UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Detección de lenguaje sexista en documentos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Pau Ramos Talens

Tutor: Carlos David Martínez Hinarejos

[2017-2018]

Resumen

Este trabajo presenta un estudio basado en modelos de clasificación automática, con el objetivo de distinguir frases sexistas dentro de documentos, especialmente documentos públicos. Se han creado y entrenado diferentes modelos de clasificación a partir de un corpus extraído directamente de documentos oficiales.

Palabras clave: Sexismo, clasificación, aprendizaje automático, Inteligencia Artificial, Igualdad.

Abstract

This research work presents an activity based on automatic classification models, focused on distinguish sexist sentences from documents, in special for public documents. Machine learning has been used together with a corpus obtained from official resources in order to analyze different classification models.

Keywords : Machine learning, sexism, classifiers, artificial intelligence, classification models.

Tabla de contenidos

1. Introducción

Los documentos hoy en día representan una gran parte de la información actual escrita. En todos sus ámbitos, su finalidad es la de comunicar. La naturaleza de cada documento varía según su finalidad: mientras que algunos son para dar a conocer productos o necesidades, otros son para entretener o informar.

Los documentos oficiales son emitidos por los organismos públicos, y cobran gran importancia por la fuente que la emite. Los artículos a tener en cuenta en el ámbito de este proyecto son los Boletines Oficiales del Estado (BOE), recogidos en la web del estado www.boe.es , donde diariamente se publican las leyes, disposiciones, actos de inserción obligatorias, normas con rango de ley, convenios internacionales, convocatorias, citaciones, resoluciones ministeriales, de órganos constitucionales del estado, oposiciones y concursos, nombramientos, contrataciones en el sector público y otros anuncios oficiales.

Anteriormente, dichos BOE solo eran impresos en papel, pero desde el 1 de enero de 2009 se han ido publicando de forma oficial y auténtica en la web pertinente, donde para confirmar su autenticidad se dispone de una firma digital avanzada que no se trata en este estudio. El formato en que se encuentran disponibles estos documentos son en PDF o bien el texto con sus metadatos en XML [1].

El lenguaje empleado en los BOE es puramente administrativo, con un bajo nivel de sexualización. Aun siendo así, en la sociedad se emplea un lenguaje más laxo, con prejuicios a nivel económico, social, cultural y sexual. De esta forma, no solo en una conversación, el mensaje se puede adulterar con un propósito de manipulación hacia el receptor. Aunque no siempre haya una voluntad directa también existe y se da la posibilidad de usar el lenguaje de forma incorrecta llegando a dañar a ciertos colectivos. Por desgracia, este tipo de eventos también suceden en la redacción de documentos públicos.

a. Objetivos

En lo que este estudio profundiza es en el área del lenguaje sexista, siguiendo para su correcta definición la guía “Guía de uso para un lenguaje igualitario” de la unidad de igualdad de la Universitat de València [2], empleada para la realización de este trabajo.

El objetivo que se ha pretendido completar es, mediante modelos de aprendizaje [3] tales como Naive Bayes, árboles de decisión o *Support Vector Machines*, la detección automática de lenguaje sexista en publicaciones del BOE. Estos modelos se han ido entrenando con un conjunto de datos cuidadosamente tratados para posteriormente validar y comparar diferentes modelos entre sí, para ver cuál de ellos se ajusta mejor al tipo de datos y cuál sería el mejor modelo de clasificación para nuevos documentos.

b. Utilidad

Estos modelos, una vez entrenados, pueden tener una gran utilidad de forma totalmente directa hacia la administración pública y la calidad de los documentos publicados. Sería una aplicación directa, por ejemplo, una vez redactado un boletín oficial del estado, antes de su publicación, poder analizar el texto actual contrastándolo contra el modelo y que, bien mediante una aplicación o bien un *plug-in* para un editor de texto, poder detectar frases posiblemente sexistas y corregir su contenido antes de su publicación, adaptándolo a las normas del lenguaje sin atentar contra valores dañinos de éste. El proceso podría completarse por la persona encargada de la redacción dotándola de los conocimientos necesarios, o bien por alguien en calidad de experto o experta que podría revisar las posibles frases detectadas y hacer los cambios que sean oportunos.

Aunque más adelante se hablará de los resultados obtenidos, si actualmente se quisiera empezar a revisar las publicaciones, no sería necesario que la persona encargada de dicha revisión tuviera que comprobar por completo los artículos, sino que, aunque no se obtengan unos resultados exactos, se podría eliminar una parte del proceso de revisión actuando solamente contra las frases destacadas si el *recall* [4] [5] es alto aunque la precisión no tanto, o por el contrario obviando una parte del texto dándolo por analizado si la precisión es alta pero no tanto el *recall*.

Por esto, una aplicación inmediata sería posible.

2. Recopilación de datos

Los datos de entrenamiento son indispensables para los modelos en cualquier proyecto de clasificación, supervisada o no [6], aunque existen diferentes conjuntos de datos, como más adelante se explica; en este estudio solo existe un conjunto de datos, en lugar de la típica disposición de tres conjuntos que son: datos de entrenamiento, validación y test. Con un solo conjunto de datos con la misma cantidad de muestras que la que habría en los diferentes conjuntos convencionales y con una serie de técnicas experimentales se consigue un mejor aprovechamiento de los mismos.

a. Obtención de los datos

Los datos se han obtenido de los boletines oficiales del estado que han sido publicados en su web correspondiente. Cada día los organismos del estado publican nuevos documentos de diferente índole, que, de forma aleatoria, se han ido utilizando para formar el corpus.

Para el corpus se han utilizado 5000 frases; la forma de proceder ha sido: a mano se han ido seleccionando frases en texto plano de unas 8 a 14 palabras aproximadamente, siempre intentando dotar de sentido completo todas las frases (es decir, se han eliminado frases con solo fecha, donde solo había entradas de texto estructural, tales como un número señalizando nueva sección o títulos breves o una enumeración de artículos legales, localizaciones, nombres o demás ámbitos nada influyentes para el estudio). Se ha intentado conseguir frases completas sin cortes, pero el lenguaje por lo general es muy complejo, con lo que la mayoría de las oraciones contienen más palabras que las necesarias para nuestro corpus; por tanto, siempre se han cortado frases largas en los signos de puntuación, como los puntos, las comas, los dos puntos, punto y coma, etc., o bien separando frases compuestas o subordinadas de forma que la pérdida de significado sea mínima.

Al final, cada una de las oraciones se ha etiquetado. La forma de etiquetación, puesto que el problema descrito consiste en dos clases, ha sido en escribir la etiqueta SEXISTA o bien NOSEXISTA siempre como última palabra; así, de esta forma, posteriormente va a ser más fácil el procesamiento del lenguaje simplemente accediendo a la última palabra o bien a la etiqueta determinada. Como ejemplo se expone una frase del corpus a analizar:

<El presente real decreto se aplicará a las organizaciones de productores. SEXISTA>

Esta muestra real del corpus empleado para el entrenamiento de los modelos se guarda en un archivo `txt` tal como se ve en el ejemplo, pero sin estar "< >" incluidos. Según el artículo "Guía de uso para un lenguaje igualitario", esta frase sería etiquetada como sexista al generalizar el sexo de los productores sin que sean productores específicos del género masculino, por no citar los nombres exactos y referirse con ello a una cantidad indeterminada de personas sin importar su género.
En cuanto a la corrección de la oración, se podría actuar de diferente manera; se puede incluir la versión femenina de esta palabra a continuación, quedando como sigue:

<El presente real decreto se aplicará a las organizaciones de productores y productoras.>

Bien podría desdoblarse el género al final de la palabra:

<El presente real decreto se aplicará a las organizaciones de productores/as.>

Otra opción sería reescribir la frase de forma que no aparezcan palabras marcadas o bien usar palabras con género neutro como puede ser "personal" en lugar de empleados, "tripulación" en lugar de marineros, "agentes" en lugar de oficiales o policías.

<El presente real decreto se aplicará a las organizaciones cinematográficas.>

Otro ejemplo con la etiqueta contraria podría ser:

< Las Partes en el Tratado del Atlántico Norte. NOSEXISTA>

Estas diferentes frases se guardan por separado junto con su etiqueta en un archivo con extensión `txt`, todo en texto plano tal cual aparece en el texto original.

Los archivos se nombran numéricamente desde el número `0001.txt` hasta el `5000.txt` y se almacenan todos dentro de una misma carpeta.

