Análisis de polaridad en textos escritos en inglés y español

Curso 2013/2014
Universidad Carlos III de Madrid
LUCÍA PLAZA SACARRERA
Tutora: Ascensión Gallardo

Tabla de contenido

Agradecimientos

Me gustaría agradecer a mi familia y amigos por estar ahí a lo largo de todos estos años en los momentos difíciles y en los buenos. Ha sido una carrera llena de obstáculos pero que ha llegado al final.

He aprendido mucho en diversos aspectos tanto académicos como personales. Durante estos años he forjado una nueva personalidad, más dura y más fuerte preparada para afrontar toda clase de retos en el futuro. Y quiero dar las gracias a todos los que estuvieron conmigo, sobre todo cuando el futuro se veía oscuro, y me dieron fuerzas y ánimos para continuar.

Y a mi tutora por ser comprensiva y hacer posible este proyecto.

A todos vosotros gracias.

Resumen

En la actualidad encontramos en Internet multitud de comentarios sobre una gran cantidad de temas, desde películas y libros hasta hoteles, restaurantes e incluso noticias. Todas estas opiniones influyen en quien las lee y muchas veces para tomar decisiones se da más prioridad a lo que se ha dicho sobre el tema, ya sea una película o un coche, que a las especificaciones. Tal cantidad de información muchas veces no está graduada por lo que se requeriría leer todas las críticas para poder sacar una conclusión general sobre el asunto destacado.

En este contexto, la idea de conseguir un motor capaz de dar la opinión sobre un tópico sin necesidad de procesar cada comentario por separado ha sido la motivación principal para la realización de este proyecto. En concreto, el proyecto se ha centrado en buscar herramientas automáticas para discriminar textos en español entre positivos y negativos. Sin embargo, dado que existen una gran cantidad de bases de datos adecuadas para este objetivo en el idioma inglés, primero se han estudiado y experimentado con diversas técnicas sobre este idioma, para posteriormente aplicar lo aprendido al caso del español.

Gran parte de los sistemas descritos en el estado del arte sobre trabajos previos utilizan una herramienta llamada WEKA que es muy potente a la hora de implementar de forma eficiente diversos algoritmos de aprendizaje máquina y que puede utilizarse para la clasificación de textos. Básicamente, su funcionamiento consiste en la generación de una clasificación de las palabras dependiendo de la clase a la que pertenecen mediante la utilización de filtros.

En el desarrollo de este proyecto se han aplicado las ventajas proporcionadas por WEKA en la realización de experimentos, variando tanto la creación de reglas para organizar las palabras según sus pesos como los clasificadores. Así se ha conseguido acotar la manera de reconocer la polaridad de un texto entre positivo o negativo y tener un punto de partida desde el que poder mejorar el sistema y ampliar el rango de sentimientos que se podrían inferir a partir de los textos.

Para los experimentos se han considerado distintos clasificadores: Naive Bayes, Naive Bayes Multinomial, Naive Bayes Multinomial Updateable y SMO. Y los filtros aplicados previamente a clasificarlos han sido una concatenación de uno no supervisado *StringToWordVector* y otro supervisado *AttributeSelection* provistos por WEKA para crear las reglas.

Summary

Nowadays, there can be found in the Internet many opinions about a plethora of topics, among them, films and books or hotels and restaurants and even news. Every one of these opinions have an influence over the reader and usually to make a decision about an issue, no matter what, a film or a car, they are taken in more consideration than the specifications. This amount of data it is not often ranked, therefore it would be needed to read all the comments to get a general conclusion over the topic.

In this situation, the idea of finding a way to be able to give an opinion over an issue without the necessity to go through every comment alone was the main motivation to complete this project. More precisely the project has been focus in finding automatic tools to infer positive or negative from a text written in Spanish. However, as there are more data bases to accomplish this objective in English, first there have been studied and experimented in this language to apply what was learnt in the Spanish one.

Most of the described systems in the state of art use WEKA, which is a very powerful tool to efficiently implement machine learning algorithms that can be used in text classification. Basically, it consists in the generation of a classification of words between the considered classes making rules with filters.

In this project the experiments have been done varying the rules that organizes and weight the words and the classifiers taking the advantages of the tool WEKA, to narrow the way to recognize the polarity of a text between positive and negative and get a starting point from which to improve the system and enlarge the type of sentiments that could be inferred from texts.

The classifiers used in the experiments have been: Naïve Bayes, Naïve Bayes Multinomial, Naïve Bayes Multinomial Updateable and SMO. And the filters applied previously have been a concatenation of one unsupervised *StringToWordVector* and a supervised *AttributeSelection* provided by WEKA to create the rules.

1. Introducción

1.1. Motivación

En la actualidad se mueve mucha información de manera escrita. Hay muchos foros en Internet donde la gente expresa su opinión sobre diversos temas, algunos de ellos tienen la posibilidad de graduar el tema con una nota, muchos otros no.

Se necesitan por lo tanto herramientas que faciliten la opinión general textual sobre un asunto para que, en las ocasiones en las que hay muchos comentarios sin valoración numérica, se pueda sacar una idea de si el pensamiento de la mayoría es positivo o negativo sin necesidad de leerse todos los comentarios.

Esta situación ha movido a lingüistas y programadores a intentar encontrar una solución mediante la creación de reglas para extraer la polaridad de las palabras y con la ayuda de algoritmos de autoaprendizaje poder clasificar los textos de manera adecuada.

La mayoría de los estudios sobre este tema son en inglés, hay algunos intentos con peores resultados en otras lenguas cómo el francés y el alemán. Se han realizado pocos estudios en español debido a la complejidad de la lengua, tiene más tiempos verbales y una gramática más extensa que en la inglesa. Además, tras ver los resultados del francés, que sería la más parecida, y que han sido los peores, es de sospechar que para el español serían unos resultados parecidos, en los que habría que invertir más en la preparación de reglas para distinguir sentimientos.

Esta escasez de estudios en español ha motivado este proyecto, potenciando la búsqueda de reglas y clasificadores que se adecuen mejor a esta lengua, inicialmente centrándonos en la polaridad de los textos, para poder afinar en sentimientos en el futuro.

1.2. Objetivos

El objetivo de este proyecto es encontrar un método fiable de reconocimiento de la polaridad de un texto en español.

Los puntos de partida de este proyecto han sido los documentos realizados por Bo Pang, Lillian Lee y Shivakumar Vaithyanathan: "Thumbs up? Sentiment Classification using Machine Learning Techniques" [1]. Otros dos documentos de Bo Pang y Lillian Lee; "Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts"[2] y "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales" [3]. Y de Erik Boiy y Marie-Francine Moens "A machine learning approach to sentiment analysis in multilingual Web Texts" [4]. De los que se sacaron las ideas para escoger los clasificadores y las reglas del lenguaje, además de la comprobación de que la dificultad de clasificar el francés puede ser aplicable al español.

Para la consecución de este objetivo se ha utilizado WEKA, una potente herramienta que cuenta con un conjunto de algoritmos de autoaprendizaje que pueden embeberse dentro de códigos JAVA y posee una interfaz en la que poder ver los resultados.

Tras las conclusiones obtenidas de los artículos de investigación consultados, se decidió utilizar en todos los casos la concatenación de dos tipos de filtro proporcionados por WEKA y que se explican en el presente documento. Primero, el *String To Word Vector*, en el que se puede variar la manera de distribuir las palabras, simplemente indicando los separadores entre las mismas,

aplicando los parámetros TFT e iDFT a la clasificación o aplicando n-gramas de una palabra, de una hasta tres o conjuntos de tres. El segundo filtro utilizado es el *Attribute Selection* que da un peso a las palabras y las ordena por ocurrencia. En este caso existe la posibilidad de cambiar la manera de ordenar modificando los parámetros, pero debido a los buenos resultados obtenidos se ha mantenido siempre el ordenamiento *Ranker* centrado en 0.0 y pesos según la evaluación *InfoGainEvaluator*.

Y los clasificadores que se han estudiado y se comentan en este documento han sido Naïve Bayes, Naïve Bayes Multinomial, Naïve Bayes Multinomial Updateable y SMO.

Los experimentos se han realizado primero sobre una base de datos en inglés y se han aplicado después sobre una base en español comparando las diferencias entre los resultados entre ambas lenguas y obteniendo los mejores parámetros para las reglas y el clasificador con resultados más acertados.

1.3. Organización del documento

Este documento consta de 6 capítulos.

- Capítulo 1: Introducción. contiene la introducción del documento contando los objetivos.
- Capítulo 2: Estado del arte. En este capítulo se explican tanto los modelos de la creación de reglas como las técnicas usadas. Además de revisar los trabajos previos de los que se parte.
- Capítulo 3: Sistema de detección de polaridad. Contiene la descripción de la herramienta, WEKA, cómo realizar el preprocesado del texto, cómo aplicar los clasificadores y los evaluadores que se van a utilizar en los experimentos.
- Capítulo 4: Experimentos. En este capítulo se describen los resultados de los experimentos.
- Capítulo 5: Conclusiones y trabajo futuro.
- Capítulo 6: Presupuesto.

Contiene también un anexo con las matrices de confusión obtenidas en todos los experimentos.

2. Estado del arte

En internet se almacenan grandes cantidades de información que resultan muy difíciles de manejar. Se han estudiado maneras de clasificar los datos dependiendo del tema del texto (i.e. deportes, política…). Pero últimamente hay discusiones online en las que la importancia del texto recae en el sentimiento más que en el tema por lo que parece que una mejor manera de organizar la información es analizar su sentimiento [1].

Esta clasificación puede ser beneficiosa para aplicaciones inteligentes de negocio y sistemas de recomendación. Además, esto puede servir para sacar una idea general sobre un tema, ya sea una película, un hotel o un restaurante, de todas las opiniones recogidas se puede hacer un resumen obteniendo frases subjetivas que se hayan incluido en los comentarios.

El problema de la organización por sentimiento es que se requiere una mayor compresión del texto más allá que simplemente averiguar el tema del texto. Para ello se ha recurrido a listas de palabras claramente positivas y claramente negativas, haciendo un recuento de ellas y la mayoría marcaría el sentimiento general del texto. Esto se demostró que no era suficiente, por eso se utilizaron métodos de aprendizaje automático como Naïve Bayes, clasificadores de máxima entropía (ME) y máquinas de vectores soporte ("*Support Vector Machines*", SVM).

Aparte de la selección de palabras, otros estudios tienen en cuenta la polaridad de las mismas. En dichos estudios, se intentó integrar la información contextual entre niveles de oraciones y las características de las palabras. Para ello se utilizaron técnicas eficientes para encontrar los mínimos cortes en grafos. Previamente al uso de esas técnicas, se realizó un estudio de objetividad, dividiendo el texto en dos conjuntos de palabras, de tal forma que se eliminaron las objetivas y se utilizan métodos de aprendizaje máquina sobre las subjetivas [2].

Muchas veces es difícil categorizar un texto solamente en positivo o negativo, por eso se han estudiado técnicas en las que se les da valor respecto a una escala, por ejemplo de uno a cinco, donde uno es lo más negativo y cinco lo más positivo. Las técnicas usadas para esto han sido versiones de SVM multiclase y regresión. Entre otras cosas se le añade un porcentaje de positivismo a la oración ("*Positive-Sentence Percentage*", PSP) con el que clasificar las frases [3]. Este porcentaje se calcula dividiendo las instancias positivas entre las subjetivas (ver subsección 2.1.4).

Además, es de destacar que la mayoría de las veces, cuando un usuario escribe una opinión suele ser gramaticalmente incorrecta y con expresiones más típicas del lenguaje hablado que del escrito. También se han tenido en cuenta diferentes lenguas, aparte del inglés, y se ha estudiado las diferencias entre ellas y cómo influyen a la hora de crear regla[4].

En este capítulo se van a explicar con más detalle las técnicas anteriormente mencionadas para llegar a una mejor comprensión de lo que se podría utilizar para un análisis de sentimientos lo más preciso posible.

En la siguiente Figura 1 se muestran los pasos genéricos del sistema que se han dado para la realización de los experimentos:

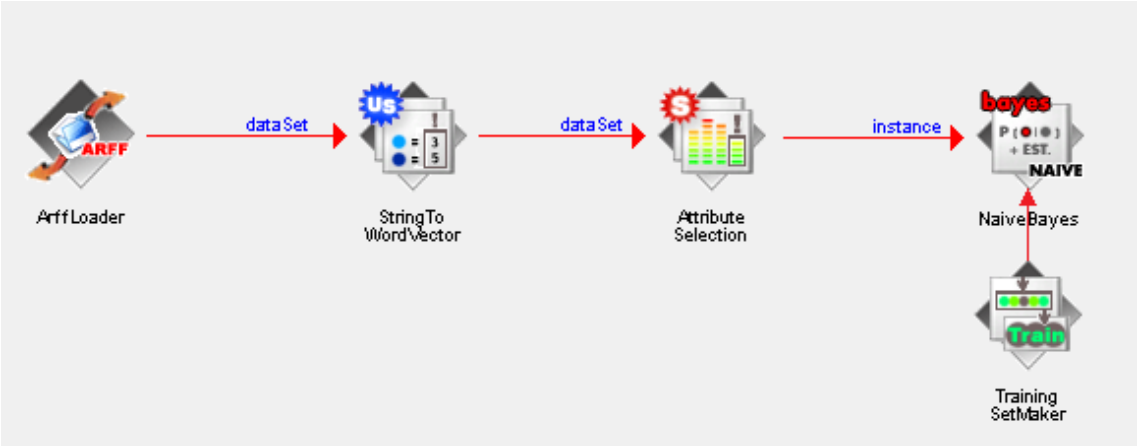


Figura 1: Esquema del sistema genérico

Explicado paso a paso:

a) Cargar las bases en inglés o en español transformadas al formato necesario (en este proyecto se va a utilizar el formato ARFF de WEKA).
b) Aplicar el filtro no supervisado ("*Unsupervised*") *StringToWordVector* de WEKA para la generación de las características textuales (se explicará en las siguientes secciones los parámetros necesarios).
c) Aplicar el filtro supervisado *AttributeSelection* para la selección de las características textuales (se describirá en las siguientes secciones).
d) Elegir la manera de evaluar.
e) Seleccionar el clasificador.

2.1. Características de los textos

2.1.1. Palabras: Modelo de lenguaje. N-Gramas [5]

Para la mayoría de los estudios se han utilizado unigramas, bigramas y en general, n-gramas como características de los textos.

En la siguiente Tabla 1 se muestran ejemplos de transformación de una frase en n-gramas aplicados a la lingüística y como unidad la palabra, en español y en inglés:

Oración	unigrama	bigrama	trigrama
Ser o no ser	Ser, o, no, ser	Ser o, o no, no ser	Ser o no, o no ser
… to be or not to be …	…, to, be, or, not, to, be, …	…, to be, be or, or not, not to, to be, …	…, to be or, be or not, or not to, not to be, …

Tabla 1: Unigramas, bigramas y trigramas

Un modelo de unigrama se usa para obtener información que puede ser tratada como la combinación de varios autómatas de estado finito de un solo estado. Divide las probabilidades de diferentes términos en un contexto desde:

$$P(t_1t_2t_3) = P(t_1)P(t_2|t_1)P(t_3|t_1t_2)$$

Hasta:

$$P_{uni}(t_1t_2t_3) = P(t_1)P(t_2)P(t_3).$$

En este modelo, la probabilidad de producir cada palabra (o término) depende sólo de sí misma, por lo que los autómatas de estado finito van a constar de un único estado como unidades. Para cada autómata, solo hay una manera de que ocurra su único estado y está asignado a una probabilidad. Teniendo en cuenta todo el modelo, la suma de todas las probabilidades debe ser uno.

$$\sum_{term\ in\ doc} P(term) = 1$$

La probabilidad generada por una petición o consulta ("*query*") específica es calculada como:

$$P(query) = \prod_{term\ in\ query} P(term)$$

Para cada documento, se pueden construir distintos modelos de unigramas con diferentes probabilidades de ocurrencia de palabras. Y se tienen en cuenta las probabilidades de distintos documentos para generar una probabilidad de ocurrencia para una consulta. Después se ordenan los documentos para una petición dada dependiendo de las probabilidades generadas.

Los pesos de los unigramas se pueden calcular en base a posición del mismo en el árbol sintáctico relativo al texto que estudiamos:

- Diferencia de profundidad.
- Distancia de camino: el peso del unigrama es inversamente proporcional a la longitud del camino entre el unigrama y la entidad.
- Distancia simple: el peso es inversamente proporcional a la distancia entre la entidad y el unigrama en la frase.

El caso de los bigramas como se ha visto en la Tabla 1, es la secuencia de dos palabras o tokens dentro de una frase o cadena. Los bigramas ayudan a calcular la probabilidad de una palabra condicionada a la palabra previa cuando se aplica la probabilidad condicional:

$$P(W_n|W_{n-1}) = \frac{P(W_{n-1}, W_n)}{P(W_{n-1})}$$

La probabilidad de un token W_n dado el precedente W_{n-1} es igual a la probabilidad del bigrama o a la ocurrencia de dos tokens a la vez $P(W_{n-1},W_n)$ dividido por la probabilidad del token anterior.

De los bigramas podemos extender a explicar los n-gramas.

En un modelo n-grama, la probabilidad $P(w_1, \ldots, w_m)$ de observar una secuencia de términos $w_1,\ldots,w_m$ se aproxima como:

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i | w_{i-(n-1)}, \ldots, w_{i-1})$$

Aquí se supone que la probabilidad de observar la i-ésima palabra (w_i) en el contexto histórico de *i*-l palabras se puede aproximar por la probabilidad de observarla en el contexto histórico acotado de las *n*-l palabras que le preceden (orden *n*).

La probabilidad condicionada se puede calcular a partir las cuentas de frecuencia (conteos) de los n-gramas correspondientes:

$$P(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{count(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{count(w_{i-(n-1)}, \ldots, w_{i-1})}$$

Sin embargo, las probabilidades de los n-gramas no se derivan directamente de su frecuencia, ya que esto da problemas con cualquier n-grama que no haya aparecido en el conjunto de entrenamiento. Para ello, es necesario un suavizado, asignando algo de la función de probabilidad a las palabras que no han salido. Por ejemplo, entre otros métodos, añadir el valor de uno a todas las palabras que faltan por salir.

2.1.2. Parts of speech (POS) [15]

Es la categoría gramatical de la palabra. Dependiendo de su morfología y su léxico, se puede determinar si esta palabra es un nombre o un verbo, etc…

En lingüística de corpus, estas palabras se pueden categorizar para evitar ambigüedad basándose en su definición y su contexto, es decir, la relación que tiene con las palabras adyacentes en una frase o párrafo.

2.1.3. Posición [1]

Se ha estudiado que la posición de una palabra dentro del texto puede variar su contenido con respecto al sentimiento que conlleva. Por ejemplo, para las críticas de cine se ha observado que suelen empezar con una frase llena de sentimiento, seguida de una discusión del argumento y un resumen del autor. Por lo que, desde el punto de vista del análisis de sentimiento, la palabra se etiqueta dependiendo de en qué fragmento del texto se encuentra, dándole más valor, en este caso, a las que están en la primera parte de los textos.

2.1.4. Positive- sentence- percentage (PSP) [3]

Otro parámetro que puede ser útil para análisis de sentimiento es el porcentaje de frases positivas, que se define como el número de frases positivas divididas entre el número de frases subjetivas.

2.1.5. Negación [1],[4]

Para tener en cuenta si las palabras han sido negadas, se pueden usar n-gramas o etiquetar las palabras que van tras una negación hasta la primera puntuación como negada.

Para el francés y el alemán son necesarias reglas más elaboradas ya que en estas lenguas existe, entre otras, la doble negación. Para estos idiomas, además, la negación se puede aplicar a las palabras anteriores a ésta, por lo que se pueden crear ventanas que engloben el máximo número de palabras que en el contexto requieren ser negadas.

También hay que tener en cuenta que a veces en las comparaciones, si lo que está en desventaja es el objeto del que se habla, el adjetivo comparativo se tiene que etiquetar como negado.

2.1.6. Características de discurso ("*discourse features*") [4]

A veces un texto está plagado de sentimientos de los que uno es el predominante y que está señalado por conectores. Por ejemplo, en español, lo que viene después de aunque, a pesar de ser adjetivos positivos, pierde importancia.

Para poder clasificar en base a esos conectores (aunque, pero,…), se hacen listas que los contengan y se especifica qué parte de la frase, la que le precede o le sigue, es la importante. Las que se marquen como no importantes se eliminan de la frase.

2.1.7. Parámetro IDFTransform [25] [26] [27]

El acrónimo IDF dentro de término de frecuencia es "*Inverse Document Frequency*". Es un estadístico numérico que intenta reflejar lo importante que es una palabra en un documento dentro de una base de datos de documentos. Se utiliza como un valor para dar peso a las palabras dentro los experimentos de "*text-mining*".

Este parámetro incrementa proporcionalmente el número de ocurrencias de una palabra dentro de un documento, pero está regulada por la frecuencia de la palabra en toda la base de datos, lo que ayuda a controlar el hecho de que unas palabras son más comunes que otras.

El parámetro idf es una medida de cuánta información da una palabra en un documento respecto a una base de datos. Es por lo tanto una fracción logarítmica que depende del número de documentos en los que aparece, quedando la siguiente fórmula:

$$\operatorname{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Dónde N es el número total de documentos en la base de datos. Y $|\{d \in D: t \in d\}|$ es el número de documentos donde *t* aparece.

Para darle a la definición forma probabilística podemos ver este parámetro dentro de una base de datos como la probabilidad de que un documento *d* contenga el término *t*:

$$P(t|d) = \frac{|\{d \in D : t \in d\}|}{N}$$

Podemos definir entonces el parámetro idf cómo:

$$idf = -\log P(t|d) = \log \frac{1}{P(t|d)} = \log \frac{N}{|\{d \in D: t \in d\}|}$$

Puesto en un lenguaje más compresible la frecuencia de las palabras en un documento se debe transformar según la regla:

$$f_{ij} \cdot \log \frac{num\ of\ Docs}{num\ of\ Docs\ with\ word\ i}$$

Donde f_{ij} es la frecuencia de la palabra *i* en el documento con instancia *j*.

2.1.8. Parámetro TFTransform

Este es otro parámetro, "*Term Frequency*", modifica las frecuencias según la ocurrencia de una palabra *i* en un documento *j*. Sigue la siguiente regla:

$$\log(1 + f_{ij})$$

Es más sencillo que el parámetro idf, y en este caso no se compara con todos los documentos de la base de datos, sino sólo dentro del documento.

2.2. Técnicas usadas

2.2.1. Naïve Bayes (NB) [6]

Un clasificador Naïve Bayes es un clasificador probabilístico basado en aplicar el teorema de Bayes asumiendo fuerte independencia.

Básicamente, el clasificador Naïve Bayes asume que la presencia o ausencia de una característica particular de una clase no está relacionada con la presencia o ausencia de cualquier otra característica dada la variable de clase.

El modelo probabilístico para un clasificador es, abstractamente, un modelo condicional:

$$p(C|F_1, \ldots, F_n)$$

Sobre una clase *C* que representa un pequeño número de clases condicionadas a varias características variables $F_1, \ldots, F_n$. Cuando el número de características variables o el número de clases es muy alto, el modelo se reformula usando el teorema de Bayes:

$$p(C|F_1, \ldots, F_n) = \frac{p(C)\ p(F_1, \ldots, F_n|C)}{p(F_1, \ldots, F_n)}.$$

Que quiere decir:

$$Posterior = \frac{Anterior * Probabilidad}{Evidencia}.$$

En la práctica sólo nos interesará el numerador de la fracción dado que la evidencia no depende ni de *C* ni de F_i, por lo que el denominador será constante. El numerador es equivalente al modelo de probabilidad conjunta.

$$p(C, F_1, \ldots, F_n)$$

La cual se puede reescribir usando la regla de la cadena y la definición de probabilidad condicional:

$$\begin{aligned}
&p(C, F_1, \ldots, F_n) \\
&\propto p(C)\ p(F_1, \ldots, F_n|C) \\
&\propto p(C)\ p(F_1|C)\ p(F_2, \ldots, F_n|C, F_1) \\
&\propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3, \ldots, F_n|C, F_1, F_2) \\
&\propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4, \ldots, F_n|C, F_1, F_2, F_3) \\
&\propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ \ldots p(F_n|C, F_1, F_2, F_3, \ldots, F_{n-1}).
\end{aligned}$$

Asumimos ahora que cada característica F_i es independiente de todas las demás F_j para $j \neq i$ en la clase *C*:

$$p(F_i|C, F_j) = p(F_i|C)$$

Para $j \neq i$, y el modelo conjunto se puede expresar como:

$$p(C|F_1,\dots,F_n) \propto p(C)\ p(F_1|C)\ p(F_2|C)\ p(F_3|C)\ \cdots$$
$$\propto p(C)\prod_{i=1}^{n} p(F_i|C).$$

Por lo tanto, bajo las anteriores suposiciones, la distribución condicional sobre la clase C se puede expresar:

$$p(C|F_1,\dots,F_n) = \frac{1}{Z}p(C)\prod_{i=1}^{n} p(F_i|C)$$

Donde Z es la evidencia y es un factor de escala que sólo depende de $F_1,\dots,F_n$.

El clasificador Naïve Bayes combina este modelo de probabilidad con una regla de decisión. Una regla común es escoger la hipótesis que es más probable, conocida como máxima a posteriori (MAP). El clasificador será de la forma:

$$\text{classify}(f_1,\dots,f_n) = \operatorname*{argmax}_c\ p(C=c)\prod_{i=1}^{n} p(F_i=f_i|C=c).$$

2.2.1.1. Naive Bayes Multinomial (NBM) [9]

Adicionalmente, el modelo Naive Bayes Multinomial considera la frecuencia de aparición de cada término en los documentos x_t en vez de una ocurrencia binaria:

$$P(d_j|c_i) = \Pi_{t=1}^{|V|} P(w_t|c_i)^{x_t}$$

El término $P(w_t|c_i)$ se calcula a partir del número de apariciones de cada término w_t en una clase c_i pero para evitar el problema de las probabilidades 0 se usa la estimación de Laplace:

$$P(w_t|c_i) = \frac{1+n(w_t,c_i)}{|V|+n(c_i)}$$

Donde $n(w_t; c_i)$ es el número de ocurrencias de w_t en c_i, $|V|$ es el tamaño del vocabulario y $n(c_i)$ es el conteo total de palabras en c_i. De este modo, la clasificación se hace buscando el argumento que maximiza la función:

$$c^*(d) = argmax_{ci} p(c_i)\Pi_{t=1}^{|V|} P(w_t|c_i)^{x_t}$$

2.2.1.2. Naïve Bayes Multinomial Updateable

Este clasificador se basa en el Naïve Bayes Multinomial, pero incorpora a su clasificación la interfaz *UpdateableClassifier* que modifica el algoritmo para tener una versión incremental.

Realiza el algoritmo instancia a instancia de manera incremental.

2.2.2. Máxima Entropía (ME) [7]

Uno de los problemas que tiene el clasificador Naïve Bayes es que depende demasiado del supuesto de independencia por lo que, para evitarlo, se han creado otras técnicas.

El clasificador de máxima entropía es una técnica de aprendizaje automático de propósito general que da la última estimación sesgada posible de una información dada. En otras palabras, es máximamente evasiva con respecto a la información que falta. Más importante, no hace la suposición de independencia condicional del Naïve Bayes.

La estimación de máxima entropía de *P(c|d)* tiene la siguiente forma:

$$P(c|d) = \frac{1}{Z(d)} \exp(\sum_i (\lambda_{i,c} F_{i,c}(d,c)))$$

Donde $\lambda_{i,c}$ son los pesos de los parámetros, donde si $\lambda_{i,c}$ es grande significa que f_i es considerado un fuerte indicador de la clase *c*.

2.2.3. Support Vector Machines (SVM) [8]

En aprendizaje automático, las máquinas de vectores soporte son algoritmos de aprendizaje supervisado que analizan información y reconocen patrones, utilizados para clasificar y para análisis de regresión. El SVM básico coge un conjunto de datos de entrada y predice para cada entrada, cuál de las dos posibles clases forma la salida, haciendo un clasificador binario, lineal y no probabilístico.

Dado un conjunto de ejemplos de entrenamiento, cada uno marcado como perteneciente a una de las dos posibles categorías, un algoritmo SVM de entrenamiento construye un modelo que asigna nuevos ejemplos en una categoría u otra. Un modelo SVM es una representación de ejemplos como puntos en el espacio, mapeados de manera que los ejemplos de las distintas categorías están divididos por una clara brecha o margen que es tan ancha como sea posible. Nuevos ejemplos se mapean en ese mismo espacio y se predicen para pertenecer a una categoría basada en qué lado de la frontera cae.

Una SVM lineal se calcula de la siguiente forma:

Dado un conjunto de entrenamiento $\mathcal{D}$, *n* puntos de la forma

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^n$$

Donde y_i es 1 o -1, indicando la clase a la que pertenece el punto $\mathbf{x}_i$. Cada $\mathbf{x}_i$ es un vector real de dimensión *p*. Queremos encontrar el hiperplano de máximo margen que divide los puntos con y_i= 1 de los que son -1. Cualquier hiperplano se puede escribir como un conjunto de puntos **x** satisfaciendo:

$\mathbf{w} \cdot \mathbf{x} - b = 0$, producto escalar entre el vector normal al hiperplano y **x** menos un offset. El parámetro $\frac{b}{\|\mathbf{w}\|}$ determina el offset del hiperplano desde el origen a lo largo del vector normal **w**. En la Figura 2 se muestra gráficamente.

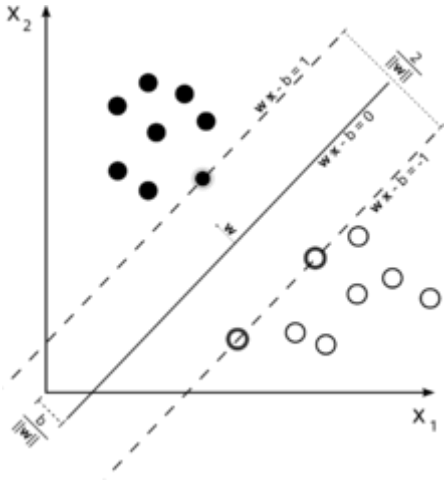


Figura 2: Hiperplano de margen máximo y márgenes para un SVM entrenado con muestras de dos clases. Las muestras en el margen son los vectores de soporte. Extraído de [5] .

Si el conjunto de entrenamiento es linealmente separable, se pueden seleccionar dos hiperplanos de manera que separen los datos sin que haya puntos entre ellos, y así poder maximizar su distancia (Figura 2). La región limitada por ellos es llamado el margen. Estos hiperplanos se puede describir según las ecuaciones: **w**· **x**-*b*=1 y **w**· **x**-*b*=-1.

Usando geometría se ve que la distancia entre los hiperplanos es $\frac{2}{\|\mathbf{w}\|}$, por lo que se quiere minimizar ||**w**||. Se necesita también una limitación para evitar que los puntos caigan dentro del margen, para cada *i*:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n.$$

Para optimizar el problema hay que minimizar ||**w**||.

Como minimizar la norma es bastante complicado, se puede sustituir ||**w**|| por $\frac{1}{2}\|\mathbf{w}\|^2$ (el factor ½ se utiliza por conveniencia matemática) y al minimizarlo la solución no cambia puesto que se utilizan la misma **w** y la misma *b*.

Si se introducen multiplicadores de Lagrange $\boldsymbol{\alpha}$, el problema se puede expresar:

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} \geq 0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1] \right\}$$

Así se busca el punto de silla. De este modo todos los puntos que pueden separarse como $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1 > 0$ dejan de importar ya que su multiplicador de Lagrange α_i se pone a cero.

La solución se puede expresar como combinación lineal de vectores de entrenamiento.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}.$$

Sólo unos pocos α_i serán mayores que cero. Los correspondientes $\mathbf{x_i}$ son exactamente los vectores soporte, que están en el margen y satisfacen $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) = 1$. De ahí se puede obtener que los vectores soporte satisfagan:

$$\mathbf{w} \cdot \mathbf{x_i} - b = 1/y_i = y_i \iff b = \mathbf{w} \cdot \mathbf{x_i} - y_i$$

Que permite definir el offset *b*:

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\mathbf{w} \cdot \mathbf{x_i} - y_i)$$

Donde N_{SV} son todos los vectores soporte.

Como extensiones del SVM se pueden encontrar:

2.2.3.1. Multiclass SVM [12]

Esta extensión de SVM pretende asignar etiquetas a instancias usando máquinas de vectores soporte, donde las etiquetas se obtienen de un conjunto finito de varios elementos.

Para hacer esto se reduce el problema de multi-clase en múltiples clasificaciones binarias. Entre los métodos utilizados para llevar a cabo esa reducción se encuentran:

- Construir clasificadores binarios que distingan entre (i) una de las etiquetas y el resto ("*one versus all*") o (ii) entre cada par de clases ("*one versus one*"). Para el primero, el clasificador con mayor puntuación asigna la clase. Para el segundo, cada clasificador asigna una instancia para una de las dos clases, y al final la clase que obtenga más votos es la elegida.
- Directed Acyclic Graph SVM (DAG SVM) [31] .
- Códigos de corrección de errores a la salida [32] .

2.2.3.2. Regression SVM [13]

Una versión de la SVM para regresión se propuso en 1996 por Vladimir N. Vapnik, Harris Drucker, Christopher J.C. Burges, Linda Kaufman y Alexander J. Smola [13] . Este método se llama regresión de vectores soporte ("*Support Vector Regression* (SVR)"). El modelo producido por la clasificación de vectores soporte depende solo de un subconjunto de entrenamiento, porque el coste para construir el modelo no tiene en cuenta los puntos que están fuera del margen. Análogamente, el modelo producido por SVR depende solo en un subconjunto de entrenamiento porque el coste de la función para construir el modelo ignora cualquier dato cercano al modelo predicho (dentro de un umbral). Otra versión de SVM es conocida como mínimos cuadrados de vectores soporte ("*Least Squares Support Vector Machine*", LS-SVM), fue propuesta por Suykens y Vandewalle [33] .

2.2.3.3. Etiquetado métrico [34]

Se basa en las distancias calculadas con los métodos anteriores. Siendo d la distancia, $nn_k(x)$ los vecinos más cercanos a *x* y *sim(x,y)* la función de semejanza, hay que encontrar el mapeado de cada elemento *x* con su etiqueta l_x que minimice la expresión:

$$\sum_{x \in test} \left[-\pi(x, l_x) + \alpha \sum_{y \in nn_k(x)} f\left(d(l_x, l_y)\right) sim(x, y) \right]$$

Donde *f* es monótonamente creciente (*f(d)=d*), y α es un factor de escala.

Este método mejora el clasificador inicial que asignaba etiquetas más divergentes a elementos similares.

2.2.4. Aprendizaje automático en cascada y agregado [4]

Los clasificadores se pueden combinar haciendo que la salida de uno sea la entrada de otro para ir filtrando información.

Para ello se pueden crear reglas de agregación para los modelos y después utilizarlas para el conjunto de entrenamiento.

El modelo de cascada utilizado en [4] es:

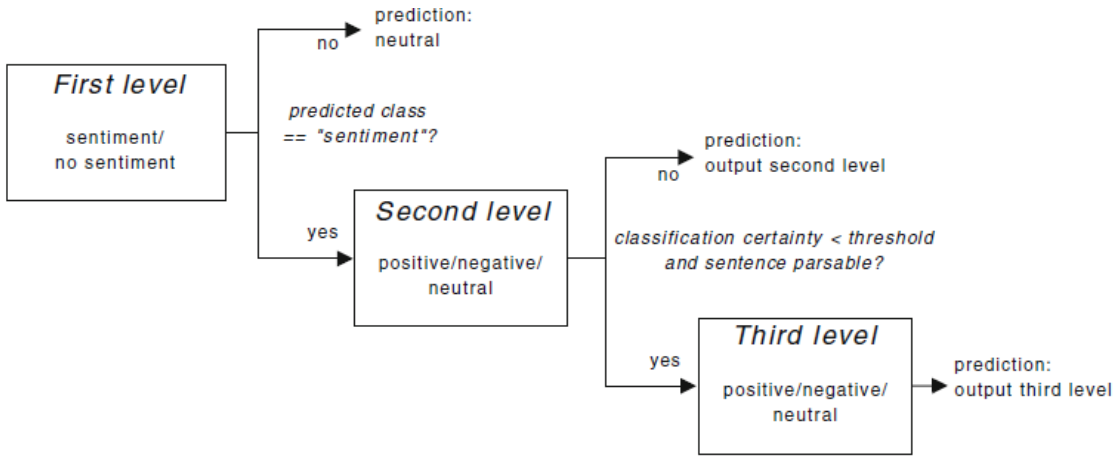


Figura 3: Modelo de cascada. Extraído de [4].

En la primera etapa se eliminan las partes objetivas, que no llevan información sobre el sentimiento. En la segunda etapa se utiliza un clasificador de 3 clases para distinguir entre positivo, negativo y neutral. Y en la última fase los casos difíciles se intentan determinar basándose en las características sintácticas más complicadas de las palabras.

2.2.5. Cut-based classification [2]

Suponiendo que se tienen *n* elementos $x_1,\ldots,x_n$ a dividir en dos clases C_1 y C_2, y se puede acceder a dos tipos de información:

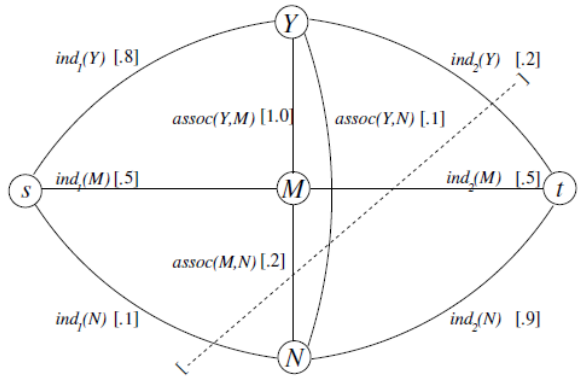
- Individual ($ind_j(x_i)$): estimación no negativa de la preferencia de x_i para pertenecer a la C_1 o la C_2 basada en las características de x_i.
- Asociación ($assoc(x_i,x_k)$): estimación no negativa de lo importante que es que x_i y x_k pertenezcan a la misma clase.

Se quiere maximizar la "felicidad neta" ("*Net happiness*") de cada elemento: su puntuación individual hacia la clase menos su puntuación individual hacia la otra. Para optimizar se puede minimizar el coste de partición:

$$\sum_{x\in C_1} ind_2(x) + \sum_{x\in C_2} ind_1(x) + \sum_{\substack{x_i\in C_1,\\ x_k\in C_2}} assoc(x_i,x_k)$$

El problema aparece intratable por las 2^n posibles particiones de x_i. Sin embargo, es posible resolverlo si se construye un grafo G indirecto con vértices $\{v_1,\ldots,v_n,s,t\}$; donde s es la fuente y t el sumidero. Se añaden n aristas (s, v_i) cada una con peso $ind_1(x_i)$, y n aristas (v_i,t) con peso $ind_2(x_i)$. Finalmente, se añaden $\binom{n}{2}$ aristas (v_i, v_k), cada una con peso $assoc(x_i, x_k)$. Entonces, los cortes en G se definen como: Un corte (S,T) de G es una partición de sus nodos en conjuntos $S=\{s\}\ U\ S'$ y $T=\{t\}\ U\ T'$, donde s no pertenece a S' y t no pertenece a T'. Su coste, $cost(S,T)$, es la suma de los pesos de todas las aristas que cruzan de S a T. Un corte mínimo de G es el de coste mínimo.

En la Figura 4 se muestra gráficamente un ejemplo.



C_1	Individual penalties	Association penalties	Cost
{Y,M}	.2 + .5 + .1	.1 + .2	1.1
(none)	.8 + .5 + .1	0	1.4
{Y,M,N}	.2 + .5 + .9	0	1.6
{Y}	.2 + .5 + .1	1.0 + .1	1.9
{N}	.8 + .5 + .9	.1 + .2	2.5
{M}	.8 + .5 + .1	1.0 + .2	2.6
{Y,N}	.2 + .5 + .9	1.0 + .2	2.8
{M,N}	.8 + .5 + .9	1.0 + .1	3.3

Figura 4: Grafo para clasificar 3 elementos. Los corchetes muestran los valores del ejemplo. Extraído de [2].

2.2.6. Aprendizaje activo [11]

El aprendizaje activo es una forma de aprendizaje automático supervisado en el que el algoritmo de aprendizaje es capaz de preguntar al usuario para obtener las salidas deseadas a nuevos puntos.

Sea T el conjunto total de información a considerar. Por cada iteración, i, T se divide en tres subconjuntos:

1. $T_{K,i}$: puntos donde la etiqueta es conocida
2. $T_{U,i}$: puntos donde la etiqueta es desconocida
3. $T_{C,i}$: Un subconjunto de $T_{U,i}$ que es elegido para etiquetar

La dificultad estriba en encontrar el mejor conjunto $T_{C,i}$. Para ello hay varios algoritmos:

- "*Uncertainty sampling*": etiqueta aquellos puntos por los que el modelo actual está menos seguro de que sea la salida correcta (se utilizan los clasificadores NBM, ME y SVM)
- "*Query by committee*": una variedad de modelos se entrenan sobre el conjunto de con etiquetas conocidas y se vota para asociar una salida a los datos no etiquetados. Los puntos del conjunto que son etiquetados son aquellos en los que el comité está más en desacuerdo.
- "*Expected model change*": etiqueta aquellos puntos que cambian más el modelo actual.

- "*Expected error reduction*": etiqueta aquellos puntos que reducen más el error del modelo.
- "*Variance reduction*": etiqueta aquellos puntos que minimizan la varianza de salida.
- "*Relevance sampling*": etiqueta aquellos puntos que tienen más probabilidad de pertenecer a la clase (utiliza el clasificador NBM).
- "*Kernel farthest first*" (KFF): la muestra más lejana del conjunto que ya está etiquetado se escoge para etiquetar. La distancia se calcula como:
  $d = \sqrt{K(x,x) + K(y,y) - 2 \times K(x,y)}$, donde *K(x,x)* es la función kernel.

2.2.7. SMO [29]

"Sequential Minimal Optimization" (SMO) es un algoritmo para solventar el problema de programación cuadrática (QP) que aparece cuando se entrenan SVM. Lo inventó John Platt en 1998 [29] y causó un gran revuelo entre la comunidad científica que utilizaba SVM dado que éstos son mucho más complejos y necesitaban de terceras partes para solucionar el problema QP.

Se ha explicado en una sección anterior el SVM, ahora se va a contar la manera en la que se ha hecho evolucionar para mejorarlo con el SMO.

Para llevar a cabo la optimización de los algoritmos SVM se consideró un problema de clasificación binaria con un conjunto $(x_1,\ldots,y_i),\ldots,(x_n,\ldots,y_n)$, dónde x_i es el vector de entrada e $y_i \in \{-1,+1\}$ es una etiqueta binaria correspondiente al mismo. Se entrena una SVM de margen suave para solucionar el problema de programación cuadrática que se expresa:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j,$$

Sujeto a las siguientes restricciones:

$$0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \ldots, n,$$
$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

Dónde C es un hiperparámetro de la SVM y $K(x_i, x_j)$ es el kernel dado por el usuario y las variables α_i los multiplicadores de Lagrange.

SMO es un algoritmo iterativo que soluciona el problema descrito descomponiéndolo en una serie de sub-problemas con el tamaño más pequeño posible que se resuelven analíticamente. Debido a la restricción de igualdad lineal que implican los multiplicadores de Lagrange, el problema mínimo incluye dos de esos multiplicadores. Por lo tanto para cualquier dos multiplicadores α_1 y α_2 las restricciones se reducen a:

$$0 \leq \alpha_1, \alpha_2 \leq C,$$
$$y_1 \alpha_1 + y_2 \alpha_2 = k,$$

Por lo que ya se puede resolver analíticamente: se debe encontrar la ecuación cuadrática mínima en una dimensión *k* que resulte negativa de la suma del resto de elementos en igualdad de restricción que se fija en cada iteración.

Los pasos son los siguientes:

1. Encontrar el multiplicador de Lagrange α_1 que viola la condición de Karush–Kuhn–Tucker (KKT) para la optimización del problema.
2. Elegir un segundo multiplicador α_2 para optimizar el par (α_1, α_2).
3. Y se repiten los pasos 1 y 2 hasta que converja.
   Cuando todos los multiplicadores satisfagan las condiciones de KKT (bajo una tolerancia descrita por el usuario), el problema se ha resuelto.

2.3. Trabajos previos

Para conseguir una mejor comprensión del sentimiento en textos, se están llevando a cabo distintas líneas de investigación que se basan, principalmente, en distinguir si el sentimiento del texto es negativo o positivo.

En esta sección se comentarán algunos trabajos relevantes sobre este tópico, indicando sus objetivos, los métodos y bases de datos utilizados y los resultados obtenidos.

2.3.1. Detección de sentimiento positivo o negativo, polaridad [1].

El objetivo de trabajo era comprobar si el análisis de sentimiento podría ser tratado como una subcategoría dentro del análisis de tópicos considerando dos únicos temas (positivo y negativo), o si sería necesario aplicar otros métodos.

