género o tendencia sexual p.e: marica, bollera y malfollada.
  + *Política* (política.txt): hace referencia al insulto, menosprecio y discriminación por la condición política de la persona p.e: podemita, fascista y facha.
  + *Etnicidad y nacionalidad* (raza.txt): hace referencia al insulto, desprecio y discriminación por la nacionalidad o etnicidad de la persona: moro, sudaca y panchito.
  + *Religión* (religión.txt): hace referencia a las creencias religiosas de los individuos p.e.: católico y ateos.

Sin embargo, no todas las palabras expresan odio en todos los contextos, por ello se ha creado una clase nueva de palabras basada en la que se contienen las palabras de soporte al odio mencionadas en (6.1.4). Estas palabras están contenidas en el archivo “maldad.txt” y algunas de ellas son: sucio, mierda, asqueroso y maldito.

Para hacer uso del conjunto de soporte de odio en los archivos: clases.txt, discapacidad.txt, genero.txt, política.txt, raza.txt y religión.txt se ha indicado el grado de odio contenido en cada término de los archivos. Hemos definidos dos grados: odio absoluto y odio relativo. Por un lado, el grado de odio absoluto especifica que la palabra sin ningún contexto ya transmite odio de por sí p.e: *feminazi*, *malfollada,* *guachupino*. Por otro lado, el grado de odio relativo indica que la palabra necesita estar en un contexto malicioso (el tuit contiene palabras de soporte al odio) para que esta se considere de odio absoluto p.e: *negro*, *retrasado* y *gitano*.

Tras desarrollar el filtro se ha procedido a su utilización y únicamente se han pasado aquellos tuits que contuvieran palabras de odio absoluto o palabras de odio relativo y palabras de soporte al odio.

Una vez que se ha finalizado el proceso de selección de tuits, se consigue reducir el número de tuits a 1971 en la primera iteración. Tras esta primera iteración se añadieron 151 palabras a los filtros y tras esta adición se realizó una segunda iteración en la que se obtuvieron 3303 tuits, es decir, pasaron 1332 tuits nuevos.

### Etiquetado de tuits

En esta etapa ya contamos con tuits limpios y filtrados. El etiquetado no se ha podido realizar en las condiciones óptimas mencionadas en el apartado de diseño de esta categoría (§6.1.5), esto es debido a que se ha considerado más cómodo que una sola persona externa al desarrollo del clasificador se encargue de leer y etiquetar los tuits filtrados. El hecho de tener a 3 personas leyendo y etiquetando tuits iba a ser muy costoso ya que el etiquetado no solo se realiza una vez sino, 3:

1. Etiquetación tras la primera iteración de filtrado.
2. Etiquetación tras la segunda iteración de filtrado.
3. Etiquetación para pruebas de validación.

El tener 3 personas realizando cada una de estas etapas podía suponer un retraso del proyecto al tener que leerse y etiquetar cada uno tantos tuits y posteriormente hacer la selección de la etiqueta correcta de cada tuit por votación mayoritaria.

Al tener a una sola persona etiquetando, se debe de tener en cuenta que el etiquetamiento es influido por la ideología de esa persona e incluso su estado de ánimo a la hora de realizar este etiquetado puede influir en el mismo. Esto es debido a que el hecho de considerar que un tuit contiene odio es muy subjetivo, salvo en algunos casos donde el odio es percibido por todo el mundo.

Durante la primera iteración del filtrado se obtuvieron 1971 tuits de los cuales 1.362 (69%) eran de no odio y 609 (31%) eran de odio. Según los resultados anteriores, se puede observar que durante la primera iteración se consiguió un equilibrio bastante significativo de ambas clases ya que en un primer momento como se mencionó en la introducción (1) la proporción de tuits de odio antes de pasar el filtro podía aproximarse a (1:1000). Tras esta primera iteración se realizó la segunda en la que se introdujeron 151 palabras nuevas en los archivos que constituyen el filtro. En esta segunda etapa se obtuvieron 3303 tuits de los cuales 2404 (73%) eran de no odio y 899 de odio (27%). Se puede observar que con respecto a la primera iteración hay un leve aumente del desequilibrio entre ambas clases, en concreto una diferencia del 4%. Esto hecho puede deberse a que las nuevas palabras añadidas al filtro no contenían tanto odio como las anteriores. Finalmente, se llevará a cabo otra fase de etiquetación en validación (7.1.8).

### Selección de atributos

Durante esta etapa se realiza la extracción de atributos de los tuits siguiendo el diseño mencionado en (§6.1.6). Se utilizan como variables de atributos tanto *tf-idf* (variable continua) como la aparición o no del término en un tuit (variable binaria: 0 o 1). En el apartado de diseño se planteaba el uso de la frecuencia del término en el tuit, pero durante esta etapa de desarrollo se ha considerado no necesario tras realizar un análisis de las pruebas realizadas en (§8.2).

Dos variables utilizadas:

* *Binaria*: el uso del modelo saco de palabras en los tuits permite determinar fácilmente si un término se encuentra o no dentro del tuit ya que cada uno de los términos se representa con un identificador, por lo tanto, se tiene que comprobar que en el tuit ese identificador aparezca al menos una vez.
* *Tf-idf*: su utilización se debe al aprovechamiento de tuits que no pasaron el filtro. Si bien se piensa, los tuits que no pasaron el filtro (una gran mayoría) no se vuelven a utilizar una vez que se ha obtenido el conjunto de clasificación filtrado, este hecho supone una pérdida de información. Por lo que se cree conveniente el aprovechamiento de estos tuits para calcular el tf-idf de cada término en los tuits. Todo este proceso se desarrolla con Gensim. Se calculará el tf-idf sobre el *corpus* entero y se aplicará posteriormente sobre cada uno de los tuits del conjunto de clasificación ya filtrado.

En esta etapa no solo se utilizarán términos individuales para la representación de los tuits sino también bigramas (2-gramas). Para construir estos bigramas es necesario tener el orden relativo de las palabras dentro de cada tuit, por lo que no se puede aplicar el modelo *saco de palabras* (BOW) en el que el orden de los términos no es relevante. El algoritmo de construcción de bigramas es sencillo, solo hay que representar el tuit como un vector de términos en orden e ir agrupándolos de dos en dos.



Figura 4‑1: Diferencias entre unigrama y bigrama

En la figura superior se muestran las diferencias entre los unigramas y bigramas.

Una realizada la transformación a bigramas, los tuits si pueden estar representados en el modelo de saco de palabras, en este caso se trataría de un saco de bigramas, ya que el orden de los mismos dentro del tuit no lo tenemos en cuenta. Esta estructura de bigramas también es compatible con el uso de la variable binarias y tf-idf ya que los bigramas pueden considerarse términos. Sin embargo, durante el desarrollo por simplicidad solo se utilizan los bigramas únicamente con la variable binaria. Los bigramas candidatos a atributos se obtendrán a partir de la unión de palabras que constituyen el filtro junto con el resto de palabras contiguas dentro de los tuits, si se utilizaran todos los bigramas existentes dentro de los tuits que han pasado el filtro, el número de bigramas sería excesivamente grande. Además, el uso de bigramas y unigramas no es exclusivo en la tabla de atributos por lo que se pueden utilizar ambas estructuras simultáneamente.