Con estos ejemplos se da a conocer el método de clasificación y se visualizan los datos para tener una idea clara de en qué consiste el corpus.

Adicionalmente, cabe mencionar que se han obtenido una proporción de 4500 frases no sexistas frente a 500 sexistas; esto es debido a que se han ido analizando BOEs de forma secuencial sin discriminar por tema. Esto ha resultado en que en algunos boletines su tema principal resulta ser normativa o procedimientos como, por ejemplo, un manual de normas marítimas o ayudas y subvenciones a vehículos, donde un ámbito sexista es muy poco probable por la naturaleza poco personal de los temas tratados. Esta proporción se ha considerado como correcta por la utilidad final pensada de analizar todos los textos antes de ser publicados sin filtrar por su naturaleza.

Se debe mencionar que la cantidad de muestras de una etiqueta y de la otra influye directamente sobre los resultados de los modelos una vez entrenados.

b. Tratamiento de los datos

Una vez obtenidos los datos como se ha explicado, se han querido tratar de forma que sea más fácil utilizarlos y manejarlos, reduciendo así su error y complicación varias como: reconocimientos erróneos de signos de puntuación debido a la codificación empleada, el cambio de `UTF-8` a `ASCII` [7] o diferentes formatos de texto como `PDF, txt, docx` [8].

Para esta tarea se ha desarrollado un software básico implementado en Java donde se selecciona una ruta específica, carga los 5000 archivos, los modifica y los guarda correctamente en otra ruta específica.

Aunque no es objetivo de esta memoria exponer el código desarrollado, sí se va a explicar la funcionalidad del mismo.

Estos documentos pasan por unas subrutinas donde en cada una de ellas se modifica una parte de las siguientes:

- Primero, se dispone el texto entero en una línea, puesto que al ir insertando frases en los `txt` no siempre quedaban todos a la misma altura.
- Segundo, se eliminan todos los caracteres en mayúscula; de esta forma se elimina el doble etiquetado que una misma palabra puede tener, por ejemplo, la palabra “aquí” y “Aquí” serían tratadas como dos palabras totalmente distintas cuando se trata de la misma, pero con la diferencia de que una está al principio de la oración y la otra no. Al ser tratadas

como distintas, posteriormente los modelos podrían tratarlas como a dos palabras sin relación entre ellas, dando más peso o etiquetándolas de forma contraria.

- Tercero, se quitan todos los signos de puntuación como tildes, comas, puntos, comillas, barras, operandos, números y demás que no sean las letras de la A a la Z.
- Cuarto, se eliminan expresiones de introducción estructural tales como enumeración de apartados como a), ii) … también elementos aislados como siglas por ejemplo NASA, BOE, etc.
- Quinto, se eliminan palabras innecesarias, tales como conjunciones y preposiciones [9] puesto que finalmente no van a tener un peso real en la decisión de las etiquetas, pero aumenta el tamaño del diccionario y así aumenta el cómputo y dificultad del problema.
- Finalmente, se elimina la etiqueta que hay al final de cada documento y, según el etiquetado cada oración, se guarda en una carpeta con su etiqueta, donde quedarían todas las frases no sexistas dentro de una carpeta llamada NOSEXISTA y todas las sexistas en una carpeta paralela llamada SEXISTA. El motivo para esto reside en que al cargar los datos posteriormente en el software para su interpretación, requiere que todos los archivos del mismo tipo deben encontrarse dentro de una carpeta con el nombre de su etiqueta, habiendo un número específico de carpetas donde cada una representa una clase; en nuestro caso serían las descritas como SEXISTA y NOSEXISTA.

Adicionalmente, se ha revisado el vocabulario restante, donde, sorprendentemente, aún quedaban muchas palabras que se suponían eliminadas y demás agrupaciones de letras sin sentido. Las palabras que se suponían eliminadas se han acabado de eliminar o reducir a un tamaño despreciable usando expresiones regulares [10] para eliminar palabras problemáticas al principio o final de una frase, palabras compuestas que eliminando su guion que las separa dan problemas, etc.

Respecto a las palabras sin sentido, se han ido eliminando de forma singular como si de siglas se trataran.

c. Detalles a tener en cuenta

El corpus creado ha sido destinado únicamente para entrenar los modelos; con esto se quiere decir que una pequeña parte del código está adaptado específicamente al corpus inicial, por lo que si se quisiera automatizar el proceso de tratamiento de los datos que serían utilizados por la aplicación

real donde se pasan nuevos datos para ser analizados, los nuevos datos no estarían igual de bien tratados, puesto que tanto siglas como palabras sin sentido formadas por letras con función estructural, errores tipográficos, etc., no serían eliminadas automáticamente. Se necesitaría un cierto tiempo para ir adaptando el filtrado o postproceso de las nuevas frases añadiendo nuevas acepciones que vayan apareciendo, o bien mejorando algo el actual tratamiento (como podría ser un diccionario de palabras sin peso, otro diccionario para siglas, otro para nombres de organizaciones o empresas), hasta el punto en que se considere aceptable.

3. Modelos de clasificación

En este apartado se va a explicar en qué consiste un modelo de clasificación, los diferentes modelos empleados en el estudio y sus características. [11].

La Inteligencia Artificial es una rama de las ciencias computacionales que trata de modelar el comportamiento humano por medio de la creación de sistemas que sean capaces de imitar la comprensión humana y que también sean capaces de aprender y reconocer. Dentro de lo que se consideraría inteligencia artificial, aparecería el concepto de modelo de clasificación; se podría definir como un conjunto de técnicas donde, otorgando una entrada a éste modelo, este pudiese analizar los datos y obtener una salida meditada con la solución. Estas técnicas se afinan o mejoran con los datos mediante una serie de algoritmos, de diferente índole según el tipo de clasificador.

a. Modelos utilizados

i. Árbol de decisión (J48)

Los modelos basados en un árbol de decisión, clasificación o regresión, deben su nombre a su particular forma cuando son trasladados al papel, empezando con un único nodo llamado raíz, el cual va ramificándose nodo tras nodo hasta llegar a un último nodo llamado hoja. Hay muchos tipos de árboles y diferentes comportamientos entre ellos; se podría hacer un estudio entero sobre ellos, pero solo se va a explicar su comportamiento básico sin entrar en detalles.

Un ejemplo muy visual, con un problema de clasificación simple y familiar, es el de la discriminación entre diferentes especies de la flor del iris, ejemplo prestado de la asignatura SIN (Sistemas Inteligentes) del grado de informática, que se presenta en la ilustración 1. [12] [13] [14]

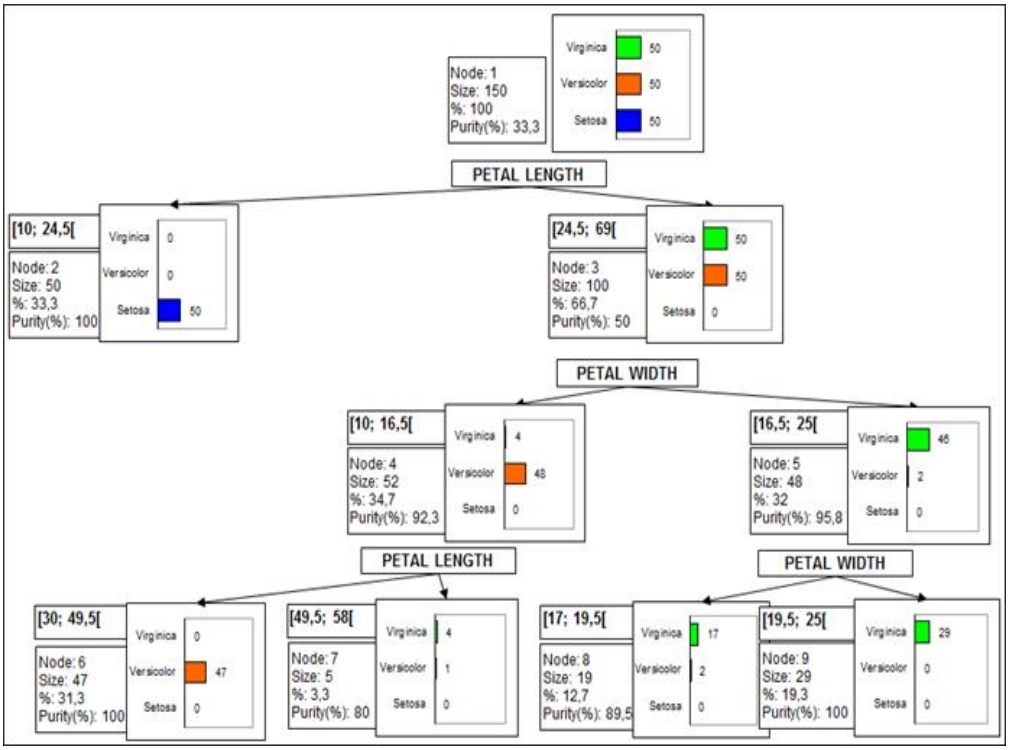

Ilustración I. Ejemplo arbol de decisión

Aquí se puede ver la formación descrita anteriormente, donde el nodo de arriba es la raíz y sus descendientes o ramificaciones llega finalmente a los últimos nodos o nodos hoja.