2.3.1.1. Hipótesis

Las hipótesis de este estudio, uno de los primeros en esta temática, se basaba en la fuerza que tienen algunas palabras para indicar el sentimiento. Para ello, debería ser suficiente con tener una lista de palabras discriminativas y contar el número de cada clase que hay en el texto.

2.3.1.2. Bases de datos

Para realizar este estudio se escogieron críticas cinematográficas en inglés que llevaran asociadas un número de estrellas o una puntuación para evitar tener que etiquetarlas manualmente. La fuente de estas críticas fue Internet Movie Database (IMDb) archivo de rec.arts.movies.reviews newsgroup (http://reviews.imdb.com/Reviews/).

Para evitar que dominara la opinión de pocos críticos se impuso un número de 20 artículos por autor y categoría sentimental, llegando a tener 752 comentarios negativos y 1301 positivos, siendo representados 144 críticos.

El conjunto se puede encontrar online en http://www.cs.cornell.edu/people/pabo/-movie-review-data/ (la URL contiene guiones sólo alrededor de la palabra “review”).

2.3.1.3. Métodos

Se crearon varias listas de palabras, unas con palabras positivas y otras negativas siguiendo dos criterios diferentes: introspección y reflexión. El segundo criterio dio mejores resultados. El funcionamiento del sistema consistía en contar el número de palabras de cada categoría y así elegir el sentimiento general que representaba la mayoría.

A continuación, se aplicaron los métodos de aprendizaje automático NB, ME y SVM (ver sección 2.2), sobre unigramas y bigramas, teniendo en cuenta la posición de la palabra en la oración (part of speech, POS, 2.1.2), si solo eran adjetivos y posición de la palabra en el texto entero (ver sección 2.1).

2.3.1.4. Resultados

Como punto de referencia se obtuvo:

	Proposed word lists	Accuracy	Ties
Human 1	positive: *dazzling, brilliant, phenomenal, excellent, fantastic* negative: *suck, terrible, awful, unwatchable, hideous*	58%	75%
Human 2	positive: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* negative: *bad, cliched, sucks, boring, stupid, slow*	64%	39%

Figura 5: Resultados de lista de palabras por introspección. Extraído de [1].

	Proposed word lists	Accuracy	Ties
Human 3 + stats	positive: *love, wonderful, best, great, superb, still, beautiful* negative: *bad, worst, stupid, waste, boring, ?, !*	69%	16%

Figura 6: Resultados de lista de palabras por reflexión. Extraído de [1].

La siguiente tabla representa los resultados obtenidos según las características de las palabras y los algoritmos de clasificación utilizados: Naïve Bayes (NB), Maximum entropy (ME), Support Vector Machine (SVM).

	Features	# of features	frequency or presence?	NB	ME	SVM
(1)	unigrams	16165	freq.	**78.7**	N/A	72.8
(2)	unigrams	"	pres.	81.0	80.4	**82.9**
(3)	unigrams+bigrams	32330	pres.	80.6	80.8	**82.7**
(4)	bigrams	16165	pres.	77.3	**77.4**	77.1
(5)	unigrams+POS	16695	pres.	81.5	80.4	**81.9**
(6)	adjectives	2633	pres.	77.0	**77.7**	75.1
(7)	top 2633 unigrams	2633	pres.	80.3	81.0	**81.4**
(8)	unigrams+position	22430	pres.	81.0	80.1	**81.6**

Figura 7: Resultados dependiendo de las características y los algoritmos. Extraído de [1].

Los primeros resultados se obtuvieron a partir de estudiar los unigramas simples y, como puede observarse, el resultado de los tres métodos de clasificación superó el punto de referencia.

Después se comprobó la importancia de la presencia de los unigramas frente a la frecuencia con la que aparecían en un texto. Para ello se utilizaron características binarias haciendo que se contase 1 si el unigrama aparecía en el texto y 0 si no. Lo que resultó fue que para el algoritmo SVM, la tasa de aciertos del sistema mejoró considerablemente teniendo sólo en cuenta la presencia. A partir de entonces sólo se tuvo en cuenta la presencia, no la frecuencia.

Queriendo estudiar cómo incluir la negación, se recurrió a los bigramas. El resultado fue que utilizando tanto la presencia de unigramas como bigramas, se observó que mejoraba respecto a sólo utilizar unigramas. Sin embargo, si solamente se usaban bigramas, el resultado era peor, llegando a la conclusión de que solo los bigramas no capturan bien el contexto.

Se quiso analizar si añadir etiquetas de POS influía en el resultado, ya que puede ayudar a eliminar la ambigüedad de las palabras. El resultado fue que el resultado no varió significativamente al incluir esta característica.

Sabiendo que los adjetivos contienen en sí mismos gran cantidad de sentimiento, se investigó si usándoles en lugar de los unigramas, se obtenían mejores resultados, pero resultó que utilizando

el mismo número de unigramas que de adjetivos se obtenían mejores tasas de acierto. De esta forma se concluyó que los unigramas proporcionaban mejor información.

Por último, teniendo en cuenta la posición de la palabra en el texto (al principio, en el medio o al final), se vio que los resultados finales tampoco diferían mucho de los unigramas solos, aunque se sugirió que si se definieran mejor las posiciones tal vez se pudiera conseguir una mejora.

2.3.1.5. Conclusiones

Observando los resultados anteriores (Figura 7), los tres algoritmos funcionan de manera parecida, siendo el Naïve Bayes el que va a peor y el SVM el que tiende a mejor.

A la hora de estudiar los tipos de características que podrían influir en el sentimiento se ve que los mejores resultados se obtuvieron con la presencia de unigramas, que son las características que típicamente se utilizan para discriminar por tema. Sin embargo, se observó que para análisis de sentimiento, es suficiente con tener en cuenta la presencia de las características en el texto, mientras que para discernir el tópico, es más importante su frecuencia de aparición.

Finalmente, este estudio concluye con que hay que intentar buscar características de discurso ya que a veces las palabras por separado dan la opinión contraria al texto en conjunto. Eso se verá en la sección 2.3.4.

2.3.2. Objetividad y subjetividad [2]

2.3.2.1. Hipótesis

La hipótesis de partida de este trabajo es la suposición de que con la parte subjetiva de un texto se pueden obtener mejores resultados, respecto al sentimiento del texto, que del texto entero.

2.3.2.2. Bases de datos

Para esta investigación se buscó información en: www.cs.cornell.edu/people/pabo/moviereview-data/ (conjunto de críticas versión 2.0). El conjunto empleado contenía 1000 críticas positivas y 1000 negativas, escritas en inglés antes de 2002 con un límite de 20 comentarios por autor (312 autores en total) por categoría. Este es el conjunto de polaridad.

Pero también era necesario un conjunto de frases etiquetadas para referirse a la subjetividad, para ello se creó un cuerpo de frases etiquetadas automáticamente. Para reunir las subjetivas se cogieron 5000 trozos de críticas de www.rottentomatoes.com. Para buscar frases objetivas, se cogieron 5000 frases de resúmenes de argumentos de www.imdb.com. Los fragmentos sólo tenían como mucho 10 palabras y de películas que salieron después de 2001, para evitar solapamiento con el conjunto de polaridad.

2.3.2.3. Métodos

Este estudio consistió en obtener la parte subjetiva del texto y aplicarle los algoritmos anteriormente mencionados de clasificación.

Para obtener la subjetividad se procedió a hacer la clasificación basada en corte (sección 2.2.5). La ventaja de utilizar grafos es que permite separar la información dada por los pares y los modelos específicos del elemento.

Después se utilizaron tanto NB como SVM como clasificadores, teniendo en cuenta la presencia de los unigramas sobre todo el texto y se compararon con los resultados de los extractos que contenían la parte subjetiva. Además se varió la longitud del fragmento de la parte subjetiva.

2.3.2.4. Resultados

Primero se entrenaron los algoritmos NB y SVM con unos conjuntos de subjetividad básicos y luego se utilizaron como detectores de subjetividad. Los resultados de NB y SVM obtenidos fueron del 92 y 90% correspondientemente. Dado el mejor funcionamiento de NB, para el resto de la experimentación se utilizaron los fragmentes extraídos con este algoritmo.

Utilizando el extracto obtenido con NB como detector y como clasificador de polaridad, se obtuvo una precisión del 86.4% frente al 82.8% correspondiente al caso en que no se tenían en cuenta extractos. Usar una SVM como clasificador de polaridad sobre el total del texto ofrece una mejora, mientras que sobre los extractos obtenidos con NB, el resultado es el mismo, tal y como se observa en la Tabla 2.

	Clasificador de polaridad NB (%)	Clasificador de polaridad SVM (%)
Texto completo	82.8	87.15
Extractos subjetivos con NB	86.4	86.4
Extractos objetivos	71	67

Tabla 2. Datos extraídos de [2].

Se probó también utilizando los extractos objetivos, y se vio claramente que la exactitud de los resultados disminuía, probando como cierta la hipótesis inicial de que quitar la parte objetiva influye en la conclusión de sentimiento final.

También se observó que al eliminar la parte objetiva, los comentarios quedaban reducidos en más o menos, un 60%, y el resultado fue aún mejor que sobre la crítica completa, por lo que investigó cómo de pequeños podrían ser los fragmentos para obtener buenos resultados.

Para ello se incluyó el parámetro N que indicaba el número de frases utilizadas de un texto. Como referencia se utilizaron las N primeras frases por un lado y las N últimas frases por otro, mientras que para el resto del estudio se utilizaron las N frases más subjetivas y las menos subjetivas. Los resultados se ven reflejados en la Figura 8 para NB y la Figura 9 para SVM.

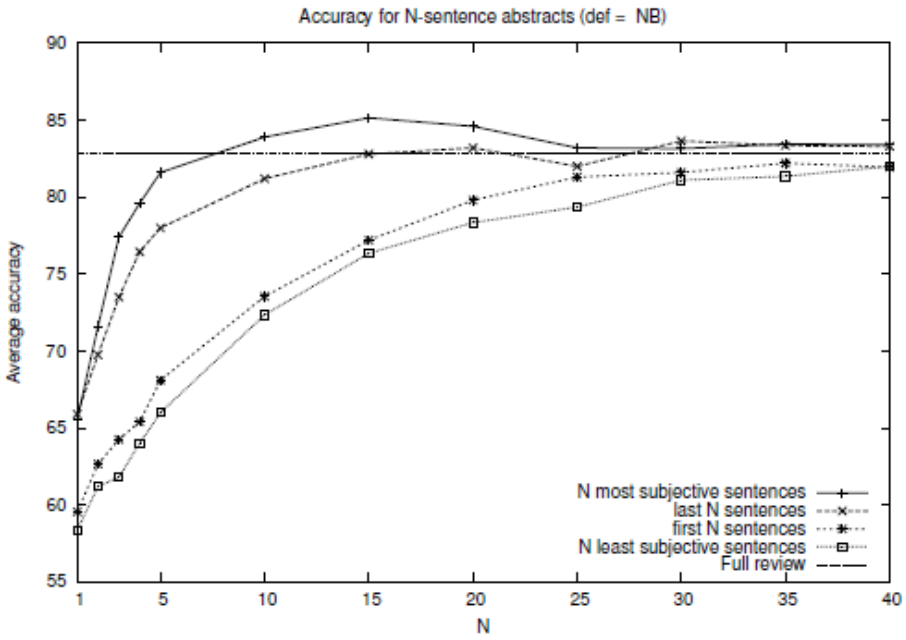


Figura 8: Resultado de la precisión usando N frases de las críticas para el clasificador de polaridad Naïve Bayes. Extraído de [2].

Se encontró que lo óptimo era utilizar las 15 frases más subjetivas, aunque con 5 frases también se obtenía una buena aproximación bastante cercana a la proporcionada con todo el texto. De las muestras de referencia se pudo ver que las últimas N frases proporcionaban un resultado bastante bueno, por lo que se pudo deducir que los autores de los textos terminaban sus críticas con su opinión.

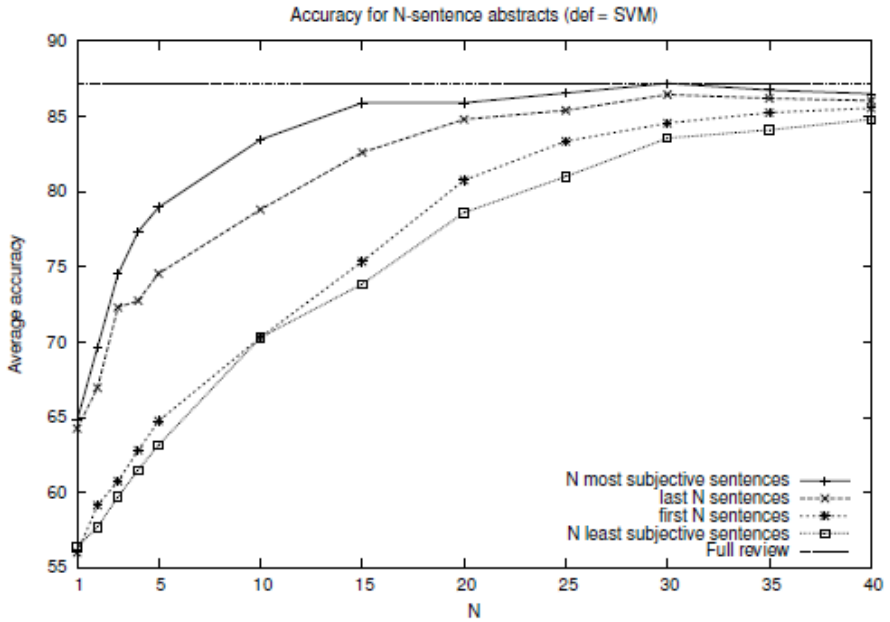


Figura 9: Resultado de la precisión usando N frases de las críticas para el clasificador de polaridad Support Vector Machine. Extraído de [2].

En el caso de la SVM, como el clasificador para el texto completo funcionaba mejor, se requerían 30 frases para obtener el resultado óptimo.

A partir de estos resultados se estudió la influencia del contexto, basado en la proximidad de las frases que se podían determinar utilizando el método basado en corte. Hay que tener en cuenta que estas características no eran entradas naturales a los algoritmos NB y SVM.

En la Figura 10, para NB, y Figura 11, para SVM, se ve la representación del mejor resultado de un conjunto de tres grupos de parámetros de pesos en aristas y proximidad.

Para el caso de clasificador Naïve Bayes, se observa una mejora significativa respecto al texto total.

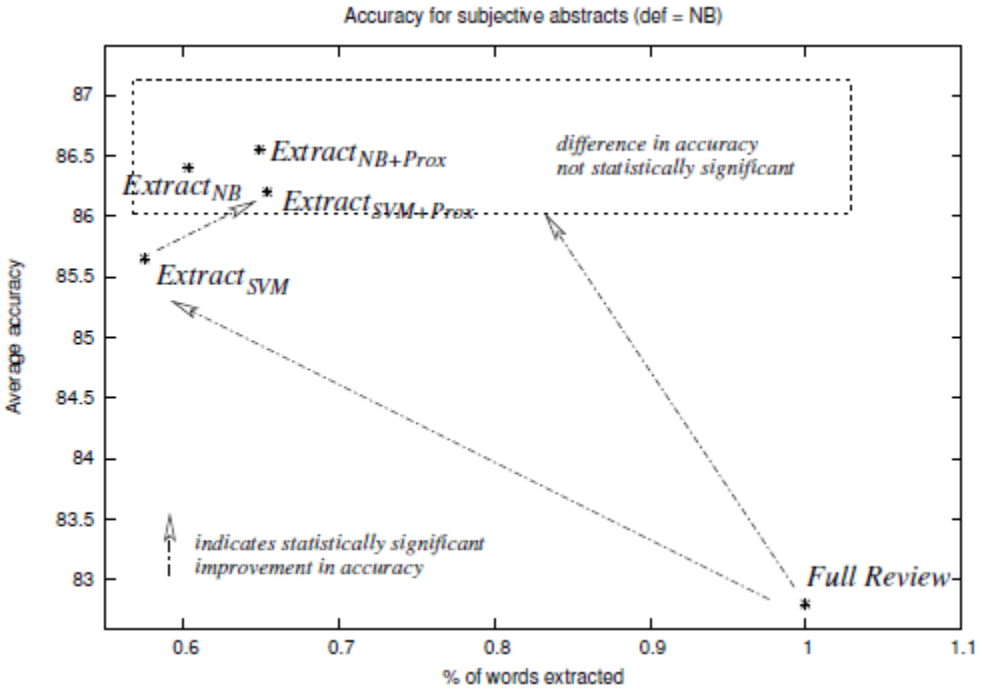


Figura 10: Precisión de los resultados según las palabras extraídas para el clasificador de polaridad NB. Extraído de [2].

En el caso del clasificador SVM, el añadir las características de proximidad mejora el resultado sobre el detector de subjetividad SVM.

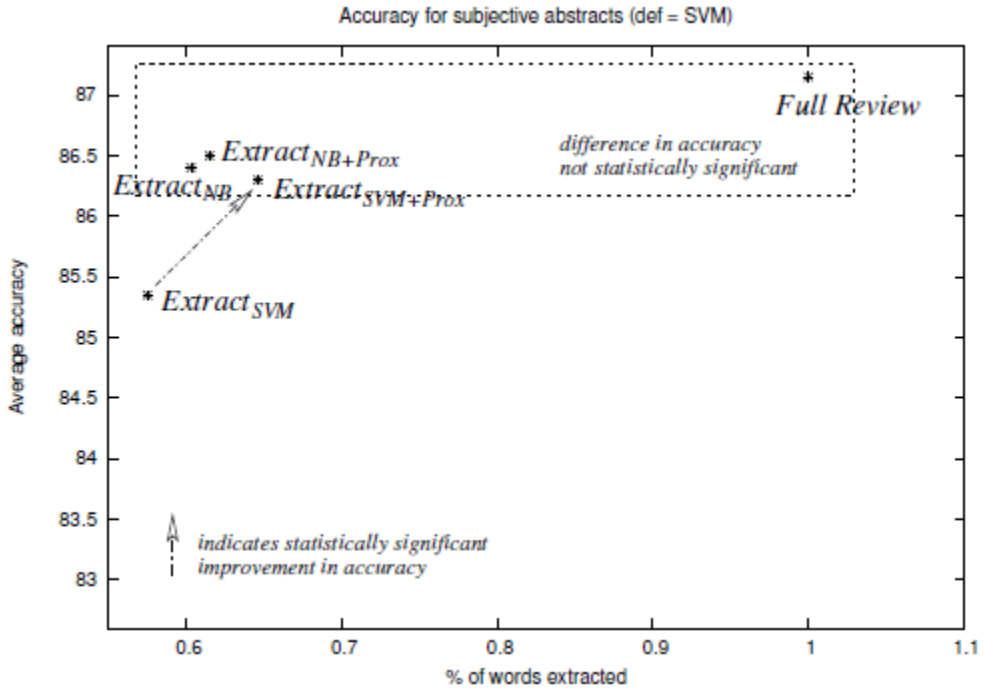


Figura 11: Precisión de los resultados según las palabras extraídas para el clasificador de polaridad SVM. Extraído de [2].

Para ambos clasificadores de polaridad se observó que para el detector de subjetividad NB, el añadir características de proximidad apenas mejora los resultados. Sin embargo, para el detector de subjetividad SVM se producía un incremento en las tasas de acierto.

2.3.2.5. Conclusiones

En este trabajo, se compararon detectores de subjetividad con clasificadores de polaridad y se observó que los detectores podían extraer fragmentos del texto mucho más pequeños de tal forma que se lograban resultados de polaridad similares a los obtenidos con el texto completo. Además en el caso de NB, se obtenían mejores resultados, por lo que se concluyó que los extractos elegidos por el del detector de subjetividad dan una información más limpia sobre el sentimiento del texto.

También se comprobó que el uso de técnicas basadas en corte que añaden contexto mejora el resultado.

2.3.3. Varias estrellas [3]

2.3.3.1. Hipótesis

El objetivo de este trabajo es conseguir un algoritmo de autoaprendizaje que sea capaz de clasificar el sentimiento de un texto en una escala de estrellas que sea, cuando menos, igual de efectivo que los humanos, teniendo en cuenta que a la hora de inferir, los humanos a veces tienen dudas. Además, una clasificación de 4 estrellas se cree que es más precisa que una escala de tres.

2.3.3.2. Bases de Datos

Se utilizaron un conjunto de críticas de cine en inglés de cuatro autores con un valor de escala asociado, de la página http://www.cs.cornell.edu/People/pabo/movie-review-data v1.0.

Para conseguir un conjunto de críticas etiquetadas con polaridad se usaron 10.662 fragmentos de comentarios (más o menos de una frase) descargados de www.rottentomatoes.com; cada fragmento etiquetado como positivo o negativo tal como indican en Rotten Tomatoes.

2.3.3.3. Métodos

Al conjunto de críticas con valoración se les quitó la puntuación explícita, y las frases objetivas (del modo explicado en la sección 2.3.2), ya que esto último se probó como favorable.

Una vez se tuvo el conjunto depurado, se procedió a la categorización del número de estrellas; para ello se basaron en la distancia métrica $d(\ell, \ell') = |\ell - \ell'|$. Se utilizaron 3 algoritmos distintos para hacer el etiquetado (ver sección 2.2.3): SVM "*one versus all"* (SVM-OVA), SVM de regresión y etiquetado métrico.

Para probar la hipótesis de que los rangos se pueden determinar usando el parámetro "*Positive-Sentence-Percentage*" (PSP) (ver sección 2.1.4), se utilizó el subconjunto de polaridad, se entrenó con NB y se aplicó al conjunto de datos de escala.

2.3.3.4. Resultados

La Figura 12 representa los valores de PSP para los cuatro autores y se puede observar que para las críticas positivas el PSP es más alto, por lo que este parámetro se puede utilizar para inferir el rango.

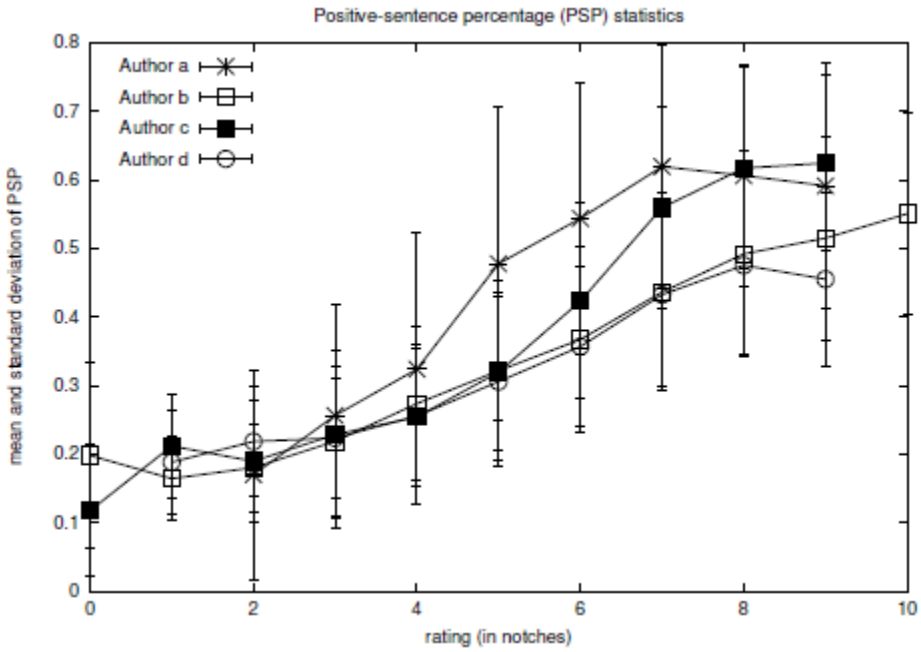


Figura 12: PSP asociado a cada autor según el número de medias estrellas. Extraído de [3].

De esta forma, la función *sim(x,y)* queda:

$$sim(x, y) = \cos\left(\overrightarrow{PSP(x)}, \overrightarrow{PSP(y)}\right)$$

Como referencia se usó la clasificación de predicción de clase mayoritaria y se comparó con los tres clasificadores antes especificados. En las siguientes figuras se muestra para cada autor y el método de SVM utilizado con o sin mejora de etiquetado métrico (PSP), la precisión de los resultados en la Figura 13 para una escala de 3 clases y la Figura 14 para 4 clases.

Para la clasificación con 4 clases el SVR obtuvo mejores resultados que el SVM-OVA para dos autores, mientras que para 3 clases, SVM-OVM mejoró a SVR para los cuatro autores. Esto se puede deber a que para el conjunto de 3 clases, éstas estaban mejor modeladas.

A la hora de tener en cuenta el etiquetado métrico se observó que la función de semejanza mejoró el resultado en todos los casos. Por lo que sí es acertado tener en cuenta el parámetro PSP a la hora de ordenar.

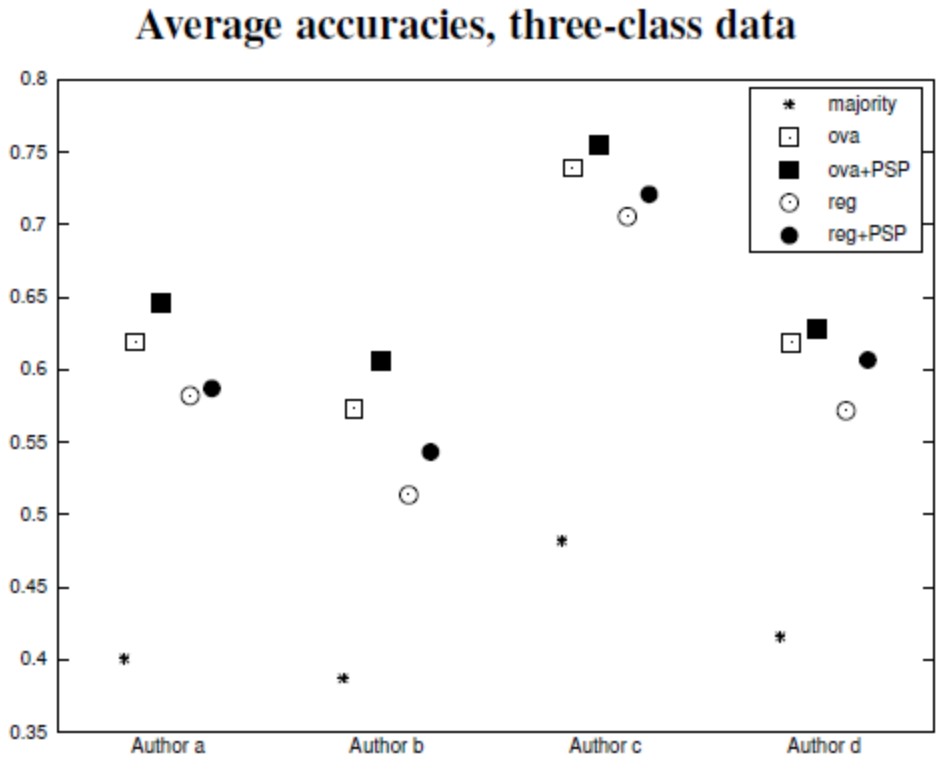


Figura 13: Tasas de clasificación para cada autor, con tres clases utilizando los distintos SVM y el factor PSP. Extraído de [3].

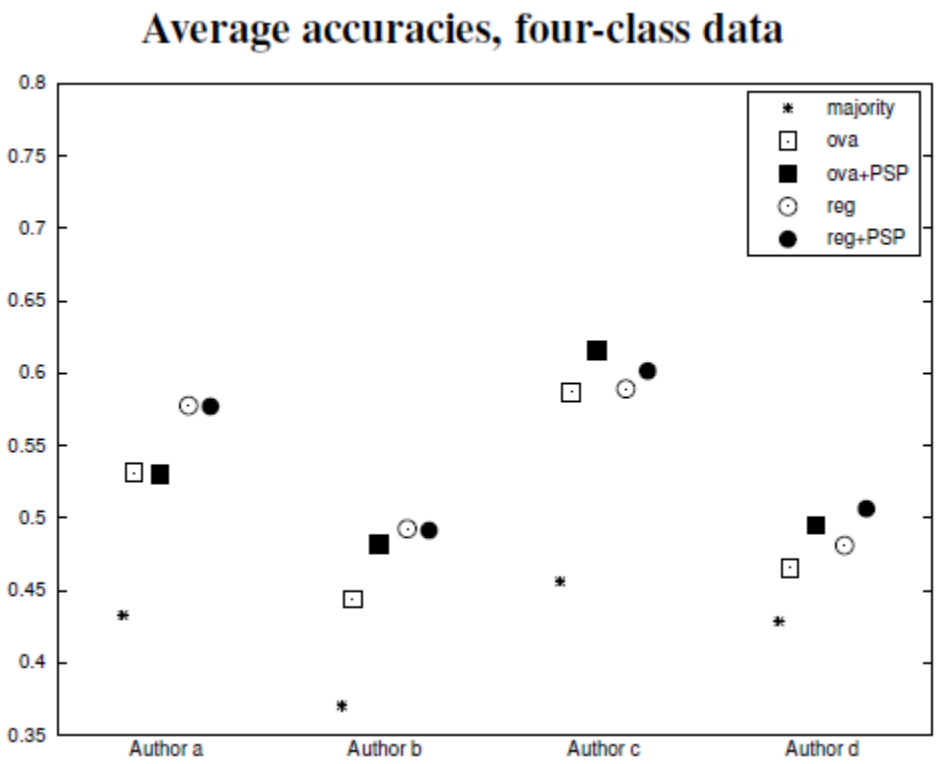


Figura 14: Tasas de clasificación para cada autor, con cuatro clases utilizando los distintos SVM y el factor PSP. Extraído de [3].

2.3.3.5. Conclusiones

En este trabajo se comprobó que el uso del etiquetado métrico basado en la semejanza mejora a los algoritmos que no utilizan PSP.

2.3.4. Utilización del aprendizaje activo para etiquetar sentimientos [4]

2.3.4.1. Hipótesis

El objetivo del trabajo es clasificar las frases de los textos en distintos idiomas en tres clases, positivo, negativo y neutral respecto a un tema de interés. Para conseguir un conjunto de entrenamiento válido se quiso validar que el uso de aprendizaje activo para este etiquetado previo mejoraba el resultado final, reduciendo el número de ejemplos sin empeorar el resultado final.

2.3.4.2. Bases de datos

Se escogieron tres conjuntos, uno en inglés otro en alemán y un último en francés, de frases de blogs (skyrock.com, livejournal.com, xanga.com, blogspot.com), foros (fok.nl, forums.automotive.com) y críticas (amazon.fr, ciao.fr, kieskeurig.nl).

Estos conjuntos contenían tanto textos claros como otros menos limpios que tal vez incluyeran anuncios.

Las entidades de estudio fueron títulos de películas y marcas de coches. La información sobre ambas entidades se distribuía uniformemente en los tres conjuntos.

Para cada frase se le añadieron dos etiquetas, una denotando la pertenencia a una clase (en este caso o coches o películas) y la otra para especificar el sentimiento (positivo, negativo). Así se consiguieron 750 frases para cada conjunto por sentimiento y clase y se añadieron 2500 neutrales.

2.3.4.3. Métodos

En este caso se buscaron textos referentes a una entidad (como el título de una película o la marca de un coche) en foros y blogs. A partir de los textos se estudiaron las características de las palabras que reflejaban mejor el sentimiento. Después se aplicaron clasificadores que, posteriormente, serían combinados con modelos de agregación y cascada. Estos modelos se entrenaron utilizando diferentes conjuntos de características y diferentes clasificadores. Para el tercer nivel de la cascada se utilizaron técnicas más refinadas para el análisis sintáctico de tal forma que a cierta profundidad las características se hacían dependientes del lenguaje. Finalmente se usaron técnicas de aprendizaje activo para reducir la carga de trabajo en las anotaciones iniciales.

2.3.4.4. Resultados

Primero se hizo un estudio de las características estudiadas anteriormente y que en este documento se explican en las secciones anteriores 2.3.1 y 2.3.2. En este caso además se estudió la subjetividad para el clasificador ME y se obtuvieron los resultados representados en la Figura 15.

Features	SVM (%)	MNB (%)	ME (%)
Unigrams	85.45	81.45	84.80
Unigrams & BSubjectivity	**86.35**	83.95	**87.40**
Bigrams	85.35	83.15	85.40
Adjectives	75.85	82.00	80.30

Figura 15: Resultados previos para establecer la referencia utilizando las distintas características de las palabras. Datos extraídos de [4].

El primer experimento se realizó sobre la base datos de los estudios anteriores y después se utilizaron los conjuntos explicados en el apartado 2.3.4.2 de esta sección. Los resultados se presentan en la Figura 16. Para cada lengua se compararon los resultados obtenidos con un solo clasificador y cascadas de dos y tres niveles. El clasificador simple *uni-lang* era el mismo utilizado para el nivel 2 de la cascada. El clasificador simple *uni-lang-dist* era el utilizado para los niveles 2 y 3 de la cascada para una distancia simple (distancia en número de palabras). El clasificador simple uni, utilizaba un clasificador basado en unigramas.

Experimentando con los tres algoritmos se llegó a la conclusión de que el mejor para el inglés era en MNB, para el alemán SVM (kernel lineal) y para el francés ME. Los resultados de la Figura 16 por cada idioma se consiguieron con los clasificadores mencionados.

Los valores representados eran la exactitud, dada por la tasa de acierto; la precisión y la exhaustividad ("*recall*"), que son el resultado de la identificación correcta de la clase; y la medida *F*, que combina las dos anteriores. Se calculan de la siguiente manera donde D_{lx} son los datos etiquetados por humanos y D_{cx}, los datos clasificados por la máquina:

$$Precision_X = \frac{|D_{lx} \cap D_{cx}|}{|D_{cx}|}, \quad Recall_X = \frac{|D_{lx} \cap D_{cx}|}{|D_{lx}|}, \quad F_\alpha = \frac{(1+\alpha) \times precision \times recall}{\alpha \times precision + recall}.$$

Architecture	Accuracy	Precision pos/neg/neu	Recall pos/neg/neu	F-measure pos/neg/neu
(a) English				
Cascade with layers 1, 2 and 3	83.30	69.09/85.48/85.93	55.73/82.40/91.84	61.70/83.91/88.79
Cascade with layers 1 and 2	83.10	70.49/87.72/84.61	54.13/79.07/93.00	61.24/83.17/88.61
SC uni-lang	83.03	69.59/86.77/85.08	56.13/79.60/92.12	62.14/83.03/88.46
SC uni-lang-dist	80.23	60.59/78.78/86.57	59.87/82.67/85.60	60.23/80.68/86.08
SC uni	82.73	68.01/85.63/85.53	58.40/78.67/91.24	62.84/82.00/88.29
(b) Dutch				
Cascade with layers 1,2 and 3	69.03	63.51/53.30/72.20	42.93/31.20/88.20	51.23/39.36/79.40
Cascade with layers 1 and 2	69.80	66.60/58.31/71.66	41.73/29.47/90.32	51.31/39.15/79.92
SC uni-lang	69.05	60.39/52.59/73.63	49.60/33.87/85.44	54.47/41.20/79.10
SC uni-lang-dist	68.85	61.08/54.52/72.20	43.73/30.53/87.88	50.97/39.15/79.27
SC uni	68.18	58.73/49.58/73.24	48.00/31.73/85.16	52.82/38.70/78.75
(c) French				
Cascade with layers 1, 2 and 3	67.68	50.74/55.88/71.90	27.47/38.67/88.44	35.64/45.71/79.32
Cascade with layers 1 and 2	67.47	52.69/53.96/71.56	26.13/38.13/88.68	34.94/44.69/79.21
SC uni-lang	65.97	47.67/50.33/72.18	30.00/40.67/84.36	36.82/44.99/77.79
SC uni-lang-dist	65.97	47.67/50.33/72.18	30.00/40.67/84.36	36.82/44.99/77.79
SC uni	65.83	45.67/50.82/72.23	28.80/41.33/84.28	35.32/45.59/77.79

Figura 16. Resultados usando para el inglés el clasificador MNB, para el alemán SVM y para el francés ME. Extraídos de [4].

Se pudo observar que los resultados mejoraban utilizando la cascada de clasificadores. Para el francés y el alemán la obtención correcta del sentimiento resultaba más difícil que para el inglés. Esto se podría deber a que estos dos idiomas tienen una sintaxis más complicada. También se pudo ver que se mejoraba respecto a la referencia SC *uni*.

Se hicieron estudios para ver la influencia de las distintas clases y se demostró que los resultados eran mejores cuando se entrenaba con las frases obtenidas del mismo dominio, pero aplicando el entrenamiento sobre películas a los textos sobre coches el resultado era peor que el del conjunto de entrenamiento sobre coches aplicado a películas.

Aparte se hizo un análisis del error del que se dedujeron las dos fuentes de error predominantes; una relacionada con la falta de patrones en el conjunto de entrenamiento y otra debido a no haber contemplado adecuadamente las diferencias entre los idiomas considerados. La primera se puede resolver añadiendo más ejemplos al entrenamiento y la segunda incorporando más reglas específicas del idioma.

Después se hizo el análisis del aprendizaje activo y cómo podía mejorar los resultados el utilizarlo. Para ello se utilizaron conjuntos de entre 10 y 150 ejemplos con una condición de parada de tamaño 500. La base de datos fue el conjunto en inglés y las características, los unigramas con negación y características de discurso. El clasificador utilizado para la evaluación fue MNB.

Las etiquetas se obtuvieron automáticamente (ya que los ejemplos ya estaban etiquetados por un experto) y los ejemplos se añadieron al conjunto de entrenamiento. Hasta que un ejemplo era escogido, su etiqueta era desconocida para el método de aprendizaje activo.

Se realizaron tests para los tres clasificadores utilizando "*Uncertainty Samples*" (US) y "*Random Sampling*" (RS) (la referencia). Los resultados se pueden ver en la Figura 17 en la que los números representados son la media de la medida *F* sobre positivo, negativo y neutral. Para todos los clasificadores US dio mejores resultados que RS. Eso puede suceder porque para problemas no separables, US escoge muestras de las zonas entre clases. Estas muestras pueden no tener mucha información, pero ayudan a afinar las regiones de decisión del clasificador.

#Ex	MNB		ME		SVM	
	RS	US	RS	US	RS	US
150	47.69	47.69	47.32	47.32	45.82	45.82
200	50.79	48.14	49.54	47.52	46.94	47.55
250	52.40	50.58	51.89	49.72	46.83	49.81
300	52.64	52.60	52.38	51.81	47.55	**51.50**
350	52.57	54.06	51.90	52.16	47.86	**52.76**
400	52.18	54.27	52.32	52.69	48.27	**53.23**
450	53.11	**55.60**	52.61	53.41	48.87	**54.56**
500	53.92	**57.01**	52.08	54.04	48.55	**54.21**

Figura 17: Comparación de clasificadores para active learning. Extraído de [4].

La semejanza se probó utilizando *relevance sampling*, y se vio que mejoraba la actuación del RS. Comparando los métodos KFF y RS para comprobar la diversidad, el primero dio peores resultados debido a la cantidad de ejemplos neutrales utilizados.

Se comparó también RS y US con un algoritmo que combinaba US con *relevance sampling*. Con esto se vio que al utilizar *relevance sampling* se obtenían más muestras negativas, lo que igualaba el balance de ejemplos, y que eso mejoraba los resultados. Además el uso de RS inicialmente podía ayudar a elegir los conjuntos más adecuados para que conseguir que el algoritmo *relevance sampling* escogiera la mejor dirección.

2.3.4.5. Conclusiones

De este estudio se concluyó que el añadir características específicas del lenguaje a los unigramas, mejora un poco los resultados comparado con el uso simple de los unigramas para los tres idiomas. El uso del sistema en cascada llevó a la conclusión de que el funcionamiento global mejora cuando previamente se hace una criba y se eliminan todos elementos neutrales, dejando así un subconjunto menor sobre el que aplicar los clasificadores. Además, el añadir una capa con características sintácticas aún más complejas ayuda a clasificar el sentimiento dependiendo de la entidad de estudio.

Entre las fuentes de error más comunes se encontró la escasez de los conjuntos de entrenamiento y la dificultad en textos escritos fonéticamente más que gramaticalmente correctas como pasa en francés, y que en español se daría con latinoamericanos o andaluces que transcribieran el ceceo o seseo, por ejemplo: "No sé qué "haser" con el dolor de "cabesa".

Las técnicas de aprendizaje activo ayudaron a etiquetar los conjuntos de entrenamiento, obteniendo los mismos resultados con una muestra menor pero mejor anotada que una al azar. Además, el uso de estas técnicas evitaba tener duplicados.

2.4. Otros trabajos realizados recientemente

Basándose en las técnicas y documentos aquí explicados, también se han utilizado técnicas para resumir textos buscando así una mejoría a la hora de elegir el sentimiento [12] y los resultados fueron que, simplemente buscando la subjetividad, se obtiene una mayor precisión que intentando resumir.

Por otro lado se han estudiado técnicas para diversificar el sentimiento [18] , no sólo dividiéndolo en positivo o negativo, sino ir más allá y buscar el aburrimiento, el entusiasmo, la hostilidad… Para ello se han basado en una estructura de ánimo que crearon el D.Watson y A. Tellegen [15] en la que en vez de representar los sentimientos como algo discreto lo intentan ver como algo continuo que construye un modelo estadístico y lo conectan con documentos y a un conjunto discreto de emociones. Este modelo lleva características derivadas de la psicología y basadas en estudios hechos con humanos. Entre las aplicaciones de este modelo están: descubrir las complejas relaciones entre emociones, agruparlas, mejorar la clasificación de las mismas y predicción de sentimiento.

Teniendo en cuenta que ahora las redes sociales son grandes fuentes de opinión, también se han hecho estudios para analizar el sentimiento de frases cortas [19] . Para ello se utilizaron técnicas para agrupar palabras que contribuyen al mismo sentimiento. Las técnicas usadas fueron: El suavizado la semántica y la extracción automática de los temas relacionados con el sentimiento Para ello se utilizó el modelo "*Joint Sentiment-Topic*" (JST). Como conclusión sacaron que ambos métodos mejoraban la referencia usada con el modelo NB entrenado solamente con unigramas, y usar "*sentiment-topic*" daba mejores resultados que utilizar características semánticas con menos características.

En español se han hecho también estudios en la universidad de Málaga dónde crearon una herramienta llamada Sentitext [16] que hace uso de la librería de análisis morfológico de Freeling. Para realizar esta herramienta se necesitaron tres bases de datos, una de palabras (Words) en la que cada palabra tiene asignada un valor, otra de expresiones formadas por conjuntos de palabras (MSWords), y por último, se tuvieron en cuenta las reglas de contexto (CRules), que tienen la capacidad de añadir o restar sentimiento a expresiones y palabras, y están basadas en Context Valence Shifters de Polanyi y Zaenen (2006) [22] . Para probar esta herramienta se hizo un estudio en el dominio de las valoraciones de hoteles, haciendo hincapié en el análisis del sentimiento independiente del dominio [23] .Tras una primera evaluación, se consiguió una discriminación de la polaridad del sentimiento del 90%, pero a la hora de afinar más con el número de estrellas el sistema no estuvo tan acertado. Se pensó por tanto que sería necesario hacer reglas de contexto dependiendo del dominio para así obtener unos resultados más certeros.

En la Universidad de Alicante se hizo un estudio sobre el corpus Emotiblog [24] , que es una colección de entradas de blogs creado y anotado para detectar expresiones subjetivas en los nuevos géneros textuales nacidos con la Web 2.0. Se hicieron experimentos para aprendizaje automático mediante validación cruzada. Las conclusiones que obtuvieron fueron que el análisis

de sentimientos es muy complejo, y que para obtener mejores resultados tendrían que profundizar en las secuencias de entrenamiento tal vez utilizando n-gramas.

3. Sistema de detección de polaridad.

En este capítulo voy a presentar la herramienta utilizada para realización de los experimentos.

Así mismo en las siguientes secciones me centraré en los filtros y clasificadores utilizados en los experimentos mostrando cómo programarlos en WEKA y explicando los parámetros utilizados y los que se van a modificar para los distintos escenarios.

3.1. WEKA [30]

WEKA da nombre a una extensa colección de algoritmos de Máquinas de conocimiento desarrollados por la universidad de Waikato (Nueva Zelanda) implementados en Java; útiles para ser aplicados sobre datos mediante los interfaces que ofrece o para embeberlos dentro de cualquier aplicación. Además WEKA contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, clustering, asociación y visualización. WEKA está diseñado como una herramienta orientada a la extensibilidad por lo que añadir nuevas funcionalidades es una tarea sencilla. Esta herramienta se puede descargar desde la página http://www.cs.waikato.ac.nz/ml/weka/ [30] dónde se encuentra la información mínima para comenzar su uso. Además al descargar la herramienta descargas la librería de JAVA.