La selección de atributos se lleva a cabo mediante el test de *Chi cuadrado* (χ2) que se encuentra disponible en la librería Scikit-learn. Se ha fijado un p-valor menor que 0.35 con el cual se consigue un menor error durante la clasificación supervisada. Sin embargo, se debe de tener en cuenta que para aplicar esta prueba los atributos tienen que ser discretos, como es el caso de la variable binaria. Por lo tanto, en primer lugar, se utiliza la variable binaria para posteriormente se aplica el test de *Chi cuadrado* (χ2). Una vez realizado el test se determina si se quiere mantener la variable binaria en representación de los atributos o si se prefiere utilizar *tf-idf*.

También se ha utilizado durante esta etapa el test de *Chi cuadrado* (χ2), para seleccionar nuevas palabras y añadirlas al filtro. Como el test devuelve aquellos atributos que depende de una de las dos clases, habrá que obtener aquellos términos cuya dependencia esté asociada a la clase de odio. Para ello se han guardado aquellos términos devueltos por el test de *Chi cuadrado* (χ2) que aparezcan más en tuits de odio que en los de no odio. Finalmente, hay que leer los términos obtenidos y determinar si contienen odio para añadirlos posteriormente al filtro. El procedimiento mencionado se utilizó en la segunda iteración de filtrado para aumentar el tamaño del mismo. Se leyeron un total de 1.261 palabras de las cuales un 12% contenían odio y pudieron ser añadidas al filtro. Este procedimiento tiene la ventaja de que el usuario puede encontrar nuevas palabras fácilmente sin mucho esfuerzo ya que el proceso es semi-supervisado.

El conjunto de entrenamiento se genera con el módulo “training\_set\_generator.py” y la selección de atributos se lleva a cabo en el mismo módulo de clasificación “hate\_classifier\_tester.py”

### Clasificación supervisada

Una vez obtenido el conjunto de clasificación con los atributos relevantes seleccionados y utilizando las variables que se desean probar (binaria y tfidf) se procede a la experimentación con los clasificadores mencionados en (6.1.7.). Para los clasificadores basados en redes neuronales se ha utilizado la librería Keras que se ejecuta sobre Tensorflow, para los demás clasificadores se ha utilizado Scikit-learn. El uso de los clasificadores se realiza en el módulo “hate\_classifier\_tester.py” del proyecto.

#### Árboles de decisión

Al realizar las pruebas para estudiar el rendimiento de este clasificador se ha utilizado la clase DecisionTreeClassifier de Scikit-learn y se ha especificado que el modelo a utilizar es el basado en la entropía. Esto último se debe a que se quiere determinar cuáles son los atributos que proporcionan mayor ganancia de información respecto a las clases de odio y no odio. Al haber dos clases se formará un árbol binario a una altura h, se propone ir aumentando la altura del árbol al realizar los experimentos. Las pruebas y el análisis de las mismas se encuentran en (5.1.1).

#### Naïve Bayes

Para desarrollar este apartado se han utilizado los clasificadores: BernoulliNB, MultinomialNB y GaussianNB. Estos clasificadores se encuentras disponibles en la librería Scikit-learn. BernoulliNB se ha utilizado cuando los atributos son binarios, es decir, está o no el término en el tuit. Si además se quiere tener en cuenta la frecuencia de cada término en los tuits se utilizar MultinomialNB. Finalmente, se propone el uso de GaussianNB para variables continuas, en nuestro caso tf-idf, suponiendo que la distribución de los atributos sigue una distribución normal. Al igual que cuando utilizamos Naive Bayes suponemos la independencia de atributos y en muchas ocasiones se obtienen buenos resultados, se ha creído en la posibilidad de obtener buenos resultados al suponer que la distribución de los atributos es una gaussiana aunque realmente esto no sea cierto. Las pruebas y el análisis de las mismas se encuentran en (5.1.2).

#### Vecinos Próximos (KNN)

Para realizar las pruebas con este clasificador se ha utilizado el clasificador proporcionado por Scikit-learn, KneighborsClassifier. Al realizar las pruebas se han variado el número de vecinos próximos y se han utilizado dos métricas para calcular las distancias: aming y coseno. La primera se ha utilizado con variables binarias, mientras que la segunda con la variable tf-idf. Al utilizar este clasificador se pretende averiguar si tuits que comparten términos pertenecen a la misma clase. Además, tf-idf proporciona más información ya que además tiene en cuenta la frecuencia documental (idf) y la frecuencia de términos dentro del tuit (tf).

En este apartado es de destacar que KNN con distancia coseno no estaba disponible en las librerías de Python utilizadas por lo que ha sido necesario la implementación del clasificador con distancia coseno. Para desarrollar este clasificador propio se ha utilizado la librería Gensim, ya que al tener el corpus con la estructura de saco de palabras (BOW) generada por la propia librería, el cálculo de tf-idf se puede realizar con mayor facilidad utilizando Gensim. Los tf-idfs calculados son sobre el corpus inicial antes de utilizar el filtro, es decir, sobre los 885.758. Esto se ha realizado debido a que posteriormente no se han vuelto a utilizar los tuits no filtrado lo cual significa que hay un desperdicio de información, si se calcula el tf-idf sobre el corpus total, se cree que se podría obtener información útil para la posterior clasificación. Además, para realizar esta clasificación no se han utilizado los atributos obtenidos por el test de *Chi cuadrado* (χ2), sino el tuit entero una vez que este está limpio, aunque solo se utilizan para clasificar los tuits del conjunto de clasificación, los 3303, ya que estos están etiquetados y nos permiten realizar pruebas con su correspondiente análisis. El clasificador propio se encuentra en el archivo “cosine\_knn.py” y su módulo de pruebas es “cosine\_tester.py”. Las pruebas y el análisis de las mismas se encuentran en (5.1.3).

#### Redes Neuronales

Para desarrollar este apartado se ha empezado utilizando en un primer momento la librería Scikit-learn y posteriormente se ha hecho uso de Keras. Esta última permite una mejor personalización de la red neuronal que se quiere diseñar, es decir, especificar mejor la estructura de la red y además proporciona una mayor cantidad de parámetros a ajustar que Scikit-learn, los cuáles son muy importantes a la hora de realizar una correcta clasificación.

### Validación de resultados

## Parte II

5

# Pruebas y resultados

Las pruebas se desarrollarán mediante *validación cruzada*, en concreto, se hará uso de 10 particiones. También se utilizará *particionamiento estratificado*. La representación de las matrices de confusión se basará en la suma de cada partición de test (10 en total). Las demás condiciones de experimentación se especificarán en cada uno de los apartados siguientes. Cuando se especifique que hay normalización de los datos esta se llevará a cabo mediante utilizando *mediana* y *rango intercuartílico* (tercer y segundo cuartil).

## Resultados de clasificación supervisada

### Árboles de decisión

Las pruebas realizadas con árboles de decisión se realizarán con: variables binarias (1 el término está o 0 si no está) y unigramas o bigramas. Además, el modelo de árbol de decisión utilizado es el basado en la entropía, de tal forma que el árbol se irá construyendo con aquellos atributos que proporcionen mayor ganancia de información.