Se aprecia en la ilustración 1 como cada nodo representa una decisión y sus ramificaciones un camino u opción. En este caso, en cada nodo se discrimina por la longitud o la amplitud de los pétalos de la flor hasta llegar a un resultado. [15].

En el caso de este estudio, para la clasificación del lenguaje en sexista o no sexista el modo de funcionamiento es algo más complicado. En primer lugar, el modelo selecciona un atributo y lo compara con el mismo atributo de un subconjunto; este proceso se repite atributo a atributo hasta hallar el atributo más importante de la muestra o con una ganancia de información mayor, y sobre este se pivota su clasificación.

ii. Naive Bayes

Naive Bayes [16] [17] es uno de los clasificadores más famoso por su simplicidad y rapidez. Como su nombre indica, está basado en el teorema de Bayes:.

$$P(A|B)=\frac{P(B|A)\cdot P(A)}{P(B)}$$

Esta fórmula representa la probabilidad de una ocurrencia (A) sabiendo a priori que otro evento (B) ha ocurrido. Aplicado a un clasificador con diferentes muestras se calcularía la probabilidad de un evento (A) comparándola a los eventos pasados ya conocidos como conjunto de entrenamiento (B1, B2, B3 …).

$$P(A|b_1,b_2,b_3,b_4)=P(A)\cdot(P(b_1|A)\cdot P(b_2|A)\cdot P(b_3|A)\cdot P(b_4|A))$$

En el caso particular de dos clases en la clasificación habría que calcular la probabilidad del evento de cada clase.

$$Solucion=arg\max_{i=1}^{n} P(c_i)\cdot\prod_{j=1}^{m} P(a_j|c_i)$$

Para crear el modelo, el algoritmo seguido es el siguiente:

- Primero, se da una probabilidad a priori a cada clase; la mayoría de veces se inicia siendo equiprobable entre ellas.
- Segundo, por cada clase, hacer un recuento de atributos y sus valores y distribuirlos en tantas tablas como clases.
- Tercero, aplicar una corrección de Laplace, para evitar problemas con los valores a cero.
- Cuarto, normalizar para tener valores entre cero y uno.

Por consiguiente, para clasificar un nuevo dato se contrastarían todos los atributos de la nueva muestra contra los de las tablas creadas.

iii. Redes Bayesianas o *Bayes Net*

Una red bayesiana [18] [19], como el modelo anterior, también depende de las ecuaciones de Bayes. Principalmente, una red bayesiana es representada con un grafo acíclico dirigido, cuyos nodos representan variables y las aristas las relaciones condicionales; cada nodo tiene asociada una función de probabilidad dependiente de los nodos que lo apuntan. Cada variable es representada de forma booleana con sus probabilidades de que el evento ocurra o no. En la Ilustración II, se ve gráficamente en ejemplo de un sistema de riego, donde la probabilidad de que la hierba esté húmeda depende del sistema de riego y de la lluvia. En las tablas anexadas a los nodos se muestran las probabilidades para cada evento y si dependen o no de otras variables.

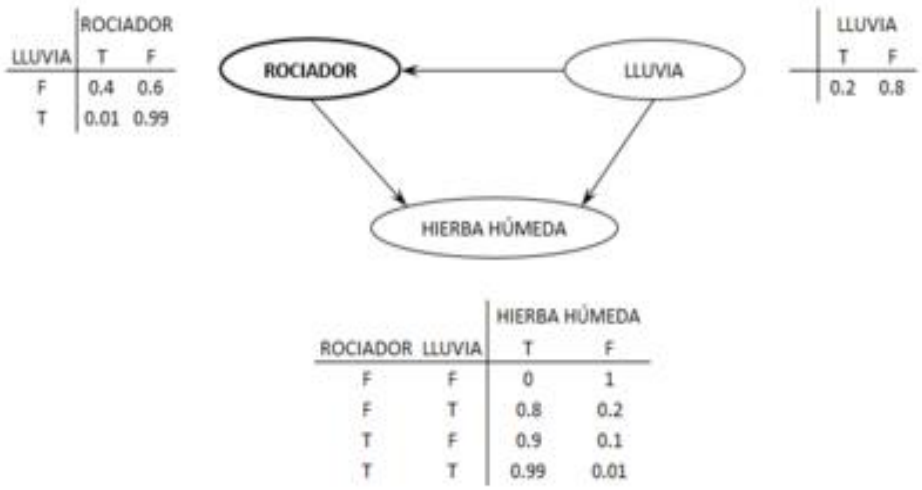

Ilustración II. Ejemplo red bayesiana.

iv. *MultilayerPerceptron* o Redes neuronales

El perceptron multicapa [20] [21] [22] [23] es un modelo basado en nodos distribuidos por capas; dichas capas, según su posición, serán llamadas capa de entrada, capas ocultas y, por último, la capa de salida.

En la capa de entrada se disponen los diferentes datos; en las capas ocultas se combinan los datos entre ellos y se les asigna un peso o error que más adelante se utilizará como un indicador de nivel de importancia de dicho nodo en el cálculo final; por último, en la capa de salida se comprueba la clase real y se hace un recorrido hacia atrás, cambiando los pesos de los nodos anteriores en la capa o capas ocultas según cual había sido su resultado.

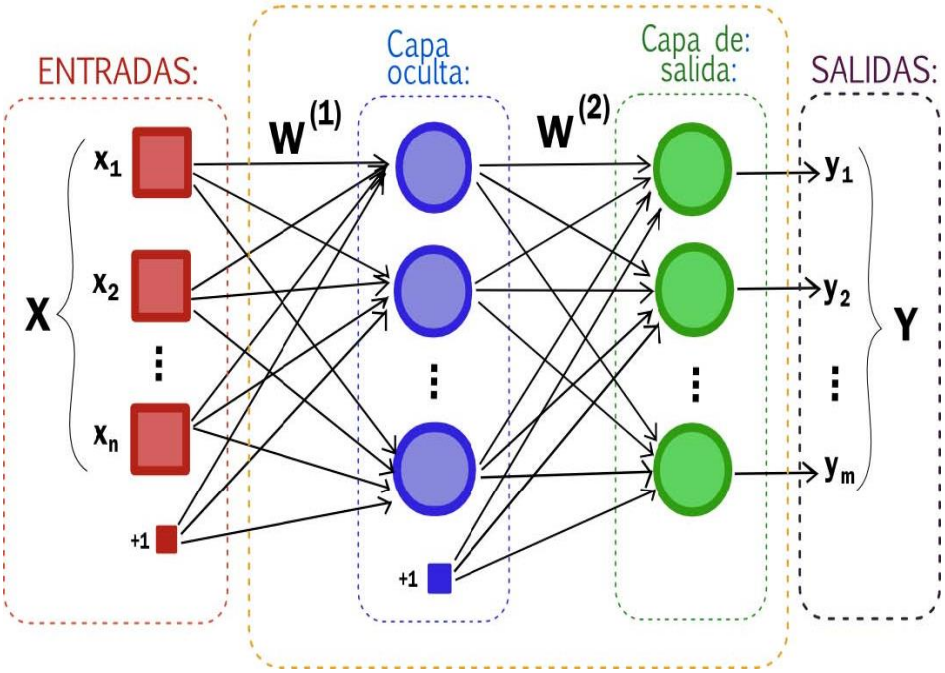

Ilustración III. Arquitectura de una red neuronal.

v. *Support Vector Machine* (SVM)

Las máquinas de vector de soporte [24] conforman una de las más eficaces y más recurrentes soluciones de clasificación.

Inicialmente, hay que tener en cuenta que los datos se representan en un espacio n-dimensional, donde las n dimensiones corresponden al número de atributos de las muestras; en el caso de este estudio sería el número de palabras del vocabulario.