Nativamente WEKA trabaja con un formato denominado ARFF, acrónimo de "*Attribute-Relation File Format*". Está compuesto por tres partes, una cabecera con el nombre de la relación, una declaración de atributos que compondrá el archivo y el tipo de los mismos y por último los atributos separando por comas, en la Figura 18 se muestra un ejemplo del archivo.

```
@relation ReviewsIngles

@attribute polarity_class {POS,NEG}
@attribute text String

@data
POS, 'with the abundance of trite , recycled movies in the late \'90s , there is a tremendous demand by movie lovers ( never mind
POS, '\" i\'d rather die today and go to heaven than live to be a hundred and go to hell , \" earnestly proclaims the pentecostal
POS, 'directed by james mangold . screenplay by mangold , lisa loomer and anna hamilton phelan , based on the book by susanna kay
POS, 'i like movies with albert brooks , and i really like movies directed and written by albert brooks . lost in america and def
POS, '* * * moviereviews . org has been asked to start collecting a list of readers in the washington , dc area who would be inte
POS, 'starring-sean connery , catherine zeta-jones , ving rhames and will pattonmovie views by jamey hughton : http : //welcome .
POS, 'armageddon ( 1998 )  starring bruce willis , billy bob thornton , ben affleck , liv tyler , will patton , steve buscemi , m
POS, 'starring : michael keaton ( bruce wayne/batman ) , danny devito ( oswald cobblepot/the penguin ) , michelle pfeiffer ( seli
POS, 'it was a crazy time in france , what with the french revolution right around the corner . king louis xv would soon die , an
POS, ' 1939 , g , 222 minutes [3 hours , 42 minutes]starring : viven leigh ( katherine scarlett o\'hara-hamilton-kennedy-butler )
POS, ' 1999 , pg , 131 minutes [2 hours , 11 minutes]starring : liam neeson ( qui-gon jinn ) ; jake lloyd ( anakin skywalker ) ;
POS, 'just in time for christmas comes a story worthy of both ebenezer and jimmy stewart , with wild at heart\'s nicholas cage ca
POS, 'in what might already be one of the \" hottest \" films of the summer , backdraft does for firefighters what goodfellas did
POS, 'movie views by jamey hughton : http : //welcome . to/movieviewsveteran actor clint eastwood has never looked as grizzled as
POS, 'an unhappy italian housewife , a lonely waiter , a goofy masseuse , lots of love , and gorgeous scenery all come together i
POS, 'months before its release , fox\'s epic animated musical anastasia had been touted as the first legitimate challenge to dis
POS, 'starring al franken , laura san giacomo , vincent d\'onofrio , and harris yulin . review by laurence mixson ( venom8@hotmai
POS, 'the amusing thing about a sequel to a sequel is that the formula usually wears off by the time you get to it . as i was wat
POS, 'after watching the first ten minutes of this japanese film , you will never eat a bowl of ramen the same way again . there
POS, 'writer-director jake kasdan\'s zero effect gets its title from its main character , daryl zero ( bill pullman ) . zero is p
POS, 'directed by michael mann . screenplay by eric roth and mann , from the article \" the man who knew too much \" by marie bre
NEG, '\" expectations \" may be an obsession of mine , but i\'ll say it : * if you can park virtually every expectation at the do
NEG, 'the formula is simple . trap a varied group of people on an isolated location , then pop in a seemingly unstoppable monster
NEG, 'every now and then , reviewers are faced with the films that are hard to be properly reviewed . most of the time it happens
NEG, 'as any reasonable human being would , i must admit that occasionally i am befuddled by certain things . i am befuddled by f
```

Figura 18: Ejemplo de archivo ARFF

Al inicio de la aplicación se puede elegir entre varias funcionalidades (Figura 19), aunque para la realización de este proyecto sólo ha sido necesaria la parte del explorador.

Figura 19: Inicio de WEKA

Mediante el explorador, sobre un archivo se puede hacer el preprocesado de los datos y la aplicación de filtros, clasificación, clustering, búsqueda de asociaciones, selección de atributos y visualización de datos. Se puede ver la interfaz en la Figura 20:

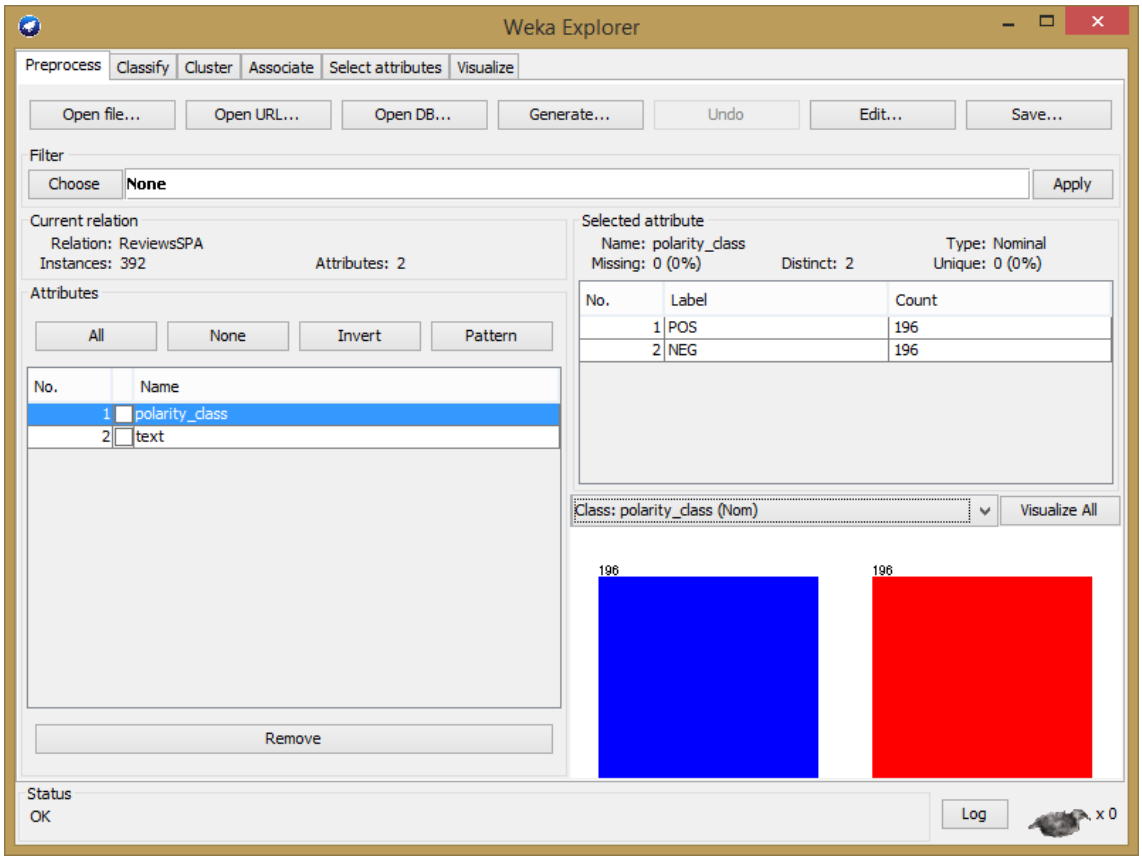


Figura 20: Explorador de WEKA

A continuación, describiremos los tipos de filtros aplicados en los experimentos.

Por un lado, dentro de los atributos no supervisados utilizamos el *StringToWordVector:* Convierte los textos de tipo "*String*" en un conjunto de atributos representando la ocurrencia de las palabras del texto. Como parámetros toma los siguientes:

- *IDFTranform* que indica si queremos que las frecuencias de las palabras sean transformadas según la regla comentada en la sección 3.2.2.3.
- *TFTransform* Otra regla de transformación vista en la sección 3.2.2.4.
- *attributeNamePrefix*, prefijo para los nombres de ficheros de atributos creados.
- Delimitadores (*delimiters*), conjunto de caracteres escape usados para delimitar la unidad fundamental (*token*). Esta opción se ignora si la opción *onlyAlphabeticTokens* está activada, ya que ésta, por defecto, asigna los tokens a secuencias alfabéticas usando como delimitadores todos los caracteres distintos a éstos.
- *lowerCaseTokens* convierte a minúsculas todos los tokens antes de ser añadidos al diccionario.
- *normalizeDocLength* selecciona si las frecuencias de las palabras en una instancia deben ser normalizadas o no.
- *outputWordCounts* cuenta las ocurrencias de cada palabra en vez de mostrar únicamente si están o no están en el diccionario.
- *useStoplist* si está activado ignora todas las palabras de una lista de palabras excluidas (stoplist).
- *wordsToKeep* determina el número de palabras (por clase si hay un asignador de clases) que se intentarán guardar.

Dentro de la categoría de atributos supervisados utilizamos el *Attribute Selection*, que tiene los siguientes parámetros:

- *AttributeEvaluator* que evalúa los atributos individualmente. Puede ser entre otros:
    - *InfoGainAttributeEval*: Evalúa la importancia de un atributo para la tarea de la clasificación midiendo la ganancia de información que se produce al considerar dicho atributo. La ganancia de información se mide a través de la entropía de la clase y la entropía de la clase dado el atributo de la siguiente manera:

$$InfoGain(Class,Attribute) = H(Class) - H(Class \mid Attribute).$$

Ecuación 1

- *RankedOutputSearch* que son métodos de búsqueda que devuelven listas ordenadas de atributos de acuerdo a su relevancia para la tarea de clasificación. Entre otros está:
    - *Ranker*: que devuelve la lista de atributos según su evaluación individual.

Una vez preprocesado el archivo ARFF, pasamos a describir la parte de los clasificadores. Las maneras de evaluar que distingue WEKA son:

- Sobre el conjunto de datos de entrenamiento ("*training set*"). Calcula el modelo y luego lo prueba sobre los mismos datos.
- Sobre un porcentaje del conjunto de datos de entrenamiento ("*Percentage Split*"). Para evaluar el clasificador, en el porcentaje restante.

- Evaluación cruzada ("*cross-validation*") dónde se puede especificar el número de particiones que se desea.

Los resultados que WEKA muestra se componen del modelo construido, las estadísticas de acierto referentes a cada clase y la matriz de confusión. Se pueden ver en la Figura 21 los resultados debidos al clasificador y evaluados en la opción seleccionada.

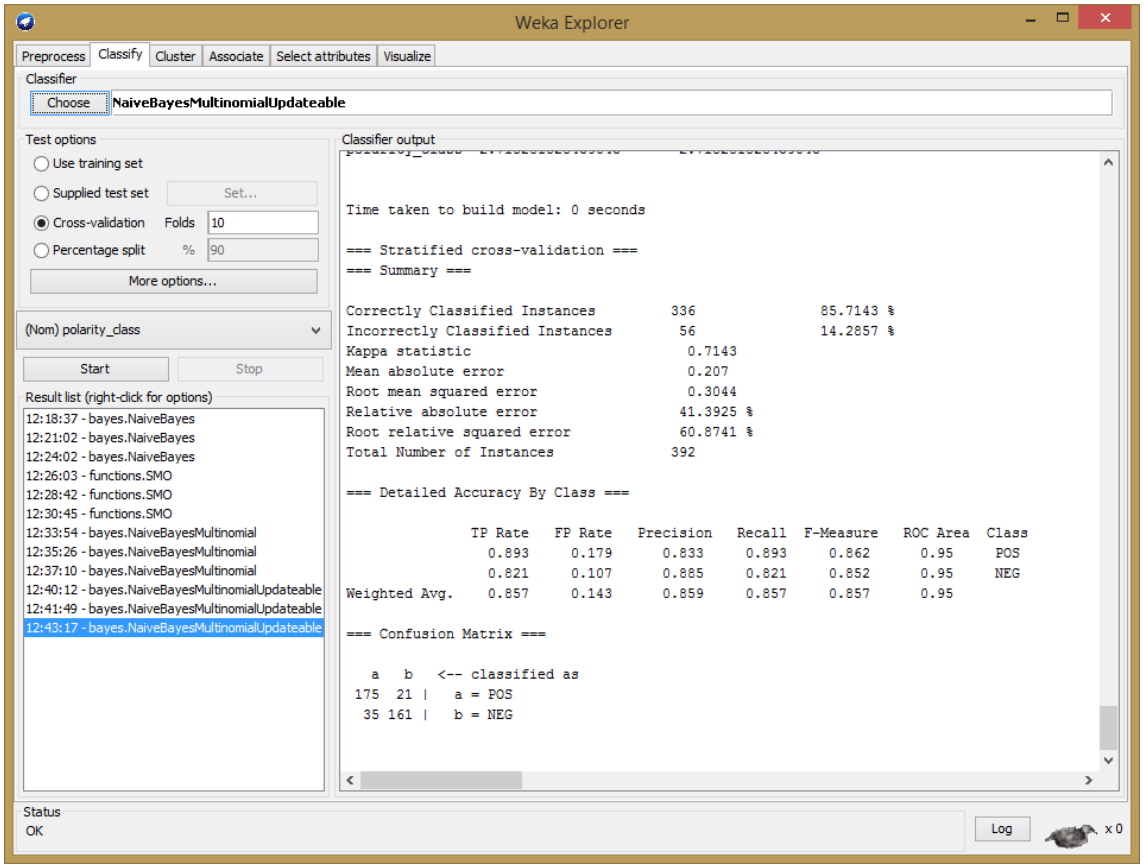


Figura 21: Clasificador de WEKA

Los modelos creados se pueden guardar para utilizarlos luego en otros programas que los necesiten.

3.2. Preprocesado de la información

En esta sección se van a contar por un lado las bases de datos sobre las que se van a realizar los experimentos y por otro lado, con los parámetros que se van a selecciona en WEKA, los filtros que preparan las bases antes de aplicar el clasificador.

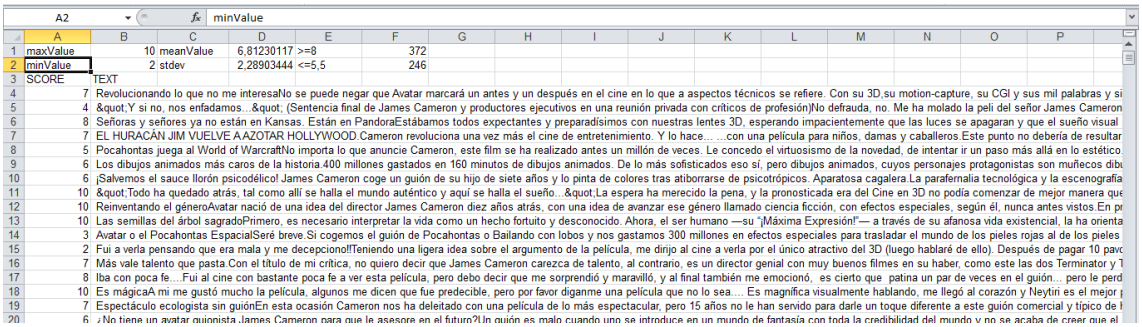
3.2.1. Bases de datos

Para poder realizar este proyecto se han utilizado dos bases de datos, una en inglés y otra en español.

La base de datos inglesa contiene dos carpetas, una con archivos de opiniones positivas y otra con las negativas. Para poder construir el formato adecuado se realizó un programa en java que leía los ficheros y los iba concatenando en un mismo archivo, llegando a tener 500 instancias positivas y 500 negativas. De esta forma, se pudo comenzar con el preprocesado del mismo y llevar a cabo los experimentos.

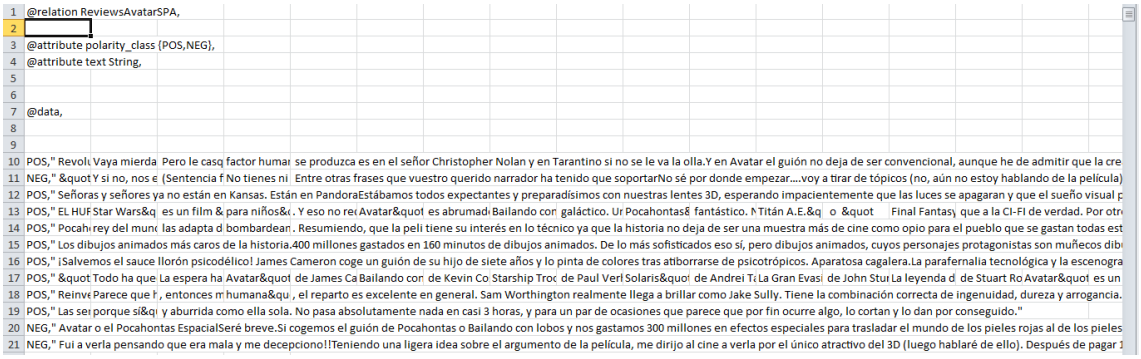
La base de datos española la componen cuatro archivos CSV de opiniones sobre espectáculos y películas. Uno de ellos, el de la película "Avatar", tiene casi 1000 instancias entre positivas y negativas y en un principio al ser de un tamaño parecido al utilizado en la base de datos inglesa, se procedió con su análisis como representación del español. Más tarde se verá que fue necesario cambiarla tras obtener los primeros resultados.

En este caso al ser archivos CSV (Figura 22), fue más fácil conseguir los archivos ARFF ya que WEKA tiene la opción de convertir de CSV a ARFF. Hubo primero que completar el archivo con la cabecera adecuada y cambiar la puntuación por las clases el resultado está en la Figura 23.



	A	B	C	D	E	F
1	maxValue	10	meanValue	6,81230117	>=8	372
2	minValue	2	stdev	2,28903444	<=5,5	246
3	SCORE	TEXT				
4	7	Revolucionando lo que no me interesaNo se puede negar que Avatar marcará un antes y un después en el cine en lo que a aspectos técnicos se refiere. Con su 3D,su motion-capture, su CGI y sus mil palabras y si				
5	4	"Y si no, nos enfadamos…" (Sentencia final de James Cameron y productores ejecutivos en una reunión privada con críticos de profesión)No defrauda, no. Me ha molado la peli del señor James Cameron				
6	8	Señoras y señores ya no están en Kansas. Están en PandoraEstábamos todos expectantes y preparadísimos con nuestras lentes 3D, esperando impacientemente que las luces se apagaran y que el sueño visual				
7	7	EL HURACÁN JIM VUELVE A AZOTAR HOLLYWOOD.Cameron revoluciona una vez más el cine de entretenimiento. Y lo hace… …con una película para niños, damas y caballeros.Este punto no debería de resultar				
8	5	Pocahontas juega al World of WarcraftNo importa lo que anuncie Cameron, este film se ha realizado antes un millón de veces. Le concedo el virtuosismo de la novedad, de intentar ir un paso más allá en lo estético.				
9	6	Los dibujos animados más caros de la historia.400 millones gastados en 160 minutos de dibujos animados. De lo más sofisticados eso sí, pero dibujos animados, cuyos personajes protagonistas son muñecos dibu				
10	6	¡Salvemos el sauce llorón psicodélico! James Cameron coge un guión de su hijo de siete años y lo pinta de colores tras atiborrarse de psicotrópicos. Aparatosa cagalera.La parafernalia tecnológica y la escenografía				
11	10	"Todo ha quedado atrás, tal como allí se halla el mundo auténtico y aquí se halla el sueño…"La espera ha merecido la pena, y la pronosticada era del Cine en 3D no podía comenzar de mejor manera que				
12	10	Reinventando el géneroAvatar nació de una idea del director James Cameron diez años atrás, con una idea de avanzar ese género llamado ciencia ficción, con efectos especiales, según él, nunca antes vistos.En pr				
13	10	Las semillas del árbol sagradoPrimero, es necesario interpretar la vida como un hecho fortuito y desconocido. Ahora, el ser humano —su "¡Máxima Expresión!"— a través de su afanosa vida existencial, la ha orienta				
14	3	Avatar o el Pocahontas EspacialSeré breve.Si cogemos el guión de Pocahontas o Bailando con lobos y nos gastamos 300 millones en efectos especiales para trasladar el mundo de los pieles rojas al de los pieles				
15	2	Fui a verla pensando que era mala y me decepciono!!Teniendo una ligera idea sobre el argumento de la película, me dirijo al cine a verla por el único atractivo del 3D (luego hablaré de ello). Después de pagar 10 pavo				
16	7	Más vale talento que pasta.Con el título de mi crítica, no quiero decir que James Cameron carezca de talento, al contrario, es un director genial con muy buenos filmes en su haber, como este las dos Terminator y T				
17	8	Iba con poca fe….Fui al cine con bastante poca fe a ver esta película, pero debo decir que me sorprendió y maravilló, y al final también me emocionó, es cierto que patina un par de veces en el guión… pero le perd				
18	10	Es mágicaA mi me gustó mucho la película, algunos me dicen que fue predecible, pero por favor diganme una película que no lo sea…. Es magnífica visualmente hablando, me llegó al corazón y Neytiri es el mejor p				
19	7	Espectáculo ecologista sin guiónEn esta ocasión Cameron nos ha deleitado con una película de lo más espectacular, pero 15 años no le han servido para darle un toque diferente a este guión comercial y típico de l				
20	6	¿No tiene un avatar guionista James Cameron para que le asesore en el futuro?Un guión es malo cuando uno se introduce en un mundo de fantasía con toda la credibilidad del mundo y no se acaba de creer que el				

Figura 22: Archivo original de opiniones de Avatar



```
1  @relation ReviewsAvatarSPA,
2
3  @attribute polarity_class {POS,NEG},
4  @attribute text String,
5
6
7  @data,
8
9
10 POS," Revol Vaya mierda  Pero le casq factor humar se produzca es en el señor Christopher Nolan y en Tarantino si no se le va la olla.Y en Avatar el guión no deja de ser convencional, aunque he de admitir que la cre
11 NEG," &quot Y si no, nos e (Sentencia f No tienes ni  Entre otras frases que vuestro querido narrador ha tenido que soportarNo sé por donde empezar....voy a tirar de tópicos (no, aún no estoy hablando de la película)
12 POS," Señoras y señores ya no están en Kansas. Están en PandoraEstábamos todos expectantes y preparadísimos con nuestras lentes 3D, esperando impacientemente que las luces se apagaran y que el sueño visual p
13 POS," EL HUF Star Wars&q es un film & para niños&( Y eso no re Avatar&quot es abrumad Bailando con galáctico. Ur Pocahontas& fantástico. N Titán A.E.&q o &quot Final Fantasy que a la CI-FI de verdad. Por otr
14 POS," Pocah rey del mun las adapta d bombardean. Resumiendo, que la peli tiene su interés en lo técnico ya que la historia no deja de ser una muestra más de cine como opio para el pueblo que se gastan todas est
15 POS," Los dibujos animados más caros de la historia.400 millones gastados en 160 minutos de dibujos animados. De lo más sofisticados eso sí, pero dibujos animados, cuyos personajes protagonistas son muñecos dib
16 POS," ¡Salvemos el sauce llorón psicodélico! James Cameron coge un guión de su hijo de siete años y lo pinta de colores tras atiborrarse de psicotrópicos. Aparatosa cagalera.La parafernalia tecnológica y la escenogra
17 POS," &quot Todo ha que La espera ha Avatar&quot de James Ca Bailando con de Kevin Co Starship Troc de Paul Verl Solaris&quot de Andrei Ta La Gran Evasi de John Stur La leyenda d de Stuart Ro Avatar&quot es un
18 POS," Reinve Parece que h, entonces m humana&qu, el reparto es excelente en general. Sam Worthington realmente llega a brillar como Jake Sully. Tiene la combinación correcta de ingenuidad, dureza y arrogancia.
19 POS," Las ser porque sí&q y aburrida como ella sola. No pasa absolutamente nada en casi 3 horas, y para un par de ocasiones que parece que por fin ocurre algo, lo cortan y lo dan por conseguido."
20 NEG," Avatar o el Pocahontas EspacialSeré breve.Si cogemos el guión de Pocahontas o Bailando con lobos y nos gastamos 300 millones en efectos especiales para trasladar el mundo de los pieles rojas al de los pieles
21 NEG," Fui a verla pensando que era mala y me decepciono!!Teniendo una ligera idea sobre el argumento de la película, me dirijo al cine a verla por el único atractivo del 3D (luego hablaré de ello). Después de pagar 1
```

Figura 23: Archivo preparado para convertir a ARFF

Esta base de datos contiene una diferencia entre el número de opiniones positivas y negativas muy grande, siendo 787 buenas opiniones y 158 malas. Por ello se creó otra base de datos que contenía las referencias de los cuatro espectáculos "Avatar", "El Cavernícola", "La venganza de Don Mendo" y "Por el placer de volver a verla". Todas ellas tenían el mismo problema que "Avatar", la desproporcionalidad de opiniones favorables, por lo que se basó el tamaño del archivo ARFF en el total de críticas negativas y se utilizaron sólo de las mejor valoradas el mismo número por espectáculo. El resultado fue una base de datos más pequeña pero con más riqueza de vocabulario al no estar basada en una sola película y con las mismas probabilidades de encontrar una referencia positiva que negativa.

3.2.2. Preprocesamiento de los archivos: StringToWordVector

En este apartado se van a comentar en detalle los procedimientos aplicados para el procesamiento de los datos. En este proyecto se van a realizar seis preprocesamientos diferentes basados en la manera de separar las palabras.

En todos los casos este preprocesamiento se realiza con el filtro no supervisado de WEKA *StringToWordVector* y son sus parámetros los que vamos a modificar. Por un lado, la utilización o no de las reglas de estadísticos numéricos y por otro lado, el modelo de *tokenizer*.

En la Figura 24 podemos ver la manera de introducir los datos en el filtro:

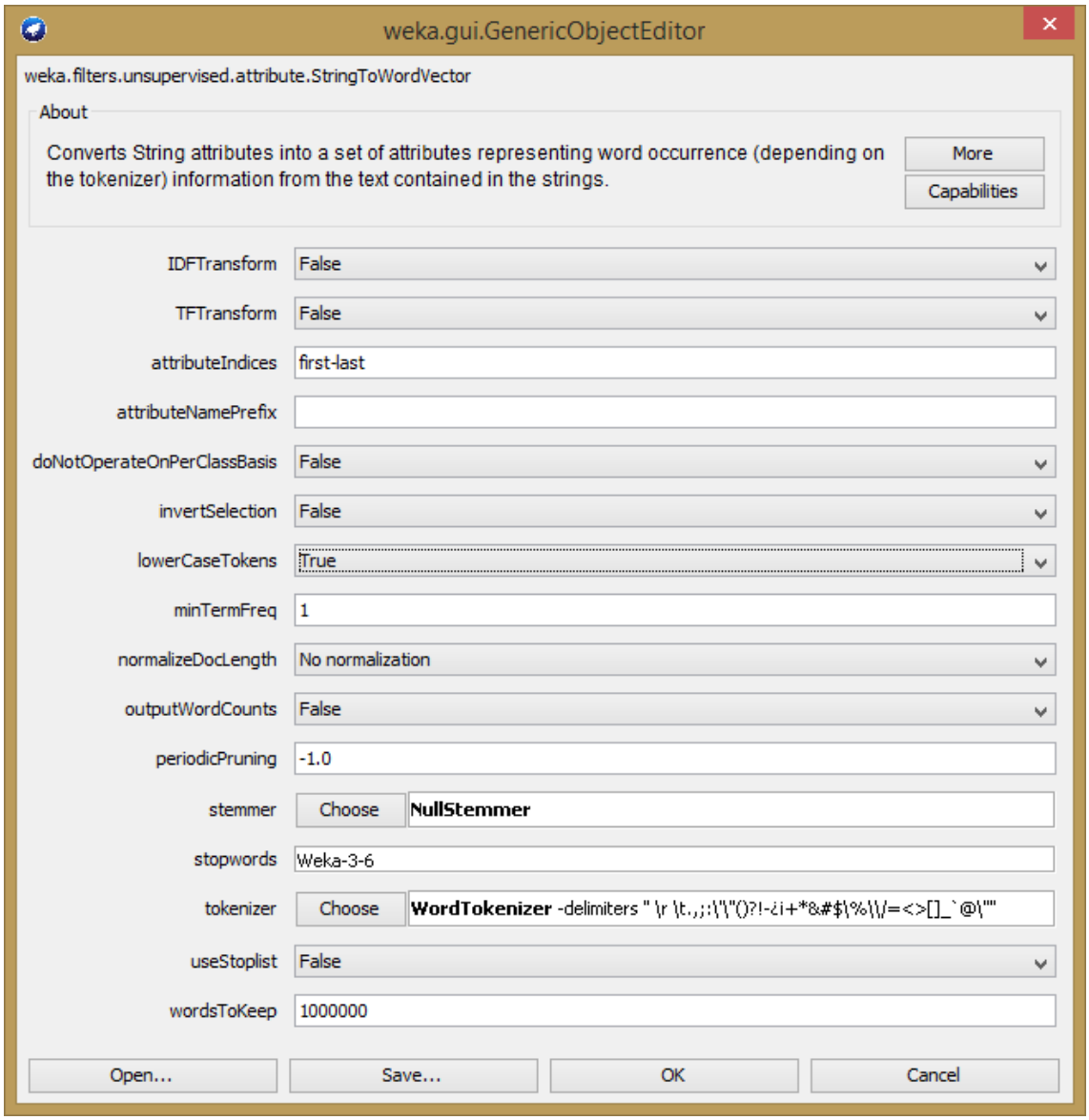


Figura 24: StringToWordVector

Los parámetros que van a mantenerse comunes en los seis casos van a ser:

- *attributeIndices*: first-last. Especifica el rango de atributos sobre los que trabajar. Es una lista separada por comas en la que se ordenan los índices de los atributos, siendo "*first*" y "*last*" valores válidos. Se pueden poner rangos separados por "-", por ejemplo: "first-3,5,6-10,last". En este caso se van a utilizar todos del primero al último.
- *AttributeNamePrefix:* vacío. Prefijo para crear los nombres de los atributos. Por defecto vacío.

- *DoNotOperateOnPerClassBasis*: false. Si estuviera puesto a verdadero, el número máximo de palabras y el término de mínima frecuencia no se ejecutarían en base a cada clase, sino considerando todos los documentos de todas las clases.
- *invertSelection:* false. Establece el modo de selección de atributos. Siendo falso sólo se procesarán los atributos seleccionados dentro del rango. Y si fuera verdadero entonces serían procesados los que se quedaran fuera.
- *lowerCaseTokens:* true. Cuando se incluyen las palabras en diccionario, al ser verdadero se transforman a minúsculas todas.
- *minTermFreq:* 1. Establece la frecuencia mínima de términos por clase.
- *NormalizeDocLength:* "No normalization". Define si la frecuencia de las palabras en un documento debe ser normalizada o no.
- *OutputWordCounts:* False. Si está en falso, en la salida establece una variable booleana 0 o 1 indicando la presencia de la palabra o no; en verdadero devuelve la cuenta de las palabras (conteo). En estos experimentos es suficiente con ver si la palabra está o no.
- *PeriodicPruning:* -1. Es el ratio (% de los datos de entrada) al cual se le recorta el diccionario periódicamente. El número de palabras que se especifica en el parámetro *WordsToKeep* se recorta tras crear el diccionario completo. Este parámetro, sin embargo, está ligado a la capacidad de memoria por lo que no se va a utilizar.
- *Stemmer:* NullStemmer. El algoritmo de localización de raíz de las palabras a utilizar. En este proyecto no se va a usar ninguno debido a que estudios han demostrado que se obtienen mejores resultados utilizando N-gramas, habiendo errores métricos cuando palabras de significado distinto se guardan bajo la misma raíz, o palabras que debieran estar bajo la misma y se clasifican como dos distintas.
- *StopWords:* weka-3-6. Listado de palabras que si no se especifica se utilizan por defecto. En estos experimentos se usará la que ha definido WEKA, basada en el sistema "*Rainbow"* [35] .
- *useStopList:* False. Si se pone a true ignora la lista de palabras que se incluye en StopWords.
- *wordsToKeep:* 1000000. El número palabras que se intentan guardar. Por defecto está puesto a 1000, pero para conseguir mejores resultados se ha incrementado.

Para los casos en los que se modifica el *Tokenizer* los siguientes parámetros son los mismos y se explican más adelante:

- *IDTFTransform:* False.
- *TFTransform:* False.

El primer paso para procesar un texto es dividirlo en palabras, frases, símbolos u otros elementos significativos que denominaremos "*tokens*", y eso es lo que hace el *Tokenizer*. En este proyecto vamos a utilizar uno más sencillo y otro con inteligencia.

3.2.2.1. Wordtokenizer

Vamos utilizar el *Tokenizer* de *WordTokenizer*, que se basa en la separación entre las palabras, que típicamente es el espacio por defecto, pero en esta herramienta existe la posibilidad de decidir los elementos que se ajustan más al texto. Por ejemplo en el caso del inglés no sería necesario tener en cuenta los símbolos "¿" y "¡", puesto que no existen en su lengua.

De todas formas y para poder hacer más comparables los resultados se han utilizado los mismos separadores en ambas lenguas, que son los siguientes: \n\r\t.,;:'"()?!-¿¡+*&#$%\/=<>[]_`@".

Este tipo de tokenizador es sencillo y sigue las reglas más básicas de transformación del texto a palabras, debido a que muchas veces el espacio no es un separador del token, por ejemplo en "Los Ángeles" o el guión en palabras compuestas, sobre todo en Inglés, o incluso el punto que comúnmente es un delimitador de frases, a veces se incluye en tokens que son abreviaturas tales como "Sr." o "U.S.A". Por otro lado, se han estudiado maneras de imprimir inteligencia al tokenizador y entre ellas está el caso de los N-gramas que es el otro *Tokenizer* que se ha utilizado y que se explica a continuación.

3.2.2.2. N-grams [5] [6]

Este tipo de *Tokenizer* está basado en un modelo más complejo que la simple separación de las palabras por símbolos. En la sección 2.1.1 se explica en qué consiste.

En los siguientes experimentos se van a utilizar tres modelos de n-gramas: Unigramas, donde los tokens van a estar formados por una sola palabra. N-gramas (3,1) para tokens desde una hasta tres palabras. Y trigramas dónde los tokens los forman tres palabras.

Distinguiendo las palabras mediante los siguientes separadores: " \r\n\t.,;:\'\"()?!". Esta lista de símbolos es más corta que la utilizada en el caso del *WordTokenizer*, esto se debe a que por la complejidad de este modelo, la herramienta no permitía el uso de cadenas más largas de caracteres delimitadores.

3.2.2.3. Parámetro IDFTransform [25] [26] [27]

Este parámetro se ha probado porque al modificar la frecuencia ayuda a mejorar la precisión de los clasificadores. En nuestro caso, se va a utilizar y aplicar sólo en el caso de que el algoritmo organizador de tokens sea *WordTokenizer*. Debido a los resultados obtenidos no se vio que fuera a mejorar los resultados de los *Unigramas*.

3.2.2.4. Parámetro TFTransform

Este parámetro como el anterior sólo se va a utilizar para el caso de que el *tokenizer* sea el *Wordtokenizer*. Los resultados obtenidos para este caso no pareció que fueran a mejorar a los n-gramas.

3.2.3. Organización de las listas de palabras: Attribute Selection

Una vez que se ha creado el diccionario de palabras vamos a utilizar el filtro supervisado *Attribute Selection*. Un filtro supervisado se utiliza para seleccionar atributos, es muy flexible y permite varios métodos de evaluación y búsqueda para combinar. Su interfaz en WEKA se muestra en la Figura 25:

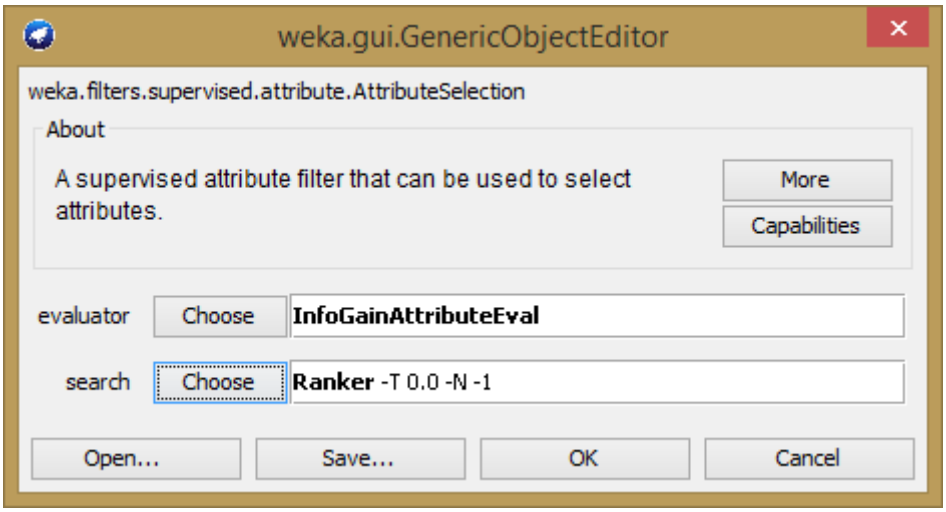


Figura 25: Interfaz en WEKA de Attribute Selection

Como podemos ver sólo hay dos variables, por un lado el evaluador que determina cómo valorar atributos y conjuntos de atributos y por otro el buscador. En este proyecto siempre se van a utilizar estos: el evaluador es el *InfoGainAttributeEval* y el buscador es el *Ranker*.

3.2.3.1. Evaluador InfoGainAttributeEval [28]

Los evaluadores de atributos según su entropía se utilizan en experimentos que utilizan algoritmos de autoaprendizaje. El propósito de estas técnicas es descartar las características irrelevantes o redundantes dado un vector de elementos, en este caso palabras.

La entropía es una medida muy común en teoría de la información, caracteriza la pureza de un conjunto de ejemplos arbitrario. Se fundamentan en las propiedades de ordenación de atributos según su ganancia de información. Se considera la entropía como una medida de lo impredecible. La entropía de Y se calcula como:

$$H(Y) = -\sum_{y \in Y} p(y) \log_2(p(y))$$

Dónde *p(y)* es la función de densidad de probabilidad marginal de *y* sobre la variable aleatoria *Y*. Si los valores de *Y* dentro de un conjunto *S* vienen repartidos según los valores de una segunda variable *X*, y la entropía de *Y* respecto a los valores inducidos de *X* es menor que la entropía de *Y* antes de las particiones hay una relación entre *X* e *Y*. La entropía de *Y* después de haber observado *X* se describe según la fórmula:

$$H(Y/X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y/x) \log_2(p(y/x))$$

Siendo *p(y|x)* la probabilidad condicionada de *y* dado *x*.

Dado que la entropía es un criterio de impureza de un conjunto de entrenamiento *S* podemos definir la medida que refleja la información obtenida de *Y* respecto a *X* que representa la cantidad en que la entropía de *Y* decrece al conocer la variable *X*. Esta medida es conocida como ganancia de información (*Information Gain*) y es dada por la siguiente fórmula:

$$IG = H(Y) - H(Y/X) = H(X) - H(X/Y)$$

Es una expresión simétrica que evalúa de la misma manera la información obtenida de *Y* respecto a *X* como de *X* respecto a *Y*. La debilidad de esta medida es que está sesgada a favor de las características más numerosas aunque no sean las que contienen más información.

Traduciendo esta fórmula a los parámetros de WEKA se evalúa el valor del atributo midiendo la ganancia de la información, respecto a la clase:

$$InfoGain(Class, Attribute) = H(Class) - H(Class|Attribute)$$

Su interfaz en WEKA se muestra en la Figura 26:

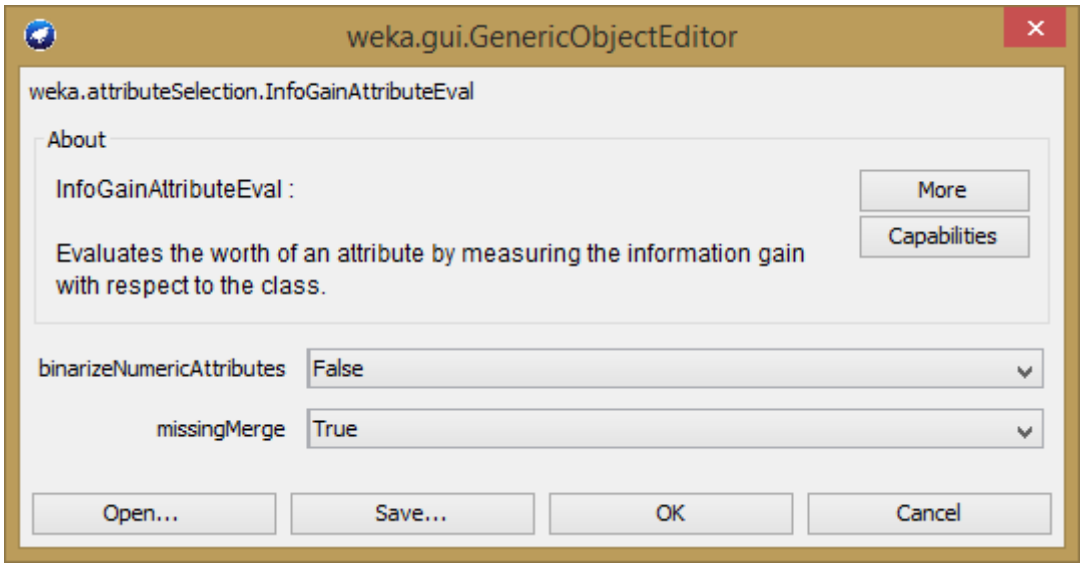


Figura 26: InfoGainAttributeEval en WEKA

Los parámetros que tiene y los valores con los que se van a utilizar son:

- *binarizeNumericAttribute*: False. Este parámetro haría dos clases de atributos numéricos binarios en vez de discretizarlos con propiedad.
- *missingMerge:* True. Distribuye las cuentas de los valores que faltan. Éstas se distribuyen sobre otros valores en proporción a su frecuencia; de otro modo los que faltaran se tratarían como valores separados.

3.2.3.2. Buscador Ranker

Este algoritmo se utiliza en conjunción con el método evaluador, y ordena los atributos según el valor individual de cada uno. Sus parámetros se seleccionan en WEKA según la Figura 27.

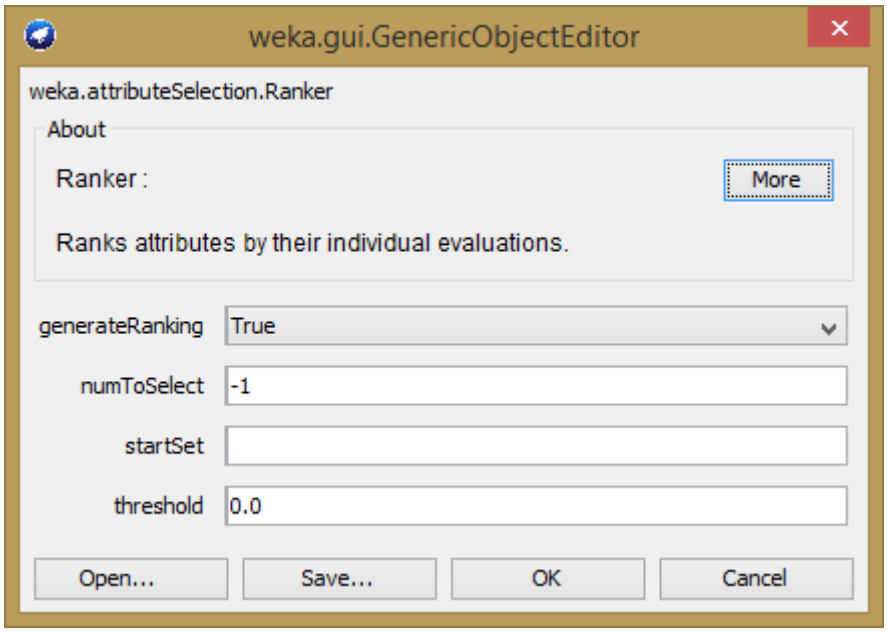


Figura 27: Ranker en WEKA

Los valores de los parámetros son:

- *generateRanking:* True. Es una opción constante, este algoritmo sólo es capaz de generar rankings de atributos.
- *numToSelect:* -1. Especificación del número de atributos a retener. El valor por defecto (-1) indica que todos los atributos se deben conservar. Se puede usar esta opción o la de umbral para reducir la lista de atributos.
- *startSet:* vacío. Especifica un conjunto de atributos a ignorar. Cuando se genera la lista, *Ranker* no evaluará los atributos de esta lista. Esta se define como una lista de índices de atributos separados por comas empezando en 1. Puede incluir rangos, por ejemplo: 1,2,5-9,17.
- *Threshold:* 0.0. Este es el umbral bajo el cual se descarta atributos. El valor por defecto (-1.7976931348623157E308) no descarta ningún atributo.

3.3. Clasificadores

En esta sección se van a describir los clasificadores que se van a utilizar en los experimentos y cómo seleccionarlos en WEKA.