Resultado utilizando unigramas con 2532 atributos y 3303 patrones:

|  |  |  |
| --- | --- | --- |
| Profundidad máxima | Error de exactitud | Desviación típica entre particiones |
| 3 | 0.2525 | 0.0121 |
| 4 | 0.2524 | 0.0146 |
| 5 | 0.2519 | 0.0101 |
| 6 | 0.2540 | 0.0139 |
| 8 | 0.2534 | 0.0161 |
| 10 | 0.2485 | 0.0133 |
| 11 | 0.2537 | 0.0195 |
| 12 | 0.2515 | 0.0165 |

Tabla 5‑1: Comparativa de error de precisión frente a profundidad

Como se puede observar en los resultados a medida que se va aumentando la profundidad máxima del árbol el error de precisión se sigue manteniendo casi igual. Para una profundidad máxima de 12, es decir, 4.096 nodos como máximo al tratarse de un árbol binario, se ha generado una matriz de confusión:

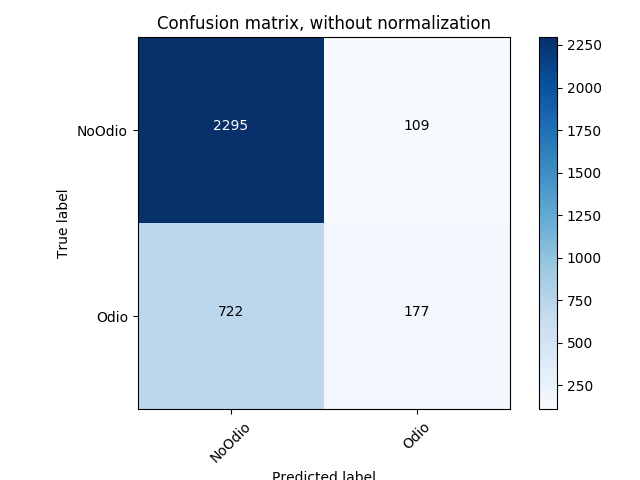


Figura 5‑1: Matriz de confusión de árbol de decisión

Como se puede apreciar en la matriz, aunque la *exactitud* sea casi el 0.75, la *precisión* (p) y la *exhaustividad* ® con respecto a la clase de odio toma los siguientes valores:

Se puede observar que la *exhaustividad* es muy baja ya que se nos escapan muchos tuits que realmente sí que son de odio, la *precisión* también podría ser bastante mejor. Además, si clasificáramos todos los tuits como de no odio obtendríamos un 0.7278 de exactitud por lo que 0.75 no se puede considerar un buen resultado.

A continuación, se muestra una pequeña imagen de los primeros nodos del árbol de decisión (aportan mayor ganancia de información):



Figura 5‑2: Árbol de decisión generado

En la Figura 5‑2 se pueden observar algunas reglas generadas por el árbol de decisión que pueden interpretarse con sentido, p.e: si aparece la palabra “*mierda*” y “*subnormal*” se puede considerar que el tuit pertenece a la clase de odio, si aparece la palabra “*mierda*”, pero no “*subnormal*”, pero sí “*político*”, el tuit no se considerará de odio y finalmente, si no aparece “*mierda*”, si aparece “*puto*” y sí aparece “eres” el tuit es considerado de odio.

Resultado utilizando bigramas con 5085 atributos y 3303 patrones:

|  |  |  |
| --- | --- | --- |
| Profundidad máxima | Error de exactitud | Desviación típica entre particiones |
| 3 | 0.2506 | 0.0118 |
| 4 | 0.2528 | 0.0126 |
| 5 | 0.2528 | 0.0108 |
| 6 | 0.2503 | 0.0085 |
| 8 | 0.2485 | 0.0130 |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |

Tabla 5‑2: Comparación de exactitud con bigramas respecto a la profundidad

Se puede observar en lasTabla 5‑1 y 5-2 que el uso de bigramas no aporta mucha mejoría en cuanto a *exactitud* respecto al uso de unigramas.

### Naive Bayes

Las pruebas realizadas con el clasificador Naive Bayes se realizarán con: variable binarias (1 el término está o 0 si no está), variable de frecuencia (se tiene en cuenta el número de apariciones de un término en el tuit), variable tf-idf y unigramas o bigramas. Además, se utilizarán tres clasificadores diferentes basados en Naive Bayes: Bernoulli (tiene en cuenta la aparición o no del término), Multinomial (tiene en cuenta la frecuencia de cada término) y Gaussiana. Para el clasificador Multinomial y Naïve Bayes se ha utilizado la constante alpha (Laplace) con el valor *2.e-06* determinada empíricamente.

Resultado utilizando Bernoulli, variable binaria, unigramas con 2532 atributos y 3303 patrones:

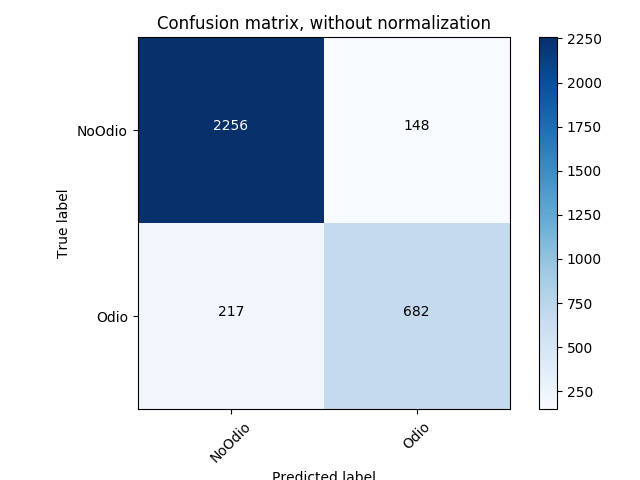


Figura 5‑3: Matriz de confusión BernoulliNB

A partir de la matriz de confusión de (8-3) se puede observar que la exactitud es de 0.8894 que comparada con la del clasificador básico (predice que todos los tuits son de no odio) es superior por 0.1616, lo cual no está nada mal. Sin embargo, hay que tener otras medidas importantes como precisión (p) y exhaustividad ®:

La exhaustividad muestra que se nos han escapado aproximadamente un 24% de tuits que sí eran de odio, pero los clasificamos como no de odio. Mientras que la precisión muestra que el 82% de tuits que se clasificaron como de odio, realmente si lo eran.

Resultado utilizando Multinomial, variable de frecuencia, unigramas con 2532 atributos y 3303 patrones:

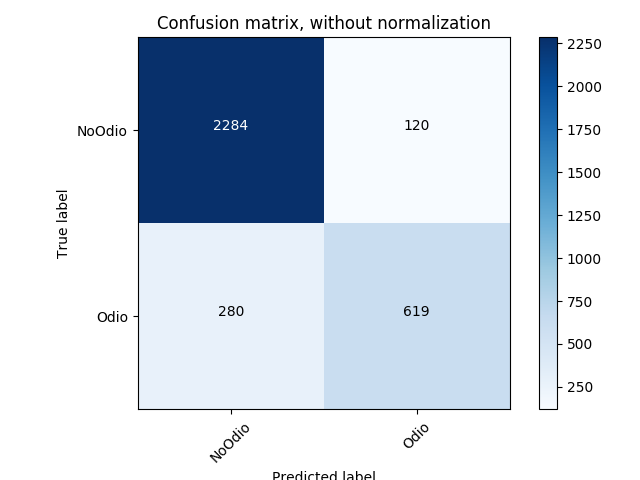


Figura 5‑4: Matriz de confusión MultinomialNB

La matriz de confusión (8-4) muestra que el clasificador Multinomial teniendo en cuenta la frecuencia de aparición de términos en un tuit tiene una exactitud de 0.8788, es decir, es mayor por 0.151 en cuanto a exactitud con respecto al clasificador básico (predice que todos los tuits son de no odio). A partir de la misma matriz se pueden calcular otras métricas de interés importantes como precisión (p) y exhaustividad ®:

Si comparamos el clasificador Bernoulli y Multinomial se puede observar que hay cierta semejanza en cuanto exactitud y precisión por lo que se puede considerar que para el cálculo de tales métricas el tener en cuenta la frecuencia o únicamente si el término aparece o no, no tiene mucha importancia, este hecho podría deberse a que un tuit no contiene muchos términos lo que implica que el conteo de la aparición de términos no tenga relevancia. Sin embargo, se puede observar cierta diferencia entre la exhaustividad utilizando el clasificador Bernoulli (0.7586) y la obtenida con el Multinomial (0.6885). Esto puede deberse a que al usar Bernoulli, este tiene en cuenta la probabilidad complementaria de aparición del término en la clase, mientras que Multinomial directamente lo considera como 0 si el término no aparece en el tuit a clasificar.