Cada muestra representa un punto en el espacio, que con todas las demás muestras compone un bosque de puntos; el objetivo del modelo es separar en tantos grupos como clases estos puntos de la mejor forma posible, formando hiperplanos de separación.

La forma de calcular estos hiperplanos viene dada por el Kernel utilizado. Estos Kernels representan funciones que equivalen al producto escalar de los datos en un espacio de mayor dimensión donde es más fácil resolver el problema de clasificación.

Algunas de estas funciones Kernel son las siguientes:

- Polinomial-homogénea: $K(x_i, x_j) = (x_i \cdot x_j)^n$
- Perceptron: $K(x_i, x_j) = || x_i - x_j ||$
- Gaussiana: $K(x_i, x_j) = \exp(-(x_i - x_j)^2 / 2(\text{sigma})^2)$
- Sigmoide: $K(x_i, x_j) = \tanh(x_i \cdot x_j - \theta)$

b. Weka

En este apartado se va a explicar el software Weka y sus características principales. [25] [26].

Primeramente, Weka es un software desarrollado en la universidad de Waikato, Nueva Zelanda. En su versión de escritorio (cuya interfaz puede verse en la ilustración IV) se trata de un programa especializado en minería de datos y aprendizaje automático capaz de importar datos, filtrarlos y utilizarlos para crear modelos, entre otras opciones. En general este software, inicialmente poco intuitivo, dispone de una cantidad de herramientas muy amplia y de gran utilidad.

Esta herramienta está creada con Java y es distribuida de forma libre. Al estar creada en Java, junto con su API y documentación aportada, resulta de una gran ayuda a la hora de su utilización, siendo 100% compatible con código Java de propia implementación tan solo importando un JAR.

Entre algunas de sus características se encuentran visualizadores de datos, tablas y agrupamientos de datos respecto a la actividad desarrollándose en el propio programa, un catálogo de filtros para los datos (Ilustración V), su propia extensión de ficheros que Weka reconoce automáticamente (`arff`) (Ilustración VI), una consola por donde ejecutar órdenes y una gran gama de clasificadores y estructuras de datos.

Ilustración IV. Interfaz Weka

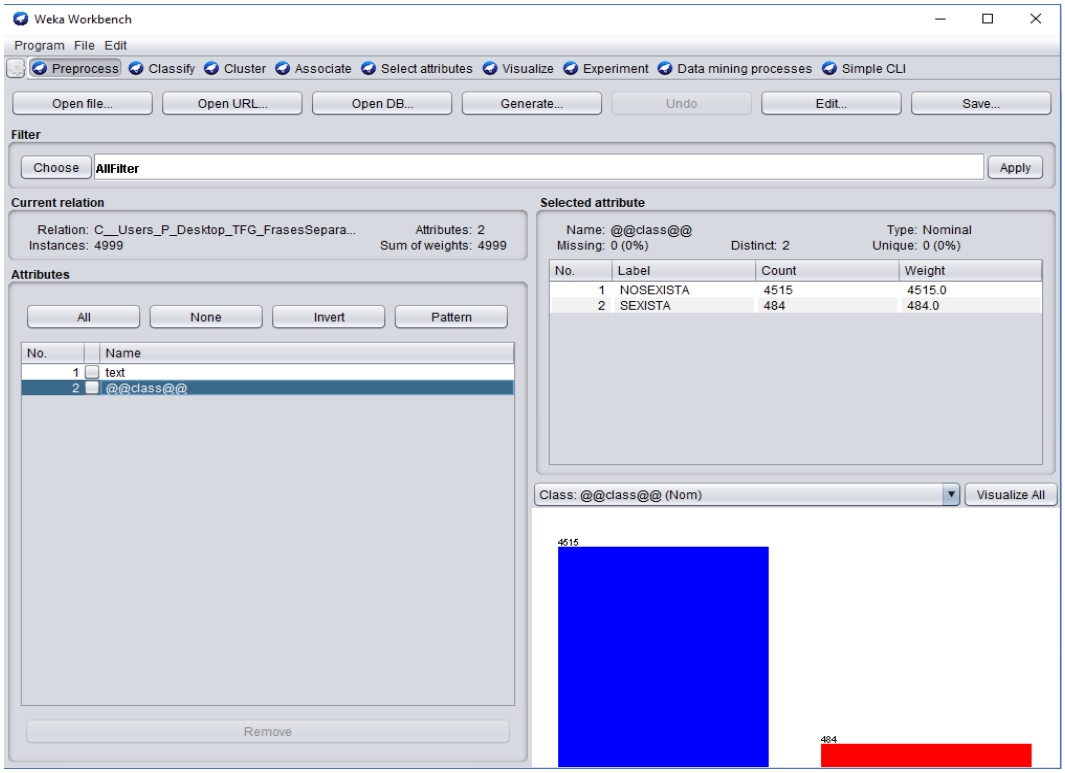

Ilustración V. Interfaz Weka clasificación frases.

```
@relation C__Users_P_Desktop_TFG_FrasesSeparadas

@attribute text string
@attribute @@class@@ {NOSEXISTA,SEXISTA}

@data

'Tengo el honor de hacer referencia a las consultas mantenidas, a nivel técnico, entre la Secretaría del Organismo Internacional de Energía Atómica.',NOSEXISTA
'El Gobierno, con sujeción a sus reglamentos y a la legislación nacional, concederá a los funcionarios y expertos del OIEA, así como a los delegados designados por los Est
y a salir de él.',SEXISTA
'La organización del Curso Regional de Capacitación en Detección y Vigilancia de los Mosquitos y Registro y Análisis de Datos con miras a la Gestión Integrada Zonal de los
2017.',NOSEXISTA
'Los visados necesarios para los funcionarios y expertos del OIEA.',SEXISTA
```

Ilustración VI. Archivo arff.

4. Experimentación y resultados

a. Software creado

Para la experimentación con los datos previamente se han filtrado y modelado; se han trasladado de 5000 archivos `.txt` a un único archivo enriquecido con formato `arff`.

Este archivo ha sido creado de forma estándar mediante la consola de Weka, donde para su creación se necesita tener separadas en carpetas diferentes cada conjunto de frases con la misma clase; después se ha ejecutado la orden adecuada para automáticamente generar un documento `arff,` que contiene unas anotaciones para Weka y las 5000 frases correctamente etiquetadas.

Una vez se ha obtenido este archivo, se ha procedido a programar la forma de crear diferentes modelos y experimentar con ellos. Para ello se explica a continuación los pasos seguidos, con la ayuda de los ejemplos proporcionados. [27].

Se importa el archivo contenedor de las 5000 frases que anteriormente se ha creado de forma estándar en cualquier archivo de texto. A continuación, se almacena en una variable dicha información, que va pasando por diferentes métodos donde se le aplica un filtro que convierte las `String` a vectores de palabras para su correcta manipulación. Sin este filtro, las librerías de Weka no serían capaces de leer los datos. El siguiente paso después del filtrado es la instanciación del modelo que se va a querer entrenar; con esto se crea un modelo vacío y sin datos; se ajustan las diferentes opciones disponibles para dicho modelo; en su siguiente paso, y mediante la técnica de validación cruzada [28] [29], se entrena el modelo creado; finalmente se estima su nivel de eficiencia mediante el nivel de aciertos y fallos y los diferentes tipos de errores, también con la precisión y *recall* posteriormente explicados.

La técnica de validación cruzada (o *cross validation*) es una manera de entrenar los modelos de forma que los datos obtenidos se utilizan al máximo. La forma normal del entrenamiento consiste en tener dos conjuntos de datos de diferente tamaño y con el de mayor tamaño (por ejemplo un 80% de los datos totales), utilizarlos para entrenar el modelo y los restantes (el 20%), para comprobar su grado de optimalidad. Mediante *cross validation*, los datos totales son particionados también en dos subconjuntos, pero esta vez se toman N particiones del total de datos, usando N-1 para entrenar y 1 para

validar. La gran diferencia es que este proceso se repite varias veces cambiando las particiones escogidas; así se consigue con un solo conjunto de datos la simulación de tener varios conjuntos.

Específicamente, en este estudio se ha utilizado la validación cruzada de 10 vueltas. Esta opción es la que viene por defecto en la librería de Weka y se ha seguido el consejo, puesto que en la mayoría de veces que se ha experimentado los parámetros estándar suelen ser óptimos.

b. Modelos y opciones

Los modelos empleados son los que se han descrito en el capítulo 3, apartado a. La implementación o algoritmo interno de estos modelos son los descritos anteriormente, aunque los detalles de la implementación usados en Weka pueden variar mínimamente.