3.3.1. Naïve Bayes [7] [8]

Este clasificador se explica en la sección 2.2.1, a la hora de utilizarlo en WEKA los parámetros que se pueden modificar son como se muestran en la Figura 28:

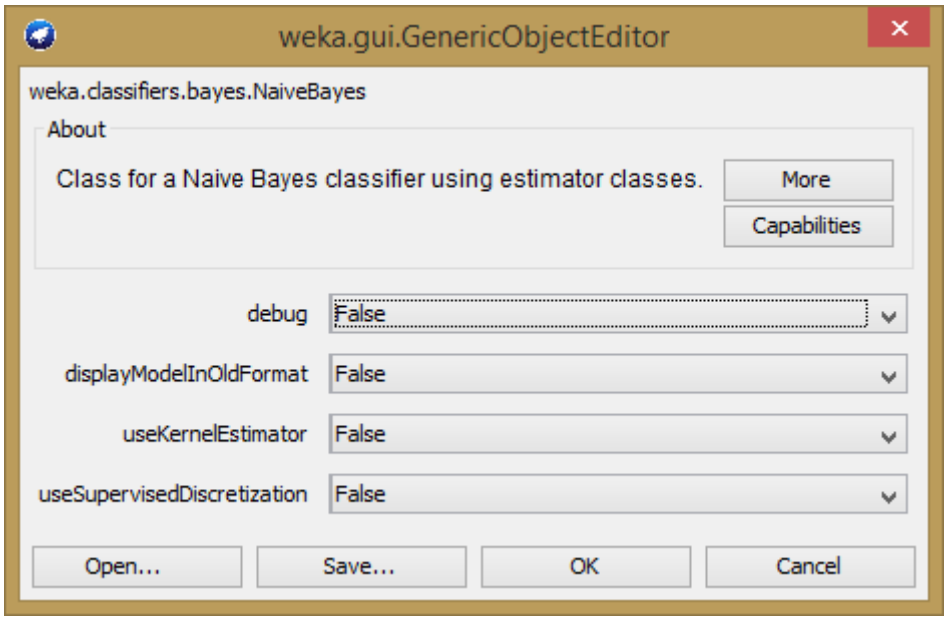


Figura 28: Clasificador Naïve Bayes enWEKA.

Dónde durante estos experimentos van a estar a falso todas.

- *Debug:* Cuando se selecciona, al realizar el experimento dará más información a la salida.
- *displayModelInOldFormat*: Utiliza el formato antiguo en la salida. Este tipo de formato es mejor cuando hay muchos valores de clases dado que el nuevo está optimizado para menor número de clases y más atributos.
- *useKernelEstimator:* Utiliza un estimador de para atributos numéricos antes que una distribución normal.
- *useSupervisedDiscretization*: Utiliza la discretización supervisada para convertir los atributos numéricos en normales.

3.3.2. SMO [29]

Este clasificador se explica en la sección 2.2.7. En WEKA esta implementación reemplaza los valores que faltan y transforma atributos nominales a binarios. También normaliza todos los atributos por defecto. (En este caso los coeficientes en la salida están basados en la información normalizada y no en la original). Sus parámetros se muestran en la Figura 29.

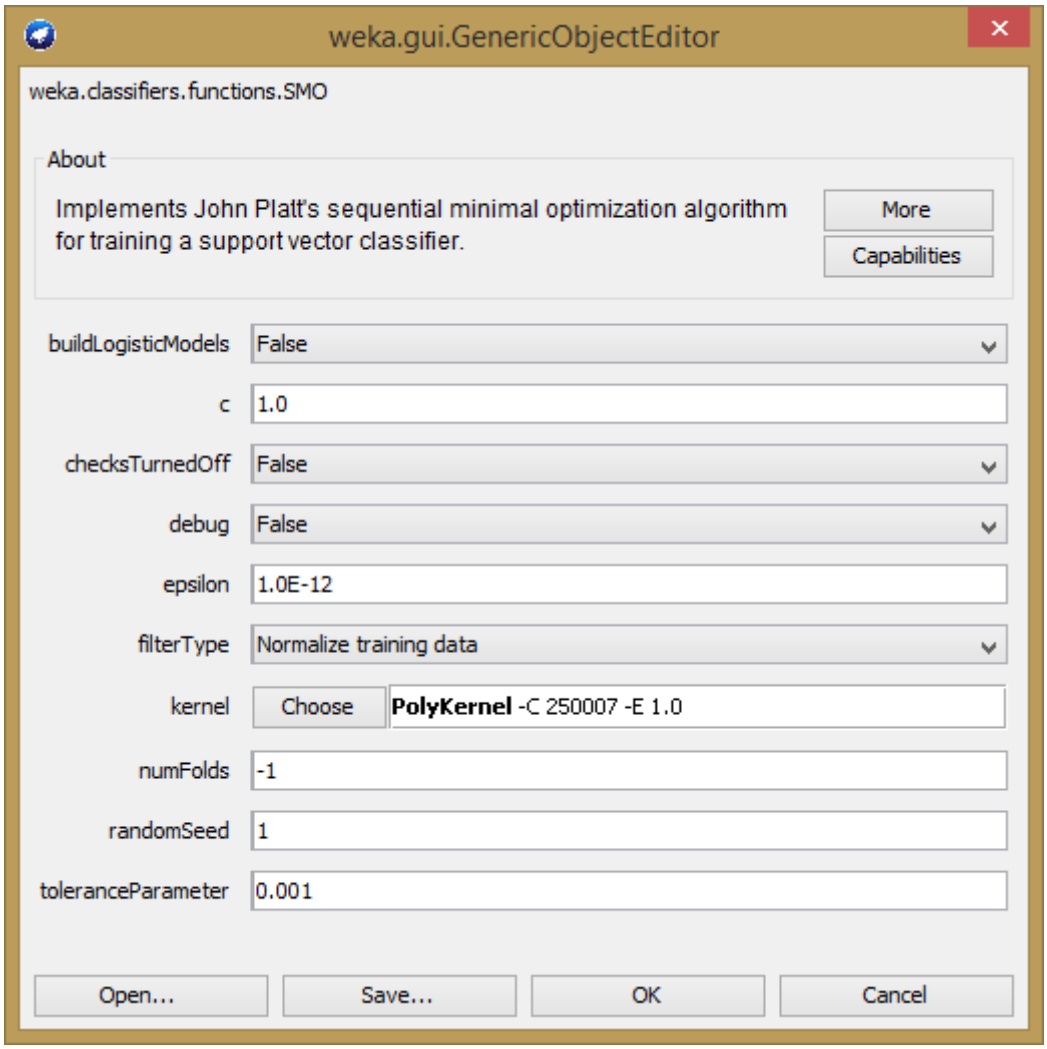


Figura 29: SMO en WEKA

Los parámetros que se han usado en los experimentos han sido los siguientes:

- *buildLogisticModels:* Falso. Para encajar los modelos logísticos a las salidas (para estimadores de probabilidad)
- *c:* 1.0. La complejidad del parámetro C.
- *checksTurnedOff*: False. Activa consumo de tiempo, se debe usar con precaución.
- *Debug:* False. Si se pone a verdadero, el clasificador dará más información a la salida.
- *Epsilon:* 1.0E-12. Este parámetro no debe cambiarse, es para calcular el error de redondeo.
- *filterType:* Normalize training data. Determina cómo o si los datos serán transformados.
- *Kernel:*PolyKernel. El kernel a utilizar.
- *numFolds:* -1. El número de partes para la validación cruzada usada para generar el conjunto de entrenamiento para modelos logísticos. (-1 significa utilizar el conjunto de entrenamiento)
- *ramdomSeed:* 1.Valor aleatorio de semilla para la validación cruzada.
- *toleranceParameter:* 0.001. El parámetro de tolerancia. No debe cambiarse nunca.

Dentro de los tipos de Kernel hay varias opciones (Figura 30), pero para este estudio el que mejor aplica es el *PolyKernel*, que es un Kernel polinómico que sigue la siguiente fórmula:

$$K(x, y) = \langle x, y \rangle^p \; o \; K(x, y) = (\langle x, y \rangle + 1)^p$$

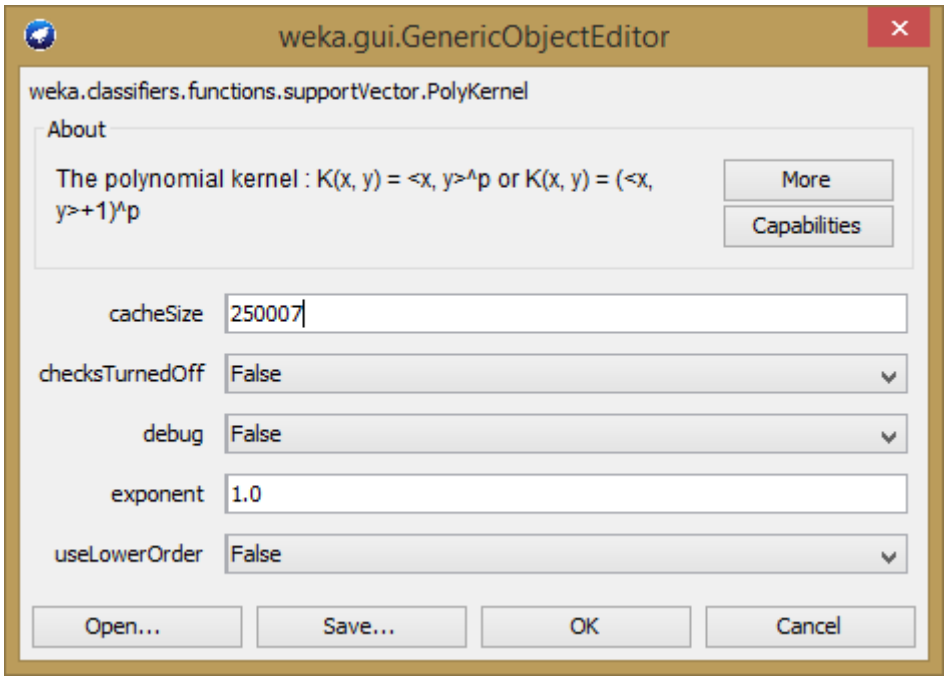


Figura 30: PolyKernel en WEKA

Las opciones a seleccionar en este caso y cómo se han dejado en los experimentos son:

- *cacheSize*: 250007. El tamaño de la caché debe ser un número primo. 0 para la caché llena y -1 para desactivarla.
- *checksTurnedOff*: False. Desactiva el consumo de tiempo de las comprobaciones. Debe usarse con precaución.
- *Debug*: Activa el modo depurador de información.
- *Exponent*: 1.0. El valor del exponente.
- *useLowerOrder*: False. Para usar términos de menor exponente.

3.3.3. Naïve Bayes Multinomial [14]

Este clasificador se ha explicado en la sección 2.2.1.1. En WEKA para este clasificador sólo existe la posibilidad de activar el depurador (Figura 31).

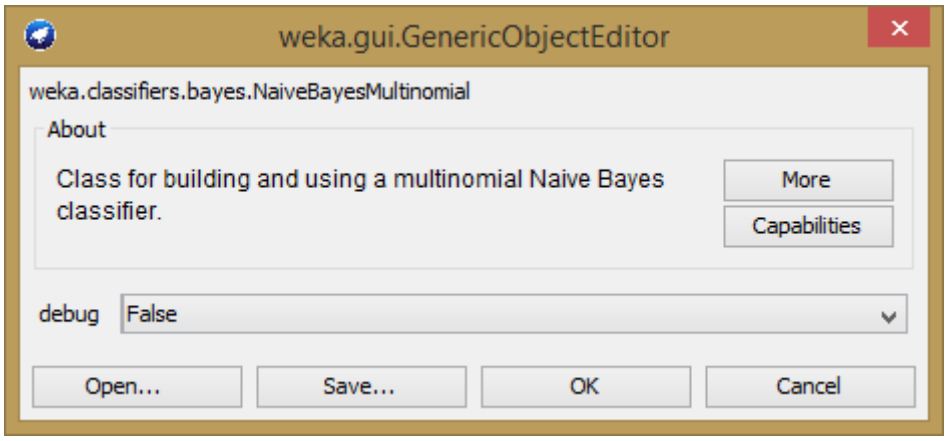


Figura 31: Naïve Bayes Multinomial en WEKA

3.3.4. Naïve Bayes Multinomial Updateable

Este clasificador se comenta en la sección 2.2.1.2. Aunque es una versión incremental del anterior.

Tampoco hay parámetros seleccionables en WEKA, sólo la activación del depurador (Figura 32):

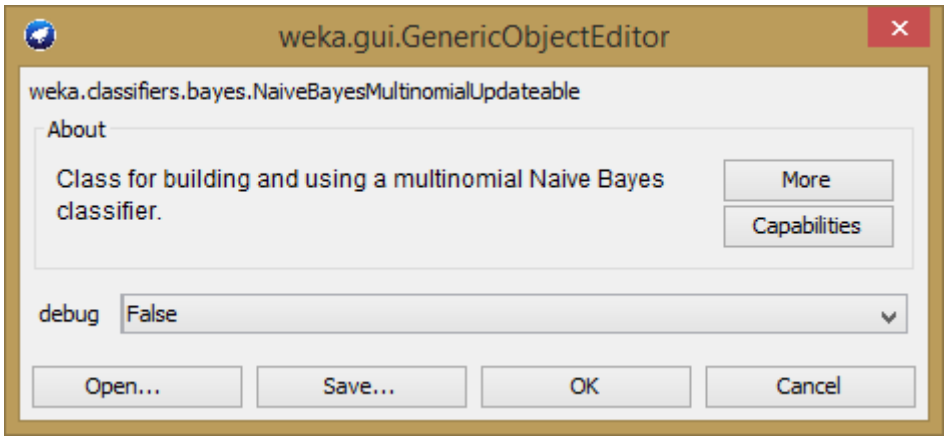


Figura 32: Naïve Bayes Multinomial Updateable en WEKA.

3.4. Opciones de testado

Para sacar los resultados de los clasificadores se han utilizado:

- Evaluación en el conjunto de entrenamiento
- Evaluación sobre un 10% del conjunto de entrenamiento.
- Evaluación con validación cruzada en 10 partes.

La evaluación sobre el conjunto de entrenamiento se ha realizado para verificar el correcto funcionamiento de los modelos. Debido a la escasez de datos la validación cruzada será la que nos dé un resultado más cercano a la realidad.

4. Experimentos

La realización de los experimentos ha sido en tres fases diferentes. La primera sobre la base de datos de textos en inglés, la segunda sobre la primera base de datos en español y, tras los resultados, una tercera en la que se balanceó la base de datos y se limpió de caracteres extraños.

En las tres fases se ha seguido el mismo procedimiento, primero se han filtrado con *wordtokenizer*, y se han clasificado según NaïveBayes, SMO, Naïve Bayes Multinomial y Naïve Bayes Multinomial Updateable. Acto seguido se han procesado las bases de datos con Unigramas, n-gramas (1,3), Trigramas y seleccionando los parámetros IDFT y TFT sobre el *wordtokenizer*, clasificándose como en el caso anterior.

Inicialmente se han evaluado todas en el conjunto de entrenamiento, pero para comparar finamente la base en inglés y la última española, se han evaluado también con una partición del 90% para entrenamiento y un 10% para probar y validación cruzada de 10 partes.

Pasamos a describir los resultados. En vez de seguir el orden en el que se realizaron voy a comenzar por contar los resultados del preprocesamiento en las bases, y después pasaré a relatar los valores obtenidos por clasificador haciendo comparaciones, una entre todos los preprocesadores sobre un mismo clasificador y otra entre todos los clasificadores con un único preprocesador. Esto dentro de una misma lengua y luego comparando ambas.

4.1. Efectos del preprocesamiento

4.1.1. Base de datos Inglesa

Primero vamos a ver los efectos sobre la base de datos en inglés.

Al pasar la base por el filtro *StringToWordVector* (STWV) independientemente de sus parámetros, se crean muchos más atributos de los que serían procesables. Sin embargo, al aplicar el *AttributeSelection* se consigue que ese número de atributos que se han generado disminuya considerablemente consiguiendo simplificar el problema a la par que organizar los atributos de una manera lógica, por pesos en vez de en orden alfabético.

4.1.1.1. StringToWordVector: WordTokenizer

Como se ha contado en secciones anteriores los separadores que se han utilizado en este filtro son: " \r\n\t.,;:\"\'()?!-¿¡+*&#$\%\\/=<>[]_`@".

La interfaz de WEKA en la que se añade la variable se muestra en la Figura 33:

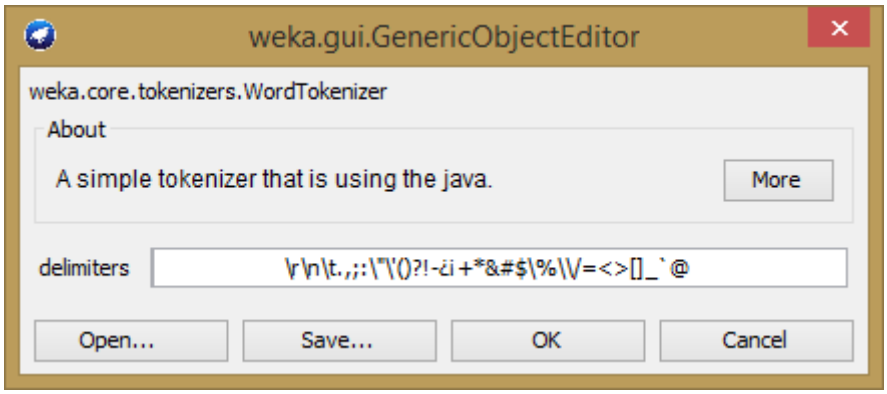


Figura 33: Wordtokenizer en Weka

El resultado de aplicar el STWV se muestra en la Figura 34:

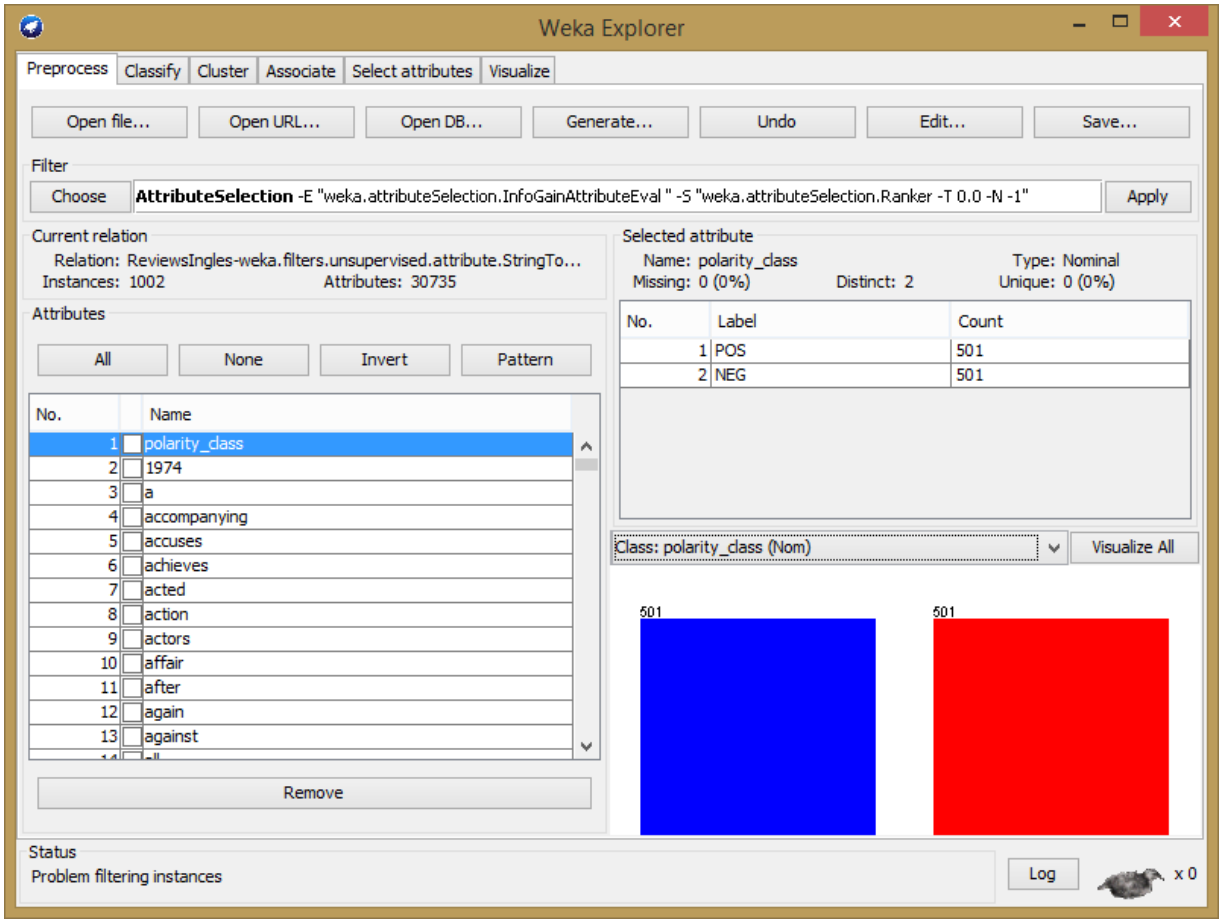


Figura 34: Resultado de aplicar el STWV con *wordtokenizer* sobre la base inglesa

Podemos ver que se generan 30.735 atributos, que hay el mismo número de instancias positivas que negativas y que la lista de palabras generada está en orden alfabético.

Tras haber aplicado el segundo filtro, este el resultado de la clasificación se muestra en la Figura 35:

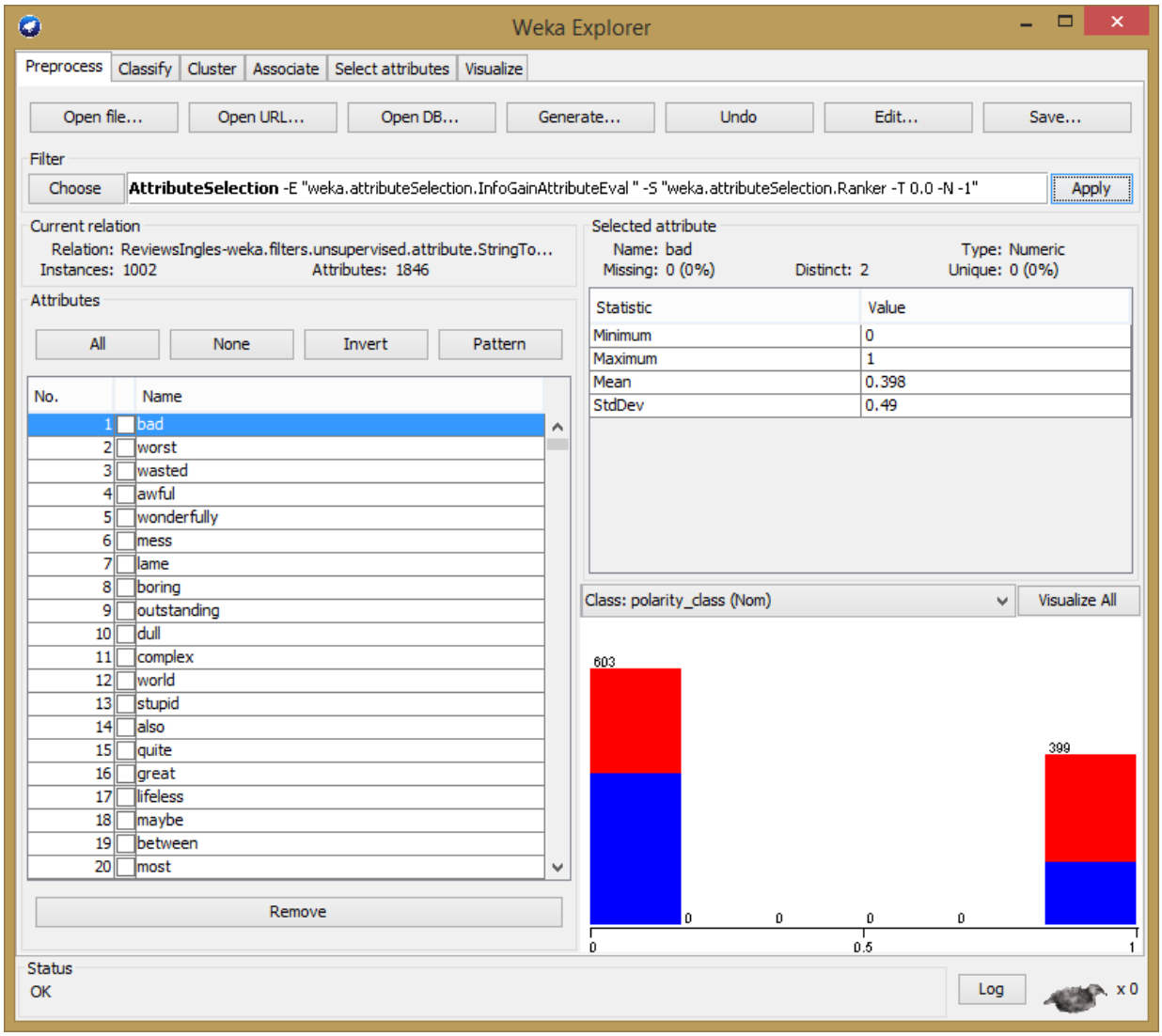


Figura 35: Resultado tras aplicar el filtro *AttributeSelecition* cuando se ha aplicado un filtro *StringToWordVector* con tokenizer igual a *Wordtokenizer* en la base inglesa.

Vemos que el número de atributos ha disminuido a 1.846. Podemos ver los parámetros del atributo seleccionado y vemos que tiene más probabilidades de ser clasificado como negativo.

4.1.1.2. StringToWordVector: Ngrams (1,1)

Como he comentado en otra sección en este caso los separadores son: \r\n\t.,;:\'\"()?!

En WEKA al seleccionar el *Tokenizer NGramTokenizer*, se debe decir cuál es el número mínimo y el máximo de palabras, además de sus separadores (Figura 36). Como se dice en el título de esta sección en este caso sólo se va a buscar una palabra, unigramas:

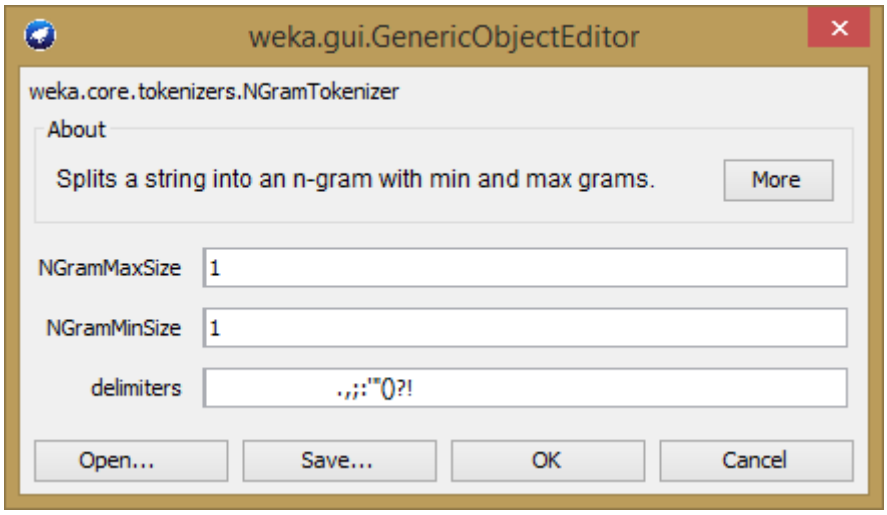


Figura 36: NGramTokenizer en WEKA

Tras aplicar los dos filtros los resultados se muestran en la Figura 37, y el número de atributos ahora es de 1.917, bastantes menos:

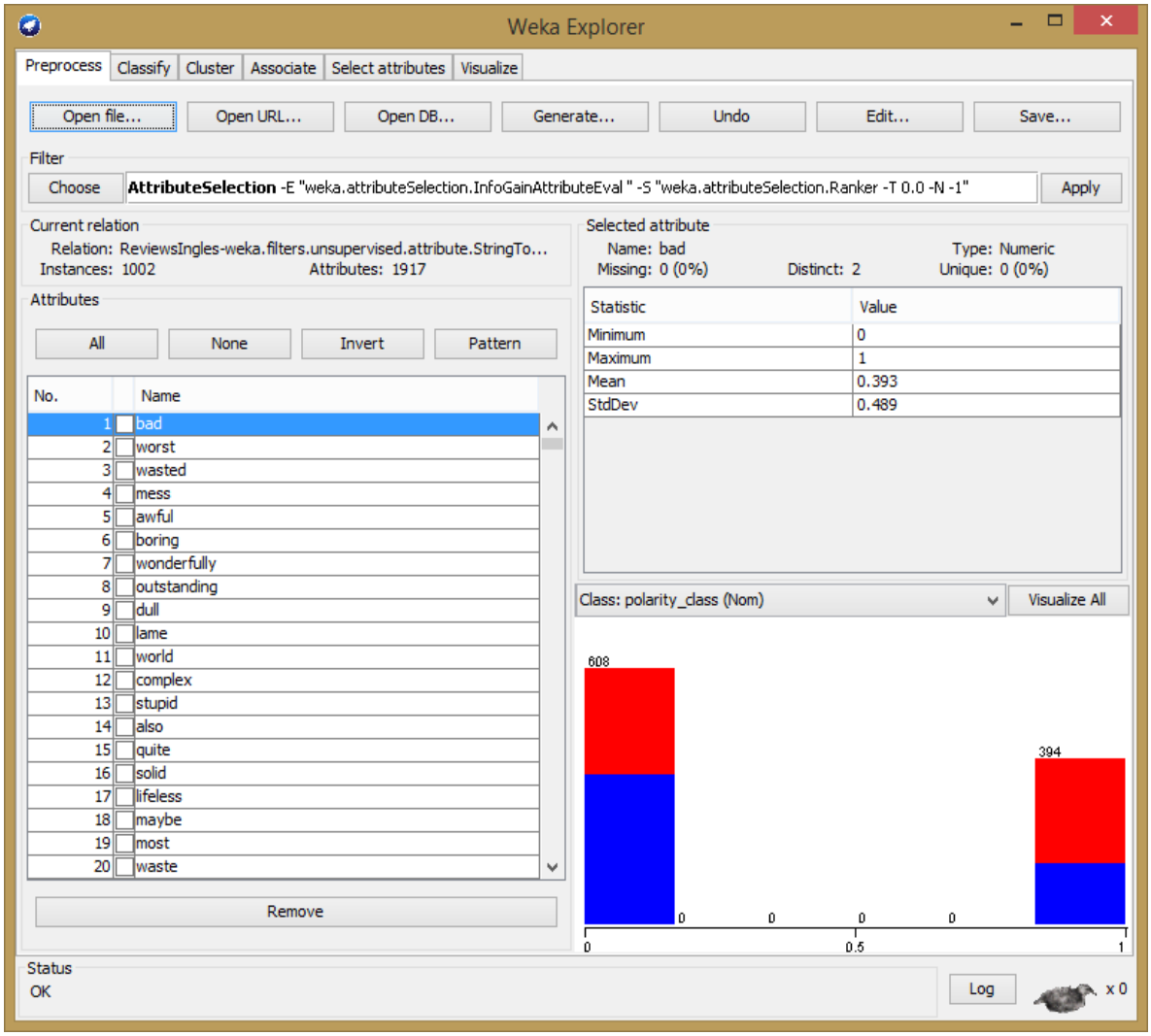


Figura 37: Resultado de aplicar los filtros *StringToWordVector* con Unigramas y el *AttributeSelection*

4.1.1.3. StringToWordVector: Ngrams (3,3)

Como en el caso anterior, sólo que en este caso se van a buscar agrupaciones de tres palabras, trigramas.

En este caso, antes de aplicar el *attributeSelection* se crean un total de 534.340 atributos. Por eso al aplicar el segundo filtro, se requiere de mayor tiempo de cómputo para realizar la selección de los mismos. Finalmente, el conjunto final de atributos consta de 3.613 elementos (Figura 38), que comparado con los casos anteriores, donde el número de atributos era 1.846 y 1.917, se comprueba que hay un mayor número de combinaciones de palabras que dan lugar casi el doble de tokens.

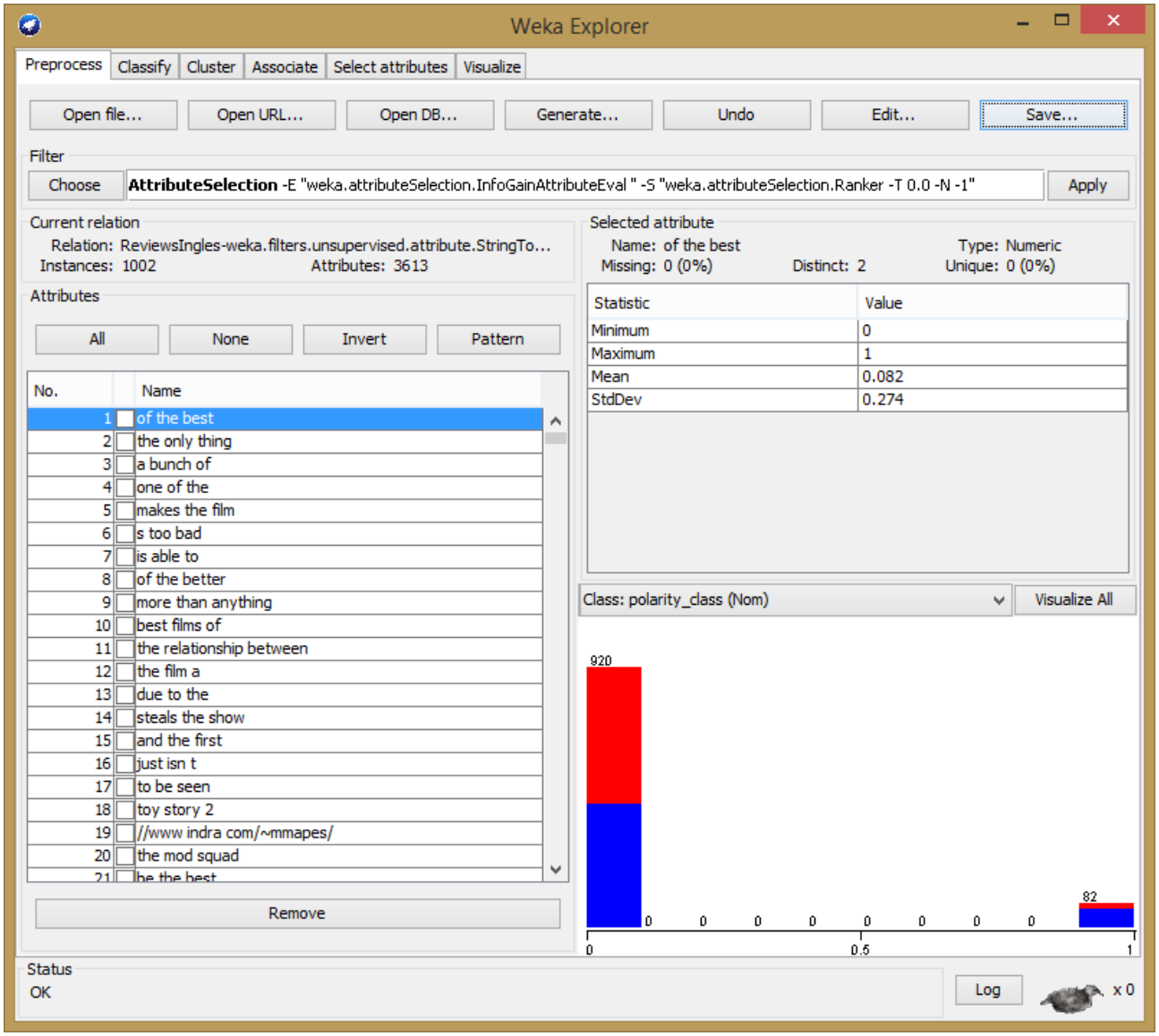


Figura 38: Resultado de aplicar los filtros *StringToWordVector* con tokenizer *NGramTokenizer(3,3)* y*AttributeSelection* en la base en inglés.

4.1.1.4. StringToWordVector: Parámetro TFT a true con wordTokenizer

El resultado de la lista generada en este caso coincide con la aplicación del *wordtokenizer*. No se observa ninguna diferencia evidente.

4.1.1.5. StringToWordVector: Parámetro IDFT a true con wordTokenizer

El resultado de la lista generada en este caso coincide con la aplicación del *wordtokenizer*. No se observa ninguna diferencia evidente.

4.1.1.6. Lista de los primeros 20 tokens según cada filtro

En la siguiente tabla se recogen los 20 primeros tokens que resultan tras aplicar los filtros. Podemos ver que entre el *WordTokenizer* y el *NgramTokenizer(1,1)* (Tabla 3) hay casi las mismas palabras, pero difieren en el orden.

WordTokenizer	**NgramTokenizer(1,1)**	**NGramTokenizer(3,3)**
Bad	Bad	of the best
Worst	Worst	the only thing
Wasted	Wasted	a bunch of
Awful	Mess	one of the
Wonderfully	Awful	makes the film
Mess	Boring	s too bad
Lame	Wonderfully	is able to
Boring	Outstanding	best films of
Outstanding	Dull	more than anything
Dull	Lame	of the better
Complex	World	the relationship between
World	Complex	the film a
Stupid	Stupid	due to the
Also	Also	steals the show
Quite	Quite	and the first
Great	Solid	just isn t
Lifeless	Lifeless	to be seen
Maybe	Maybe	we re supposed
Between	Most	than just a
Most	Waste	the characters the

Tabla 3: Primeros 20 tokens resultado de aplicar los filtros *StringToWordVector* y *AttributeSelction* en los tres casos en la base inglesa.

4.1.2. Base de datos española

Ahora siguiendo las mismas directrices que en la base de datos inglesa pasamos a comprobar el efecto de aplicar los filtros sobre la base de datos en español. Como se ha comentado en la sección 3.2.1 la base de datos española escogida para realizar los experimentos ha pasado por tres fases hasta conseguir la que más se acerca a la realidad. Veremos ahora las diferencias que han hecho que se fuera optimizando la base.

4.1.2.1. StringToWordVector: WordTokenizer

En el primer caso vamos a comprobar la lista que se generó al aplicar los filtros sobre la base que constaba únicamente de opiniones sobre le película "Avatar" (Figura 39):

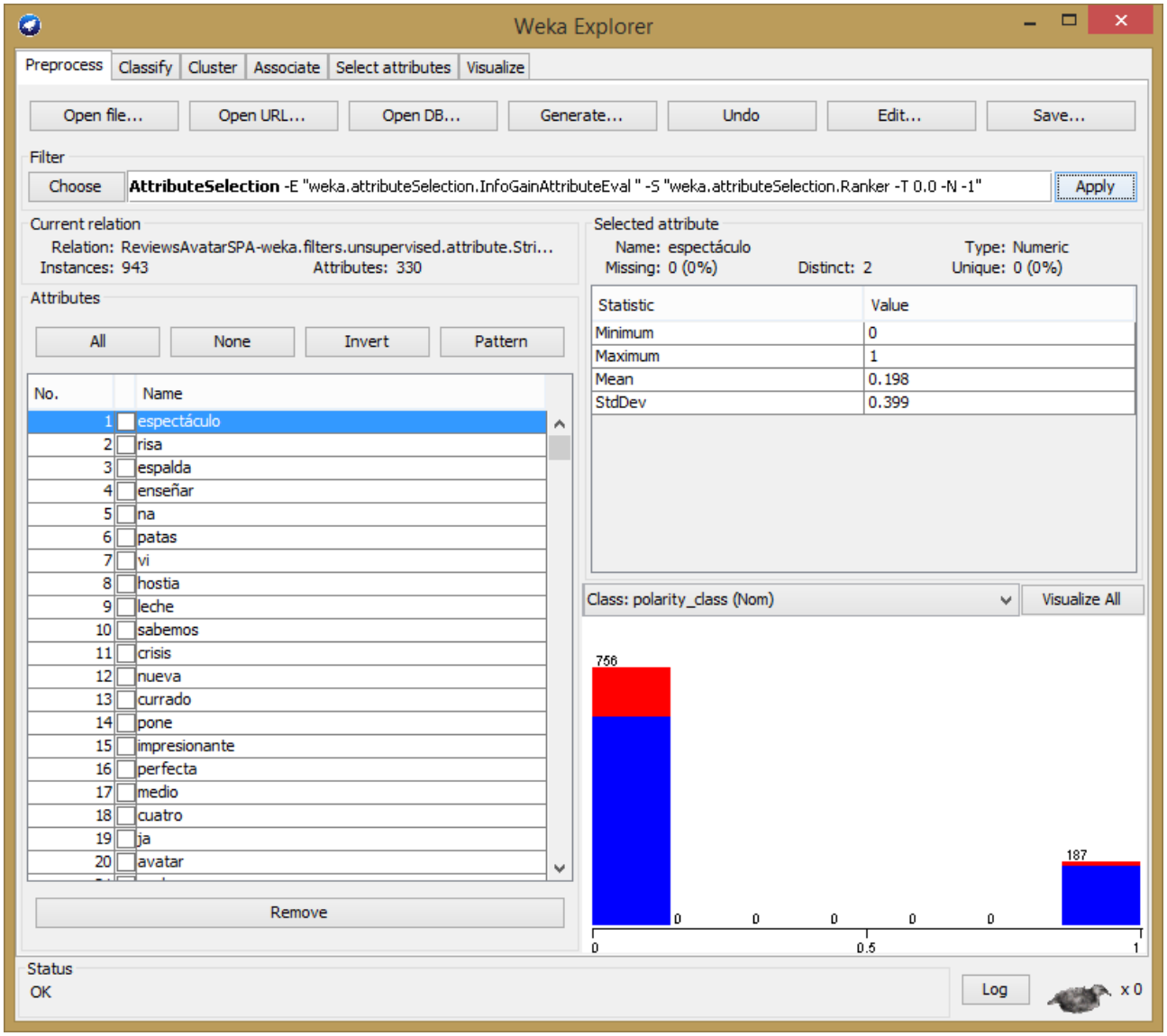


Figura 39: Resultado de aplicar los filtros *StringToWordVector* con *wordtokenizer* y *AtributteSeleccion* sobre la base española de opiniones de Avatar.

Como puede observarse, esta base de datos está desbalanceada ya que hay muchos más ejemplos positivos que negativos. Este hecho puede dar lugar a que la clase mayoritaria esté mejor modelada con lo que se produciría un sesgo hacia dicha clase. Por este motivo, se creó una nueva base de datos balanceada que constaba de opiniones de cuatro películas y espectáculos distintos. Los resultados con esta nueva base de datos se muestran en la Figura 40:

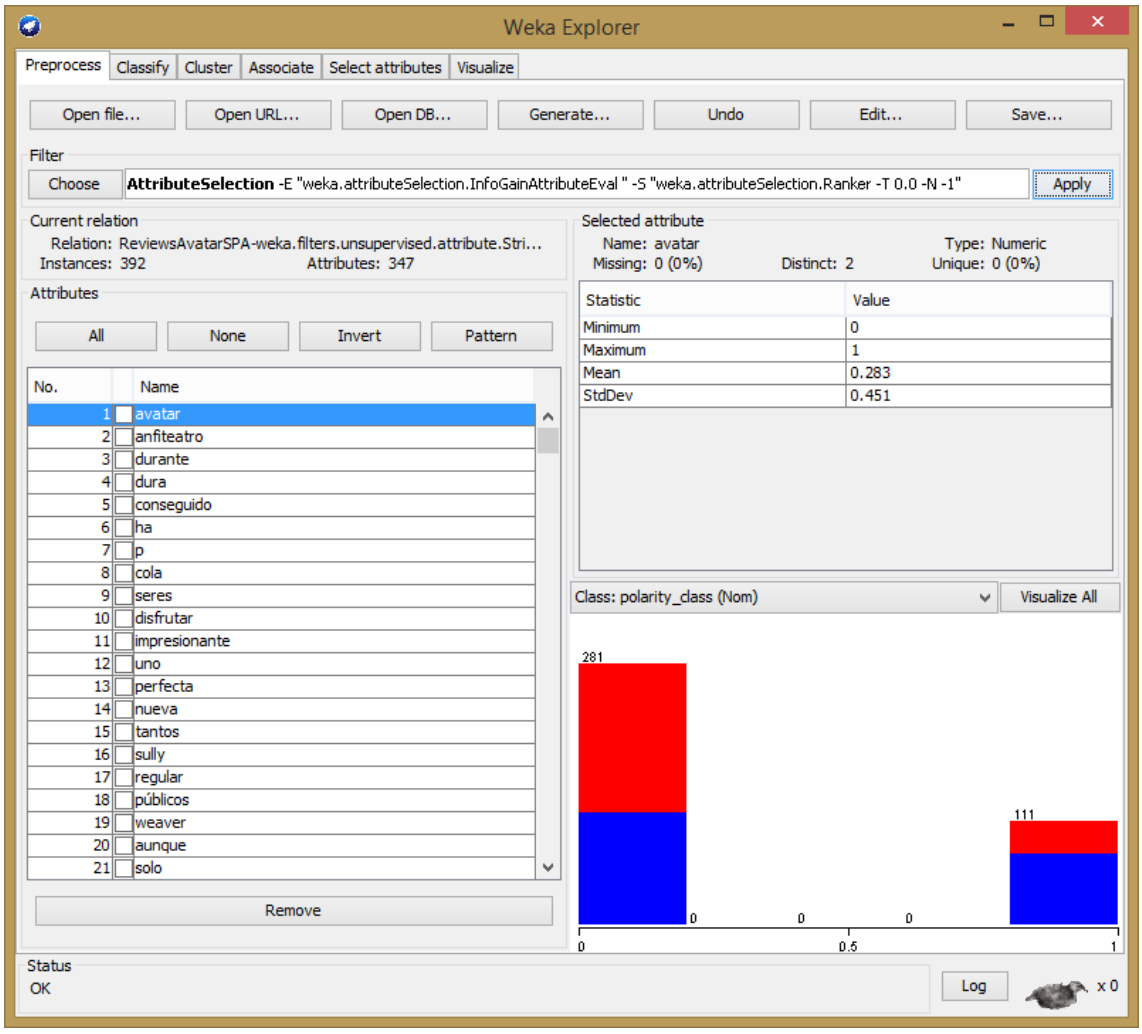


Figura 40: Resultado de aplicar los filtros *StringToWordVector* con *wordtokenizer* y *AtributteSeleccion* sobre la base española de opiniones de varios espectáculos.