Resultado utilizando Bernoulli, variable binaria, bigramas con 5085 atributos y 3303 patrones:

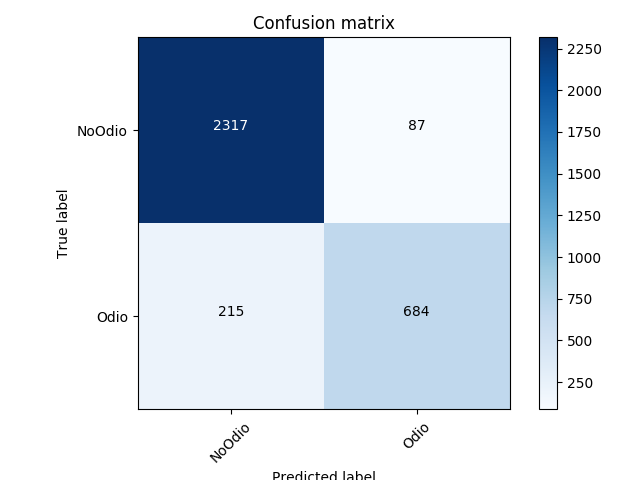


Figura 5‑5: Matriz de confusión usando bigramas y BernoulliNB

En este caso la matriz de confusión (8-5) obtenida usando bigramas muestra una exactitud de 0.9085, es decir, hay menos de un 10% de tuits que se clasifican mal. Además, la precisión (p) y exhaustividad ® son:

Se puede considerar una mejoría con respecto al uso del clasificador Bernoulli con unigramas. El incremento de la exactitud usando bigramas es de 0.0297, el de precisión, 0.0495 y finalmente, la exhaustividad aumenta 0.0723. Los incrementos anteriores, muestran una mejoría del clasificador BernoulliNB usando bigramas en todas las métricas. Esto hecho tiene sentido, pues ahora podemos obtener más información acerca del contexto de los términos, información necesaria puesto que la semántica del tuit dependerá del contexto en el que se utilicen las palabras.

Resultado utilizando Gaussiana, variable tf-idf (normalizada), unigramas con 2532 atributos y 3303 patrones:

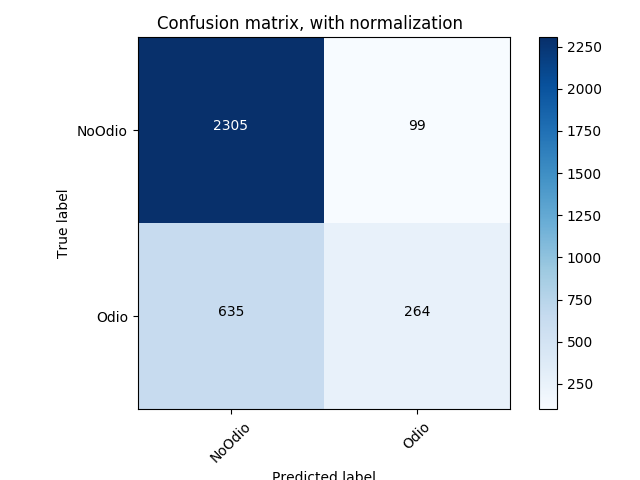


Figura 5‑6Matriz de confusión utilizando tfidf y GaussianNB

La exactitud proporcionada por la matriz de confusión es de 0.7778, sin embargo, solo es mayor que la del clasificador básico (predice todos de no odio) por 0.0499. Los valores de precisión (p) y exhaustividad ® respecto a los tuits de odio son los siguientes:

Se puede observar que la exhaustividad es bastante baja, hay más de un 70% de odio que no se han considerado de tal clase. El error puede deberse a que el considerar que los atributos continuos (tfidf) siguen una distribución gaussiana es incorrecto.

### Vecinos Próximos (KNN)

Las pruebas realizadas con el clasificador Vecinos Próximos se realizarán con: variable binarias (1 el término está o 0 si no está), variable tf-idf y unigramas. Las distancias utilizadas son: Haming (variable binaria) y coseno (variable tf-idf).

Resultado utilizando KNN distancia Hamming, variable binaria, unigramas con 2532 atributos y 3303 patrones:

|  |  |  |
| --- | --- | --- |
| Número de vecinos | Error de exactitud | Desviación típica entre particiones |
| 3 | 0.2652 | 0.0074 |
| 5 | 0.2664 | 0.0058 |
| 7 | 0.2664 | 0.0045 |
| 13 | 0.2688 | 0.0018 |

Tabla 5‑3: Error de exactitud de KNN por vecinos

Se observa en la tabla (8-3) que los errores de exactitud se mantienen iguales a medida que se aumenta el número de vecinos próximos. Además también se ha realizado una matriz de confusión para k=3:

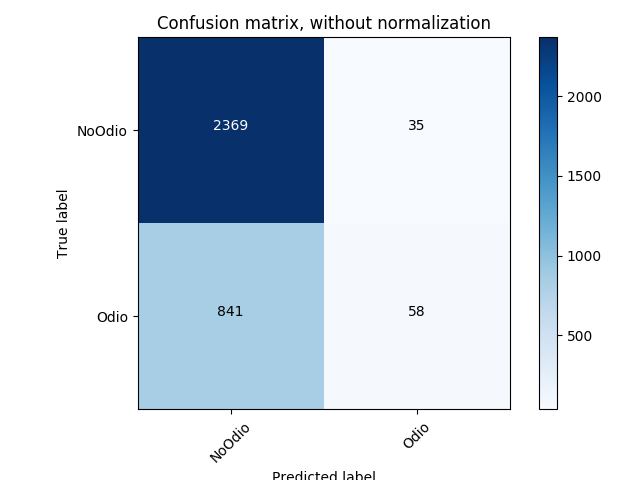


Figura 5‑7: Matriz de confusión de KNN (distancia Hamming)

La precisión (p) y exhaustividad ® obtenidas de la matriz de confusión (8-7) son las siguientes:

Por lo general los resultados de las métricas utilizadas son muy pobres, destacado la exhaustividad que únicamente detecta el 6% de tuits que son de odio. Estos resultados tan malos pueden ser debidos a que realmente no importa que los tuits posean términos similares, también se debe de tener en cuenta el contexto que rodean a esos términos. Además, habría que añadir la cantidad de ruido que puede haber en los tuits.