Los parámetros disponibles para cada modelo son diferentes, pero todos ellos se aplican de la misma forma, en forma de atributos en una `String` en la función definida para las opciones.

Las opciones disponibles para cada modelo son las siguientes:

- J48: [30].

  -U. No podar el árbol.

  -O. No comprimir la estructura del árbol.

  -C<pruning confidence> default 0.25. Cambia el valor de la poda, siendo más o menos optimista. [31].

  -M<minimum number of instances> default 2. Mínimo número de instancias por hoja.

  -R. Usar reducción de error en la poda.

  -N<number of folds> default 3. Asignar número de vueltas en la reducción de error de poda.

  -B. Usar solo particiones binarias.

  -S. No realizar reemplazo por sub árboles.

  -L. No limpiar al construir el árbol.

  -A. Usar un filtro de suavizado de Laplace.

  -J. No usar corrección MDL para la ganancia de información en atributos numéricos.

  -Q<seed> default 1. Cambiar la semilla.

  -doNotMakeSplitPointActualValue. No utilizar un valor real como punto de división.

- Naive Bayes: [32].

  -K. Usar el estimador de densidad del kernel en lugar de la distribución para atributos numéricos.

  -D. Usar discretización supervisada para atributos numéricos.

  -O. Visualizar el modelo en formato antiguo.

- Bayes Net: [33]

  -D. No usar la estructura de datos ADTree.

  -B<BIF file> cargar un archivo BIF para compararlo. [34]

  -Q weka.classifiers.bayes.net.search.SearchAlgorithm. Algoritmo de búsqueda.

  -E weka.classifiers.bayes.net.estimate.SimpleEstimator. Se usa para estimar probabilidades en las tablas de una red de Bayes. [35].

- SMO o SVM:

  -no-checks. Desactivar comprobaciones.

  -C<double> default 1. La constante de complejidad C.

  -N. 0 para normalizar, 1 para estandarizar y 2 para ninguna.

  -L<double> default 1.0e-3. Parámetro de tolerancia.

  -P<double> default 1.0e-12. Epsilon para el error de redondeo.

  -M. Calibración de la salida de los SVM.

  -V <double> default -1. Numero de vueltas para el *cross validation* interno.

  -W<double> default 1. Semilla.

  -K. El tipo de kernel a usar. [36].

  -calibrator<scheme>. Calibrador de modelo.

  -output-debut-info. Habilitar modo de información.

  -do-not-check-capabilities. No verificar capacidades.

  -num-decimal-places default 2. Numero de lugares para la salida de los números del modelo.

c. Modo de experimentación

Una vez expuesto y explicado las opciones de los diferentes clasificadores, se va a detallar el modo en que se ha experimentado con los modelos. El objetivo seguido ha sido, mediante las pruebas empíricas, mejorar los

resultados ya obtenidos de los modelos estándar sin haber cambiado ningún parámetro. Para conseguir dicha mejora con la cantidad de diferentes opciones que existen se ha procedido a realizar una técnica bastante empleada, que consiste en barrer opción a opción buscando el mejor valor para cada una de ellas. Un ejemplo más visual sería: se dispone de un modelo con dos tipos de parámetros, el parámetro X e Y, el modo en que se ha operado ha sido escoger uno de las opciones al azar y probar sus extremos. Se selecciona el atributo X cuyo valor por defecto es 0,5 en un rango del 0 al 1; se procedería a probar el 0 y luego el 1, a continuación, y dependiendo de los resultados se probaría el 0,2 y el 0,7; una vez se han obtenido los resultados se comparan entre ellos y se sigue con la experimentación de otros valores. Pongamos el ejemplo de que el mejor resultado obtenido ha sido con el valor X a 0; sabiendo esto ahora se elegiría el 0,1; de esta forma se irían discriminando valores hasta llegar al más idóneo; si esta opción acepta más decimales, también sería posible discriminar de forma más precisa del mismo modo anterior, hasta llegar a la precisión decimal que sea adecuada. Una vez el parámetro X sea óptimo, con el valor de éste fijo, se empieza el mismo procedimiento con el valor Y.

Para poder valorar los resultados obtenidos en cada clasificador con sus diferentes opciones, los datos claves van a ser tanto la precisión como el *recall*.

Tanto la precisión como el *recall* se pueden obtener de un modelo entrenado de forma directa en Weka con una simple instrucción. Para conocer mejor la optimalidad de un modelo y por qué se basa en estos dos datos, es necesario que sepamos qué representan: [37] [38]

- La precisión como en la ilustración VII se ve, representa la cantidad de datos que se han clasificado correctamente del total de ellos.

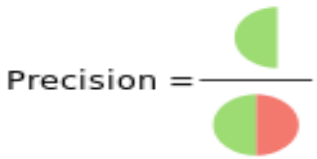

Ilustración VII. Definición gráfica de precisión.

- El *recall,* ilustración VIII, se define como el número de datos clasificados correctamente como una clase determinada, del total de datos existente de esta clase.

Ilustración VIII. Definición grafica de recall

Para no disponer de dos datos independientes, se puede calcular un único valor derivado de ellas, conocido como *f1 score,* siendo:

$$f1 = 2 \cdot \frac{\text{precisión} \cdot \textit{recall}}{\text{precisión} + \textit{recall}}$$

d. Resultados

Una vez expuestos las diferentes opciones de los modelos se procede a exponer los diferentes datos obtenidos a través de la experimentación con los modelos, teniendo en cuenta que se han ido probando los parámetros de cada modelo más principales y de lo que se podría esperar un cambio de resultado.

Tabla 1. Resultados iniciales modelo J48

	J48	J48 -U	J48 -A
Aciertos	91.55 %	91.1382 %	91.55 %
Precisión:	0.758	0.598	0.758
Recall:	0.188	0.258	0.188
F1 score:	0.301	0.360	0.301

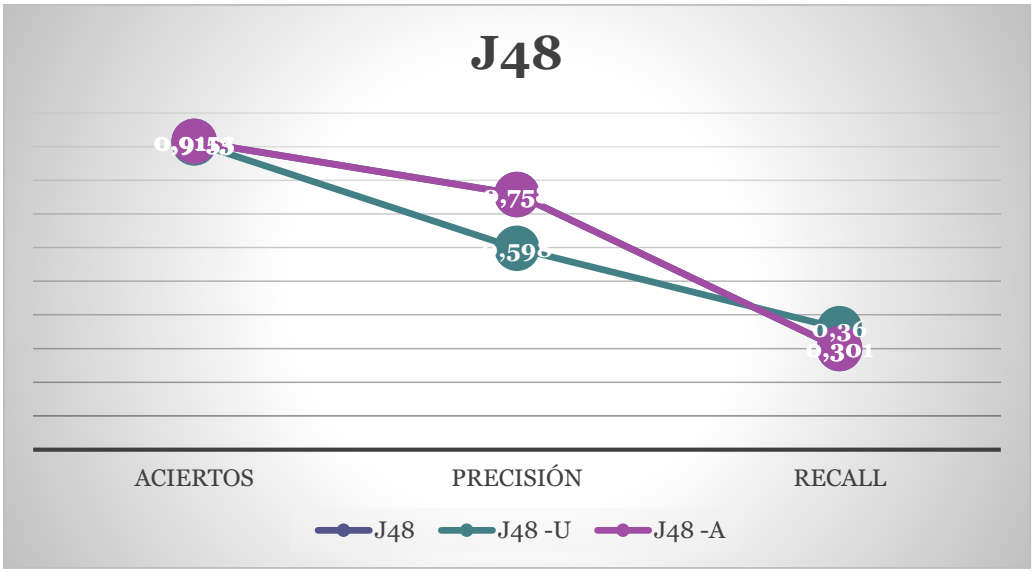

Gráfico I. Modelo J48.

Tabla 2. Modelo J48 modificando el valor -C .