Llama la atención en este caso que la primera palabra sea avatar, cuando en la lista anterior estaba en el puesto 20 y era una base de datos sólo con comentarios para esa película. La explicación es breve, como se ha comentado en la sección de bases de datos, se ha explicado que el número de instancias venía limitado por las opiniones negativas de todos los espectáculos, y siendo Avatar la que más tenía, la repetición del nombre en comparación con el resto es mayor.

Por otro lado, cabe recalcar que el número de atributos generado en el primer caso es de 330 sobre un número instancias de 943, lo que quiere decir que muchas palabras se repetían en las distintas opiniones. En el segundo caso, el número de atributos es mayor (347 sobre 392 instancias), lo que deja en evidencia que al ser una base variada hay una mayor riqueza de vocabulario, consiguiendo con menos opiniones un número mayor tokens.

Vemos en este segundo caso (Figura 40) que el atributo 7 es una “p”. El motivo de que haya sido contado como una palabra es que deriva del símbolo de punto y final “<\p>” y fue la motivación para realizar un preproceso de los textos de modo que se eliminara este símbolo, producido al pasar de unos formatos de texto a otros.

Recogemos ahora los resultados de la base española balanceada de positivos y negativos y sin caracteres extraños en la Figura 41:

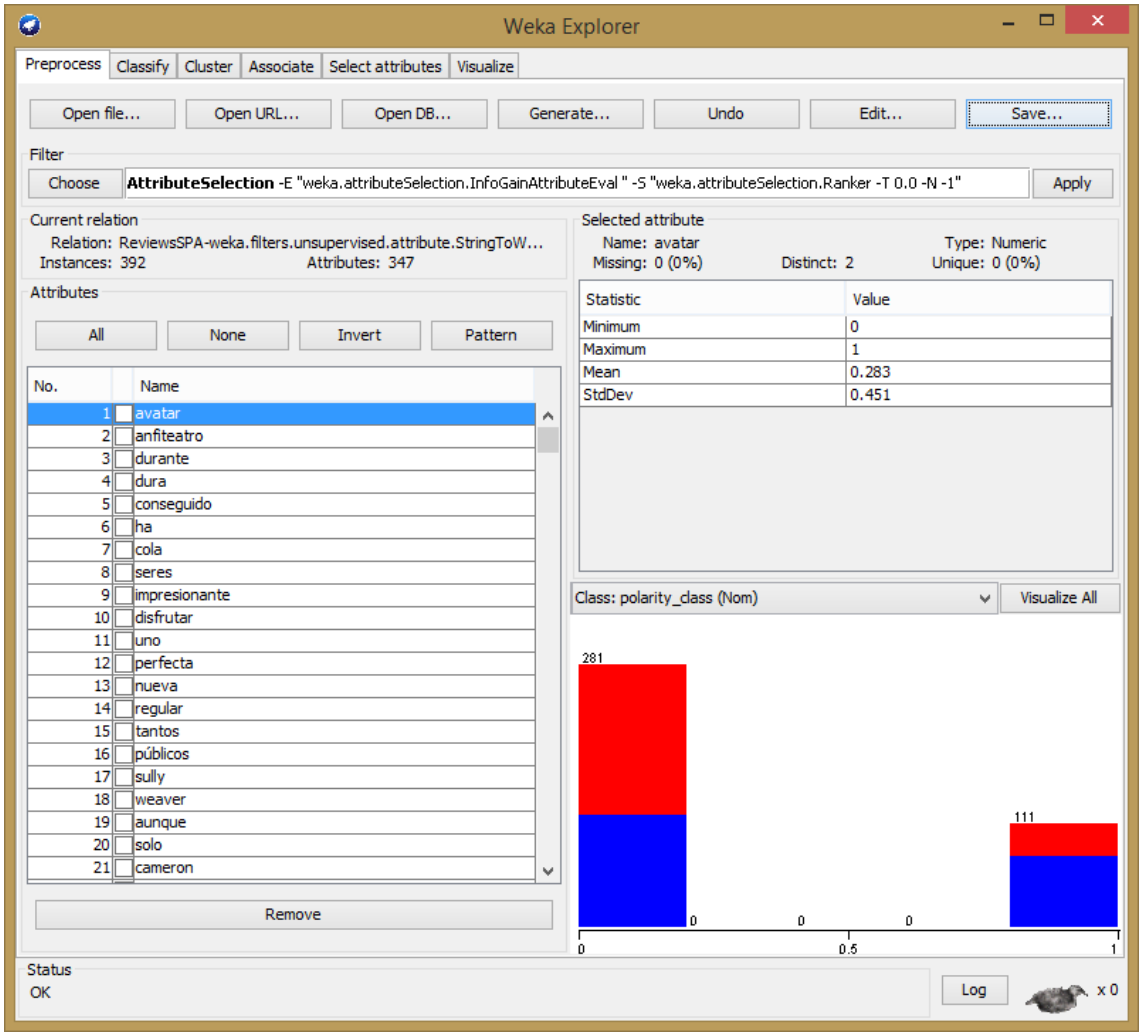


Figura 41: Resultado de aplicar los filtros *StringToWordVector* con *wordtokenizer* y *AtributteSeleccion* sobre la base española de opiniones de varios espectáculos limpia de caracteres extraños.

Se ve que finalmente el token "p" que no tenía sentido ha desaparecido dejando hueco a otra palabra.

Comparamos las tres listas en la Tabla 4:

Avatar	**Variada**	**Variada limpia**
Espectáculo	Avatar	Avatar
Risa	Anfiteatro	anfiteatro
Enseñar	Durante	Durante
Espalda	Dura	Dura
Na	Conseguido	conseguido
Patas	Ha	Ha
Vi	P	Cola
Leche	Cola	Seres

Hostia	Seres	**impresionante**
Sabemos	**Impresionante**	Disfrutar
Crisis	Disfrutar	Uno
Nueva	Uno	**Perfecta**
Currado	**Perfecta**	Nueva
Pone	Nueva	Públicos
Impresionante	Públicos	Sully
Perfecta	Sully	Tantos
Medio	Tantos	Regular
Cuatro	Regular	Weaver
Ja	Weaver	Aunque
Avatar	Aunque	Solo

Tabla 4: 20 primeras palabras tras haber filtrado con *StringToWordVector* con *tokenizer wordtokenizer* y *AttributeSelection* sobre las tres bases de datos en español.

De la comparación de las listas vemos que de las palabras de la base de Avatar que salían entre las 20 primeras posiciones, en la base variada sólo salen tres. Como se ha comentado anteriormente el token Avatar pasa a ser el primero. Y los atributos "perfecta" e "impresionante" ganan más peso puesto que puede ser que entre las otras opiniones también se mencione.

4.1.2.2. StringToWordVector: NGrams(1,1)

Realizamos los mismos pasos con el *tokenizer* de la forma *NGrams(1,1)*.

Primero sobre la base de avatar en la Figura 42:

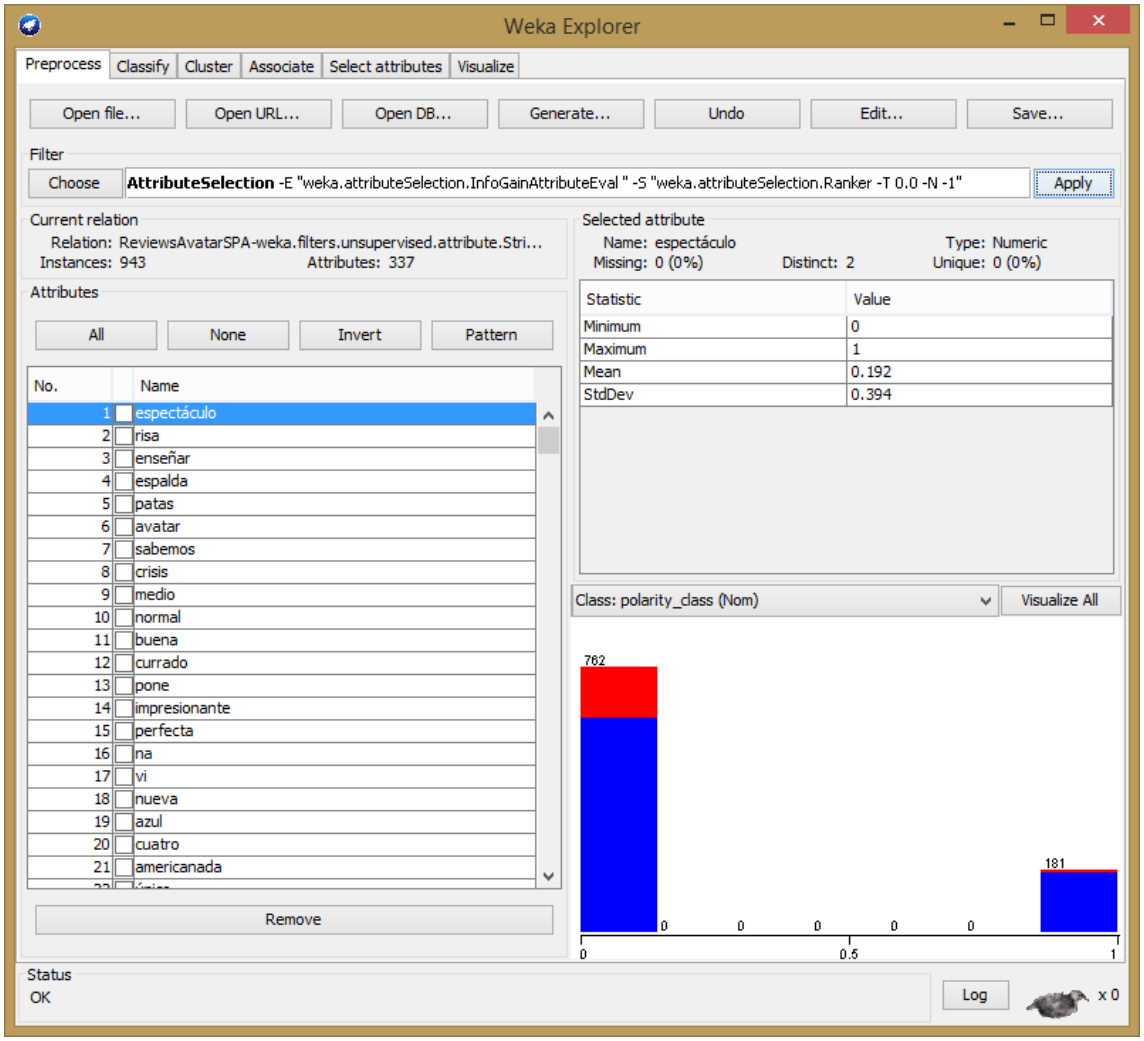


Figura 42: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(1,1)* y *AtributteSeleccion* sobre la base española de opiniones de avatar.

Al aplicar Unigramas vemos que el número de atributos aumenta de 330 a 337.

Pasamos a ver cómo afecta a la base balanceada antes de limpiar en la Figura 43.

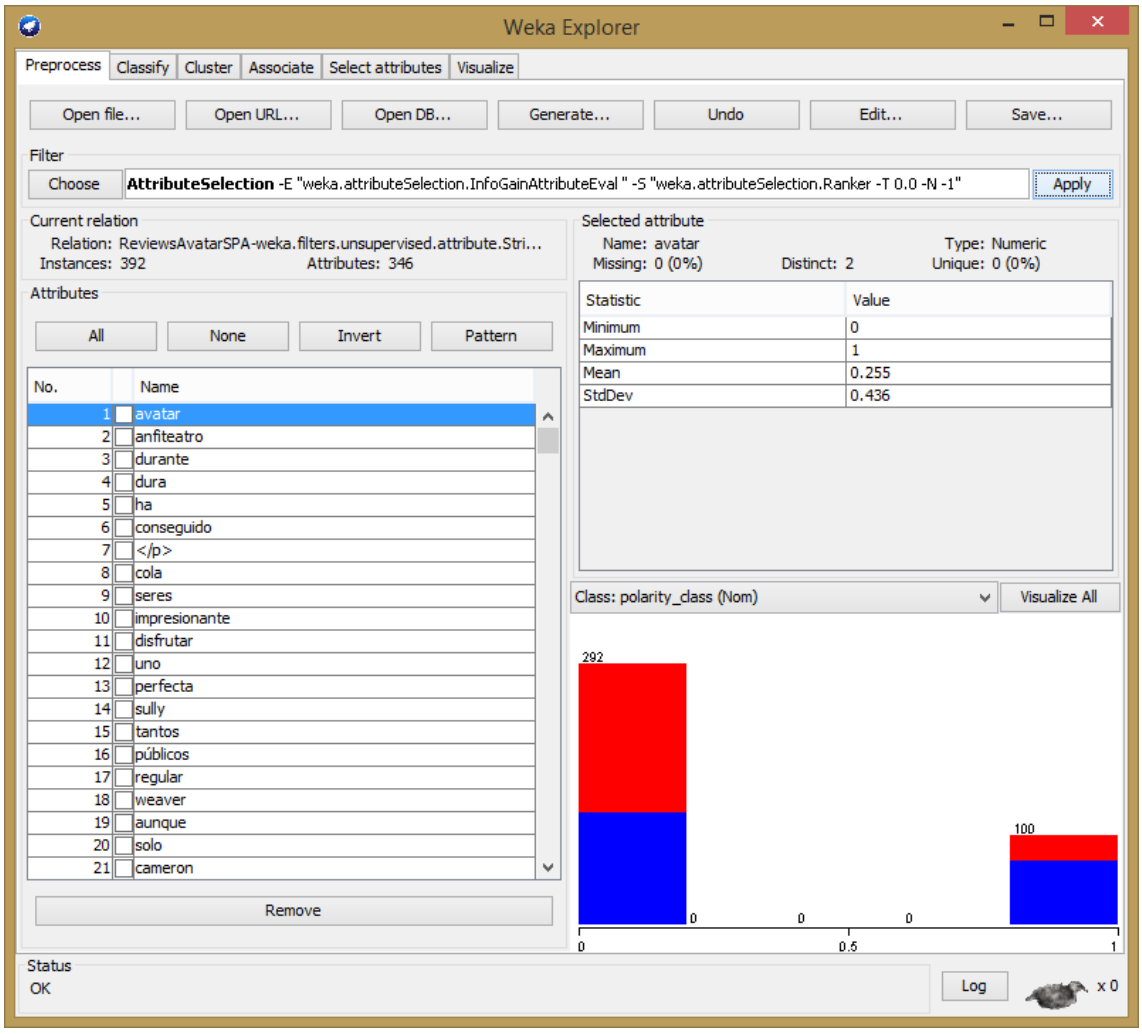


Figura 43: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(1,1)* y *AtributteSeleccion* sobre la base española de opiniones mixtas.

Vemos en este caso claramente que el token 7 no es una “p”, sino “</p>” (Figura 43). En este caso el número de atributos ha disminuido en uno.

Finalmente vemos lo que ha pasado con la base balanceada y limpia en la Figura 44:

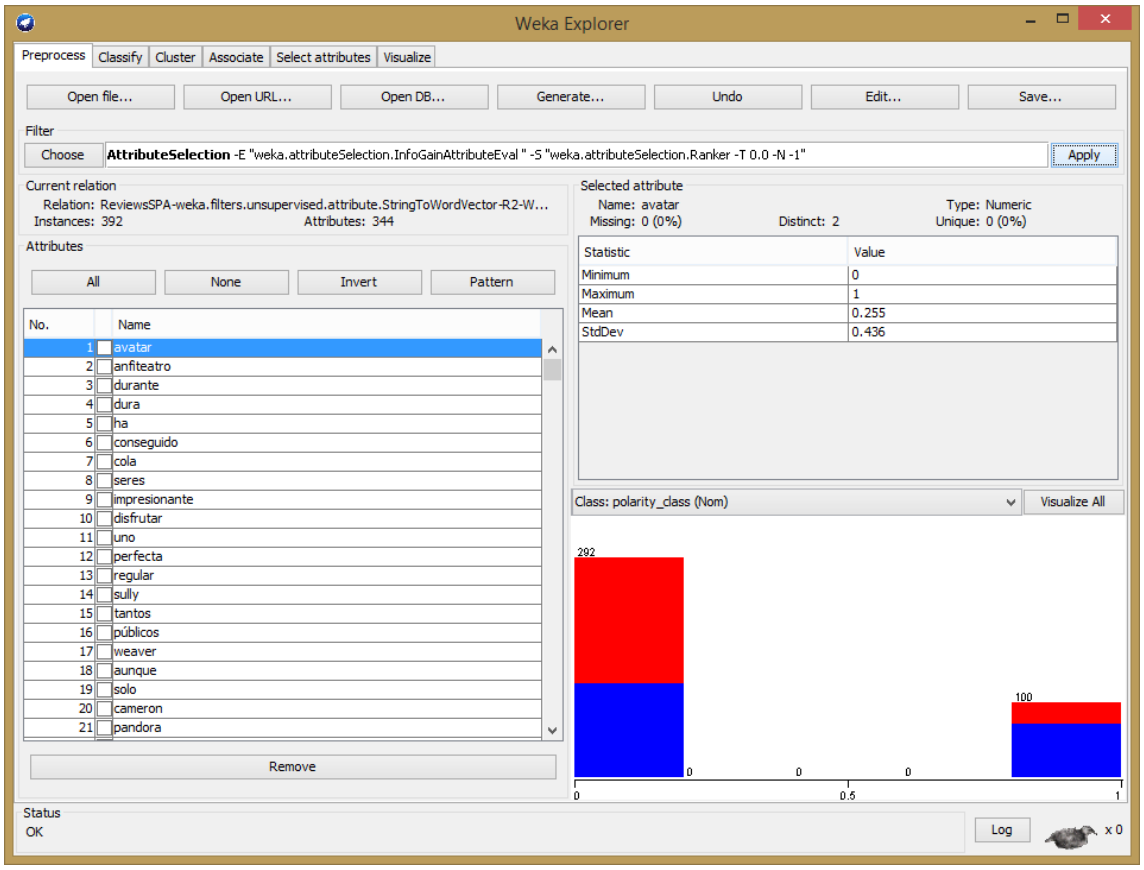


Figura 44: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(1,1)* y *AtributteSeleccion* sobre la base española de opiniones mixtas y sin caracteres extraños.

Vemos que en este último caso también disminuye el número de atributos de 347 a 344. Esta disminución es poco apreciable, pero se da en el caso de palabras compuestas, que en el caso de los unigramas se cuentan como un único token en vez de dos (como, por ejemplo, en el caso de na´vi).

Y la comparación de las listas en la Tabla 5:

Avatar	**Variada**	**Variada limpia**
Espectáculo	**Avatar**	**avatar**
Risa	Anfiteatro	anfiteatro
Espalda	Durante	durante
Enseñar	Dura	Dura
Patas	Ha	Ha
Avatar	Conseguido	conseguido
Sabemos	</p>	Cola
Crisis	Cola	seres
Medio	Seres	**impresionante**
Normal	**Impresionante**	disfrutar
Buena	Disfrutar	Uno
Currado	Uno	**perfecta**
Pone	**Perfecta**	Tantos

Impresionante	Tantos	Públicos
Perfecta	Públicos	Sully
Na	Sully	Regular
Vi	Regular	Weaver
Nueva	Weaver	Aunque
Azul	Aunque	Solo
Cuatro	Solo	Cameron

Tabla 5: 20 primeras palabras tras haber filtrado con *StringToWordVector* con *tokenizer NGram(1,1)* y *AttributeSelection* sobre las tres bases en español.

4.1.2.3. StringToWordVector: NGrams(3,1)

En este apartado vemos lo que ocurre al aplicar el *tokenizer* con valor máximo 3 y mínimo 1. Dónde se crea una combinación de unigramas, bigramas y trigramas.

Comenzamos como en los casos anteriores por la base con las opiniones de Avatar en la Figura 45.

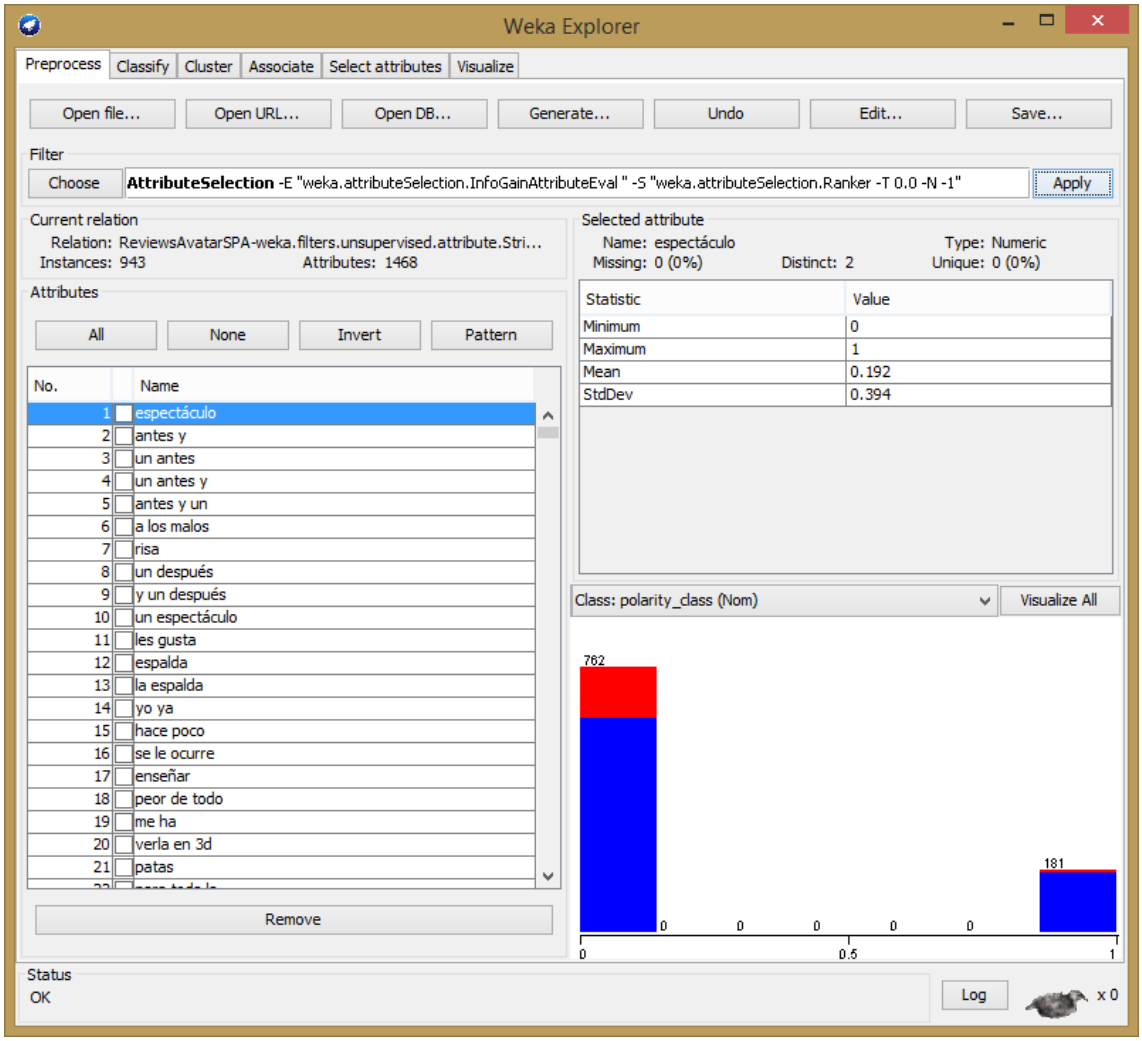


Figura 45: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(3,1)* y *AtributteSeleccion* sobre la base española de opiniones de avatar.

En este caso los tokens van en combinaciones desde una palabra hasta tres, por eso el número de atributos ha pasado de 330 o 337 a 1.468. Al ver esta diferencia podemos comprender por qué en el caso de la base inglesa no llega a fin con sus cálculos al tener ésta un número 6 veces más elevado de tokens.

Comprobamos ahora si en el caso de la base española de varias opiniones sufre un aumento en su número de atributos también.

Vemos primero antes de aplicar el *AttributeSelection* para dejar claro que el aumento de atributos es enorme (Figura 46):

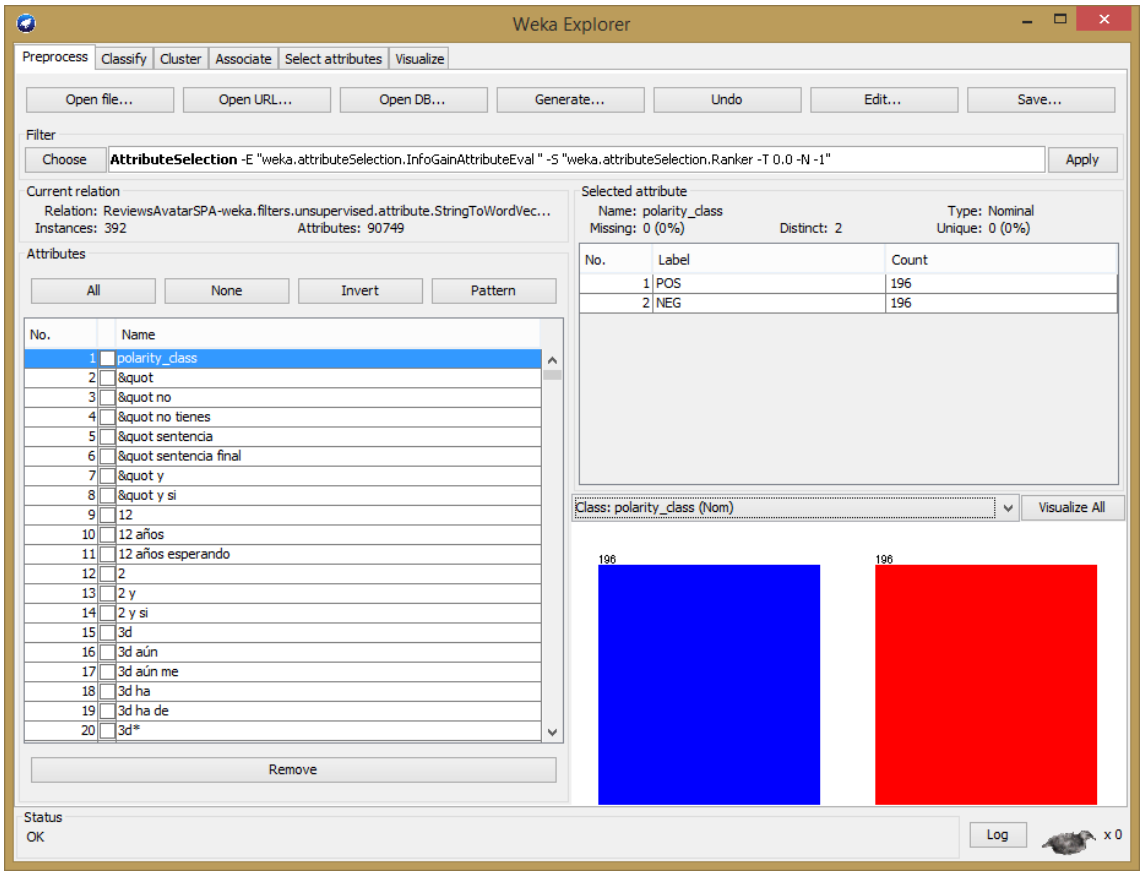


Figura 46: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(3,1)* sobre la base española de opiniones mixtas.

El número de atributos generados de 392 instancias es de 90.749, aunque muchos de ellos son eliminados tras aplicar el filtro *AttributeSelection*, como veremos a continuación.

Quería dejar también esta imagen para demostrar que aparecen más tokens que en la base limpia debido a que muchos contienen "&quot" entre otros que es la traducción de las comillas al pasar de un formato a otro.

Ahora se muestran los resultados en la Figura 47 tras aplicar el filtro *AttributeSelection* en el que los tokens se ordenan por ganancia de información y se les da valor.

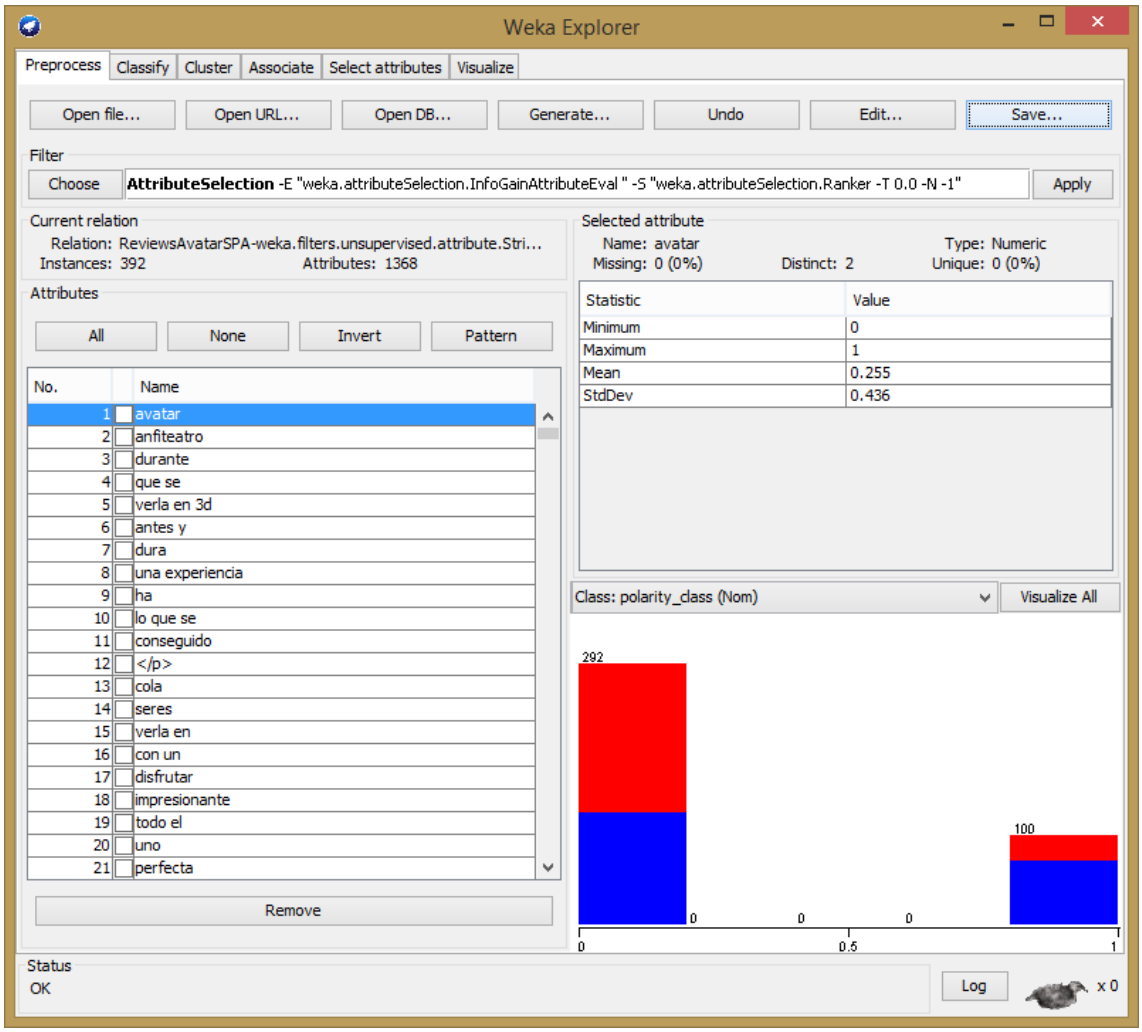


Figura 47: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(3,1)* y *AtributteSeleccion* sobre la base española de opiniones mixtas.

En este caso también se incrementan bastante, de 346 o 347 atributos a 1.368, mucho menor que los 90.749 antes del filtro supervisado. Aunque no tanto como en el caso de la base de avatar, en este caso el mayor número de instancias, aunque sobre el mismo tema, incorpora más combinaciones.

Seguimos viendo, aunque en este caso en la posición 12, el token de fin de párrafo.

Limpiando la base se han obtenido los resultados de la Figura 48:

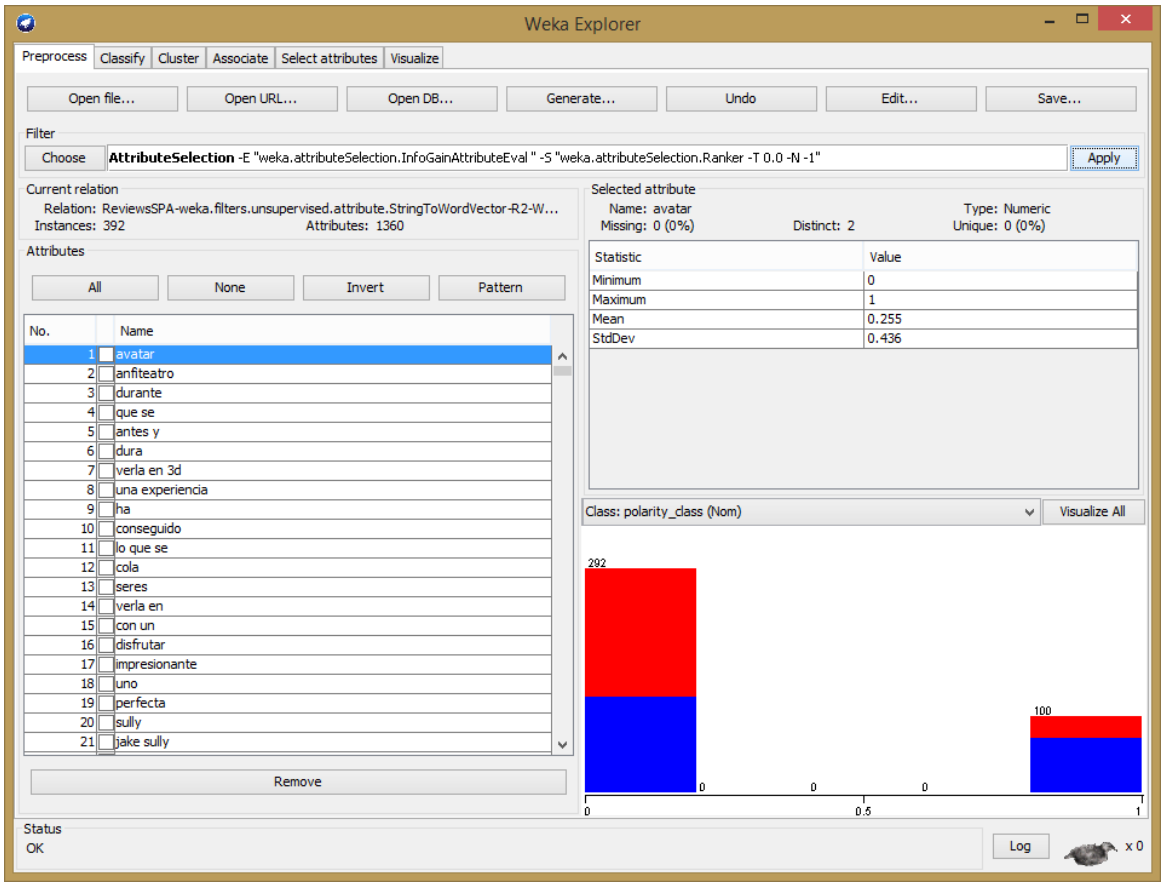


Figura 48: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(3,1)* y *AtributteSeleccion* sobre la base española de opiniones mixtas sin caracteres extraños.

En este caso el número de atributos es 1360, menos que antes de limpiarla puesto que se quitaron posibles tokens que aparecerían como palabras y realmente no lo eran.

Comparamos las listas de estas tres bases en la Tabla 6:

Avatar	**variada**	**variada limpia**
Espectáculo	avatar	Avatar
antes y	anfiteatro	anfiteatro
un antes	durante	Durante
un antes y	que se	que se
antes y un	**verla en 3d**	**verla en 3d**
a los malos	una experiencia	una experiencia
Risa	**antes y**	**antes y**
un después	dura	Dura
y un después	ha	Ha
un espectáculo	lo que se	lo que se
Enseñar	conseguido	Conseguido
yo ya	</p>	Cola
peor de todo	cola	Seres
hace poco	seres	verla en

se le ocurre	verla en	con un
la espalda	con un	Impresionante
les gusta	impresionante	Disfrutar
Espalda	disfrutar	Uno
me ha	todo el	Perfecta
verla en 3d	uno	ha conseguido

Tabla 6: 20 primeras palabras tras haber filtrado con *StringToWordVector* con *tokenizer NGram(3,1)* y *AttributeSelection* sobre las tres bases en español.

En este caso se ve que el token "verla en 3d" viene sólo de las referencias de Avatar dado que el resto son obras de teatro.

4.1.2.4. StringToWordVector: NGrams(3,3)

Probamos con el último caso de *NGrams* que vamos a experimentar en este proyecto, en el que consideran un máximo de 3 palabras y un mínimo de 3, trigramas.

Presento los resultados de la lista generada sobre la base de opiniones de Avatar en la Figura 49:

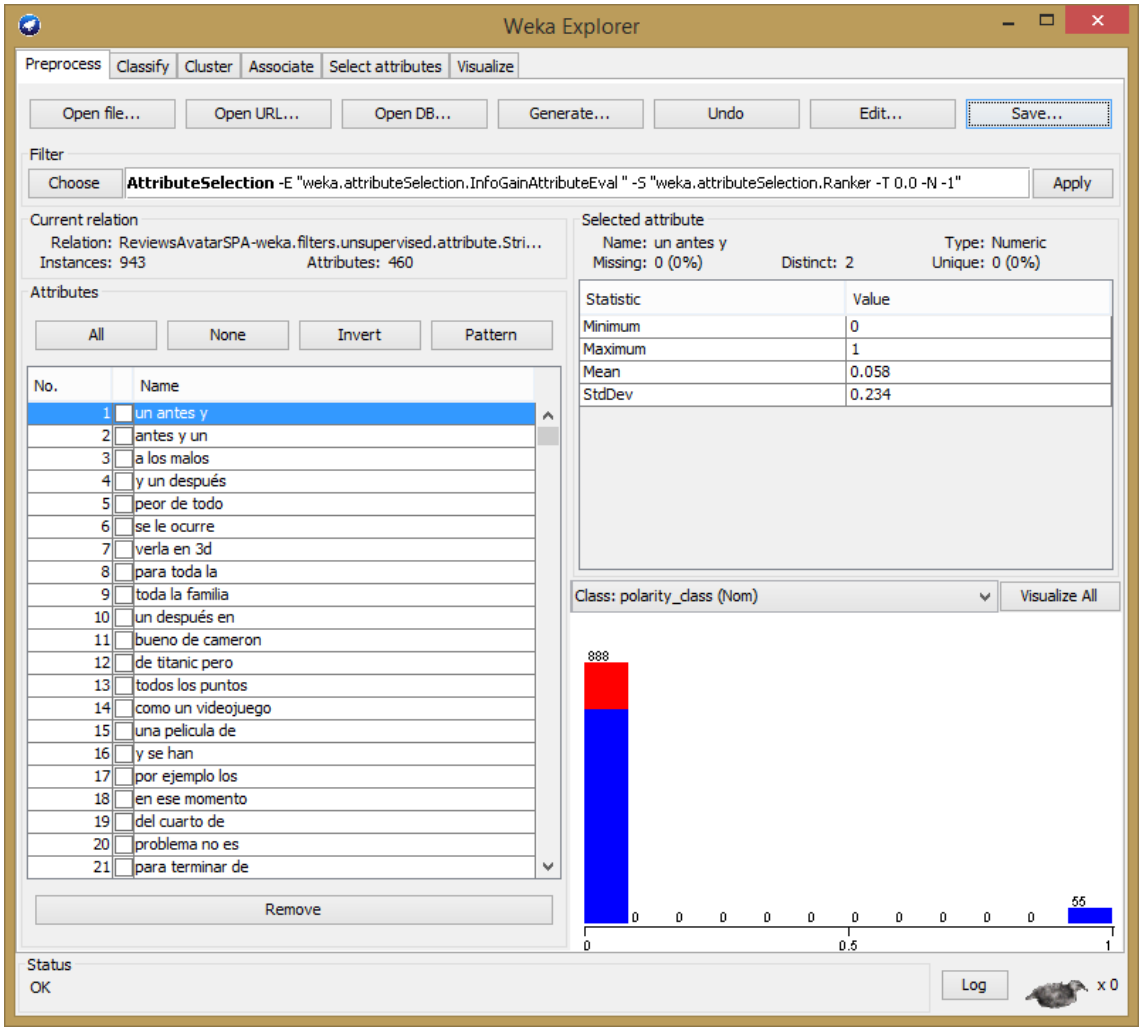


Figura 49: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(3,3)* y *AtributteSeleccion* sobre la base española de opiniones de avatar.

El número de atributos generado ahora es de 460, mayor que en los primeros experimentos pero mucho menor que el caso de *NGrams(3,1)* ya que en este caso sólo se tienen en cuenta las combinaciones de 3 en 3 palabras, mientras que en el caso anterior también de una y de dos. Comentar que en este caso, antes de aplicar el filtro supervisado, había 2516 atributos.

Comprobamos ahora la base con varias opiniones en la Figura 50:

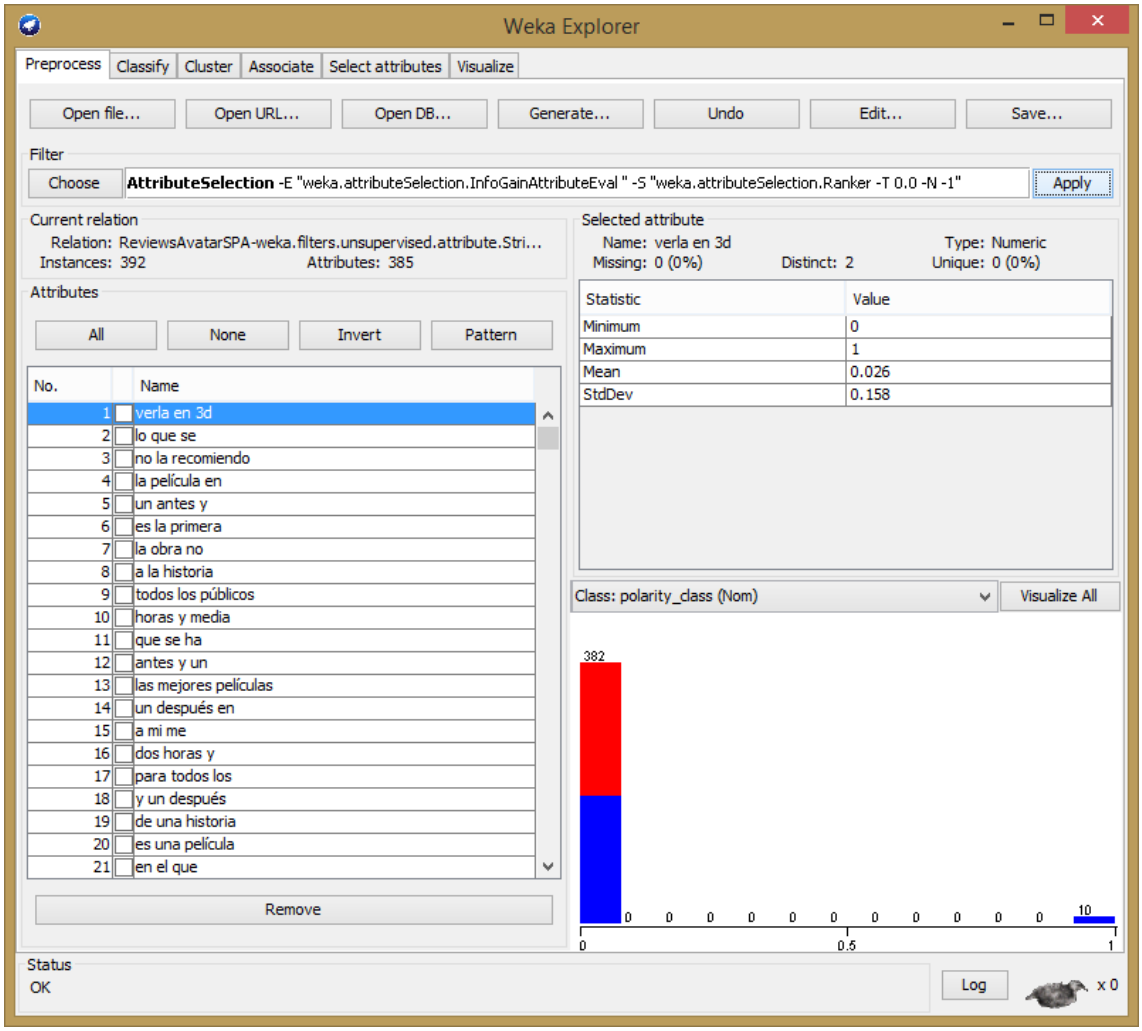


Figura 50: Resultado de aplicar los filtros *StringToWordVector* con *NGrams(3,3)* y *AtributteSeleccion* sobre la base española de varias opiniones.