Resultado utilizando KNN distancia coseno, variable tf-idf, texto limpio del tuit original y 3303 patrones:

|  |  |  |
| --- | --- | --- |
| Número de vecinos | Error de exactitud | Desviación típica entre particiones |
| 3 | 0.2721 | 0.0206 |
| 5 | 0.2667 | 0.0239 |
| 7 | 0.2582 | 0.0110 |
| 13 | 0.2570 | 0.0054 |

Tabla 5‑4: Error de exactitud de KNN (distancia coseno)

Como se puede observar en la tabla (8-4) esta es muy similar a la obtenida utilizando la distancia de Hamming. En este caso también se ha realizado una matriz de confusión para k=3:

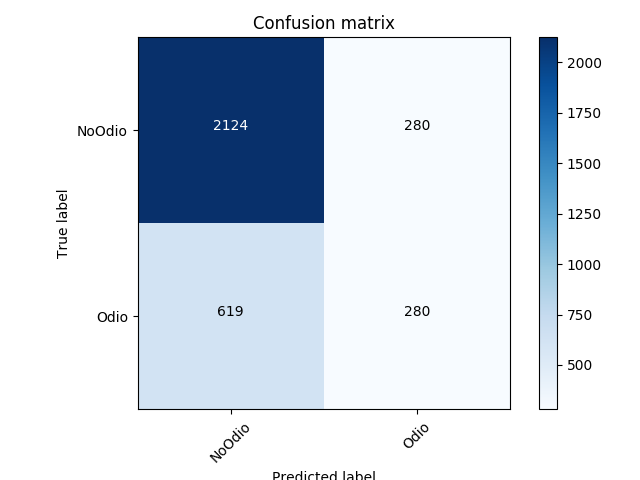


Figura 5‑8: Matriz de confusión KNN (distancia coseno)

La precisión (p) y exhaustividad ® obtenidas de la matriz de confusión (8-7) son las siguientes:

La exhaustividad es bastante mayor en este caso en comparación con los resultados obtenidos con la distancia de Hamming (0.0645), sin embargo, la precisión es menor con distancia coseno, una disminución de 0.12. A pesar de que con la distancia coseno haya cierta mejoría con respecto al uso de KNN con distancia Hamming, los resultados no se pueden considerar buenos. A pesar de que ahora se tenga en cuenta el tf e idf y la distancia coseno permita obtener los k tuits más parecidos, esto no significa que los tuits pertenezcan a la misma clase. Puede deberse al ruido o incluso por el propio contexto, es decir, que este último no sea vea lo suficientemente reflejado en la distancia coseno.

### Redes Neuronales

Las pruebas realizadas con los clasificadores basados en redes neuronales se realizarán con: variable binarias (1 el término está o 0 si no está), variable tf-idf y unigramas. Los 3 modelos de redes neuronales utilizados son: perceptrón simple, perceptrón de una capa oculta y perceptrón multicapa de tres capas ocultas.

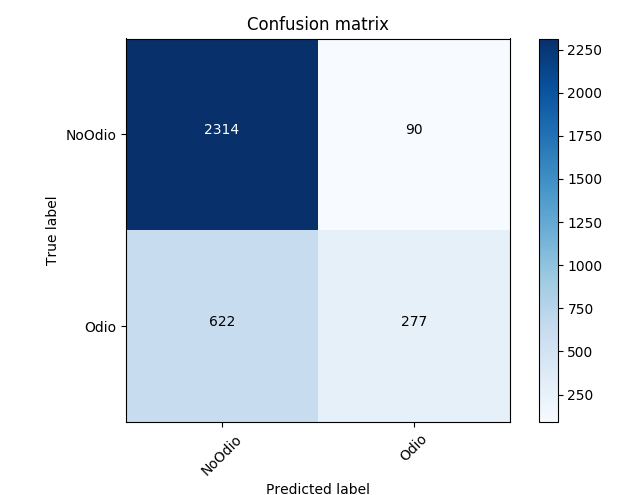
#### Perceptrón Simple

Para realizar las pruebas correspondientes con este clasificador se han tenido en cuenta dos parámetros muy importantes que son el número de épocas y la tasa de aprendizaje. Además, estas pruebas se han realizado con 3303 patrones y 2532 atributos binarios.

Para calcular la tasa de aprendizaje óptima se ha establecido un número máximo de iteraciones (1000) y esta se ha ido variando. Al realizar el experimento se obtuvo la siguiente gráfica:



Esta gráfica muestra que el error mínimo de exactitud se encuentra con una tasa de aprendizaje de 0.1, es decir, el valor más pequeño con el que se ha probado. Utilizando esa tasa de aprendizaje se ha obtenido la siguiente matriz de confusión de una de las particiones:



El error de exactitud obtenido es 0.2155 si se compara con la del clasificador básico (0.2721) solos se rebajaría el error un 6%. La precisión (p) y exhaustividad ® obtenidas de la matriz de confusión (-) son las siguientes:

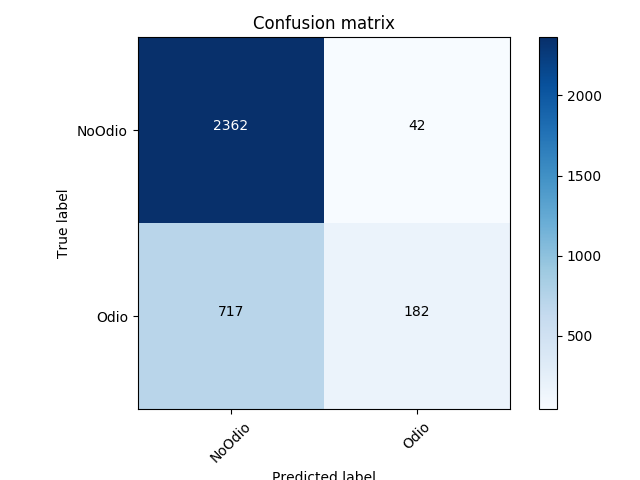
El error de precisión, exhaustividad y exactitud puede deberse a que se trata de un problema que no es linealmente separable por lo que un perceptrón no es capaz de clasificar correctamente los tuits en sus correspondientes clases.

#### Perceptrón de una capa oculta

Al utilizar un perceptrón de una sola capa oculta hay que ajustar el número de neuronas en la capa oculta aparte de la tasa de aprendizaje. Para ello se propone una búsqueda codiciosa de parámetros, se fija la tasa de aprendizaje mientras se varía el número de neuronas en la capa oculta y una vez encontrado el número que minimice el error de exactitud se fija ese número de neuronas y se varía la tasa de aprendizaje hasta encontrar la mínima:

Se puede observar en las gráficas que error de exactitud mínimo se consigue con 230 neuronas en la capa oculta y una tasa de aprendizaje de 0.001. Clasificando con estos parámetros y fijando el número máximo de iteraciones a 1000 se ha generado una matriz de confusión:

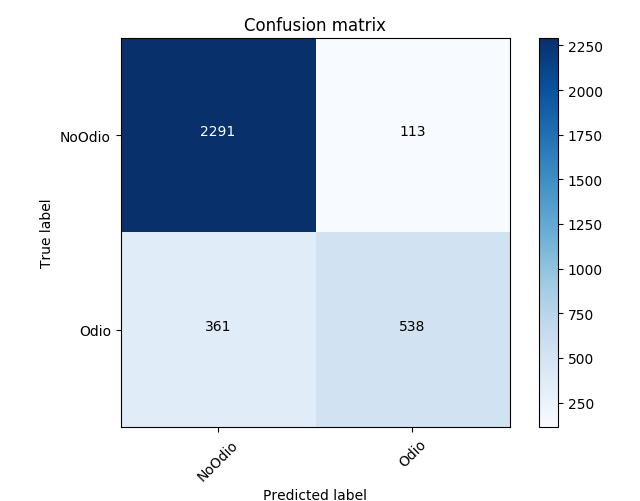


Como se puede observar en la matriz de confusión, el error de exactitud es de 0.2297, mientras que la precisión (p) y la exhaustividad toman los siguientes valores:

La precisión ha aumentado respecto al perceptrón simple mientras que la exhaustividad ha disminuido. Puede deberse a que una única capa no sea suficiente para resolver el problema al que nos enfrentamos.