	J48 -C 0.1	J48 -C 0.3	J48 -C 0.355	J48 -C 0.5
Aciertos	91.3383 %	91.7383 %	91.7784 %	91.5383 %
Precisión:	0.774	0.755	0.741	0.674
Recall:	0.148	0.216	0.231	0.243
F1 score:	0.242	0.335	0.352	0.357

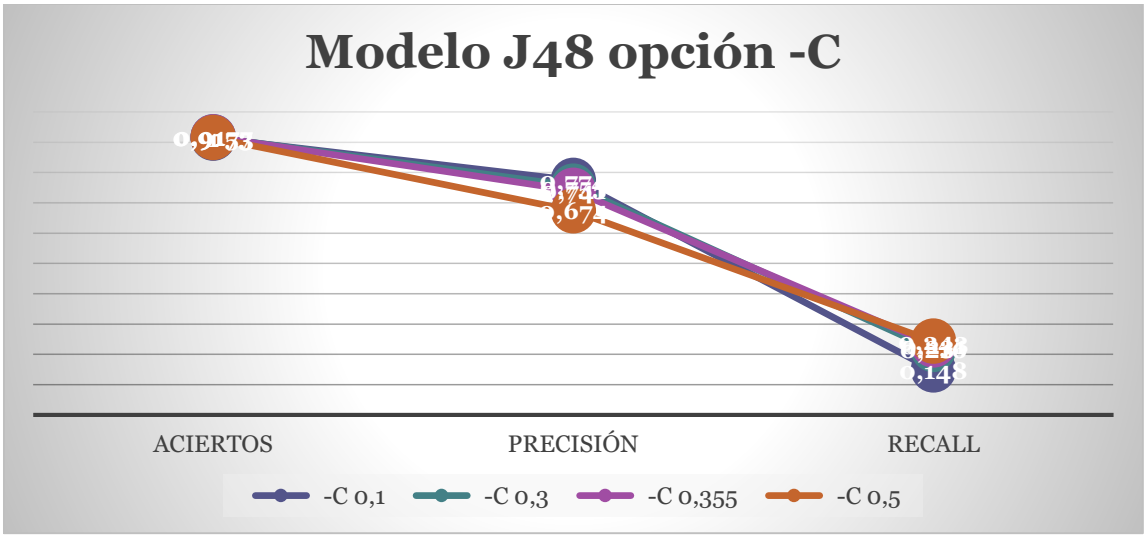

Gráfico II. Modelo J48 opción –C.

Tabla 3. Resultados modelo J48 con valor –C 0.355 para diferentes valores de -M

	J48 -M 0	J48 –M 3	J48 -M 10	J48 -M 25
Aciertos	91.67 %	91.61 %	91.07 %	90.97 %
Precisión:	0.686	0.768	0.771	0.923
Recall:	0.258	0.192	0.111	0.0743
F1 score:	0.374	0.307	0.192	0.137

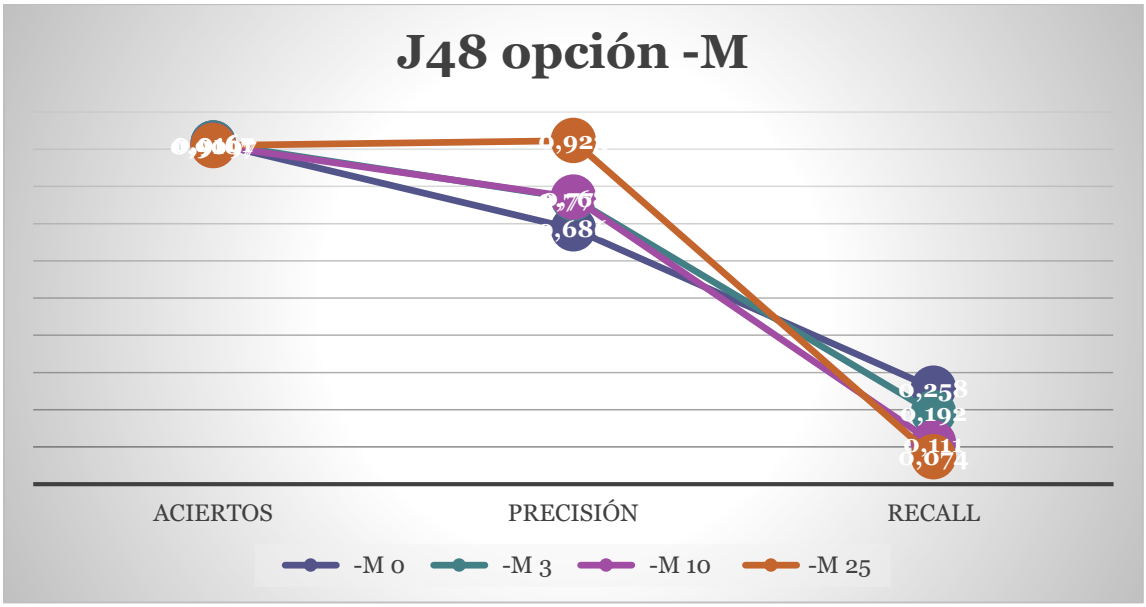

Gráfico III. Modelo J48 opción –M

Tabla 4. Resultados del modelo Naive Bayes con diferentes opciones.

	Naive Bayes	Naive Bayes -D	Naive Bayes -K	Naive Bayes setUseKernel Estimator(true)
Aciertos	91.6183 %	93.4787 %	93.1986 %	93.1986 %
Precisión:	0.696	0.821	0.797	0.797
Recall:	0.237	0.417	0.398	0.398
F1 score:	0.353	0.553	0.552	0.530

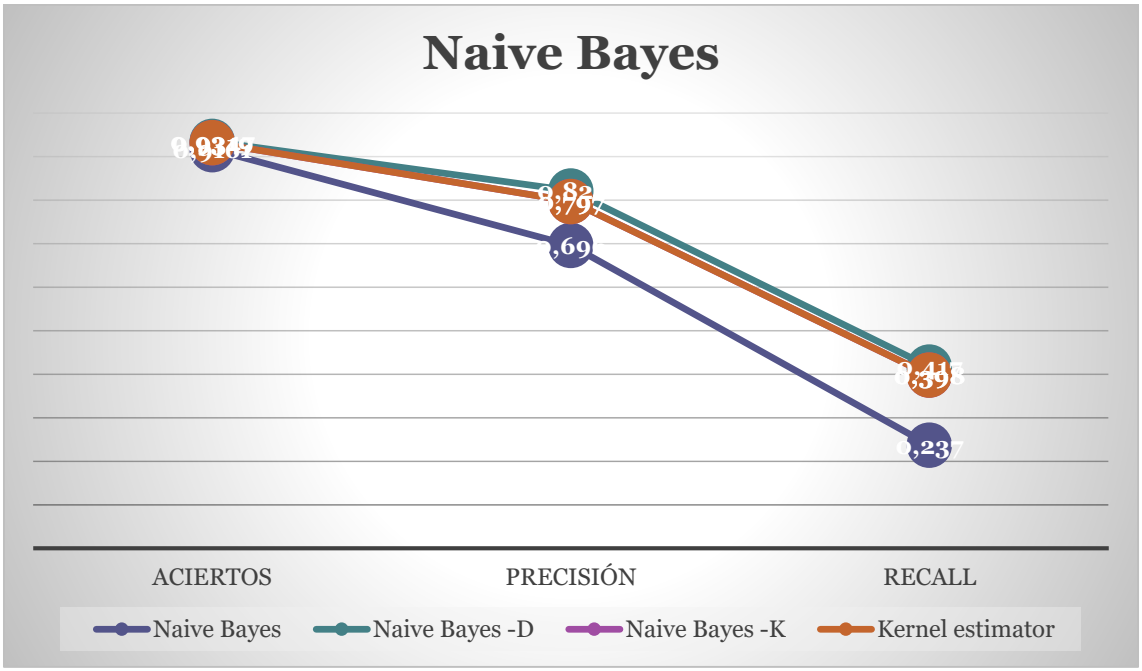

Gráfico IV. Modelo Naive Bayes.

Resultados modelo Bayes Net:

Tabla 5. Resultados modelo Bayes Net con diferentes opciones.