En este caso el número de atributos es 385, también mayor que en los casos de *wordtokenizer* y *NGrams(1,1)* aunque mucho menor que el caso de *NGrams(3,1)*, si comparamos con la base de "Avatar" vemos que el número de tokens de esta es mayor porque tiene más instancias. también este caso tiene menos tokens que en la base de Avatar, ocurre como antes, el número de instancias ahora se hace significativa.

En este caso no se ven caracteres raros y he comprobado que la solución con la base limpia, en estos primeros 20 tokens da el mismo resultado. El número de atributos que se generan es 380, que sigue las mismas implicaciones descritas previamente.

Por este motivo sólo voy a comparar la base de Avatar con la de varios eventos en la Tabla 7:

Avatar	Variada
un antes y	**verla en 3d**
antes y un	lo que se
a los malos	no la recomiendo
y un después	la película en
peor de todo	**un antes y**
se le ocurre	es la primera
verla en 3d	a la historia
para toda la	horas y media
toda la familia	todos los públicos
un después en	la obra no
bueno de cameron	que se ha
de titanic pero	para todos los
todos los puntos	dos horas y
como un videojuego	a mi me
una pelicula de	las mejores películas
y se han	de una historia
por ejemplo los	**un después en**
en ese momento	**y un después**
del cuarto de	**antes y un**
problema no es	es una película

Tabla 7: 20 primeras palabras tras haber filtrado con *StringToWordVector* con *tokenizer NGram(3,1)* y *AttributeSelection* sobre las tres bases en español

Vemos en este caso que es curioso que la frase "un antes y un después en" se ve reflejada en distintas partes en ambas bases, por lo que debe de predominar en la base de Avatar.

4.1.2.5. Lista de los 20 primeros tokens por base

Ya hemos visto en los apartados previos la comparación entre las distintas bases en español y los motivos por los que se tuvo que generar una nueva base de datos balanceada. Ahora quiero comparar los efectos de los distintos tratamientos sobre la misma base en la Tabla 8. Como se ha explicado y demostrado, los resultados de la base limpia son mejores que antes de eliminar los caracteres no procedentes, por ese motivo sólo se van a comparar la base de avatar y la base limpia, como se seguirá en los resultados de los clasificadores.

WordTokenizer	Ngrams(1,1)	Ngrams(3,1)	Ngrams(3,3)
Espectáculo	**espectáculo**	**espectáculo**	**un antes y**
Risa	**risa**	antes y	**antes y un**
Espalda	**espalda**	un antes	**a los malos**
Enseñar	enseñar	**un antes y**	y un después
Na	patas	**antes y un**	peor de todo
Patas	avatar	**a los malos**	se le ocurre
Vi	sabemos	**risa**	verla en 3d
Leche	crisis	un después	para toda la
Hostia	medio	y un después	toda la familia

Sabemos	normal	un espectáculo	un después en
Crisis	buena	les gusta	bueno de cameron
Nueva	currado	**espalda**	de titanic pero
Currado	pone	la espalda	todos los puntos
Pone	impresionante	yo ya	como un videojuego
Impresionante	perfecta	hace poco	una pelicula de
Perfecta	Na	se le ocurre	y se han
Medio	Vi	enseñar	por ejemplo los
Cuatro	nueva	peor de todo	en ese momento
Ja	azul	me ha	del cuarto de
Avatar	cuatro	verla en 3d	problema no es

Tabla 8: 20 primeros tokens obtenidos de los distintos procesamientos de la base de Avatar

Comparando las cuatro columnas, vemos que las dos primeras en las que los tokens los forman sólo una palabra contienen prácticamente las mismas palabras, y que difieren poco en el orden.

En el caso de los *NGrams(3,1)* comprobamos que los primeros tokens que tienen una palabra coinciden con los de las dos primeras columnas (espectáculo, risa y espalda). Del mismo modo, los tres primeros tokens de tres palabras coinciden con los tres primeros de los correspondientes a *NGrams(3,3)*.

Vemos ahora el caso de la base de opiniones de varios eventos y sin tokens no procedentes en la Tabla 9:

WordTokenizer	Ngrams(1,1)	Ngrams(3,1)	Ngrams(3,3)
Avatar	**avatar**	**avatar**	**verla en 3d**
Anfiteatro	**anfiteatro**	**anfiteatro**	**lo que se**
Durante	**durante**	**durante**	**no la recomiendo**
Dura	dura	que se	la película en
Conseguido	ha	antes y	un antes y
Ha	conseguido	dura	es la primera
Cola	cola	**verla en 3d**	a la historia
Seres	seres	una experiencia	horas y media
Impresionante	impresionante	ha	todos los públicos
Disfrutar	disfrutar	conseguido	la obra no
Uno	uno	**lo que se**	que se ha
Perfecta	perfecta	cola	para todos los
Nueva	regular	seres	dos horas y
Públicos	sully	verla en	a mi me
Sully	tantos	con un	las mejores películas
Tantos	públicos	disfrutar	de una historia
Regular	weaver	impresionante	un después en
Weaver	aunque	uno	y un después
Aunque	solo	perfecta	antes y un
Solo	cameron	**no la recomiendo**	es una película

Tabla 9: 20 primeros tokens obtenidos de los distintos procesamientos de la base de varios eventos.

Como en la base de Avatar los tres primeros tokens de *wordtokenizer* y *NGrams(1,1)* aparecen en la lista de *NGrams(3,1)*, *así* como los tres primeros de *NGrams(3,3).* Y se comprueba que las dos primeras listas son muy parecidas.

4.2. Resultados con los distintos clasificadores

En esta sección sólo vamos a tener en cuenta la base inglesa y la española de varias opiniones que fue limpiada para obtener mejores resultados.

Primero vamos a comparar dentro de las mismas bases qué clasificador obtiene mejores resultados, y después compararemos entre las dos lenguas.

Por cada una vamos a realizar dos comparaciones, por clasificador qué preprocesamiento hace mejor y la comparación por procesamiento sobre clasificador.

4.2.1. Base de datos Inglesa

4.2.1.1. Resultados según clasificador y procesamiento.

En la siguiente gráfica muestro los resultados obtenidos según ambos el clasificador y el preprocesamiento para el conjunto de entrenamiento de la base de datos en inglés.

Para entenderlo hay que ver que los clasificadores están puestos en horizontal y las siglas significan:

- NB: Naïve Bayes
- NBM: Naïve Bayes Multinomial
- NBMU: Naïve Bayes Multinomial Updateable
- SMO: Sequential Minimal Optimization

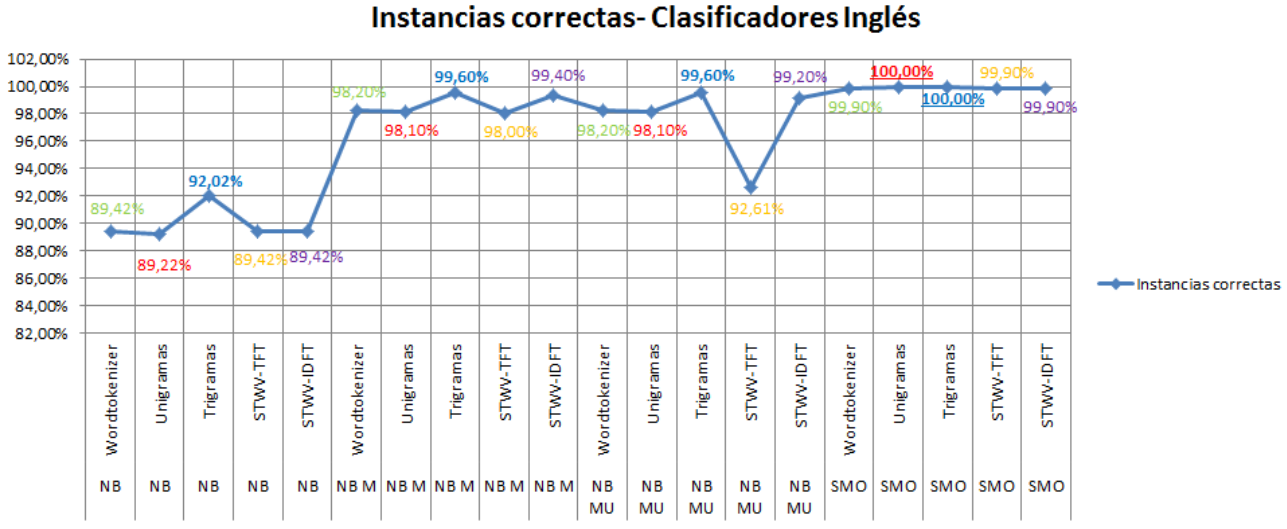


Figura 51: Resultados de la aplicación de los clasificadores sobre el conjunto de entrenamiento en inglés.

A simple vista en la Figura 51 vemos que el Naïve Bayes es el que peor resultados obtiene en todos los casos. Por el contrario, el SMO es el que mejor actuación tiene, y su tendencia es casi constante, sin grandes diferencias en los preprocesadores. El Naïve Bayes Multinomial también tiene muy buenos resultados siendo casi iguales que el Naïve Bayes Multinomial Updateable.

Si nos fijamos en los preprocesamientos los mejores resultados son cuando se utiliza el *tokenizer* con *NGrams(3,3)*. Para los cuatro clasificadores con este método se obtienen porcentajes más altos de acierto.

En la evaluación sobre la base de entrenamiento, los mejores resultados se obtienen con el clasificador SMO, donde clasifica perfectamente cuando se utilizan n-gramas.

Se sacaron también los resultados tomando una parte de la base como datos de entrenamiento y los datos restantes para realizar el test. En este caso se utilizó un 90% para entrenar y un 10% para el test (Figura 52).

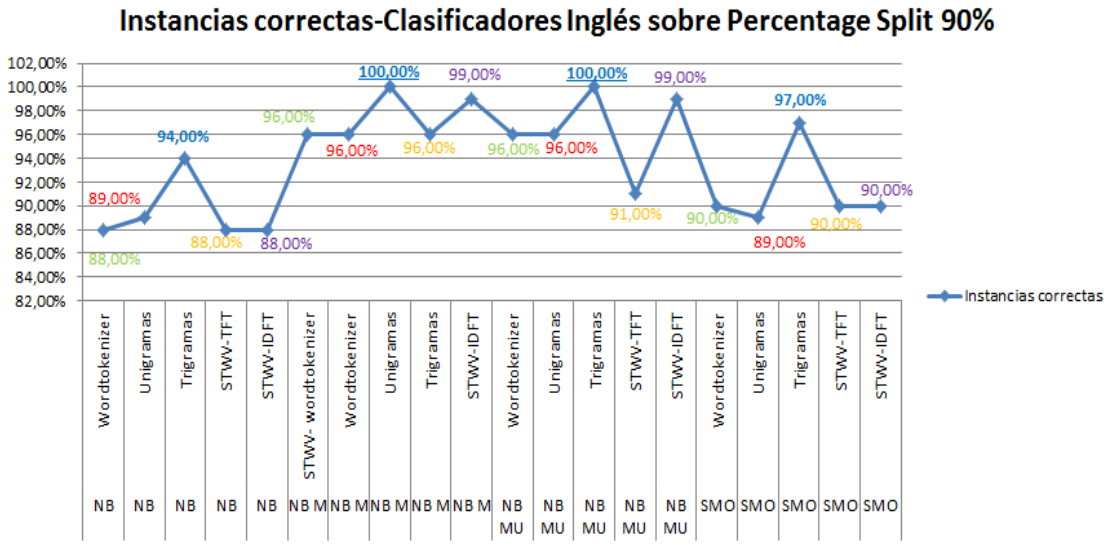


Figura 52: Resultados de la aplicación de los clasificadores evaluados en un porcentaje del 90% en inglés.

Estos resultados se acercan más a la realidad, y lo que sorprende es que el SMO empeora mucho. Eso refleja que este algoritmo funciona mejor sobre los datos que conoce (es decir, que no generaliza bien), mientras que el Naïve Multinomial en ambos modelos mejora hasta la obtener un 100% de tasa de acierto en el caso de *NGrams(3,3)*.

Finalmente se evaluaron los resultados utilizando Validación cruzada con 10 partes; se muestran en la Figura 53.

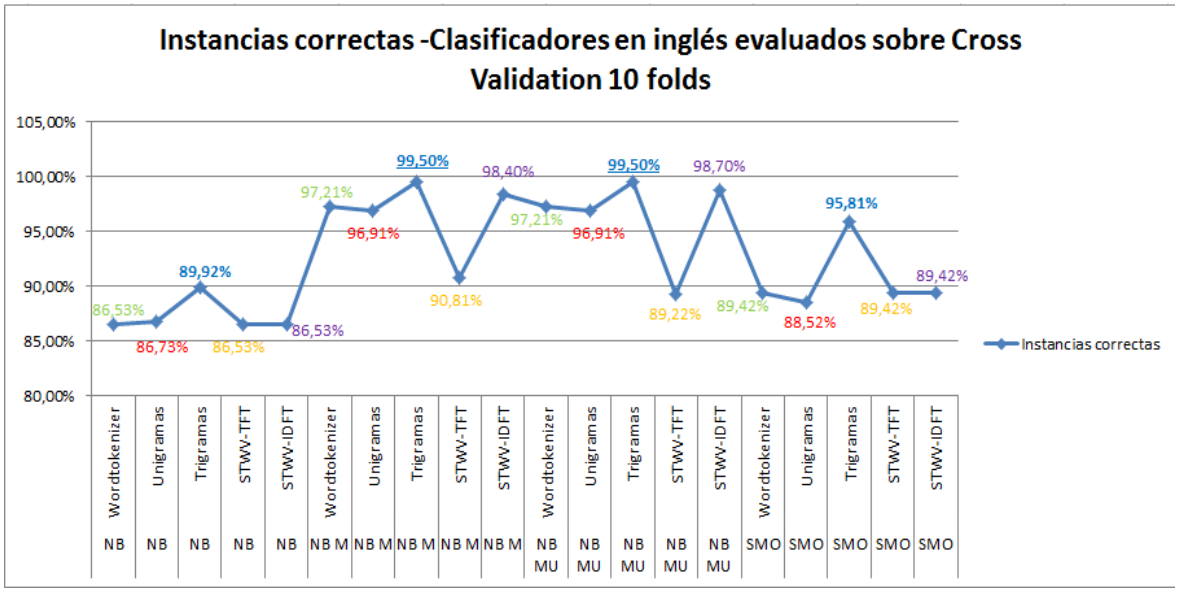


Figura 53: Resultados de la aplicación de los clasificadores evaluados con validación cruzada de 10 partes en inglés.

Los resultados son ligeramente peores pero más parecidos a los obtenidos con el porcentaje del 10% de test.

Comparamos las tres opciones de testado en la Figura 54:

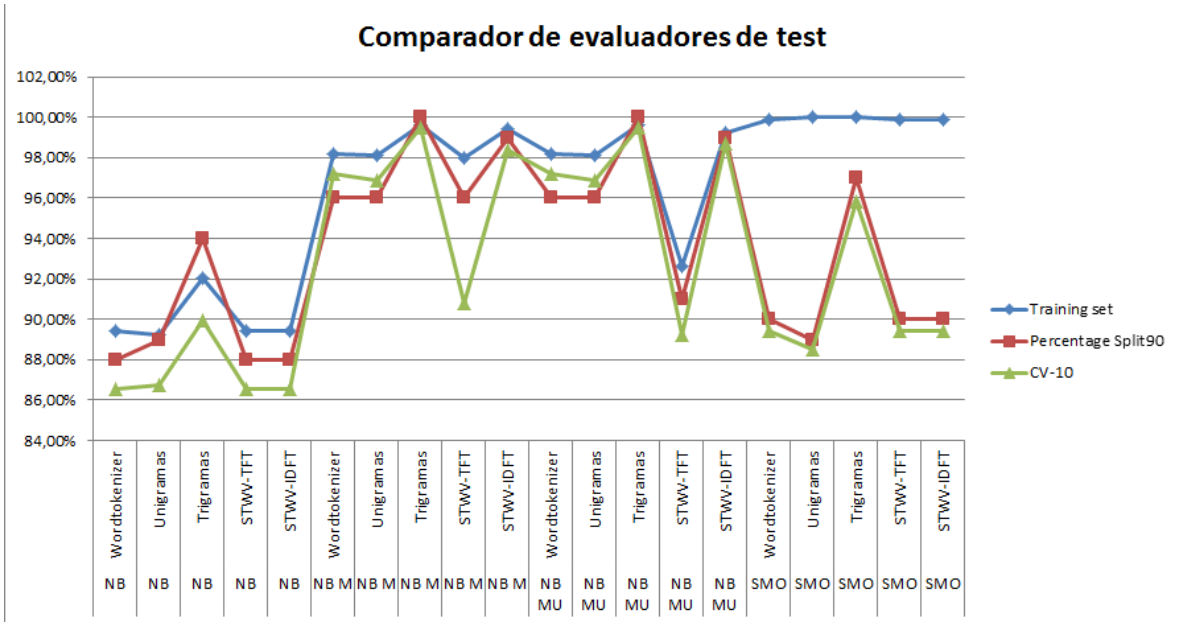


Figura 54: Comparación de resultados según las opciones de testado en la base en inglés.

Vemos que siguen la misma forma para los tres tipos de Naïve Bayes y que para el SMO en el caso del conjunto de entrenamiento no se parece mucho. Vemos que en general para el conjunto de entrenamiento, las diferencias entre unos métodos y otros es menor que en los otras dos formas de evaluación.

En la siguiente Tabla 10 se pueden ver los valores:

Clasificador	Tokenizer	Training set	Percentage Split 90	CV-10
NB	Wordtokenizer	89,42%	88,00%	86,53%
	Unigramas	89,22%	89,00%	86,73%
	Trigramas	**92,02%**	***94,00%***	**89,92%**
	STWV-TFT	89,42%	88,00%	86,53%
	STWV-IDFT	89,42%	88,00%	86,53%
NB M	Wordtokenizer	98,20%	96,00%	97,21%
	Unigramas	98,10%	96,00%	96,91%
	Trigramas	**99,60%**	***100,00%***	**99,50%**
	STWV-TFT	98,00%	96,00%	90,81%
	STWV-IDFT	99,40%	99,00%	98,40%
NB MU	Wordtokenizer	98,20%	96,00%	97,21%
	Unigramas	98,10%	96,00%	96,91%
	Trigramas	**99,60%**	***100,00%***	**99,50%**
	STWV-TFT	92,61%	91,00%	89,22%
	STWV-IDFT	99,20%	99,00%	98,70%
SMO	Wordtokenizer	99,90%	90,00%	89,42%
	Unigramas	***100,00%***	89,00%	88,52%
	Trigramas	***100,00%***	**97,00%**	**95,81%**
	STWV-TFT	99,90%	90,00%	89,42%
	STWV-IDFT	99,90%	90,00%	89,42%

Tabla 10: Comparación de clasificadores, preprocesamientos y evaluadores en la base en inglés.

De esta tabla podemos sacar conclusiones a simple vista según los clasificadores.

4.2.1.2. Naïve Bayes

Si nos fijamos en la Tabla 10 en la parte Naïve Bayes las conclusiones que sacamos son que en todos los casos los mejores resultados son para *NGrams(3,3)* evaluado por el *Percentage Split 90%*, sin embargo para *wordtokenizer* y *NGrams(1,1)* los resultados que mejor se obtienen son dentro del conjunto de entrenamiento. Siendo la validación cruzada la peor.

Comprobamos que en este caso los parámetros TFT e IDFT aplicados sobre *wordtokenizer* no producen ningún efecto en ningún caso, dando igual seleccionarlo que no. Podemos verlo más claro en la Figura 55.

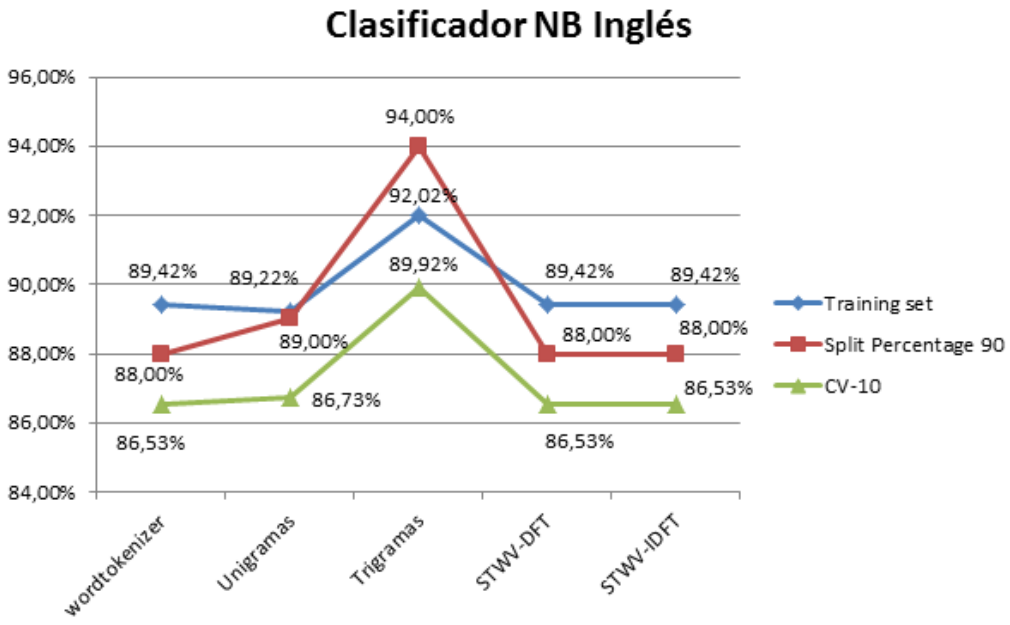


Figura 55: Resultados del clasificador NB en inglés según el preprocesamiento y en las tres maneras de testado.

Comparado con el resto de clasificadores, el Naïve Bayes es el que peor resultados tiene en todos los casos.

Según la generación del modelo podemos ver los valores que ha aplicado a las palabras para determinar si es positivo o negativo. Este modelo saca como parámetros la media y la desviación típica de la palabra según la clase, que recuerdo en este proyecto son Negativa y Positiva.

En la Tabla 11 se muestran los resultados del modelo de las primeras palabras:

=== Run information === Scheme:weka.classifiers.bayes.NaiveBayes Relation: ReviewsIngles-weka.filters.unsupervised.attribute.StringToWordVector-R2-W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r \t\r\n\t.,;:\"\'()?!-¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N -1 Instances: 1002 Attributes: 1846 [list of attributes omitted] Test mode:evaluate on training data === Classifier model (full training set) === Naive Bayes Classifier

Class Attribute ==============	POS (0.5)	NEG (0,5)	

Bad mean std. dev. weight sum precision	 0.2914 0.4544 501 1	 **0.505** 0.5 501 1	*Negativa*
Worst mean std. dev. weight sum precision	 0.0539 0.2258 501 1	 **0.1896** 0.392 501 1	*Negativa*
Wasted mean std. dev. weight sum precision	 0.012 0.1667 501 1	 **0.0938** 0.2916 501 1	*Negativa*
Awful mean std. dev. weight sum precision	 0.014 0.1667 501 1	 **0.0958** 0.2943 501 1	*Negativa*
Wonderfully mean std. dev. weight sum precision	 **0.0739** 0.2615 501 1	 0.006 0.1667 501 1	*Positiva*

Tabla 11: Modelo NB sobre preprocesamiento con *wordtokenizer* en la base inglesa.

4.2.1.3. Naïve Bayes Multinomial

Pasamos ahora a analizar los datos obtenidos en este caso.

Según la Tabla 10, comprobamos que este clasificador mejora al NB en todos los casos, y que para *N-Grams(3,3)* tiene un acierto del 100% en el caso de que se evalúe sobre un pequeño porcentaje de la base.

En este clasificador se ven claramente evidencias del uso de los parámetros TFT e IDFT sobre el *wordtokenizer* en el primer caso los resultados empeoran, siendo claramente la diferencia máxima en el evaluador de validación cruzada. Por el contrario, el IDFT mejora notablemente los resultados, sobretodo en el caso de *Percentage Split 90%.*

Vemos claramente los resultados en la Figura 51.

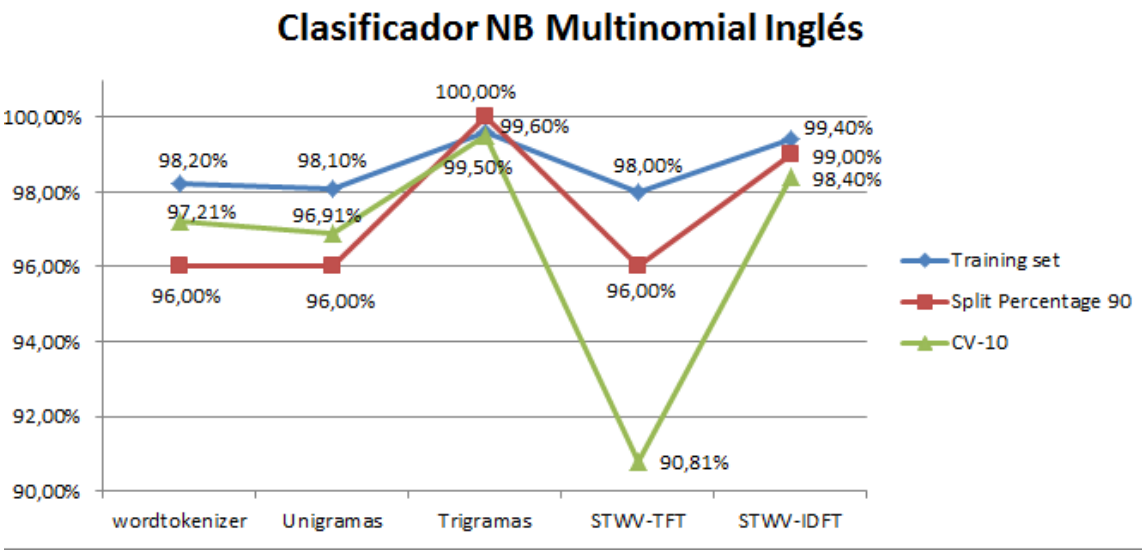


Figura 56: Resultados del clasificador NB Multinomial en inglés según el preprocesamiento y en las tres maneras de testado.

En general este clasificador es de los mejores sobre todo a la hora de utilizar datos de test en situaciones más reales que no sobre el conjunto de entrenamiento.

Los resultados del modelo nos dan los pesos de las palabras desde los que podemos deducir a qué clase le ha asignado.

En este caso, no se dan los mismos valores que en Naïve Bayes, en vez de la media y la desviación típica del token, se da directamente la probabilidad de la palabra dada la clase.

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayesMultinomial
Relation:     ReviewsIngles-weka.filters.unsupervised.attribute.StringToWordVector-R2-
W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-
tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r \t\r\n\t.,;:\"\'()?!-
¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-
Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N
-1
Instances:    1002
Attributes:   1846
[list of attributes omitted]
Test mode:evaluate on training data

=== Classifier model (full training set) ===

The independent probability of a class
--------------------------------------
POS     0.5
NEG     0.5

The probability of a word given the class
-----------------------------------------
```

class	POS	NEG	

bad	0.006209868198715784	**0.013221591796366661**	*Negativa*
Worst	0.0011828320378506252	**0.004997137056894485**	*Negativa*
wasted	2.957080094626565E-4	**0.002498568528447245**	*Negativa*
Awful	3.379520108144643E-4	**0.0025506220394565615**	*Negativa*
Wonderfully	**0.001605272051368705**	2.082140440372703E-4	*Positiva*

Tabla 12: Modelo NB Multinomial sobre preprocesamiento con *wordtokenizer* en la base inglesa

4.2.1.4. Naïve Bayes Multinomial Updateable

Según la Tabla 10, podemos ver los resultados de este clasificador en relación con el resto. Este clasificador mejora a Naïve Bayes y SMO salvo para el conjunto de entrenamiento. Se comporta igual que el NB Multinomial siendo los parámetros TFT e IDFT los que modifican el resultado empeorándolo en gran medida en el primer caso y ligeramente en el segundo.

Como en los casos anteriores su mejor resultado es con *NGrams (3,3)* lo vemos más claro en la Figura 57:

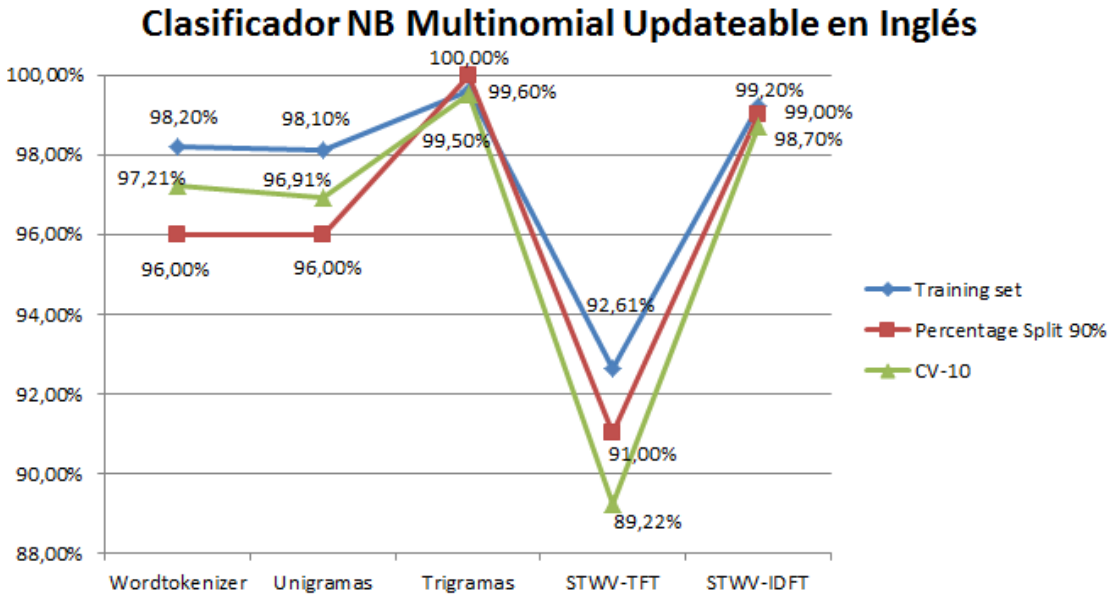


Figura 57: Resultados del clasificador NB Multinomial Updateable en inglés según el preprocesamiento y en las tres maneras de testado.

Estos resultados son muy buenos, aunque no mejoran nada el Naïve Bayes Multinomial y en el caso de los parámetros TFT e IDFT lo empeoran.

Comprobamos en este caso cómo se clasifican las palabras. Estos resultados están tomados del preprocesamiento de *wordtokenizer.*

En este caso también se calcula sobre como la probabilidad de una palabra dada una clase, pero en este caso la probabilidad independiente de la clase es absoluta, como cabe suponer de los resultados idénticos obtenidos, lo vemos en la Tabla 13.

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayesMultinomialUpdateable
Relation:     ReviewsIngles-weka.filters.unsupervised.attribute.StringToWordVector-R2-
W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-
```

tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r \t\r\n\t.,;:\"\'()?!-¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N -1
Instances: 1002
Attributes: 1846
[list of attributes omitted]
Test mode:evaluate on training data

=== Classifier model (full training set) ===

The independent probability of a class

POS 502.0
NEG 502.0

The probability of a word given the class
---

class	POS	NEG	
bad	6.938871417758404E63	**2.045494911349825E110**	*Negativa*
Worst	1.446257064291475E12	**4.923458286012058E41**	*Negativa*
wasted	1096.6331584284585	**7.016735912097631E20**	*Negativa*
Awful	2980.9579870417283	**1.9073465724950998E21**	*Negativa*
Wonderfully	**3.1855931757113756E16**	54.598150033144236	*Positiva*

Tabla 13: Modelo NB Multinomial Updateable sobre preprocesamiento con *wordtokenizer* en la base inglesa

Coincide con las conclusiones sacadas a priori.

4.2.1.5. Sequential Minimal Optimization

Según la Tabla 10, vemos que el SMO tiene los mejores resultados cuando se prueba sobre el conjunto de entrenamiento, mientras que en el resto de evaluadores se ve superado por los dos NB Multinomial.

Dentro de la ejecución del SMO vemos que el preprocesado que mejor resultados tiene es como en el resto de casos para *NGrams(3,3)*. Lo vemos más claro en la Figura 58.

Cabe destacar que para el caso del conjunto de entrenamiento ambos tipos de n-gramas obtenían la mejor performance mientras que para los otros dos evaluadores los *NGrams (1,1)* lo hacen peor que en NB.

Vemos que también en este caso los parámetros TFT e IDFT no afectan en nada a la obtención de resultados como pasaba en el caso del Naïve Bayes.

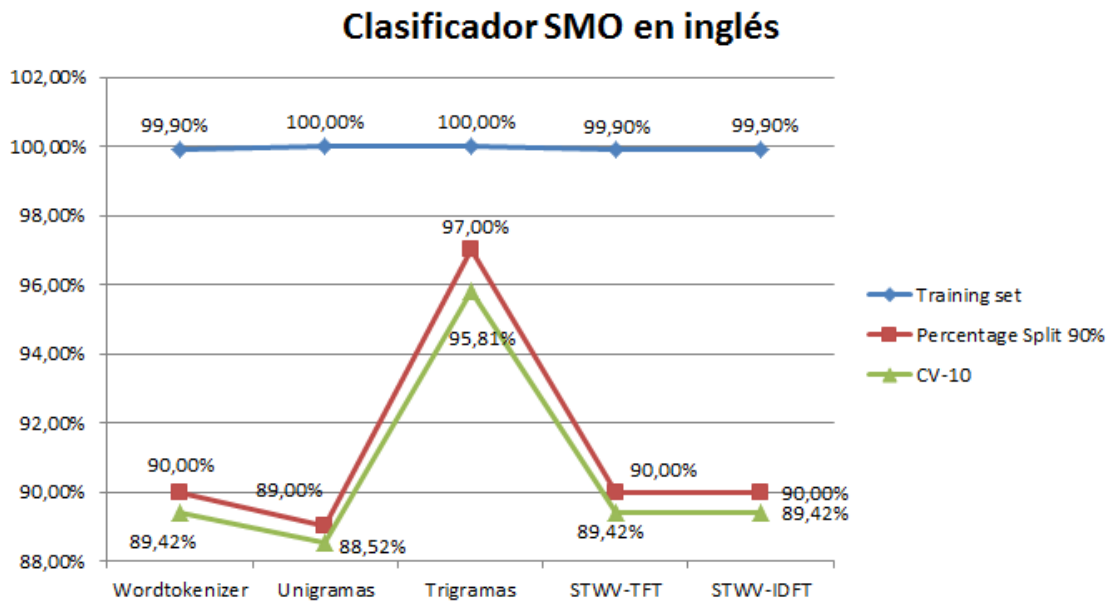


Figura 58: Resultados del clasificador SMO en inglés según el preprocesamiento y en las tres maneras de testado

Comprobamos que este clasificador dentro de conjuntos de entrenamiento, lo que quiere decir para información conocida, es el mejor. Y en el resto de evaluadores tiene una muy buena actuación aunque no óptima en el caso de trigramas.

Comprobamos ahora cómo se genera el modelo del SMO.

En este caso en vez de medias y desviaciones o probabilidades se muestran los pesos de las palabras como se muestra en la Tabla 14, siendo las negativas las que pertenecen a la dirección positiva.

```
=== Run information ===

Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K
"weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
Relation:  ReviewsIngles-weka.filters.unsupervised.attribute.StringToWordVector-R2-
W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-
tokenizerweka.core.tokenizers.WordTokenizer  -delimiters  "  \r  \t\r\n\t.,;:\"\'()?!-
¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-
Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N
-1
Instances:   1002
Attributes:   1846
[list of attributes omitted]
Test mode:evaluate on training data

=== Classifier model (full training set) ===

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>
```

Classifier for classes: POS, NEG BinarySMO Machine linear: showing attribute weights, not support vectors.		
bad	-0.0159 * (normalized) bad	*Positiva*
Worst	+ 0.3922 * (normalized) worst	*Negativa*
wasted	+ 0.6383 * (normalized) wasted	*Negativa*
Awful	+ 0.4155 * (normalized) awful	*Negativa*
Wonderfully	+ -0.1034 * (normalized) wonderfully	*Positiva*

Tabla 14: Modelo SMO sobre preprocesamiento con *wordtokenizer* en la base inglesa

Las que están marcadas con el signo negativo son de la clase positiva, en este caso el primero debería ser negativo pero aparece positivo, posiblemente porque esta palabra puede estar en frases de carácter positivo, como "*Not that bad"*.

Tras el estudio de estas cuatro secciones podemos llegar a la conclusión que el preprocesamiento más adecuado para la base inglesa es *NGrams(3,3)* dado que en prácticamente todos los casos era la que proporcionaba mejores resultados.

En las siguientes secciones vamos a comparar directamente los preprocesamientos contra los clasificadores.

4.2.1.6. Wordtokenizer

En este apartado vamos a ver qué clasificador habría que utilizar en el caso de que el preporcesamiento incluyera el *tokenizer* en el *StringToWordVector* de *wordtokenizer*.

Presentamos la siguiente gráfica con los porcentajes de las instancias correctas según cada uno de los cuatro clasificadores en los diferentes evaluadores.

Donde las siglas NB son por Naïve Bayes, NB M por Naïve Bayes Multinonial, NB MU por Naïve Bayes Multinomial Updateable y SMO de Sequential Minimal Optimization, de ahora en adelante en todas las gráficas.

Y los tres evaluadores estudiados (Figura 59), *Training Set* sobre conjunto de entrenamiento. *Percentage Split 90*; generado el modelo sobre el 90% de la base y testado en el 10% restante. *CV-10* validación cruzada en diez partes.

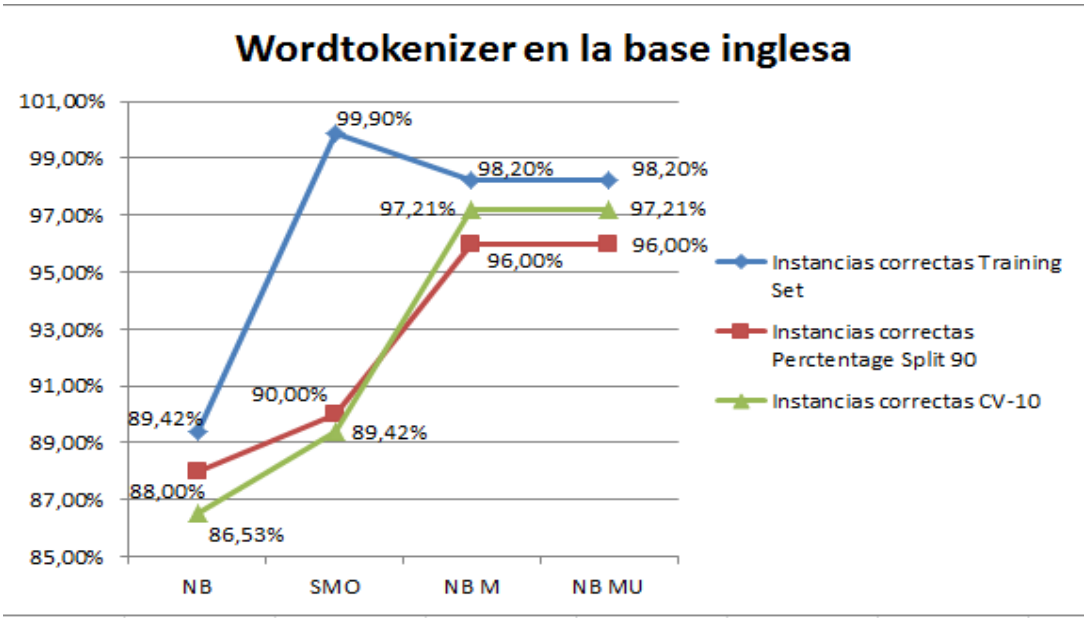


Figura 59: Resultados del preprocesamiento *wordtokenizer* sobre los clasificadores, según el método de evaluación en la base inglesa.

Comprobamos que en general los resultados son mejores en el caso de evaluar sobre el conjunto de entrenamiento, y en este caso destaca el clasificador SMO con unos resultados casi perfectos.

Entre los clasificadores para caso de *Percentage Split* tienen mejores resultados que la validación cruzada en el caso de NB y SMO, sin embargo son superados por el CV-10 en el caso de los NB Multinomial, que tienen la misma actuación, consiguiendo para ambos evaluadores los mejores resultados para el *wordtokenizer* con los dos clasificadores NB M y NB MU independientemente de qué evaluador sea mejor en este caso.

Lo que coincide en todos los casos es que el peor clasificador es el Naïve Bayes. Los resultados en general son buenos para los casos de Naïve Bayes Multinomial y Naïve Bayes Multinomial en los tres evaluadores.

4.2.1.7. N-Grams(1,1)

Veamos ahora qué ocurre en el caso de que el *tokenizer* del STWV sea *NGrams(1,1)*. Para ello nos apoyamos en la gráfica usada en el caso anterior (Figura 60).

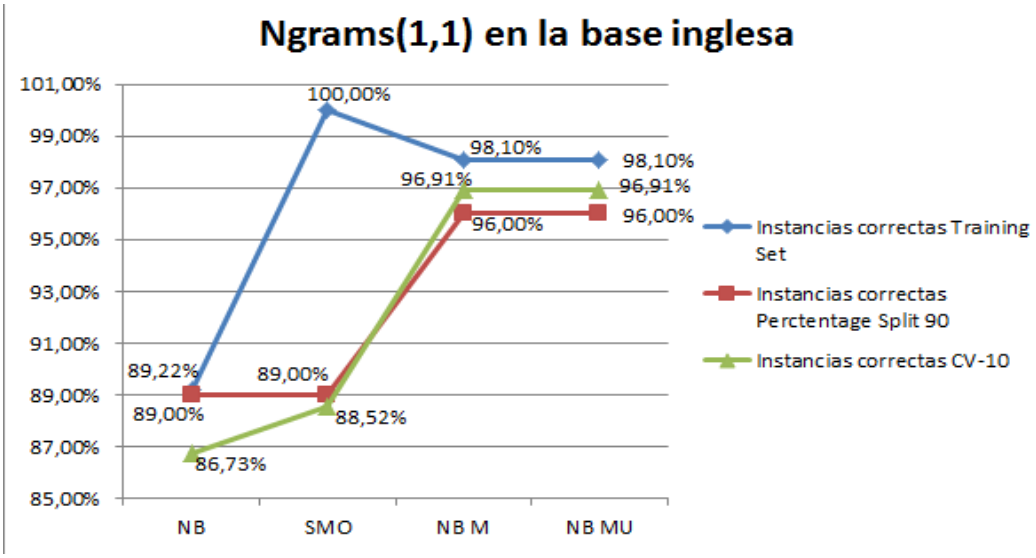


Figura 60: Resultados del preprocesamiento *NGrams(1,1)* sobre los clasificadores, según el método de evaluación en la base inglesa.

La forma de estos resultados es muy parecida a la del *wordtokenizer*, siendo los mejores resultados con el evaluador sobre el conjunto de entrenamiento, y en este caso para el clasificador SMO. Y para los otros dos evaluadores con mejores resultados para los Naïve Bayes Multinomial, que tanto éste como el Updateable obtienen los mismos valores, y en estos casos para el evaluador de validación cruzada.

En general en todos los evaluadores para los clasificadores NB M y NB MU tienen buenos resultados.

4.2.1.8. N-Grams(3,3)

Estudiamos ahora el caso de que el *tokenizer* del STWV sea n-gramas con máximo de tres y mínimo de tres (trigramas).

Presentamos el gráfico en la Figura 61como en las secciones previas:

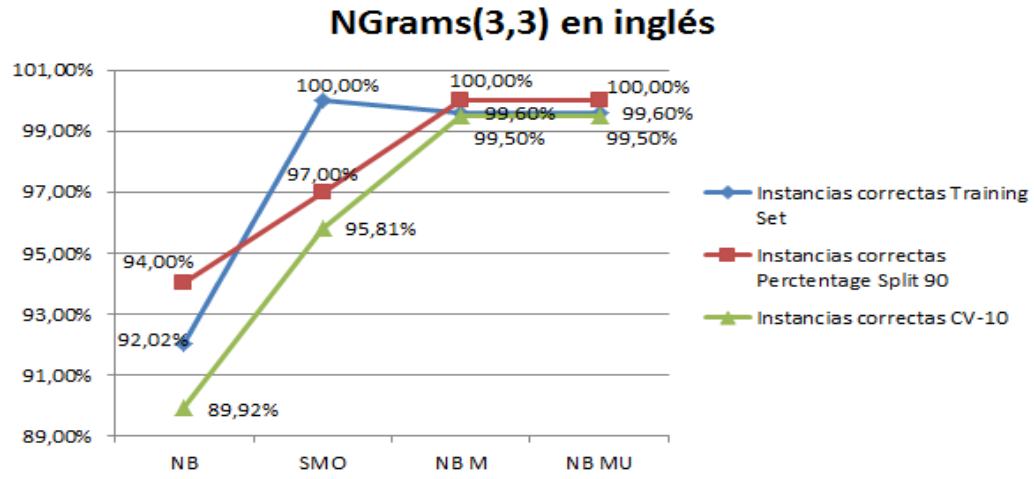


Figura 61: Resultados del preprocesamiento *NGrams(3,3)* sobre los clasificadores, según el método de evaluación en la base inglesa.