#### Perceptrón multicapa de tres capas ocultas

Finalmente se experimenta con un perceptrón de tres capas ocultas con 1600 neuronas en cada una y la estructura mencionada en TODO obteniendo la siguiente matriz de confusión:



Claramente se puede observar que la exactitud se ha incrementado con respecto a los dos clasificadores basados en redes neuronales examinados anteriormente, ya que ahora el error de exactitud es de 0.1435. Mientras que la precisión (p) y la exhaustividad ® toman los siguientes valores:

La precisión se ha mantenido muy similar a los casos anteriores, aunque haya aumentado, sin embargo, se puede apreciar un gran aumento en la exhaustividad, es decir, se nos escapan menos tuits de odio. Esta mejoría del clasificador puede deberse a los nuevos parámetros mencionados en la estructura de esta red neuronal y de la variación de la misma con respecto a los anteriores experimentos.

# 6

# Pruebas.

# 7

# Resultados.

# 8

# Utilización.

Los programas y documentación del proyecto se encuentran contenidos en [repositorio GitHub]. Tal repositorio ha sido dividido en dos partes:

1. La primera contiene los programas utilizados en la fase de ensayo y selección de modelos y
2. Una segunda en la que se contienen los programas operativos, es decir aquellos que son de aplicación directa a los tuits y que permiten distinguir aquellos microblogs que pueden considerarse de odio hacia algún grupo o persona.

En este segundo directorio existe un *wiki* que contiene las instrucciones necesarias para descargar e instalar los programas y correrlos para clasificar tuits en las categorías mencionadas.

## Bibliografía

|  |  |
| --- | --- |
| [1] | «BOE,» 30 Marzo 2015. [En línea]. Available: https://www.boe.es/boe/dias/2015/03/31/pdfs/BOE-A-2015-3439.pdf. |
| [2] | G. S. Becker, «Crime and Punishment: An Economic Approach,» *Journal of Political Economy,* vol. 76, nº 2, pp. 169-217, Marzo-Abril 1968. |
| [3] | J. S. S. R. M. R. A. J. S. V. García, «The class imbalance problem in pattern classification and learning,» 2007. |
| [4] | Pete Burnap , Matthew L. Williams, «Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making.,» *Policy & Internet,* vol. 7, nº 2, pp. 223-242,, 2015. |
| [5] | T. Fawcett, «An introduction to ROC analysis,» 2005. |
| [6] | A. Downs, «Up and Down with Ecology - The Issue Attention Cycle',» *Public Interest (28),* pp. 28-50, 1972. |
| [7] | A. Huang, «Similarity Measures for Text Document Clustering,» de *proceedings of the New Zealand Computer Science Research Student Conference 2008*, Christchurch, 2008. |
| [8] | M. J. J. P. K. y. M. J. Tomas Borovicka, «Selecting Representative Data Sets,» 2012. [En línea]. Available: http://dx.dio.org/10.5772/50787. |
| [9] | H. L. y. H. Motoda, «On Issues of Instance Selection,» *Data Mining and Knowledge Discovery 6(2),* pp. 115-130, 2002. |
| [10] | J. A. O.-L. e. al., «A review of instance selection methods,» *Artif Intell Rev (2010) 34,* p. 133–143, 2010. |
| [11] | P. L. C. D. Marco Saerens, «Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure,» *Neural computation 14(1),* pp. 21-41, 2002. |
| [12] | Youngjoong Ko y Jungyun Seo, «Automatic Text Categorization by Unsupervised Learning,» 1997. |
| [13] | Yiming Yan y Jan O. Pedersen, «A Comparative Study on Feature Selection in Text Categorization,» de *ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning*, San Francisco, 1997. |
| [14] | «MIT Libraries,» [En línea]. Available: http://libguides.mit.edu/c.php?g=176061&p=1159432. |
| [15] | M. F. Porter, «An algorithm for suffix stripping,» *Program , Vol. 14 Issue: 3,* pp. 130-137, 1980. |
| [16] | «Snowball,» [En línea]. Available: http://snowball.tartarus.org/. |
| [17] | V. K. J. M. C. e. a. Hollink, «Monolingual Document Retrieval for European Languages,» *Information Retrieval,* vol. 7, nº 4, 2004. |
| [18] | E. K. a. E. L. Steven Bird, «Natural Language Processing with Python,» 2009. [En línea]. Available: www.nltk.org/book/. |
| [19] | R. a. P. S. Řehůřek, «Software Framework for Topic Modelling with Large Corpora,» de *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*, Valetta, Malta, 2010. |
| [20] | Gael Varoquax et al., «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research,* pp. 2825-2830, 2011. |
| [21] | Google Research, «TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,» 2015. [En línea]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf. |
| [22] | Christopher D. Manning et al., «The Stanford CoreNLP Natural Language Processing Toolkit,» [En línea]. Available: http://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf. |
| [23] | A. McCallum, «MALLET: A Machine Learning for Language Toolkit,» [En línea]. Available: https://people.cs.umass.edu/~mccallum/mallet/. |
| [24] | B. Carpenter, «LingPipe for 99.99% Recall of Gene Mentions,» de *Proceedings of the 2nd BioCreative workshop*, Valencia,Spain, 2007. |
| [25] | Ian H. Witten et al., «Weka: Practical Machine Learning Tools and Techniques,» de *Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems.*, 2007. |
| [26] | «yTextMiner,» [En línea]. Available: http://informatics.yonsei.ac.kr/yTextMiner. |
| [27] | «https://lucene.apache.org/core/,» [En línea]. |
| [28] | E. K. Dan Garrette, «An Extensible Toolkit for Computational Semantics,» de *Proceedings of the Eighth International Conference on Computational Semantics*, Tilburg University, Netherlands, 2009. |

## Glosario

A

AI. *Véase* Inteligencia Artificial.

análisis exploratorio de datos

Proceso al que se someten los datos antes de su modelización. Suele consistir en resumir sus características principales, con frecuencia usando métodos gráficos. Su objetivo es permitir una planificación más adecuada tanto del proceso de recolección como de su tratamiento posterior., 6

aprendizaje estadístico

Rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. Se trata de crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos., 13

Aprendizaje Máquina. *Véase* Aprendizaje Estadístico

B

*business analytics*

Conjunto de habilidades, técnicas y prácticas para la exploración iterativa del desempeño pasado de una empresa a fin de obtener una mejor comprensión de su funcionamiento y su desarrollo futuro. Utiliza tanto la minería de datos como la inteligencia artificial., 20