	Bayes Net	Bayes Net -D	Bayes Net *
Aciertos	93.47 %	93.47 %	93.55 %
Precisión:	0.816	0.816	0.849
Recall:	0.421	0.421	0.407
F1 score:	0.555	0.555	0.550

Bayes Net *= -Q weka.classifiers.bayes.net.search.local.K2 -- -P 2 -S ENTROPY –E

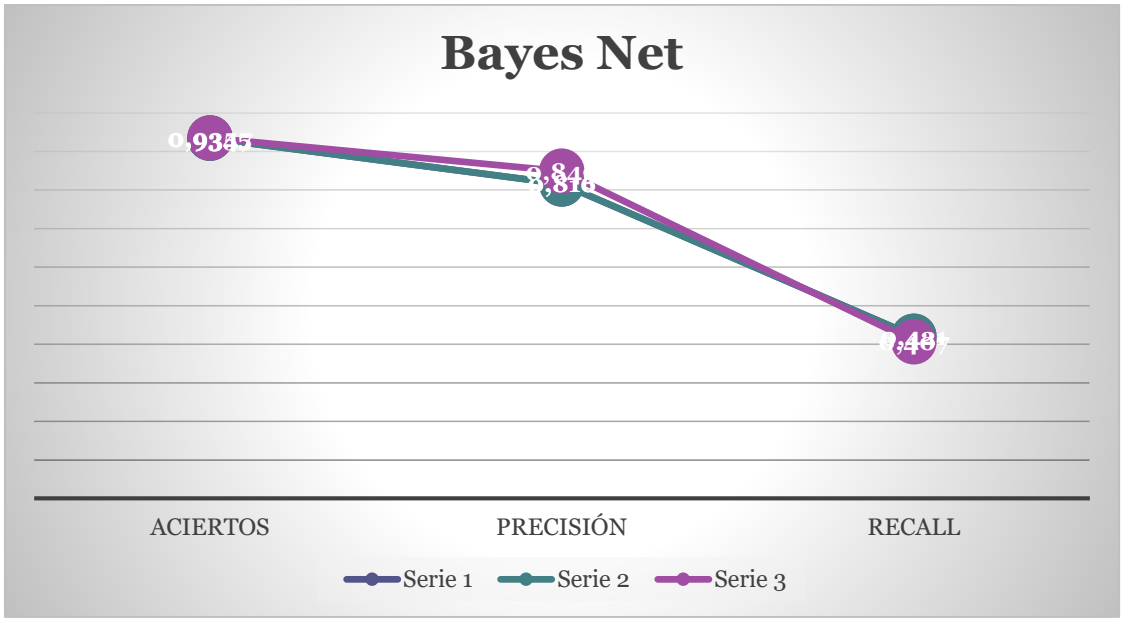

Gráfico V. Modelo Bayes Net.

Tabla 6. Resultados para el modelo SMO con PolyKernel y diferentes opciones para parámetro –C.

	SMO	SMO –C .01	SMO –C 2	SMO –C 0.4
Aciertos	94.23%	93.37%	93.81%	94.60%
Precisión:	0.781	0.900	0.735	0.829
Recall:	0.562	0.355	0.564	0.551
F1 score:	0.653	0.509	0.638	0.662

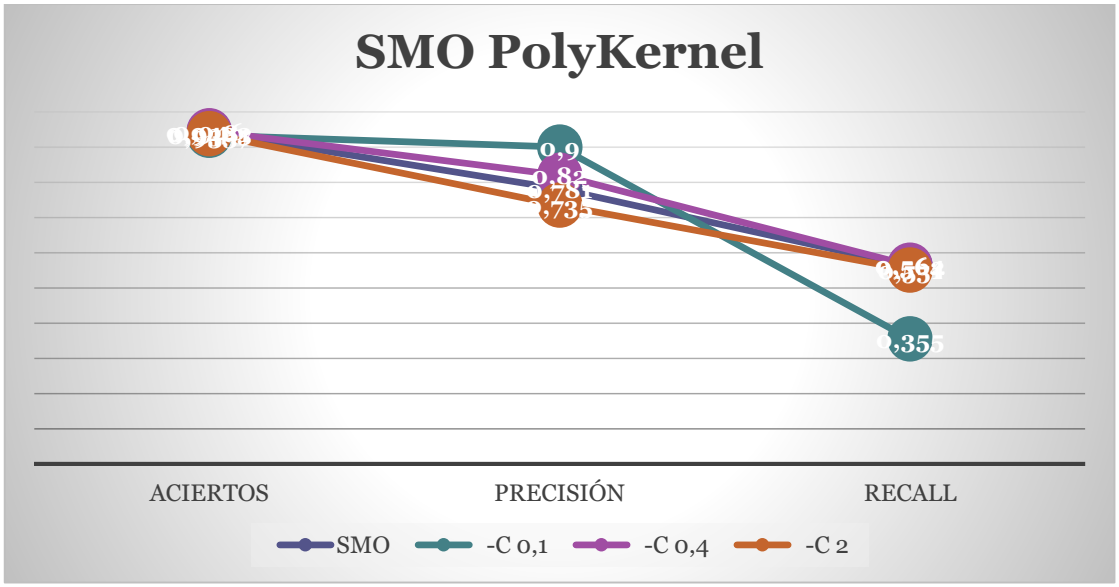

Gráfico VI. Modelo SVM con PolyKernel.

Tabla 7. Resultados para el modelo SMO con RBF Kernel variando parámetro –C.

	SMO –C 2	SMO –C 10
Aciertos	91.19%	94.51%
Precisión:	0.94	0.852
Recall:	0.097	0.524
F1 score:	0.176	0,649

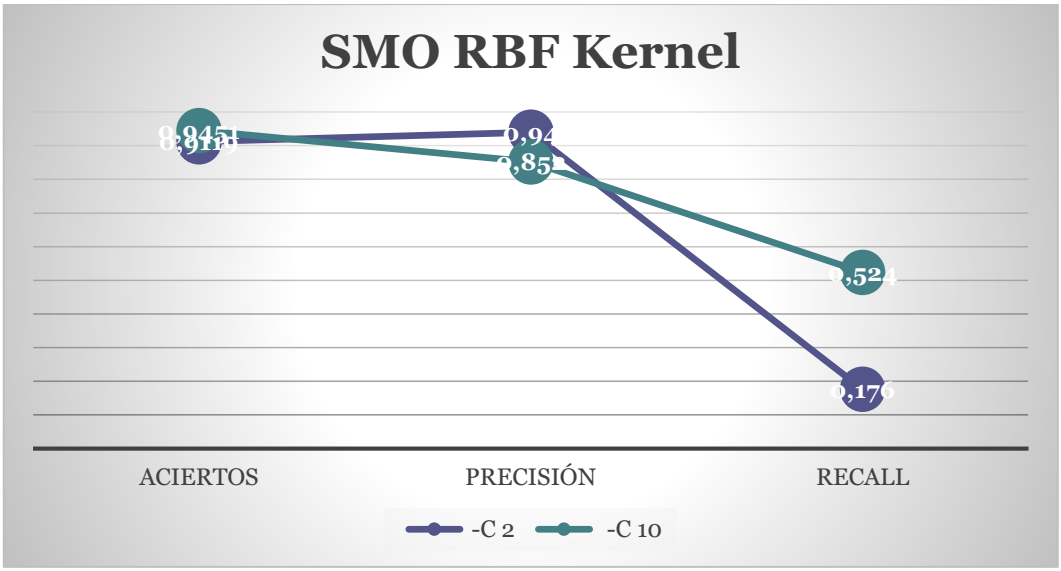

Gráfico VII. Modelo SVM con RBF Kernel.

Los resultados obtenidos varían bastante solo por tratarse de modelos diferentes. Para cada modelo sus opciones proporcionan diferentes resultados, coincidiendo en que cuando su precisión aumenta, disminuye el *recall* y, al contrario. Se ha dado una importancia parcialmente mayor al *recall* que a la precisión, dado que con una precisión perfecta pero un *recall* muy bajo, la cantidad de trabajo aprovechable realizada por el clasificador sería muy baja y, aunque sea así, se suele premiar un mayor número de resultados con un error bajo que según qué aplicación se podrá asumir o despreciar, aunque la decisión final será tomada por el tipo de servicio que sería prestado.

Todos los modelos en su modalidad estándar han resultado ser óptimos o cerca del resultado más óptimo alcanzado en cualquier variante de sus opciones.

El árbol de decisión J48 presenta los peores resultados de los diferentes modelos, además que también, de forma anecdótica, ha sido el modelo que

más tiempo emplea entrenándose, de unos 10 a 15 minutos, dependiendo de sus opciones. Este modelo ha sido el más experimental, debido a su cantidad de opciones ya que no es muy empleado para este tipo de clasificaciones se ha querido dar una opción diferente. Los resultados obtenidos se pueden ver en la tabla 1, la tabla 2 y la tabla 3; también en el archivo descargable en el apartado contenidos de este documento.