En este caso las formas cambian respecto a los dos *tokenizers* anteriores. Ya no son siempre los mejores resultados con el evaluador sobre el conjunto de entrenamiento. En este caso y como sí que pasaba en los dos casos anteriores el mejor clasificador es el SMO.

Vemos ahora que para los casos de los clasificadores NB M y NB MU el mejor evaluador es el *Percentage Split 90%*, siendo la validación cruzada la peor aunque en el caso de estos clasificadores casi igual que el *Training Set.*

En este caso se obtienen muy buenos resultados para los clasificadores Naïve Bayes Multinomial y Naïve Bayes Multinomial Updateable aunque no se aprecia diferencia entre estos dos.

4.2.1.9. Parámetro TFT

Veamos qué diferencias aplica el poner este parámetro en verdadero en el STWV cuando el *tokenizer* elegido es el *wordtokenizer.*

Representamos los resultados en la siguiente Figura 62:

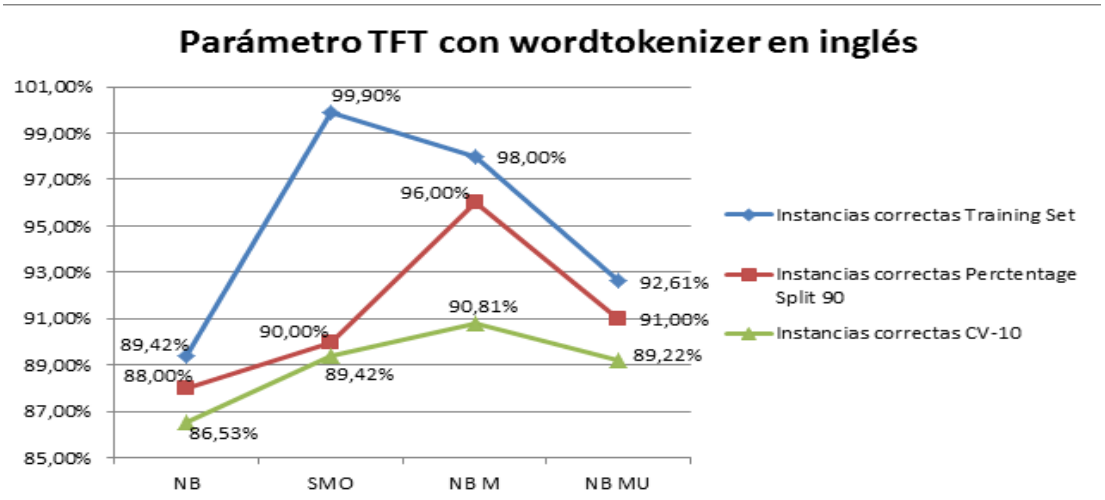


Figura 62: Resultados del preprocesamiento *wordtokenizer* y el parámetro TFT sobre los clasificadores, según el método de evaluación en la base inglesa.

Lo que primero llama la atención de esta figura respecto a las anteriores es que este parámetro influye en el clasificador Naïve Bayes Multinomial Updateable empeorándolo respecto a casos anteriores y con referencia al NB M.

Lo segundo es que en este caso el evaluador de validación cruzada es el que peores resultados obtiene y dentro de un rango mucho menor, no se ven grandes picos como en los otros evaluadores, en el caso del SMO para el conjunto de entrenamiento es el mejor con diferencia, lo mismo se podría decir del evaluador *Percentage Split 90%* con el clasificador Naïve Bayes Multinomial, que es mucho mejor, mientras que el CV-10, aunque sea el mejor en el mismo clasificador, no por tanta diferencia.

En general para el clasificador NB M se obtienen resultados buenos con los evaluadores de *Training Set* y haciendo el test sobre el 10% de la base de datos.

4.2.1.10. Parámetro IDFT

Comprobamos en esta sección el efecto de poner a verdadero este parámetro IDFT cuando el *tokenizer* en el *StringToWordVector* es el *wordtokenizer*.

Vemos los resultados en la Figura 63.

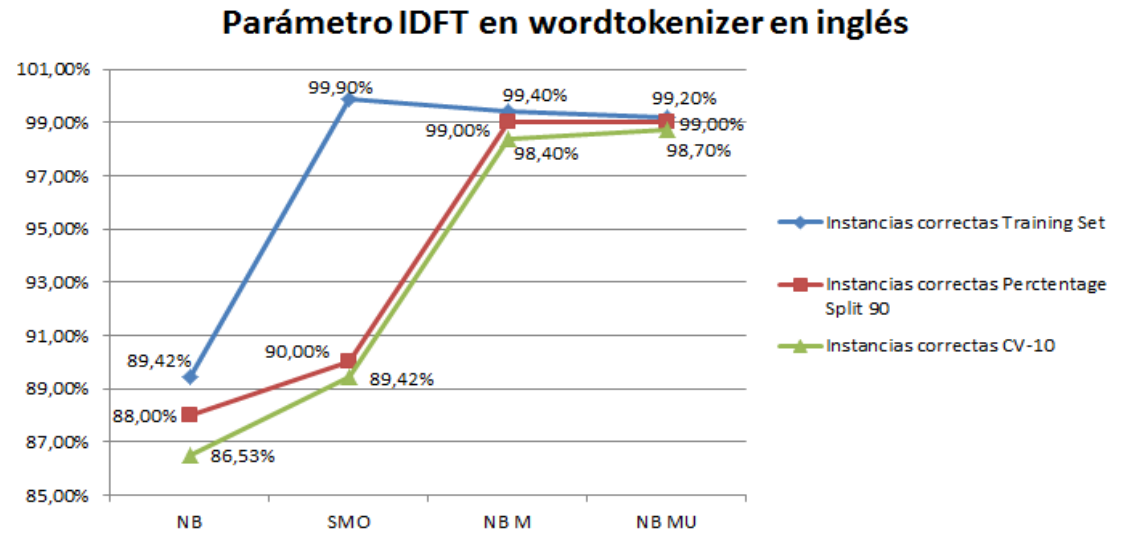


Figura 63: Resultados del preprocesamiento *wordtokenizer* y el parámetro IDFT sobre los clasificadores, según el método de evaluación en la base inglesa.

En este caso el clasificador NB MU también empeora en los casos de evaluar sobre el conjunto de entrenamiento al NB M, pero muy ligeramente. Lo interesante es que para la validación cruzada este parámetro consigue mejorar un poco los resultados del Naïve Bayes Multinomial.

Como viene ocurriendo para el conjunto de entrenamiento el mejor clasificador es el SMO, aunque en este caso los resultados de los Naïve Multinomial no se alejan mucho.

Para el *Percentage Split 90%* los resultados idénticos y mejores son para los dos casos de NB M. Y para la CV-10 el clasificador NB MU.

En general son resultados muy buenos para los clasificadores NB M.

En conclusión para los casos en los que se vaya a evaluar sobre un conjunto de entrenamiento, que es un escenario poco realista, elegiría sin lugar a dudas el clasificador SMO en todos los casos. Pero cuando evalúo en bases de test es preferible el Naïve Bayes Multinomial.

4.2.2. Base de datos española

En esta sección se van a describir los resultados en español. Recordamos que se han estudiado tres bases españolas tal y como se comentó en la sección 3.2.1. Como se explicó en dicha sección se descartó la primera base de datos (opiniones sobre la película Avatar) porque estaba muy desbalanceada.

De hecho, dicha base de datos tenía muy pocas opiniones negativas. La mejor manera de ver cómo esto afectaba a los resultados es poniendo la matriz de confusión de cualquier caso puesto que en todas era parecido.

Miramos en el caso del clasificador SMO sobre la base de entrenamiento cuando el STWV utiliza el *NGrams(1,1)* que se vio en la base inglesa que conseguía un 100%.

```
    a   b   <-- classified as
786   0 |   a = POS
 21 136 |   b = NEG
```

Está claro que prácticamente todos los errores se cometen en la clase negativa que sólo consta de 157 instancias. Y no sólo para el *NGrams(1,1)*, sino para el resto de n-gramas y el *wordtokenizer*. Esto es debido a que esta base de datos no cuenta con suficientes muestras negativas como para modelar la clase correspondiente de forma adecuada. Con esto queda comprobado que el análisis de los resultados no iba a ser muy preciso puesto que cuando el resultado de la clasificación era del 97.77% se debía a las instancias de la clase negativa, la positiva era perfecta.

Para darle un peso más parecido a ambas clases se procedió a la mezcla de las bases de datos fijando el número de instancias con las negativas de todas. Los resultados no son tan buenos como con la de Avatar, pero son más reales y están más repartidos.

A partir de ahora sólo se contarán los resultados obtenidos en español con la base de datos variada y a la que se quitaron los caracteres extraños debidos a los cambios de formatos.

4.2.2.1. Resultados según clasificador y procesamiento

Comenzamos por una comparación general de todos los métodos utilizados. En este caso, por lo reducido de la base, se van a exponer también los resultados con *NGrams(3,1)*.

Primero mostramos un gráfico con los resultados del evaluador *Training Set* en la Figura 64.

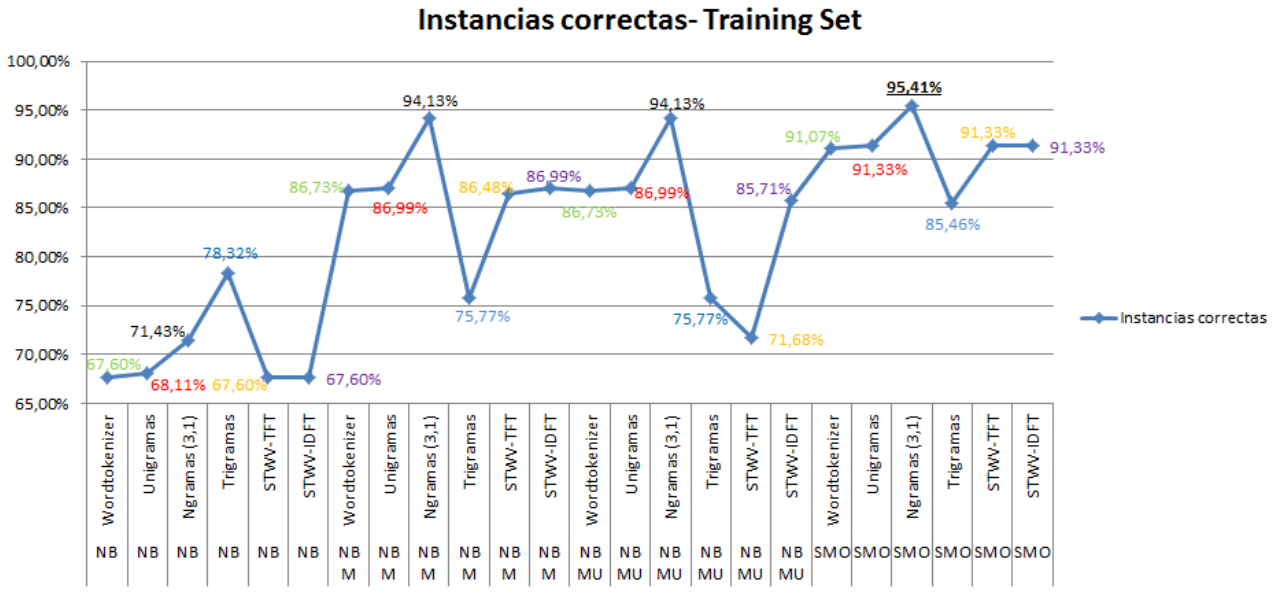


Figura 64: Resultados de la aplicación de los clasificadores sobre el conjunto de entrenamiento en español

Para entenderlo hay que ver que los clasificadores están puestos en horizontal y las siglas significan:

- NB: Naïve Bayes
- NB M: Naïve Bayes Multinomial
- NB MU: Naïve Bayes Multinomial Updateable
- SMO: Sequential Minimal Optimization

Lo primero que podemos decir que el peor clasificador independientemente del preprocesamiento es el Naïve Bayes, que además tiene unos resultados muy pobres de como máximo el 78%.

El Naïve Bayes Multinomial y el Naïve Bayes Multinomial Updateable tienen una actuación idéntica cuando no entran en juego los parámetros TFT e IDFT.

En el caso del SMO, es el clasificador que produce los mejores resultados y dentro de un rango más estrecho que el resto.

El mejor preprocesamiento en la mayoría de los casos es para *NGrams(3,1)* y el peor para *NGrams(3,3).*

Vemos cómo actúan los otros evaluadores en comparación con el *Training Set* en la Figura 65.

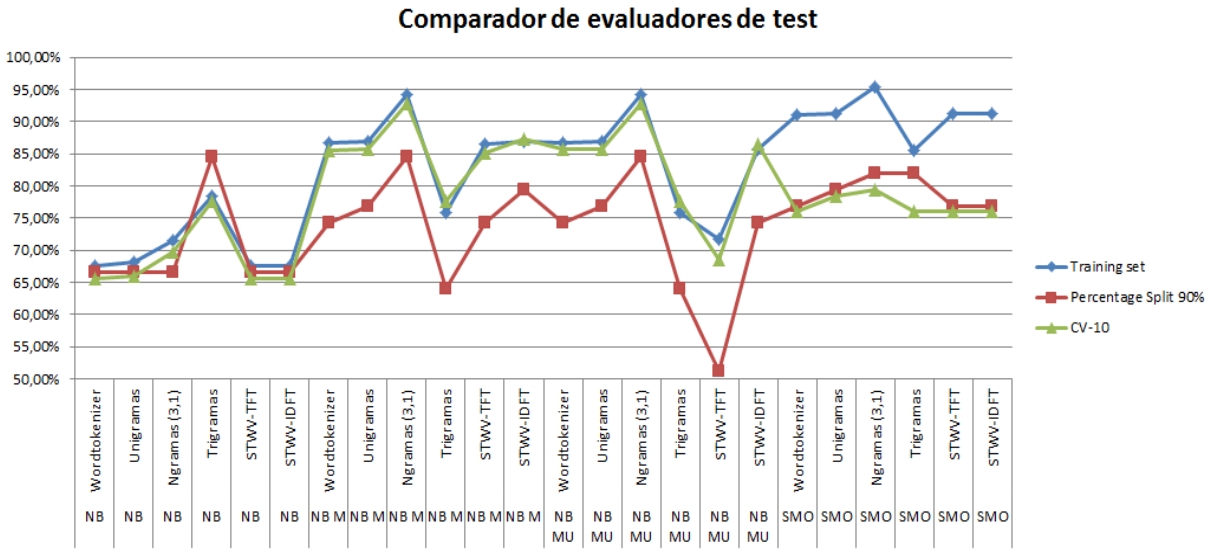


Figura 65: Comparación de evaluadores de test en la base en español

En este caso la validación cruzada y el usar el 10% de la base de datos como test tienen un comportamiento parejo salvo en el caso del SMO, que como en la base en inglés mejora mucho en el *Training set.* El evaluador *Percentage Split 90%* obtiene unos resultados mucho peores, superando únicamente a CV-10 para el clasificador SMO y a ambos en el caso de NB para *Ngrams(3,3)* donde tiene una diferencia a mejor.

Los valores de esta gráfica se muestran en la siguiente Tabla 15:

Clasificador	Tokenizer	Training set	Percentage Split 90	CV-10
NB	**Wordtokenizer**	67,60%	66,67%	65,56%

	Unigramas	68,11%	66,67%	66,07%
	Ngramas(3,1)	71,43%	66,67%	69,64%
	Trigramas	**78,32%**	***84,62%***	**77,55%**
	STWV-TFT	67,60%	66,67%	65,56%
	STWV-IDFT	67,60%	66,67%	65,56%
NB M	**Wordtokenizer**	86,73%	74,36%	85,46%
	Unigramas	86,99%	76,92%	85,71%
	Ngramas(3,1)	**94,13%**	***84,62%***	***92,86%***
	Trigramas	75,77%	64,10%	77,55%
	STWV-TFT	86,48%	74,36%	85,20%
	STWV-IDFT	86,99%	79,49%	87,24%
NB MU	**Wordtokenizer**	86,73%	74,36%	85,71%
	Unigramas	86,99%	76,92%	85,71%
	Ngramas(3,1)	**94,13%**	***84,62%***	***92,86%***
	Trigramas	75,77%	64,10%	77,55%
	STWV-TFT	71,68%	51,28%	68,62%
	STWV-IDFT	85,71%	74,36%	86,48%
SMO	**Wordtokenizer**	91,07%	76,92%	76,02%
	Unigramas	91,33%	79,49%	78,32%
	Ngramas(3,1)	***95,41%***	**82,05%**	**79,34%**
	Trigramas	85,46%	82,05%	76,02%
	STWV-TFT	91,33%	76,92%	76,02%
	STWV-IDFT	91,33%	76,92%	76,02%

Tabla 15: Comparación de clasificadores, preprocesamientos y evaluadores en la base en español.

En las siguientes secciones vamos a estudiar cada caso más detalladamente valiéndonos de los datos de esta tabla.

4.2.2.2. Naïve Bayes

Basándonos en la Tabla 15 podemos ver que los resultados de este clasificador son los peores con diferencia, sin importar el evaluador. Salvo para el caso únicamente de que el *tokenizer* del filtro de preprocesamiento STWV sea *NGrams(3,3)*, donde supera a todos los demás clasificadores y siendo el evaluador con el mejor resultado el *Percentage Split 90%* con un 84% de acierto que no se puede considerar muy bueno.

Vemos en la Figura 66 los valores:

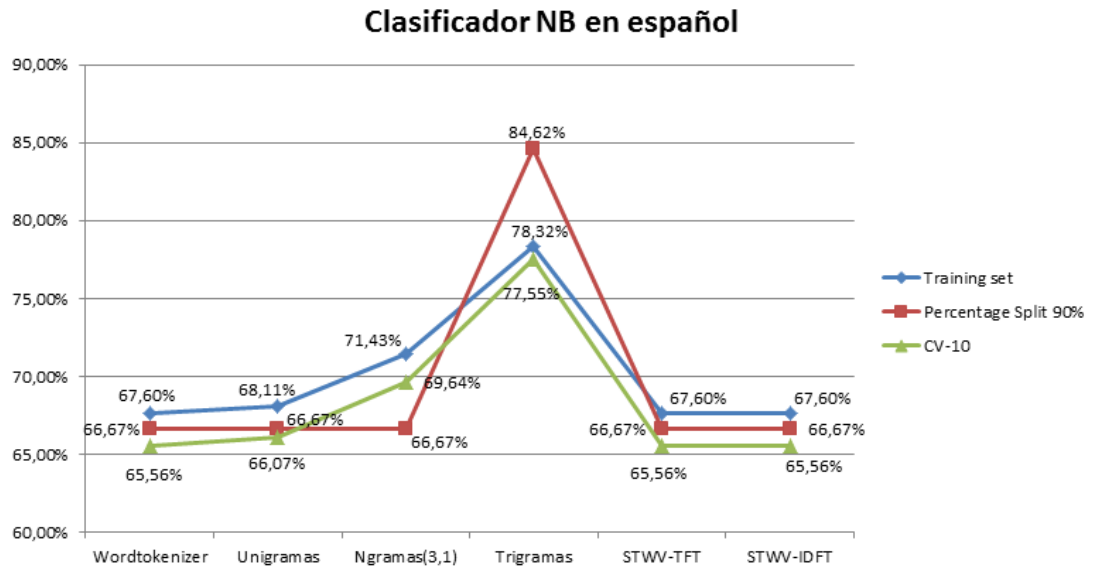


Figura 66: Resultados del clasificador NB en español según el preprocesamiento y en las tres maneras de testado.

Los resultados son malos en general y sólo utilizaría el clasificador con preprocesamiento que incluyera trigramas aunque ni en este caso los resultados serían óptimos.

Podemos ver con este clasificador los valores que ha dado a la media y la desviación típica de las primeras palabras del modelo y la clase en la que se interpretaría que está.

Este resultado del modelo es para el preprocesamiento con *wordtokenizer* y evaluador *Training set* en la Tabla 16.

=== Run information === Scheme:weka.classifiers.bayes.NaiveBayes Relation: ReviewsSPA-weka.filters.unsupervised.attribute.StringToWordVector-R2-W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r \t\r\n\t.,;:\"\'()?!-¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N -1 Instances: 392 Attributes: 347 [list of attributes omitted] Test mode:evaluate on training data === Classifier model (full training set) === Naive Bayes Classifier			
Class Attribute ==============	POS (0.5)	NEG (0,5)	
Avatar mean std. dev.	 **0.3878** 0.4872	 0.1786 0.383	*Positiva*

weight sum precision	196 1	196 1	
Anfiteatro mean std. dev. weight sum precision	0 0.1667 196 1	**0.0663** 0.2489 196 1	*Negativa*
Durante mean std. dev. weight sum precision	**0.1122** 0.3157 196 1	0.0153 0.1667 196 1	*Positiva*
Dura mean std. dev. weight sum precision	**0.051** 0.22 196 1	0 0.1667 196 1	*Positiva*
Conseguido mean std. dev. weight sum precision	**0.0714** 0.2575 196 1	0.0051 0.1667 196 1	*Positiva*

Tabla 16: Modelo NB sobre preprocesamiento con *wordtokenizer* en la base española.

Marcados en negrita los valores mayores de la media que dan la clase, en este caso la polaridad.

4.2.2.3. Naïve Bayes Multinomial

Según los valores de la Tabla 15 podemos ver que este clasificador es mejor que el NB y prácticamente igual que el Naïve Bayes Multinomial Updateable. Y sólo superado por el SMO en el caso de evaluar sobre el conjunto de entrenamiento.

Dentro de este clasificador, los mejores resultados se dan para *NGrams(3,1)* y los parámetros TFT e IDFT influyen sobre el *wordtokenizer*. Lo vemos en la Figura 67:

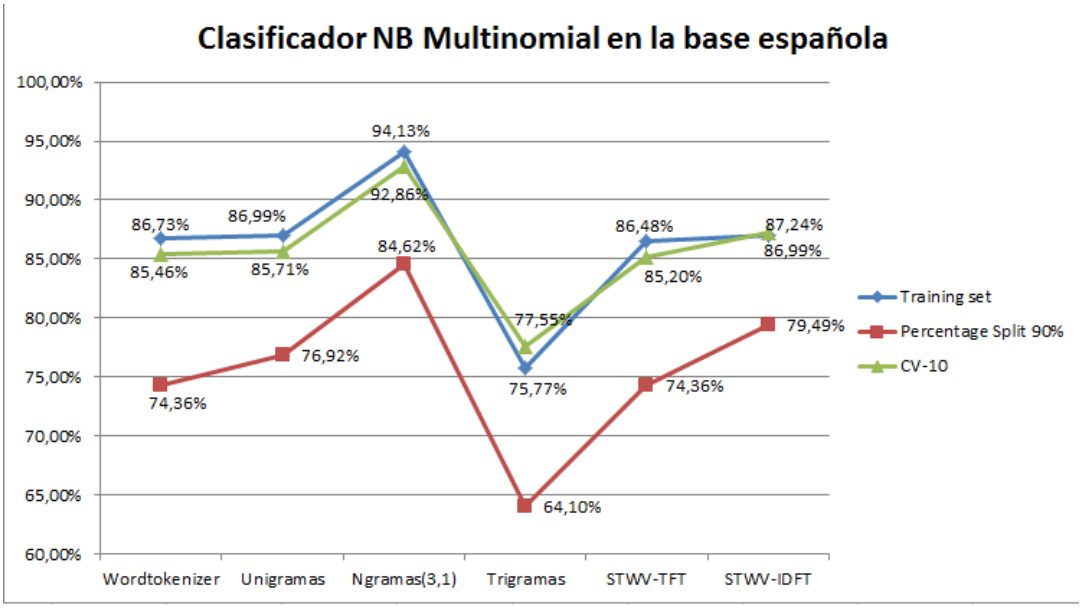


Figura 67: Resultados del clasificador NB Multinomial en español según el preprocesamiento y en las tres maneras de testado.

El evaluador *Percentage Split 90%* es el que peor actuación tiene. Los otros dos más o menos parecido.

Vemos que el parámetro TFT empeora levemente el resultado del *wordtokenizer*. Mientras que en el caso del IDFT mejora el resultado, claramente en el caso del *Split Percentage 90%* aunque en los otros dos casos también.

Vemos ahora los resultados del modelo de las primeras palabras. Para el *tokenizer* de *wordtokenizer* en la Tabla 17.

En este caso, no se dan los mismos valores que en Naïve Bayes, en vez de la media y la desviación típica del token, se da directamente la probabilidad de la palabra dada la clase.

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayesMultinomial
Relation: ReviewsSPA-weka.filters.unsupervised.attribute.StringToWordVector-R2-
W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-
tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r \t\r\n\t.,;:\"\'()?!-
¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-
Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N
-1
Instances:  392
Attributes:  347
[list of attributes omitted]
Test mode:evaluate on training data

=== Classifier model (full training set) ===

The independent probability of a class
--------------------------------------
POS  0.5
```

NEG 0.5

The probability of a word given the class
---

class	POS	NEG	
avatar	**0.02650602409638555**	0.024456521739130432	*Positiva*
anfiteatro	3.442340791738383-4	**0.009510869565217388**	*Negativa*
durante	**0.00791738382099822**	0.002717391304347825	*Positiva*
dura	**0.0037865748709122217**	6.793478260869568E-4	*Positiva*
conseguido	**0.00516351118760753**	0.0013586956521739126	*Positiva*

Tabla 17: Modelo NB Multinomial sobre preprocesamiento con *wordtokenizer* en la base española

En el caso de Avatar las probabilidades están muy próximas aunque se decantaría por positiva.

4.2.2.4. Naïve Bayes Multinomial Updateable

Vemos en la Tabla 15 que este clasificador es mejor que el NB salvo para el caso de *NGrams(3,3)* que supera al SMO en los evaluadores de *Cross Validation* y *Percentage Split*. Siendo su actuación muy parecida al NB M, salvo cuando entran en juego los parámetros TFT e IDFT sobre el *wordtokenizer*.

Vemos en la Figura 68 cómo se comporta este clasificador respecto a los preprocesamientos y los evaluadores:

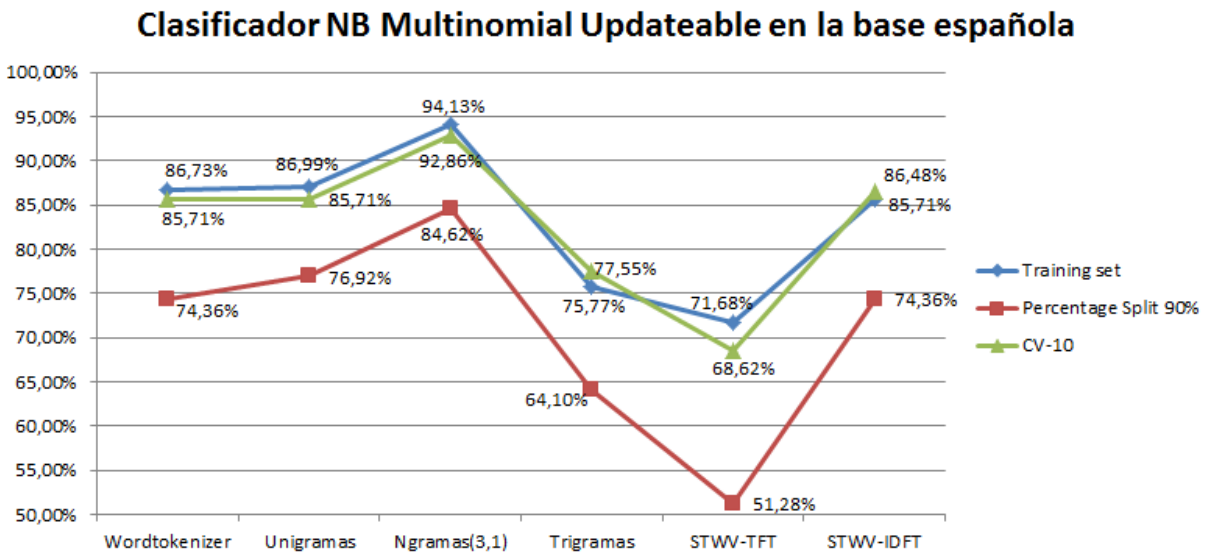


Figura 68: Resultados del clasificador NB Multinomial Updateable en español según el preprocesamiento y en las tres maneras de testado.

Los mejores resultados se dan para *NGrams(3,1)* siendo para los evaluadores sobre el conjunto de entrenamiento y validación cruzada unos valores aceptables de resultado por encima del 90%.

Vemos que el efecto del parámetro TFT es malísimo, todos los resultados caen en picado. Sin embargo para el caso de evaluar con *cross Validation* el parámetro IDFT mejora los resultados en casi un pp, esa misma cantidad es lo que baja el resultado si el evaluador es *Training Set*.

En este caso en el de Naïve Bayes Multinomial Updateable se calculan las probabilidades absolutas de un token dada una clase (Tabla 18).

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayesMultinomialUpdateable
Relation: ReviewsSPA-weka.filters.unsupervised.attribute.StringToWordVector-R2-
W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-
tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r \t\r\n\t.,;:\"\'()?!-
¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-
Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N
-1
Instances: 392
Attributes: 347
[list of attributes omitted]
Test mode:evaluate on training data

=== Classifier model (full training set) ===

The independent probability of a class
-------------------------------------
POS 197.0
NEG 197.0

The probability of a word given the class
-----------------------------------------------------------------------------
```

class	POS	NEG	
avatar	**2.7585134545231703E33**	4.311231547115195E15	*Positiva*
anfiteatro	2.718281828459045	**1202604.2841647768**	*Negativa*
durante	**9.744803446248903E9**	54.598150033144236	*Positiva*
dura	**59874.14171519782**	2.718281828459045	*Positiva*
conseguido	**3269017.3724721107**	7.38905609893065	*Positiva*

Tabla 18: Modelo NB Multinomial Updateable sobre preprocesamiento con *wordtokenizer* en la base española

Se obtiene la misma clasificación que en el caso anterior.

4.2.2.5. Sequential Minimal Optimization

Como hemos comentado al comparar el resto de clasificadores, y vemos en la Tabla 15, este tiene los mejores resultados sobre el conjunto de entrenamiento. En el caso de CV-10 va un poco por detrás de los NB Multinomial y en el *Percentage Split 90%* en unos casos es mejor el SMO y en otros los NB M y NB MU.

Vemos su actuación sobre los preprocesamientos en la Figura 69:

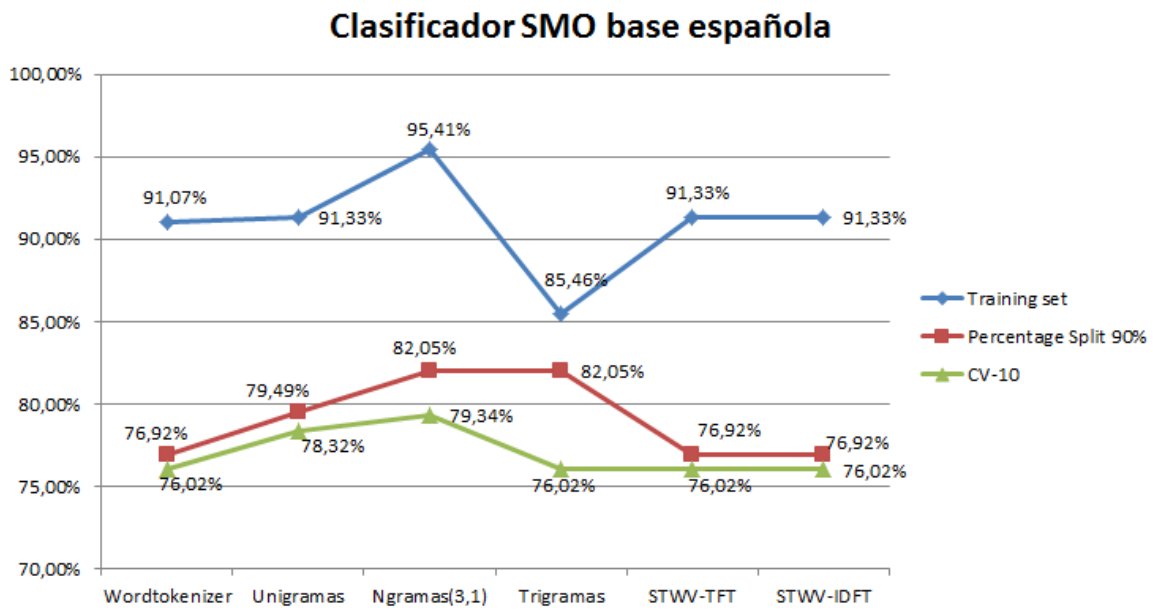


Figura 69: Resultados del clasificador NB Multinomial Updateable en español según el preprocesamiento y en las tres maneras de testado

Vemos que sobre el conjunto de entrenamiento no tiene rival. Notamos también que los parámetros TFT e IDFT no influyen para nada en su actuación.

Los valores obtenidos para los evaluadores de validación cruzada y *Percentage Split 90%* no son buenos puesto que no llegan ni al 85% de instancias correctas.

Vemos cómo sale el clasificador en este caso en la Tabla 19, recordamos que aquí se muestran los pesos:

```
=== Run information ===

Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K
"weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
Relation: ReviewsSPA-weka.filters.unsupervised.attribute.StringToWordVector-R2-
W1000000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-M1-
tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r \t\r\n\t.,;:\"\'()?!-
¿¡+*&#$\%\\/=<>[]_`@"-weka.filters.supervised.attribute.AttributeSelection-
Eweka.attributeSelection.InfoGainAttributeEval-Sweka.attributeSelection.Ranker -T 0.0 -N
-1
Instances:  392
Attributes:  347
[list of attributes omitted]
Test mode:evaluate on training data

=== Classifier model (full training set) ===

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>
```

Classifier for classes: POS, NEG BinarySMO Machine linear: showing attribute weights, not support vectors.		
Avatar	0.1375 * (normalized) avatar	*Negativa*
Anfiteatro	+ 1.2935 * (normalized) anfiteatro	*Negativa*
durante	+ -0.5863 * (normalized) durante	*Positiva*
Dura	+ -1.6306 * (normalized) dura	*Positiva*
Conseguido	+ -0.9871 * (normalized) conseguido	*Positiva*

Tabla 19: Modelo SMO sobre preprocesamiento con wordtokenizer en la base española

En este caso Avatar es negativa y su peso es bajo.

Tras ver los resultados según el preprocesamiento vemos que las mejores tasas de acierto se consiguen con los *NGrams(3,1)* en prácticamente todos los casos, obteniendo en ocasiones resultados por encima del 90%.

Vamos a estudiar en las siguientes secciones cómo se comportan los preprocesamientos según los clasificadores y evaluadores viendo qué solución es preferible en cada caso.

4.2.2.6. Wordtokenizer

Comenzamos con el caso en el que aplicamos en el filtro *StringToWordVector* como distribuidor de tokens el *wordtokenizer.*

En la Figura 70 como en las secciones de la base de inglés se muestran los resultados según los clasificadores, siendo NB Naïve Bayes, NB M Naïve Bayes Multinomial, NB MU Naïve Bayes Multinomial Updateable y SMO Sequential Minimal Optimization.

Y los evaluadores *Training Set*, testeado sobre el conjunto de entrenamiento, *Percentage Split 90%* construye el modelo sobre el 90% de la base y lo evalúa en el restante 10%, *CV-10* validación cruzada de diez partes.

Esto será igual en las siguientes secciones.

Vemos los resultados en este caso en la Figura 70:

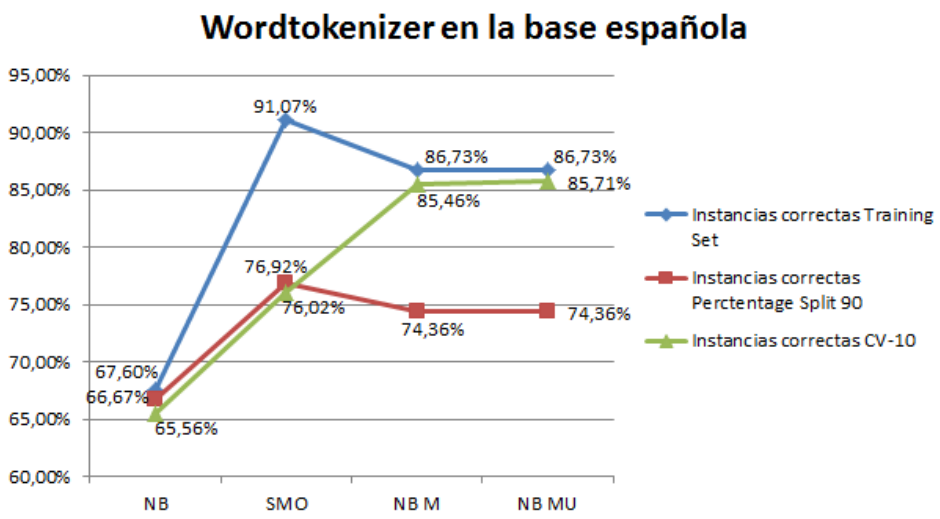


Figura 70: Resultados del preprocesamiento *wordtokenizer* sobre los clasificadores, según el método de evaluación en la base española.

Vemos que los mejores resultados se dan para el conjunto de entrenamiento, especialmente en el uso del SMO, el resto de valores se parecen al CV-10, pero en general son valores malos que sólo en el caso de SMO para *Training Set* supera el 90%.

Por acercarse más a la realidad el evaluador de validación cruzada obtiene el mejor resultado para Naïve Bayes Multinomial Updateable, por lo que en este caso ese sería el clasificador a utilizar.

4.2.2.7. N-Grams(1,1)

Comprobamos en este caso, para el *tokenizer* con n-gramas con máximo y mínimo de 1, unigramas, en el STWV frente a los evaluadores los resultados se ven en Figura 71.

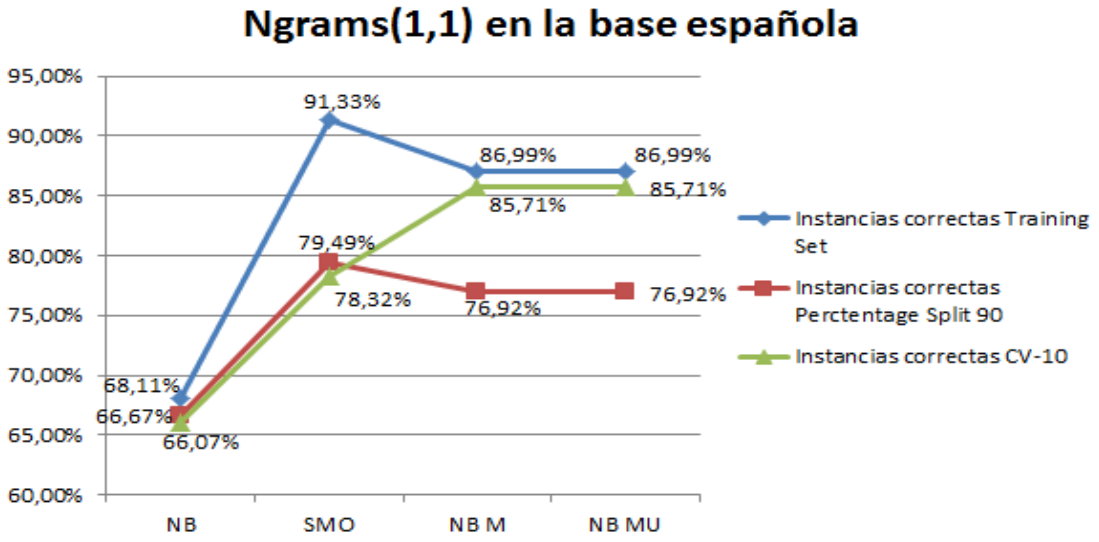


Figura 71: Resultados del preprocesamiento *NGrams(1,1)* sobre los clasificadores, según el método de evaluación en la base española.

Vemos que tiene una forma parecida a la del *wordtokenizer*, donde el clasificador NB tiene una pobre actuación, sólo para el SMO en el conjunto de entrenamiento se obtienen resultados por encima del 90%.

En este caso el NB Multinonial y el NB Multinomial Updateable tienen la misma actuación en el caso de los tres evaluadores, y serían los que escogería dado que para CV-10 dan los mejores resultados.

4.2.2.8. N-Grams(3,1)

Vemos que ocurre con este tipo de *tokenizer* los resultados se espera que sean mejores que los en los dos casos anteriores, pero lo vemos mejor en la Figura 72:

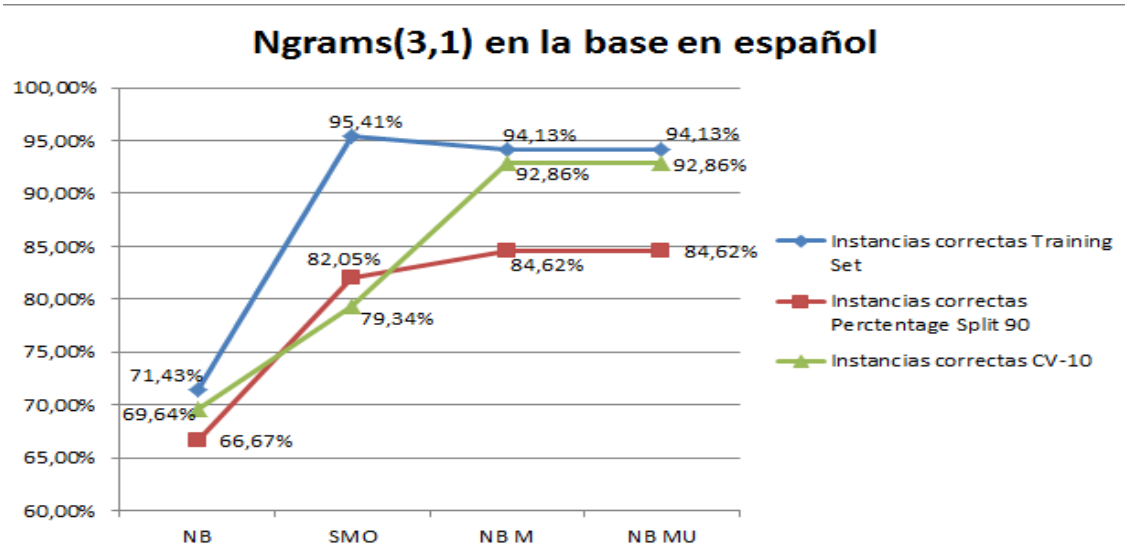


Figura 72: Resultados del preprocesamiento *NGrams(3,1)* sobre los clasificadores, según el método de evaluación en la base española.

Sigue teniendo unos malos resultados el clasificador de Naïve Bayes. Pero en este caso hay más clasificadores que obtienen resultados por encima del 90% para tanto el evaluador *Training Set* como la validación cruzada.

En este caso tampoco se ve diferencia entre los dos tipos de NB Multinomial, el uso de cualquiera de ellos sería bueno puesto que las instancias correctas superan el 90%.

4.2.2.9. N-Grams(3,3)

En este caso se podrían esperar muy buenos resultados como ocurría en la base inglesa, pero comprobamos en la Figura 73 que para español este tipo de organizador de tokens no es el apropiado:

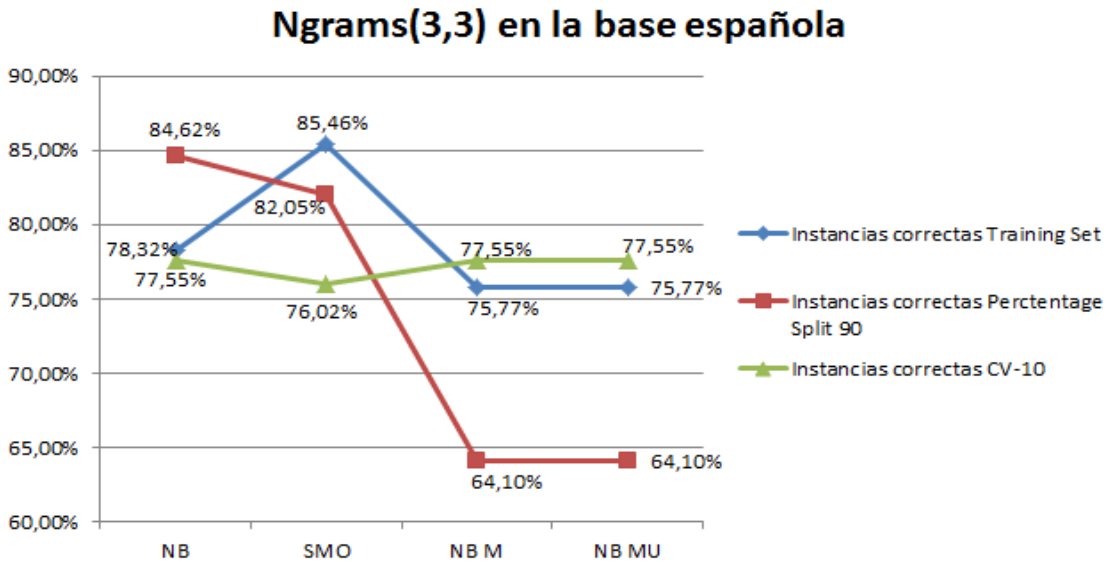


Figura 73: Resultados del preprocesamiento *NGrams(3,3)* sobre los clasificadores, según el método de evaluación en la base española.