C

*Ciencia de los Datos*

Campo interdisciplinario que comprende los procesos y sistemas para extraer conocimiento ,,de grandes volúmenes de datos en sus diferentes formas (estructurados o no estructurados) y formatos (.txt, .dat, .doc, .jpg, etcétera, 6

clasificación de textos

Tarea consistente en asignar un documento a una categoría determinada., 20

*clasificaciónsupervisada*

Este tipo de clasificación cuenta con un conocimiento *a priori*, es decir para la tarea de clasificar una instancia dentro de una categoría contamos con modelos ya clasificados (instancias agrupadas que tienen características comunes), 7

Se parte de un conjunto de clases conocido a priori. Estas clases deben caracterizarse en función del conjunto de variables mediante la medición de las mismas en individuos cuya pertenencia a una de las clases no presente dudas., 13

conjunto de entrenamiento

Es el histórico de datos que se usa para entrenar al sistema que detecta los patrones. El conjunto de datos se compone de instancias, y las instancias de factores, características o propiedades, 8

*conjunto de prueba*

Es el usado para evaluar la bondad d las predicciones del modelo., 8

*conjunto de validación*

Es el usado para evaluar y seleccionar modelos *entrenados* sobre el conjunto de entrenamiento, 8

contraste de hipótesis

Procedimiento utilizado en inferencia estadística para juzgar si una propiedad que se supone en una población es compatible con lo observado en una muestra de dicha población., 18

*corpus*

En NLP, colección de documentos., 13

D

*delitos de odio*

Aquellos motivados por prejuicios respecto a la víctima del mismo y tienen lugar cuando el perpetrador del delito elige a la víctima en base a su pertenencia a un cierto grupo., 6

detonador

Suceso que desencadena una serie de actitudes, sentimientos y acciones en el público de modo que aumenta la relevancia del tema con que se relaciona., 11

E

*entidad*

Producto, persona, evento, organización o tópico., 6

error tipo I

En contraste de hipótesis es la probabilidad de aceptar la hipótesis alternativa siendo cierta la que se contrasta., 18

error tipo II

En contraste de hipótesis estadísticas es la probabilidad de elegir la hipótesis que se contrasta siendo cierta la alternativa., 18

***especificidad***

Proporción de falsos negativos sobre el total de instancias negativas que proporciona un clasificdor., 10

estado del arte

Una de las primeras etapas dentro de un proyecto es la construcción de su estado del arte, ya que permite determinar la forma como ha sido tratado el tema, cómo se encuentra el avance de su conocimiento en el momento de realizar una investigación y cuáles son las tendencias existentes, en ese momento, para el desarrollo de proyectos en el mismo campo., 7

*etiquetado*

Asignar una etiqueta (clase) a cualquier dato del conjuntoada., 15

*etiquetado gramatical*

Proceso de asignar a cada una de las palabras de un texto su categoría gramatical., 17

etnicidad

etnicidad

Una etnia es un conjunto de personas que tienen en común rasgos culturales, idioma, religión, vestimenta, nexos históricos, tipo de alimentación, y, muchas veces, un territorio. Dichas comunidades, a veces, reclaman para sí una estructura política y el dominio de un territorio, 6

*Exactitud*

Medida de desempeño de un clasificador. Es el porcentaje de instancias que se clasifican en su clase real., 8

*Exhaustividad*

Porcentaje de instancias clasificadas en su clase real sobre el total de las existentes en dicha clase., 9

F

f.de D

Función de distribución de una varaiable a leatoria., 16

G

grupo de control

Grupo al que no se aplica el factor que se prueba en diseño de experimentos., 15

grupo experimental

Grupo al que se aplica el factor que se prueba en diseño de experimentos., 15

I

identidad de género

Percepción subjetiva que un individuo tiene sobre sí mismo en cuanto a sentirse hombre, mujer, o de un género no binario, sin considerar características físicas o biológicas., 6

instancias

Una instancia es cada uno de los datos de los que se disponen para hacer un análisis., 7, 9, 11, 13, 15

*Inteligencia Artificial*

*ciencia de hacer máquinas que actúan racionalmente. • Racional es todo agente que busca alcanzar unos objetivos de manera tal que optimiza el valor de una función de utilidad .*, 6

La Inteligencia Artificial es la ciencia de hacer máquinas que actúan racionalmente., 13

L

*lema*

Forma única con la que se nombra un, 16

*lexema*

Parte que se mantiene invariable en todas las palabras de una misma familia, 16

*Ley de Moore*

La ley de Moore expresa que aproximadamente cada dos años se duplica el número de transistores en un microprocesador., 13

M

*matriz de confusión*

Herramienta que permite la visualización del desempeño de un algoritmo que se emplea en *aprendizaje supervisado*. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa las instancias en la clase real., 8

*métrica*

Conjunto de patrones de medida mediante los cuales se evalúa el rendimiento o la calidad de un clasificador., 8

*microblogs*

Servicio que permite a sus usuarios enviar y publicar mensajes breves, generalmente solo de texto., 6

modelizado temático

(*topic model*) Modelo de aprendizaje sin supervisión que permite detectar el tema o asunto del que trata un documento., 20

N

NLP. *Véase* Procesado de Lenguaje Natural

O

*orientación*

Tipo de opinión respecto a una entidad

favorable, desfavorable o inexistente., 4, 6

orientación sexual

orientación sexual

Patrón de atracción sexual, erótica, emocional o amorosa a determinado grupo de personas definidas por su sexo., 4, 6

P

*POS Tagging*. *Véase* etiquetado gramatical

*Precisión*

Porcentaje de instancias clasificadas en su clase real sobre el total de las clasificadas en dicha clase., 9

Procesado de Lenguaje Natural

Técnicas para conseguir que los ordenadores lleven a cabo tareas que involucran el uso del habla humana, tales como comunicación por voz entre hombre y máquina, procesamientomde texto o de voz., 6

*Procesamiento del Lenguaje Natural*

Campo de las ciencias de la computación, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano., 13

*proporción de falsos positivos*

Fracción de las instancias negativas que se clasifican como positivas., 10

S

SAS

Acrónimo de Statistical Analysis Systems

Lenguaje de programación desarrollado por SAS Institute a finales de los años sesenta., 20

***sensibilidad***. *Véase* exhaustividad

*stemming*

Forma básica de lematización que prescinde del contexto., 17

SVD

*D*escomposición en *V*alores *S*ingulares de la matriz término-documento que permite simplificar las tareas de minería de datos., 20

T

*Tasa de error*

Medida de desempeño de un clasificador. Es el porcentaje de instancias que se clasifican en una clase equivocada, 8

tasa de prevalencia

Número de personas que padecen de una enfermedad determinada en un punto determinado de tiempo por cada 1.000 habitantes., 15

*token*

Secuencia de letras entre delimitadores, 17

U

*umbral* de corte

Puntuación de una instancia que actúa como frontera entre dos clases. Las instancias con puntuación superior al umbral se clasifican en una clase y los de puntaje inferior en otra., 18

## **Anexos**

### **Anexo I**: **R**eceiver **O**perating **C**haracteristics

Un *gráfico ROC* es una técnica para visualizar, organizar y seleccionar clasificadores basados en su desempeño. Su nombre se deriva de su primera aplicación a la detección de señales campo desde el que se ha venido extendiendo su uso a la medicina y a los problemas de clasificación en general.