El clasificador Naive Bayes sigue de cerca los resultados obtenidos por el modelo anterior, superando su puntuación significativamente con su opción –D con discretización. También se ha reducido el tiempo de entrenamiento considerablemente, como máximo 5 minutos. Sus resultados se pueden ver en la tabla 4 o en el archivo descargable en el apartado contenido.

Bayes Net ha obtenido unos resultados muy parecidos a su homólogo Naive Bayes, pero en este caso su versión inicial y sus demás versiones no han diferido apenas. Este modelo ha tardado en su entrenamiento un tiempo estimado al clasificador anterior igualando o superando sus resultados. Sus resultados se pueden ver en la tabla 5 y en su archivo descargable.

El modelo SMO ha sido el que mejor resultados ha conseguido, superando a todos los demás clasificadores con un amplio margen. A pesar de la naturaleza de los diferentes *kernels* que se han utilizado, los resultados obtenidos han sido proporcionales entre ellos e igual de buenos en la mayoría de los casos. También ha sido el modelo que menos tiempo consume en su entrenamiento, aproximadamente entre los 3 y los 6 minutos por ejecución. Los resultados obtenidos se pueden ver en las tablas 6 y 7, sus resultados ampliados se pueden descargar en el apartado contenidos.

Finalmente, se encuentra la red neuronal, que a pesar de las esperanzas que se tenían depositadas en este modelo y después de intentar su correcto entrenamiento con sus diferentes opciones de nodos y capas ocultas, al llevar un tiempo con su ejecución, el proceso lanza un error por falta de memoria en todos y cada uno de los casos probados.

5. Conclusiones y trabajos futuros

a. Conclusiones

En este estudio se ha demostrado la posibilidad real de poder obtener una aplicación funcional para evitar el lenguaje sexista de los textos oficiales del estado.

Con el montaje y validación de unos modelos de clasificación a nivel de estudio se abre la posibilidad de que, en un futuro, se puedan obtener documentos libres de prejuicios lingüísticos de forma automática y así elevar su calidad y su nivel de corrección.

Después de haber recopilado y preparado un conjunto de datos de entrenamiento de 5000 frases se han obtenido unos resultados muy positivos a pesar de que hoy en día los procesos de entrenamiento pueden llegar a contar con miles o millones de datos extraídos para su elaboración.

También se ha podido comprobar que no todos los modelos responden igual ante la misma situación, haciendo que unos sean más óptimos que otros según el trabajo a realizar.

b. Trabajos futuros

Una vez realizado este estudio, basándose en los datos obtenidos y con perspectiva de futuro, se podría trazar una senda a seguir.

La obtención de un conjunto de datos de entrenamiento más amplio sería la primera mejora que impactaría directamente en la calidad de los clasificadores; al ampliar el número de muestras en que basar una clasificación, el resultado a obtener sería mejor contrastado. Inicialmente, doblar el corpus actual sería un buen comienzo.

Otra medida a seguir sería el comprobar más modelos de clasificación, con el objetivo de tener una mejor visión de cual o cuales son los modelos que mejor se adaptan a la clasificación de lenguaje natural. Incluso se podría construir una herramienta de clasificación multi-modelo con clasificadores de diferente naturaleza y arbitrando la salida de un dato comparando los resultados de todos ellos.

Otro trabajo futuro sería la implementación de una herramienta accesible a todo el mundo con la que se dispondría de esta función. Esta herramienta se podría materializar de diferentes maneras, por ejemplo, una aplicación de escritorio independiente que se dedique a analizar documentos; también una

página web donde estaría disponible la subida o escritura de documentos en la misma y sea capaz de analizar el texto, marcando los cambios a realizar. Otra forma sería mediante una opción embebida dentro de un editor de texto, como puede ser la aplicación de Office, de forma que mediante la activación de esta opción se marcarían las frases sexistas, con un funcionamiento similar al corrector de ortografía ya existente en muchos editores.

Finalmente, sería necesario formar a las personas en el lenguaje y aprender herramientas para evitar estas faltas.

6. Contenidos

Enlace a GitHub con el código implementado.

Enlace al corpus empleado en el aprendizaje.

Enlace al archivo `arff` creado en Weka y utilizado en los modelos.

Enlace al archivo con datos del modelo J48

Enlace al archivo con datos del modelo Naive Bayes

Enlace al archivo con datos del modelo Bayes Net

Enlace al archivo con datos del modelo SVM

7. Bibliografía

[1] <El Pais> 2008, fin del BOE en papel | Edición impresa | EL PAÍS. 3 de enero de 2008.

[2] “Guía de uso para un lenguaje igualitario” de la unidad de igualdad de la Universitat de València [2] ISBN: 978-84-695-3602-5.

[3] <Wiki-Weka> https://weka.wikispaces.com/Making+predictions#Command line-Classifiers.

[4] <scikit learn> http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.

[5] Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008).

[6] <Blog/Web> http://analisisydecision.es/entrenamiento-validacion-y-test/.

[7] <Blog/Web> https://www.w3.org/International/articles/definitions-characters/index.es.

[8] <Blog/Web> http://www.proyectoautodidacta.com/comics/formatos-de-texto-ii/.

[9] <Blog/Web> http://www.auladiez.com/ejercicios/16_preposiciones.php.

[10] <Blog/Web> http://programacion.net/articulo/expresiones_regulares_en_java_127.

[11] Russell, S. J. Norvig, P. (2003). Inteligencia Artificial: Un Enfoque Práctico. Russell. (2ª ed.) Madrid: Pearson.

[12] <Universidad Autónoma de Madrid> http://www.uam.es/personal_pdi/ciencias/jspinill/CFCUAM2014/Trees-CFCUAM2014.html.

[13] <Wikipedia> https://es.wikipedia.org/wiki/Aprendizaje_basado_en_%C3%A1rboles_de_decisi%C3%B3n.

[14] <Web> https://www.xlstat.com/es/soluciones/funciones/arboles-de-clasificacion-y-de-regresion.

[15] <Universitat oberta de Catalunya> http://data-mining.business-intelligence.uoc.edu/home/j48-decision-tree.

[16] <Blog/Web> http://naivebayes.blogspot.com.es/.

[17] <Wikipedia> https://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo.

[18] Pearl, Judea (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press. ISBN 0-521-77362-8.

[19] Luis M. de Campos, Juan M. Fernández-Luna and Juan F. Huete (2004). «Bayesian networks and information retrieval: an introduction to the special issue». *Information Processing & Management* (Elsevier) **40** (5): 727-733. ISBN 0-471-14182-8. doi:10.1016/j.ipm.2004.03.001

[20] <Wikipedia> https://en.wikipedia.org/wiki/Multilayer_perceptron.

[21] <Xataka> https://www.xataka.com/robotica-e-ia/las-redes-neuronales-que-son-y-por-que-estan-volviendo.

[22] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.

[23] Haykin, Simon (1998). *Neural Networks: A Comprehensive Foundation* (2 ed.). Prentice Hall. ISBN 0-13-273350-1.

[24] <scikit learn> http://scikit-learn.org/stable/modules/svm.html.

[25] <Universidad de Waikato> https://www.cs.waikato.ac.nz/ml/weka/.

[26] <Universidad Politécnica de Valencia> http://users.dsic.upv.es/~cferri/weka/

[27] <Wiki-Weka> https://weka.wikispaces.com/Use+WEKA+in+your+Java+code-.

[28] <scikit learn> http://scikit-learn.org/stable/modules/cross_validation.html.

[29] <Carnegie Mellon University>
https://www.cs.cmu.edu/~schneide/tut5/node42.html

[30] <API Weka> http://weka.sourceforge.net/doc.stable-3-8/weka/classifiers/trees/J48.html.

[31] <Foro Weka> http://weka.8497.n7.nabble.com/Pruning-confidence-in-C4-5-Decision-Tree-td13886.html.

[32] <API Weka> http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/NaiveBayes.html.

[33] <API Weka> http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/BayesNet.html.

[34] <Blog/Web> http://www.torsten-schoen.de/2013/02/sample-instances-from-a-bif-xml-file-using-weka/.

[35] <API Weka>
http://weka.sourceforge.net/doc.stable/weka/classifiers/functions/MultilayerPerceptron.html.

[36] <API Weka>
http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/supportVector/package-summary.html

[37] *Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" (PDF). Journal of Machine Learning Technologies.* ***2*** *(1): 37–63.*

[38] Perruchet, P.; Peereman, R. (2004). "The exploitation of distributional information in syllable processing". *J. Neurolinguistics*. **17** (2–3): 97–119.