Ahora a la *exhaustividad* citada en §1.4. -- se la denomina ***sensibilidad****.* Como se ve*,* es la

y se introduce la ***especificidad****:*

De manera que

Y es la *proporción de falsos positivos* ***fp****:*

Por el teorema de la probabilidad total sabemos que:

|  |  |
| --- | --- |
|  | Ec. ( 8‑1) |

Si dibujamos el gráfico que relaciona ambas magnitudes, obtenemos la Figura 8‑1 [5] en que ***AUC*** es el *área bajo la curva* que puede tomar valores entre 0 (no acierta nunca) y 1 (la predicción acierta siempre).



Figura 8‑1: ROC

La Tabla 8‑2 pone de manifiesto que mientras que los indicadores usuales (*p*, *r*) usan valores de ambas filas de la matriz, los de ROC usan *valores de la misma fila*.

Por tanto, en el primer caso cuando cambian las proporciones de las clases estas medidas cambian aunque no lo haga el rendimiento del clasificador, cosa que no sucede en ROC como se ve en la Tabla 8‑2 al cambiar la proporción de clases.



Tabla 8‑1 Métrica ROC frente a Métrica Convencional

### **Anexo II:** Clasificación tras Filtrado y Exhaustividad

Como hemos dicho, debido a la dificultad de etiquetado manual de un conjunto de tan gran tamaño y tan baja proporción de tuits de odio, hemos utilizado un procedimiento que filtra el conjunto inicial extraído de Twitter mediante el uso de un vocabulario con términos de odio obtenidos de diferentes fuentes y el conjunto filtrado, de una cardinalidad mucho menor, se *etiqueta* y usa como conjunto de *entrenamiento* (Figura 1‑4).

Seguidamente, el conjunto de entrenamiento se usa para estimar los parámetros del clasificador y, con este, se clasifican los tuits del conjunto de test.

## Prefiltrado de tuits para clasificación.

Para la utilización del clasificador en otros conjuntos de tuits, caben dos opciones:

1. *Aplicarlo tras filtrar* el nuevo conjunto, asignando a todos los tuits que no pasan el filtro a *C-.*
2. *Aplicarlo al conjunto completo* con lo cual algunos de los tuits que no pasaron el filtro, se clasifiquen en *C+* (correcta o incorrectamente).

En el primer caso – el seguido por nosotros- ( Figura 8‑2), la *matriz de confusión* cuando se ***aplica a la totalidad del conjunto inicial***, muestra como FN a las instancias que no pasan el filtro que debieran estar etiquetadas con (+), una en nuestro ejemplo y, evidentemente, estas instancias que no pasan el filtro



Tabla 8‑2: Matriz de Confusión del Conjunto Inicial



Figura 8‑2: Clasificación tras Filtrado

## Medidas de desempeño de clasificadores.

Ya se ha mencionado (§1.4) que la *matriz de confusión* permite medir el rendimiento de un clasificador y se expresa en el caso de que solo existan dos clases como:



Tabla 8‑3: matriz de confusión.

Las medidas de evaluación del modelo cuando existe un claro *desequilibrio entre clases* son:

* **Precisión** (p) es el porcentaje de los *tuits* clasificados *correctamente* como de odio - *TP* -del total de los asignados a dicha clase por el clasificador - *TP+FP* - (% de aciertos).
* **Exhaustividad** (r) es el porcentaje de los *tuits* de la clase odio existente en el fichero que han sido clasificados correctamente.

## Exhaustividad del método.

Al proceder a la clasificación de tuits tras su filtrado, *inevitablemente, se produce una exhaustividad muy baja* como veremos a continuación.

Supongamos que disponemos de un conjunto de 100.000 instancias en el cual hay 1.200 instancias *C+* y 88.000 *C-*. Se aprecia claramente el desequilibrio de clases (1,2%) frente a 98,8 %).

Sea que hemos seguido el proceso de filtrado para la obtención del conjunto de entrenamiento:

1. Para el ***etiquetado***, se parte del conjunto S (muestra) de tamaño 100.000 que *se filtra* mediante una lista de vocabulario obteniendo un nuevo conjunto R de mucha menor cardinalidad (1.000) que *se etiqueta manualmente*, produciendo el conjunto de entrenamiento T de tamaño 1.000, a partir del cual se procederá a la estimación de los modelos de clasificación como muestra la Figura 8‑3.



Figura 8‑3: Prefiltrado de tuits

Seguidamente se procede a la prueba de los modelos (Bayes, K-NN, Redes Neuronales,…) a fin de seleccionar el más adecuado.

1. Para la ***clasificación***, se opera también en dos fases:
   1. *Filtrado* del conjunto inicial;
   2. *Clasificación* del conjunto filtrado.



Figura 8‑4: Clasificación tras filtrado

Este procedimiento presenta un serio e inevitable problema: su *exhaustividad* es siempre muy pequeña, como se ve en el siguiente ejemplo:

1. ***Matriz de confusión del conjunto F***:



Figura 8‑5: Matriz de confusión de conjunto prefiltrado

* + Precisión: 95%
  + Exhaustividad (*respecto al conjunto F*) : 90%

1. **Matriz de confusión en el conjunto *S***: En este caso, los casos que no pasan el filtro se clasifican como C+, por lo cual:

Figura 8‑6: Matriz de confusión con prefiltrado para el conjunto S

Y se obtiene:

* + La misma precisión: 95%
  + Exhaustividad (*sobre conjunto S*): 15%

Con lo cual, el método es de ***baja exhaustividad***.

## Clasificación sin prefiltrado.

Cabe pensar en una clasificación sobre el conjunto sin filtrar



Figura 8‑7: Clasificación sin prefiltrado

Sin embargo, es evidente que al aumentarse el ruido en el conjunto de entrada, la precisión debe reducirse y, a cambio, aumentará la exhaustividad, como p.e. según los resultados siguientes:



Figura 8‑8: Matriz de confusión sin prefiltrado previo

De donde se obtiene:

* + Precisión: 61%
  + Exhaustividad: 66%

## Conclusión.

Es claro que:

1. Resulta ***imposible la clasificación no supervisada****.*
2. El gigantesco desequilibrio existente entre las clases implica la ***necesidad de*** ***etiquetar un conjunto prefiltrado*** que se usará como conjunto de entrenamiento.
3. Si se hace una clasificación de tuits tras el filtrado, *la precisión puede ser elevada*, pero la ***exhaustividad*** de la clasificación será ***necesariamente pequeña*** ya que ***los tuits*** que no pasan el filtro se califican como C-, con independencia de su clase. Esto es un problema relativo para aplicaciones como la detección de tuits de odio en que entiendo que se persigue identificar el mayor número posible de ***fuentes*** de mensajes de odio, lo que no pareces afectado: *se pierde exhaustividad en mensajes*, pero este problema *no afecta tanto a las fuentes*.

1. En España en 2015 [1] [↑](#footnote-ref-1)
2. Utilizamos la terminología económica de G. Becker [2] [↑](#footnote-ref-2)
3. O *fuente de opinión.* [↑](#footnote-ref-3)
4. Es decir, diferencias significativas entre las probabilidades *a priori*. [↑](#footnote-ref-4)
5. ***R****eceiver* ***O****perating* ***C****haracteristics*. [↑](#footnote-ref-5)
6. Con *SnowballStemmer.* [↑](#footnote-ref-6)
7. Utilizaremos en castellano *troncalización* para la palabra *stemming*. [↑](#footnote-ref-7)
8. Existe una versión - *SAS university edition* - que puede utilizarse temporalmente de manera libre (<https://www.sas.com/en_us/software/university-edition.html>) pero no incluye las herramientas de minería de datos. [↑](#footnote-ref-8)