En este caso los resultados son malos en general, ninguno llega al 90%. Lo que sorprende también en este caso es que el clasificador NB no es el que peor resultados obtiene para ninguno de los evaluadores, es más, en el caso de *Percentage Split 90%* es el mejor con diferencia, sólo superado por el SMO sobre el conjunto de entrenamiento que como en los casos anteriores es el que da el mejor resultado, aunque en este caso por debajo del 90% que sería aceptable.

Tampoco se observan diferencias entre los clasificadores NB Multinomial que tienen una actuación como mínimo igual a la del Naïve Bayes.

4.2.2.10. Parámetro TFT

Estudiamos en este caso la influencia del parámetro TFT sobre los clasificadores en la Figura 74:

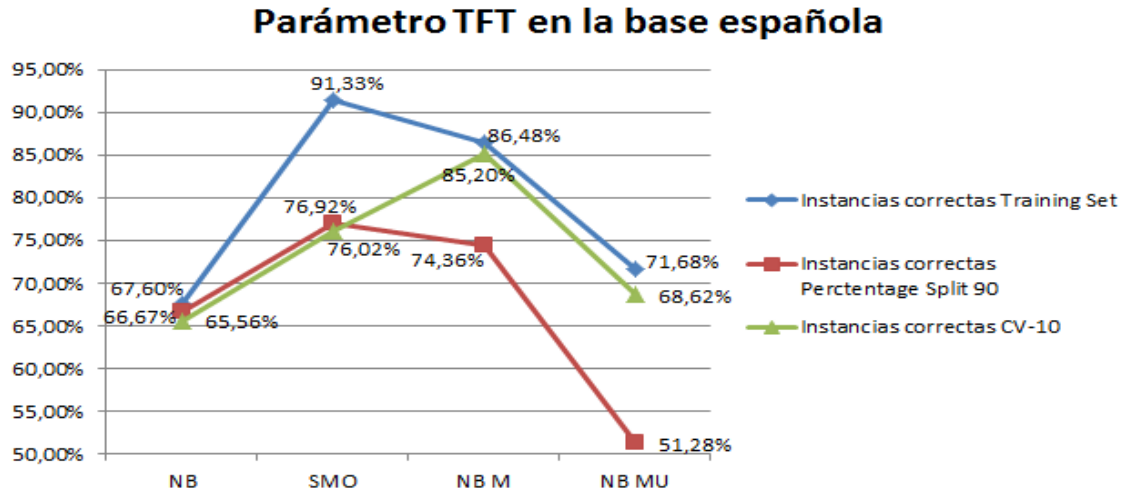


Figura 74: Resultados del preprocesamiento *wordtokenizer* y el parámetro TFT sobre los clasificadores, según el método de evaluación en la base española.

A la vista los resultados no son buenos, sólo para el caso de SMO sobre *Training Set* el resultado es mayor del 90%.

El resto de los casos tiene resultados muy pobres, y la influencia del parámetro TFT se ve clara en el caso de Naïve Bayes Multinomial Updateable que empeora los resultados del NB M, mientras que anteriormente se veía que se mantenían. Y en el caso del evaluador *Percentage Split 90%* lleva a bajarlo al 50%, un resultado inaceptable.

Si el preprocesamiento llevara este parámetro el clasificador sería el Naïve Bayes Multinomial, dado que para el evaluador CV-10 es el que mejor resultados obtiene.

4.2.2.11. Parámetro IDFT

Comprobamos ahora la influencia del parámetro IDFT sobre el *tokenizer* del filtro STWV de *wordtokenizer.*

Vemos el resultado en la Figura 75:

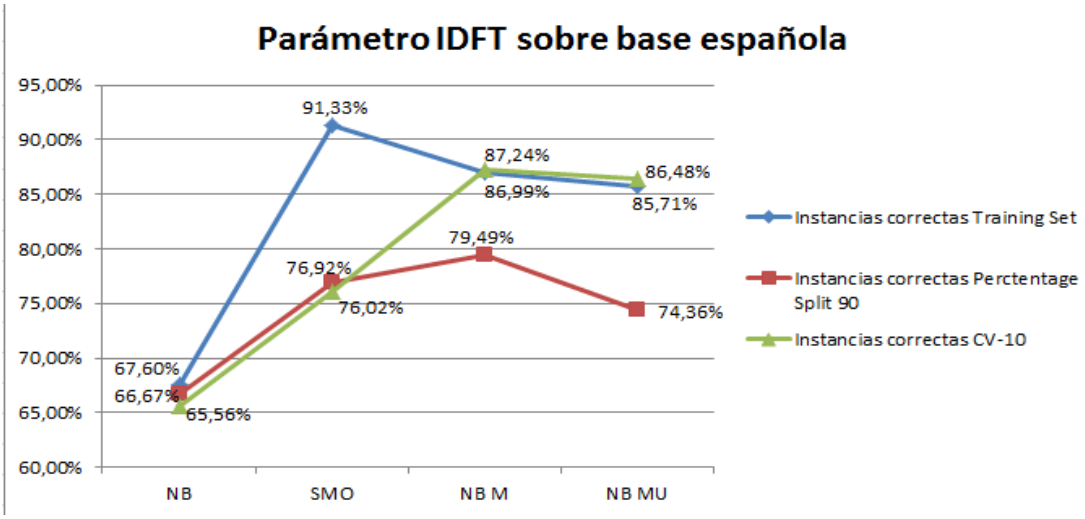


Figura 75: Resultados del preprocesamiento *wordtokenizer* y el parámetro IDFT sobre los clasificadores, según el método de evaluación en la base española.

La utilización de este parámetro empeora ligeramente el resultado del NB MU en relación con el NB M. Sin embargo mejora la actuación del evaluador de validación cruzada para estos dos clasificadores, aunque su resultado no llega al 90% en el caso de NB Multinomial se acerca.

Y como en la mayoría de los casos el clasificador NB es el que peor actúa y el SMO sobre el conjunto de entrenamiento el que mejor.

Tras el estudio de todos los preprocesamientos, en la mayoría el mejor resultado lo obtiene el NB Multinomial para el evaluador CV-10 que es más real. Y en todos los casos el SMO para el conjunto de entrenamiento.

4.3. Comparación de resultados en inglés y en español

A partir de los experimentos realizados, se ha visto que los resultados son mejores para la base de datos inglesa que para la española en todas las combinaciones. ¿A qué se puede deber?

Lo primero que cabe destacar son las dificultades para encontrar una base española adecuada, y eso ha implicado que al final se dispusiese de una base de datos pequeña. Con esto quiero decir que la base española cuenta con 392 instancias mientras que la inglesa tiene 1002, por lo que se puede ver que con menos instancias hay menos atributos que generar y por lo tanto un modelo menos preciso.

Por otro lado, gracias a que la base española es más pequeña se han podido realizar los experimentos con los n-gramas que contienen un máximo de 3 y un mínimo de 1, que han sido finalmente con los que mejores resultados se han obtenido.

Otro punto que sorprende es el caso de los *NGrams(3,3)*; hemos visto y comprobado que en la base inglesa, este *tokenizer* obtiene los mejores resultados sin lugar a dudas, mientras que en la española obtiene los peores resultados.

Esta diferencia puede ser debida a lo distintas que son estas lenguas. Los unigramas se construyen como se ha contado en la sección 2.1.1, dando el peso al n-grama según su probabilidad de aparición en un documento. La concatenación de palabras para que hagan secuencias con sentido se realiza según el número de repeticiones que tenga esa cadena. En inglés hay más combinaciones de tres palabras que se repiten y que tengan un significado por sí mismo, como por ejemplo, en la lista de tokens generados nos encontramos " *Of the best*", "*the only thing*", "*A bunch of*", "*Steals the show*", "*The worst movie*". Estas frases dan clara referencia de positividad o negatividad y se encuentran entre los 20 primeros tokens de la base inglesa. Sin embargo en la española encontramos: "Verla en 3d", "lo que se", "La película en", como las tres primeras de la lista de tokens, que no dan en ningún caso información específica sobre polaridad, ya que pueden tener connotaciones tanto positivas como negativas. También hay tokens como "no la recomiendo", que tienen un significado más concreto en cuanto a polaridad, pero son menos. Por otra parte, la diferencia de funcionamiento de los Trigramas en los dos idiomas, también puede deberse a la escasez de datos de entrenamiento en la base de datos española para modelar correctamente los n-gramas correspondientes. Y el hecho de utilizar la lista *StopWords* por defecto de WEKA que está en inglés, pero que se utilizó porque se hicieron experimentos sin ella y los resultados eran peores.

Entonces, ¿por qué se dan los mejores resultados en español con *NGrams(3,1)* si van a incluir todos los n-gramas con máximo y mínimo de tres y éstos no funcionaban bien? Por la sencilla razón que con el abanico de tokens de una, dos o tres palabras se generan un total de 1360 atributos con 392 instancias, que consigue que a la hora de generar los modelos, éstos sean mucho más precisos.

Respecto a los clasificadores queda claro que si se evalúa sobre el conjunto de entrenamiento el que hay que utilizar es el SMO en ambos casos, como se vio en los trabajos previos de la sección 2.3 los resultados del SVM eran mejores cuánto mayor era la base, y en este caso es cuando se evalúa sobre el conjunto de entrenamiento. Pero como se ha visto con otras formas de testado, más cercanas a la realidad, el SMO no generaliza bien y es preferible el uso de Naïve Bayes Multinomial que ofrece mejores prestaciones que el NB y el SMO en situaciones nuevas.

Para apoyar estos comentario se muestran las siguientes figuras (Figura 76 y Figura 77) en las que se ve que el comportamiento según preprocesamientos y clasificadores es más o menos parecido en ambas lenguas salvo para los n-gramas, especialmente para los de máximo y mínimo de tres.

Evaluados sobre el conjunto de entrenamiento la Figura 76:

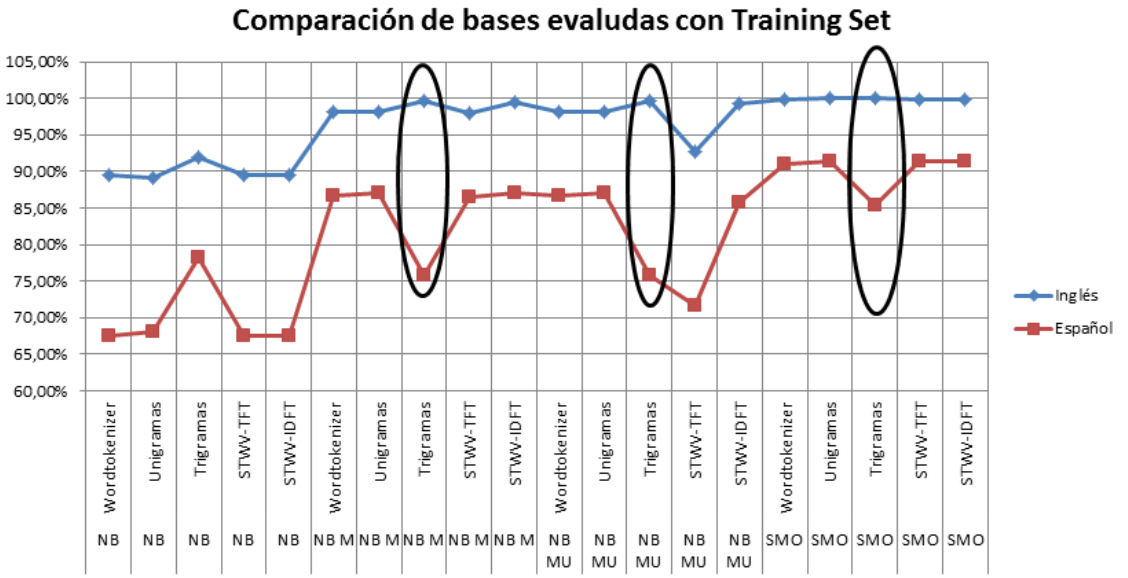


Figura 76: Comparación de las dos bases evaluadas sobre *Training Set.*

Evaluados con validación cruzada en 10 partes en la Figura 77:

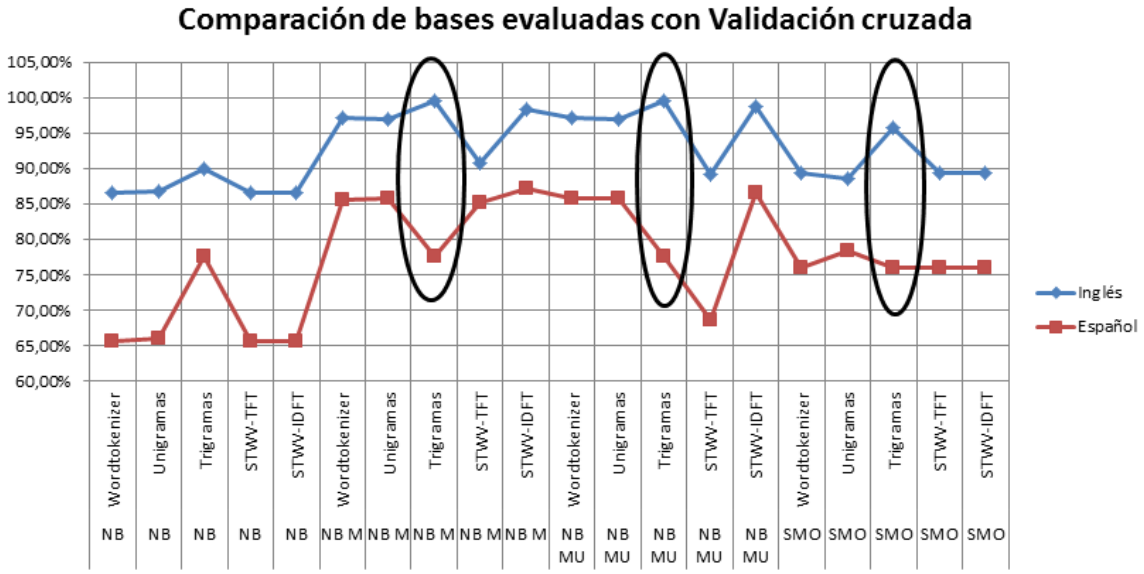


Figura 77: Comparación de las dos bases evaluadas con *Cross Validation 10 folds.*

Comparamos también cómo en los mejores casos reales, evaluados con validación cruzada, salen las matrices de confusión para ver si los errores se cometen de forma equilibrada o sólo en una clase.

En el caso de inglés se muestra el resultado de *NGrams(3,3)* clasificados con NB Multinomial:

```
  a   b   <-- classified as
501   0 |   a = POS
  5 496 |   b = NEG
```

Clasifica mejor las positivas pero casi perfectamente las negativas también.

En el caso de español se muestra el resultado de *NGrams(3,1)* clasificados con NB Multinomial:

```
   a   b   <-- classified as
 178  18 |   a = POS
  10 186 |   b = NEG
```

Clasifica peor las positivas que las negativas pero sin una gran diferencia.

Se pueden ver el resto de matrices de confusión en el anexo de la sección 8.

5. Conclusiones y trabajo futuro

5.1. Conclusiones

Tras todo el trabajo realizado analizando las bases de datos de textos consideradas, los distintos tipos de preprocesamiento de las mismas y los clasificadores utilizados, primero queda claro que los resultados son mucho mejores en la base de datos inglesa.

Como ya se ha comentado previamente, esta diferencia entre los resultados de inglés y español se debe básicamente al tamaño de las bases de datos, que es mucho menor en el caso del español.

Con respecto al preprocesamiento o extracción de atributos, es evidente que la simple división de los documentos en tokens de una sola palabra indicando nada más que los separadores, no es suficiente para obtener unos buenos resultados. Las dos alternativas más convenientes consisten en utilizar el *wordtokenizer* activando el parámetro IDFT o cambiar de *tokenizer* a n-gramas.

En concreto, el parámetro TFT es mejor no utilizarlo porque en general empeora los resultados o no produce cambios significativos en la tasa de clasificación. Por el contrario el IDFT mejora los resultados del Naïve Bayes Multinomial y el Naïve Bayes Multinomial Updateable para algunos preprocesamientos. Es por eso que dependiendo del tipo de preprocesado es recomendable su utilización. Como se comentó en la sección 2.1.7 este parámetro ayuda a mejorar el peso asignado a las palabras dentro del documento, lo que favorece los casos de Naïve Bayes Multinomial dónde se miden las probabilidades de las palabras según los documentos de la base de datos. También mejora más al Updateable que es la versión incremental del clasificador anterior.

Con respecto a los clasificadores, se ha observado que en la mayoría de los experimentos el NB ha dado los peores resultados. Hay que tener en cuenta que este clasificador es el más sencillo de los experimentados ya que crea sus modelos a partir de la media y desviación típica de los atributos. Por tanto, tiene competidores más complejos que consiguen mejores resultados, no sólo la evolución de éste, Naïve Bayes Multinomial, sino el SMO cuyo principio es distinto y se basa en SVM.

Con respecto a los clasificadores, el SMO es el que ofrece mejores prestaciones sobre el conjunto de entrenamiento casi independientemente del preprocesamiento aunque funciona mejor con los n-gramas. Sin embargo, el SMO no generaliza lo suficientemente bien ante textos nuevos, de forma que, en los experimentos más realistas y fiables (con validación cruzada) los mejores resultados se obtienen con el Naïve Bayes Multinomial.

Como resumen final para la base de datos española, los mejores resultados se han obtenido con un preprocesado en el que el organizador de tokens del filtro *StringToWordVector* es el *NGrams(3,1)* y el clasificador a utilizar el Naïve Bayes Multinomial.

Con este trabajo se concluye que la mejor combinación de atributos y clasificadores depende de la base de datos de origen que se va a estudiar, pero que, sin embargo, es posible dar algunas recomendaciones generales:

- Para una base de datos inglesa de tamaño adecuado, es aconsejable utilizar *NGrams(3,3)*. Aunque en este caso no ha podido probarse (en otros trabajos previos sí

se hizo), por la estructura de esta lengua, y viendo que los resultados de unigramas no conseguían los mejores resultados, es posible deducir que los trigramas son la mejor opción.
- Para una base de datos española es recomendable utilizar *NGrams(3,1)*.
- Para un escenario realista en el que los textos a clasificar sean distintos a los utilizados para la obtención de los modelos, es aconsejable utilizar el Naïve Bayes Multinomial, y si el preprocesamiento se basa en *wordtokenizer,* activar el parámetro IDFT.

5.2. Trabajo futuro

Como trabajo futuro, proponemos estudiar otras bases de datos en español que estén relacionadas con otros temas y comprobar si las conclusiones obtenidas de este trabajo también aplican.

Con respecto al proceso de extracción de atributos, sería interesante utilizar los parámetros del filtro STWV de *StopWords* donde se puede incluir una lista de palabras objetivas (que no impliquen sentimiento), pero que se repitan mucho, y así restringir los atributos de modo que, previsiblemente, se obtengan unos resultados más precisos, además de crear una en español para mejorar sus resultados. Por otra parte, el sistema también podría mejorar, cambiando el *Ranker* del filtro *AttributeSelection* de forma que se aumente su umbral para dejar fuera palabras que no tienen un peso significativo y probando con el evaluador *GainRatioAttributeEval* que sigue la línea del utilizado en estos experimentos pero puede que obtenga resultados interesantes.

Con los resultados obtenidos, una vez generado el modelo de polaridad, se puede crear un programa en Java en el que incluirlo y poder utilizarlo en aplicaciones concretas. Es una de las ventajas de WEKA, que proporciona la posibilidad de integrar los modelos en Java.

Una vez desarrollado el análisis de polaridad se puede extender a construir grados entre positivo y negativo, dando una escala de valores. Y como paso más avanzado definir reglas para emociones, como aburrimiento, alegría, hostilidad, etc…

En general cada vez más se hace necesario tener un discriminador de emociones para distintas aplicaciones y maneras de funcionamiento en la red. Hay distintos proyectos en los que se está utilizando, por ejemplo para redes sociales como Twitter [38] y Facebook, en el campo de e-learning [36] o en el de e-commerce [37] . Pero aún es necesario mejorar los métodos, y adaptarlos a otras lenguas.

6. Presupuesto

El cálculo del presupuesto es necesario en todo proyecto para reflejar el tiempo dedicado al mismo, el coste de las herramientas utilizadas y aplicarle un margen de beneficio.

Además del esfuerzo de ejecución de los experimentos, para la completa realización de este proyecto es necesario tener en cuenta unas tareas previas que fueron necesarias para realizarlo y que se han considerado para calcular su duración:

- Estudios de antecedentes y documentación: búsqueda de información relacionada con el proyecto, antecedentes, revisión de documentación útil, etc. Esta tarea es el inicio del proyecto pero continuará en todo su desarrollo.
- Tiempo dedicado al aprendizaje de la herramienta usada: WEKA.
- Realización de los experimentos.
- Preparación, análisis y presentación de los resultados obtenidos.
- Redacción de la memoria. El presupuesto de este trabajo se va a calcular en base a los costes ocasionados:

6.1. Costes del personal

En este apartado se van a contar los costes del proyectando y de la persona encargada de su supervisión.

- El proyectando, persona encargada de realizar los experimentos y sacar las conclusiones:

Nombres y apellidos	**Categoría profesional**	**Dedicación (persona/mes)**	**Coste (persona/mes)**	**Coste (euros)**
Lucía Plaza Sacarrera	Ingeniero Superior	12	1350	16200

- Costes de dirección del proyecto:

Descripción	**Coste/Euros**	**% dedicación**	**Dedicación (meses)**	**Coste imputable**
Dirección	10050	10	12	10050

6.2. Costes derivados del equipamiento utilizado

Ordenador personal del proyectando cuyo coste se detalla a continuación:

Descripción	**Coste/Euros**	**% dedicación**	**Dedicación (meses)**	**Periodo depreciación**	**Coste imputable**
Intel Core i5	500	100	12	60	100

Utilizado para la realización de los experimentos y la memoria del proyecto.

6.3. Costes de funcionamiento

Otros costes relacionados con la realización del proyecto:

- *Microsoft Office 2010:* Necesario para la organización de los datos obtenidos de los experimentos y el documento final de la memoria del proyecto.
- *WEKA:* Herramienta utilizada para la realización de los experimentos. Se puede descargar gratuitamente.
- *Tarifa ADSL:* Una conexión a Internet ha sido necesaria para la fase de documentación y la descarga de las bases de datos. El coste de la tarifa es de 36 €/mes utilizada durante 12 meses.
- *Sistema Operativo Windows:* el sistema operativo utilizado en el ordenador, no genera coste adicional por venir de serie.
- *Base de datos TOKENS:* Base de datos de textos en inglés. Se puede descargar gratuitamente,
- *Bases de datos en Español:* Las bases de datos que incluyen las opiniones de "Avatar", "La venganza de Don Mendo", "El Cavernícola" y "Por el placer de volver a verla".

Descripción	**Coste Imputable**
Microsoft Office 2010	60
WEKA	0
Tarifa de ADSL	432
Sistema Operativo Windows	0
Base de datos TOKENs	0
Bases de datos en español	0
Total	**492**

6.4. Resumen de los costes

Para el cálculo de los costes se ha tenido en cuenta una tasa de costes indirectos del 2.5%.

El coste total del proyecto ha sido:

Presupuesto costes totales	**Presupuesto costes**
Personal	16.200
Amortización	100
Subcontratas	
Costes funcionamiento	492
Costes indirectos	405
Costes dirección	10.050
Total	**27.247**

7. Referencias

[1] Bo Pang and Lillian Lee and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceeding, EMNLP '02 Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Volume 10, Pages 79-86. doi>10.3115/1118693.1118704

[2] Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume, pages 271–278. DOI 10.3115/1218955.1218990.

[3] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. Proceedings of ACL, pp. 115--124 .

[4] Erik Boiy and Marie-Francine Moens. 2008. A machine learning approach to sentiment analysis in multilingual Web texts. Information Retrieval, Volume 12, Issue 5, pp 526-558. DOI 10.1007/s10791-008-9070-z

[5] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze: An Introduction to Information Retrieval, pages 237-240. Cambridge University Press, 2009.

[6] F. Song and W. B. Croft (1999). "A General Language Model for Information Retrieval". *Research and Development in Information Retrieval*. pp. 279–280.

[7] Zhang, Harry.2004. "The Optimality of Naive Bayes". FLAIRS Conference 2004: pp. 562-567.

[8] George H. John and Pat Langley (1995). Estimating Continuous Distributions in Bayesian Classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338-345. Morgan Kaufmann, San Mateo.

[9] Andrew McCallum, and Kamal Nigam. 1998. "A comparison of event models for Naive Bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752.

[10] Kamal Nigam, John Lafferty, Andrew McCallum.1999. Using Maximum Entropy for text classification. IJCAI'99 Workshop on Information Filtering.

[11] Corinna Cortes, Vladimir Vapnik. 1995. Support Vector Machines. Machine Learning, 20, pp.273-297.

[12] Kai-Bo Duan1 and S. Sathiya Keerthi. 2005. Which Is the Best Multiclass SVM Method? An Empirical Study. Proceedings of the Sixth International Workshop on Multiple Classifier Systems.

[13] Drucker, Harris; Burges, Christopher J. C.; Kaufman, Linda; Smola, Alexander J.; and Vapnik, Vladimir N. (1997); "Support Vector Regression Machines", in *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155–161, MIT Press.

[14] Emmanuel Anguiano-Hernández. 2009. Naive Bayes Multinomial para Clasificación de Texto Usando un Esquema de Pesado por Clases.

[15] Steven J. DeRose. 1988. Grammatical Category Disambiguation By Statistical Optimization. Journal Computational Linguistics, Volume 14, Issue 1, Pages 31-39.

[16] Burr Settles. 2005. Active Learning Literature Survey. Computer Science Technical Report 1648. University of Wisconsin-Madison.

[17] Marco Bonzanini, Miguel Martinez-Alvarez, y Thomas Roelleke. 2012. Investigating the Use of Extractive Summarisation in Sentiment Classification. *2012 IIR* , p45-52

[18] Seungyeon Kim, Fuxin Li, Guy Lebanon, and Irfan Essa. 2012. Beyond Sentiment: The Manifold of Human Emotions. Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)

[19] Hassan Saif, Yulan He y Harith Alani. 2012. Alleviating Data Sparsity for Twitter Sentiment Analysis. Workshop: The 2nd Workshop on Making Sense of Microposts (#MSM2012): Big things come in small packages at World Wide Web (WWW) 2012, Lyon, France.

[20] D.Watson and A. Tellegen. Toward a consensual structure of mood. *Psychological bulletin*, 98(2):219–235, September 1985.

[21] Antonio Moreno Ortiz, Álvaro Pérez Pozo, Sergio Torres Sánchez. 2010. Sentitext: Sistema de análisis de sentimiento en español. Procesamiento del Lenguaje Natural 45: pp 297-298.

[22] L. Polanyi y A. Zaenen. 2006. Contextual Valence Shifters. In Computing Attitude and Affect in Text: Theory and Applications. The Information Retrieval Series Volume 20, pp 1-10.

[23] Antonio Moreno Ortiz, Francisco Pineda Castillo, Rodrigo Hidalgo García. 2011. Análisis de Valoraciones de Usuario de Hoteles con Sentitext: un sistema de análisis de sentimiento independiente del dominio. Procesamiento del lenguaje natural, volumen 45: pp 31-39.

[24] Javi Fernández, Ester Boldrini, José Manuel Gómez, Patricio Martínez-Barco. 2011. Análisis de Sentimientos y Minería de Opiniones: el corpus EmotiBlog. Procesamiento del lenguaje natural, volumen 47.

[25] Rajaraman, A.; Ullman, J. D. (2011). Data Mining. Mining of Massive Datasets. pp. 1–17. doi:10.1017/CBO9781139058452.002. ISBN 9781139058452.

[26] Manning, C. D.; Raghavan, P.; Schutze, H. (2008). "Introduction to Information Retrieval". p. 100. doi:10.1017/CBO9780511809071.007. ISBN 9780511809071.

[27] Robertson, Stephen. Understanding inverse document frequency: On theoretical arguments for IDF. Journal of Documentation 60 (5): 503–520. doi:10.1108/00220410410560582.

[28] Jasmina Novakovic. 2009. "Using Information Gain Attribute Evaluation to Classify Sonar Targets".

[29] Platt, John (1998), Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, CiteSeerX: 10.1.1.43.4376

[30] http://www.cs.waikato.ac.nz/ml/weka/

[31] Platt, John; Cristianini, Nello; Shawne-Taylor, John. 2000. Large Margin DAGs for Multiclass Classification. In Advances in Neural Information Processing Systems (NIPS), vol. 12.

[32] Dietterich, Thomas G.; Bakiri, Ghulum. 1995. "Solving Multiclass Learning Problems via Error-Correcting Output Codes". Journal of Artificial Intelligence Research, Volume 2 Issue 1, Pages 263-286.

[33] J.A.K. Suykens; J. Vandewalle. 1999. "Least Squares Support Vector Machine Classifiers". Neural Processing Letters, Volume 9, Issue 3, pp 293-300.

[34] Kleinberg, Jon; Tardos, Éva. 2002. "Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov Random fields". *Journal of the ACM*, 49(5):616-639.

[35] Rainbow System: http://www.cs.cmu.edu/~mccallum/bow/rainbow/. StopList de WEKA.

[36] Ortigosa, Alvaro; Martín, Jose M.; Carro, Rosa M. 2014. "Sentiment Analysis in Facebook and its application to e-learning". Computers in Human behaviour, Volume 31, Pages 527-541. DOI: 10.1016/j.chb.2013.05.024

[37] G. Vinodhini; RM Chandrasekaran. 2014. "Measuring the quality of hybrid opinion mining model for e-commerce application". Measurement, Volume 55, Pages 101-109. DOI: 10.1016/j.measurement.2014.04.033.

[38] Montejo-Ráez, Arturo; Martinez-Cámara, Eugenio; Matín-Valdivia, M.Teresa; Ureña-López, L.Alfonso. 2014. "Ranked WordNet graph for Sentiment Polarity Classification inTwitter". Computer Speech & Language,Volume 28, Issue 1, Pages 93-107. DOI: 10.1016/j.csl.2013.04.001.

8. Anexo: Matrices de confusión

Este anexo contiene las matrices de confusión generadas en todos los casos junto a su precisión de clase:

8.1. Base inglesa

8.1.1. Naïve Bayes

Podemos ver cómo fue el resultado de clasificación según la clase con las matrices de confusión.

El parámetro de precisión según la clase también es un parámetro que se obtiene al ejecutar los test. En el conjunto de entrenamiento se clasifican mejor las instancias positivas.

	Training Set	Precisión NEG	Precisión POS
Wordtokenizer	a b <-- classified as 453 48 | a = POS 58 443 | b = NEG	0,886	0,902
Unigramas	a b <-- classified as 452 49 | a = POS 59 442 | b = NEG	0,885	0,9
Trigramas	a b <-- classified as 466 35 | a = POS 45 456 | b = NEG	0,912	0,929
TFT	a b <-- classified as 453 48 | a = POS 58 443 | b = NEG	0,886	0,902
IDFT	a b <-- classified as 453 48 | a = POS 58 443 | b = NEG	0,886	0,902

Tabla 20: Matrices de confusión para el clasificador NB y evaluado sobre el conjunto de entrenamiento en inglés.

Veamos si en el caso más real de *Perctentage Split 90%* ocurre lo mismo.

En la tabla a continuación vemos las matrices de confusión y en este caso se clasifican mejor las negativas, salvo para el caso de n-gramas (3,3). Vemos también que el número absoluto de instancias es mucho menor debido a que es sólo el 10% del conjunto entero.

	Percentage Split 90%	Precisión NEG	Precisión POS
Wordtokenizer	a b <-- classified as 39 8 | a = POS 4 49 | b = NEG	0,906	0,86

Unigramas	a b <-- classified as 39 8 | a = POS 3 50 | b = NEG	0,929	0,862
Trigramas	a b <-- classified as 45 2 | a = POS 4 49 | b = NEG	0,918	0,961
TFT	a b <-- classified as 39 8 | a = POS 4 49 | b = NEG	0,907	0,86
IDFT	a b <-- classified as 39 8 | a = POS 4 49 | b = NEG	0,907	0,86

Tabla 21: Matrices de confusión para el clasificador NB y evaluado sobre el *PercentageSplit 90%* en inglés.

Por último vemos los resultados de las matrices de confusión cuando se evalúa con Validación cruzada de diez partes.

	Cross validation 10 folds	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 445 56 | a = POS 77 424 | b = NEG	0,857	0,873
Unigramas	a b <-- classified as 480 21 | a = POS 10 491 | b = NEG	0,852	0,883
Trigramas	a b <-- classified as 458 43 | a = POS 58 443 | b = NEG	0,888	0,912
TFT	a b <-- classified as 482 19 | a = POS 13 488 | b = NEG	0,857	0,873
IDFT	a b <-- classified as 439 62 | a = POS 73 428 | b = NEG	0,857	0,873

Tabla 22: Matrices de confusión para el clasificador NB y evaluado sobre el *Cross Validation 10 folds* en inglés.

8.1.2. Naïve Bayes Multinomial

Veamos ahora cómo han salido las matrices de confusión en este caso.

Como en la sección anterior se van a comentar según al método de evaluación utilizado.

	Training Set	Precisión NEG	Precisión POS
Wordtokenizer	a b <-- classified as 490 11 | a = POS 7 494 | b = NEG	0,986	0,978
Unigramas	a b <-- classified as 489 12 | a = POS 7 494 | b = NEG	0,986	0,976
Trigramas	a b <-- classified as 501 0 | a = POS 4 497 | b = NEG	0,992	1
TFT	a b <-- classified as 490 11 | a = POS 9 492 | b = NEG	0,982	0,978
IDFT	a b <-- classified as 498 3 | a = POS 3 498 | b = NEG	0,994	0,994

Tabla 23: Matrices de confusión para el clasificador NB Multinomial y evaluado sobre el conjunto de entrenamiento en inglés.

Seguimos confirmando los mejores resultados para el caso de Trigramas, en los que para los tokens positivos no hay error. Y vemos también cómo el parámetro IDFT mejora los resultados del *wordtokenizer*, siendo el que segunda mejor actuación tiene.

	Percentage Split 90%	Precisión NEG	Precisión POS
Wordtokenizer	a b <-- classified as 45 2 | a = POS 2 51 | b = NEG	0,957	0,962
Unigramas	a b <-- classified as 45 2 | a = POS 2 51 | b = NEG	0,957	0,962
Trigramas	a b <-- classified as 47 0 | a = POS 0 53 | b = NEG	1	1
TFT	a b <-- classified as 45 2 | a = POS 2 51 | b = NEG	0,957	0,962
IDFT	a b <-- classified as 47 0 | a = POS 1 52 | b = NEG	0,979	1

Tabla 24: Matrices de confusión para el clasificador NB Multinomial y evaluado sobre el *percentage Split 90%* en inglés.

En este caso los resultados son los mismos e incluso mejores al alcanzar el 100% de acierto en más situaciones.

	Cross Validation 10 folds	Precisión NEG	Precisión POS
Wordtokenizer	a b <-- classified as 484 17 | a = POS 11 490 | b = NEG	0,978	0,966
Unigramas	a b <-- classified as	0,98	0,959

	480 21 | a = POS 10 491 | b = NEG		
Trigramas	a b <-- classified as 501 0 | a = POS 5 496 | b = NEG	0,99	1
TFT	a b <-- classified as 482 19 | a = POS 13 488 | b = NEG	0,974	0,963
IDFT	a b <-- classified as 491 10 | a = POS 6 495 | b = NEG	0,988	0,98

Tabla 25: Matrices de confusión para el clasificador NB Multinomial y evaluado sobre validación cruzada de 10 partes en inglés

Por último en el caso de validación cruzada se mantienen los resultados que en los casos anteriores si bien su precisión es ligeramente más baja.

8.1.3. Naïve Bayes Multinomial Updateable

	Training Set	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 490 11 | a = POS 7 494 | b = NEG	0,986	0,978
Unigramas	a b <-- classified as 489 12 | a = POS 7 494 | b = NEG	0,986	0,976
Trigramas	a b <-- classified as 501 0 | a = POS 4 497 | b = NEG	0,992	1
TFT	a b <-- classified as 501 0 | a = POS 74 427 | b = NEG	1	0.871
IDFT	a b <-- classified as 499 2 | a = POS 6 495 | b = NEG	0,988	0,996

Tabla 26: Matrices de confusión para el clasificador NB Multinomial Updateable y evaluado sobre el conjunto de entrenamiento en inglés.

Seguimos confirmando los mejores resultados para el caso de Trigramas, en los que para los tokens positivos no hay error. Y vemos también cómo el parámetro IDFT mejora los resultados del *wordtokenizer*, siendo el que segunda mejor actuación tiene. Los mismos valores en los tres primeros casos en NB Multinomial, pero distintos en los dos últimos.

	Percentage Split 90%	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 45 2 | a = POS 2 51 | b = NEG	0,957	0,962
Unigramas	a b <-- classified as 45 2 | a = POS 2 51 | b = NEG	0,957	0,962

Trigramas	a b <-- classified as 47 0 | a = POS 0 53 | b = NEG	1	1
TFT	a b <-- classified as 47 0 | a = POS 9 44 | b = NEG	0,839	1
IDFT	a b <-- classified as 47 0 | a = POS 1 52 | b = NEG	0,979	1

Tabla 27: Matrices de confusión para el clasificador NB Multinomial Updateable y evaluado sobre *percentage Split 90%* en inglés.

En este caso los resultados son los mismos e incluso mejores al alcanzar el 100% de acierto en más situaciones.

	Cross Validation 10 folds	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 484 17 | a = POS 11 490 | b = NEG	0,978	0,966
Unigramas	a b <-- classified as 480 21 | a = POS 10 491 | b = NEG	0,98	0,959
Trigramas	a b <-- classified as 501 0 | a = POS 5 496 | b = NEG	0,99	1
TFT	a b <-- classified as 501 0 | a = POS 108 393 | b = NEG	0,823	1
IDFT	a b <-- classified as 497 4 | a = POS 9 492 | b = NEG	0,982	0.992

Tabla 28: Matrices de confusión para el clasificador NB Multinomial Updateable y evaluado sobre la validación cruzada con 10 partes en inglés

En el caso del parámetro TFT cabe destacar la gran diferencia entre las clases positiva y negativa donde una clasifica perfectamente, pero en el otro caso lo hace bastante mal.

8.1.4. SMO

Repasemos ahora las matrices de confusión obtenidas.

	Training Set	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 500 1 | a = POS 0 501 | b = NEG	1	0.998
Unigramas	a b <-- classified as 501 0 | a = POS 0 501 | b = NEG	1	1
Trigramas	a b <-- classified as 501 0 | a = POS 0 501 | b = NEG	1	1
TFT	a b <-- classified as	1	0.998

	500 1 | a = POS 0 501 | b = NEG		
IDFT	a b <-- classified as 500 1 | a = POS 0 501 | b = NEG	1	0.998

Tabla 29: Matrices de confusión para el clasificador SMO y evaluado sobre el conjunto de entrenamiento en inglés.

En este caso, como se viene viendo, no hay casi errores.

	Percentage Split 90%	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 40 7 | a = POS 3 50 | b = NEG	0,93	0,877
Unigramas	a b <-- classified as 39 8 | a = POS 3 50 | b = NEG	0,929	0,862
Trigramas	a b <-- classified as 45 2 | a = POS 1 52 | b = NEG	0,978	0,963
TFT	a b <-- classified as 40 7 | a = POS 3 50 | b = NEG	0,93	0,877
IDFT	a b <-- classified as 40 7 | a = POS 3 50 | b = NEG	0,93	0,877

Tabla 30: Matrices de confusión para el clasificador SMO y evaluado sobre el *percentage Split 90%* en inglés.

En general se confunden más las palabras positivas, habiendo más falsos positivos que negativos.

	Cross Validation 10 folds	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 442 59 | a = POS 47 454 | b = NEG	0,904	0,885
Unigramas	a b <-- classified as 445 56 | a = POS 59 442 | b = NEG	0,883	0,888
Trigramas	a b <-- classified as 469 32 | a = POS 10 491 | b = NEG	0,979	0,939
TFT	a b <-- classified as 442 59 | a = POS 47 454 | b = NEG	0,904	0,885
IDFT	a b <-- classified as 442 59 | a = POS 47 454 | b = NEG	0,904	0,885

Tabla 31: Matrices de confusión para el clasificador SMO y evaluado sobre la validación cruzada con 10 partes en inglés

En este caso se clasifican mejor las negativas salvo en el caso de *NGrams(1,1)* donde la precisión es prácticamente la misma pero peor para las negativas.

8.2. Base Española

8.2.1. Naïve Bayes

Comprobamos los resultados de las matrices de confusión para analizar mejor el comportamiento según la clase.

	Training Set	Precisión NEG	Precisión POS
Wordtokenizer	a b <-- classified as 108 88 | a = POS 39 157 | b = NEG	0,735	0,641
Unigramas	a b <-- classified as 110 86 | a = POS 39 157 | b = NEG	0,738	0,646
N-gramas (3,1)	a b <-- classified as 123 73 | a = POS 39 157 | b = NEG	0,759	0,683
Trigramas	a b <-- classified as 121 75 | a = POS 10 186 | b = NEG	0,924	0,713
TFT	a b <-- classified as 108 88 | a = POS 39 157 | b = NEG	0,735	0,641
IDFT	a b <-- classified as 108 88 | a = POS 39 157 | b = NEG	0,735	0,641

Tabla 32: Matrices de confusión para el clasificador NB y evaluado sobre el conjunto de entrenamiento en español.

Clasifica un poco peor la clase positiva, aunque la negativa no obtiene muy buenos resultados.

Vemos ahora aplicando el evaluador de *Percentage Split 90%.*

	Percentage Split	Precisión NEG	Precisión POS
Wordtokenizer	a b <-- classified as 6 8 | a = POS 5 20 | b = NEG	0,545	0,714
Unigramas	a b <-- classified as 6 8 | a = POS 5 20 | b = NEG	0,545	0,714
N-gramas (3,1)	a b <-- classified as 7 7 | a = POS 6 19 | b = NEG	0,538	0,731
Trigramas	a b <-- classified as 8 6 | a = POS 0 25 | b = NEG	1	0,806
TFT	a b <-- classified as 6 8 | a = POS 5 20 | b = NEG	0,545	0,714

IDFT	a b <-- classified as 6 8 | a = POS 5 20 | b = NEG	0,545	0,714

Tabla 33: Matrices de confusión para el clasificador NB y evaluado sobre *Percentage Split* en español.

Con este evaluador al tener una muestra tan pequeña los resultados son mucho peores, siendo en los tres casos del *wordtokenizer* el número de falsos positivos mayor que los aciertos en la clase Positivo, viéndose que el error de los *Ngrams(3,3)* se produce sólo en la clase POS.

Estudiamos el caso de la validación cruzada:

	Cross Validation 10 folds	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 107 89 | a = POS 46 150 | b = NEG	0,699	0,628
Unigramas	a b <-- classified as 109 87 | a = POS 46 150 | b = NEG	0,703	0,633
N-gramas (3,1)	a b <-- classified as 118 78 | a = POS 41 155 | b = NEG	0,742	0,665
Trigramas	a b <-- classified as 118 78 | a = POS 10 186 | b = NEG	0,922	0,705
TFT	a b <-- classified as 107 89 | a = POS 46 150 | b = NEG	0,699	0,628
IDFT	a b <-- classified as 108 88 | a = POS 39 157 | b = NEG	0,699	0,628

Tabla 34: Matrices de confusión para el clasificador NB y evaluado sobre el validación cruzada de 10 partes en español.

Para el caso de la validación cruzada ocurre como en el de *Percentage Split 90%* que en general clasifica mal la clase positiva y en el caso de *NGrams(3,3)* el error prácticamente sólo es debido a ésta.

8.2.2. Naïve Bayes Multinomial

Estudiamos ahora las matrices de confusión para ver el comportamiento por clase:

	Training Set	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 187 9 | a = POS 43 153 | b = NEG	0,813	0,944
Unigramas	a b <-- classified as 188 8 | a = POS 43 153 | b = NEG	0,814	0,95
N-gramas (3,1)	a b <-- classified as 189 7 | a = POS 16 180 | b = NEG	0,922	0,963
Trigramas	a b <-- classified as	0,674	1

	196 0 | a = POS 95 101 | b = NEG		
TFT	a b <-- classified as 188 8 | a = POS 45 151 | b = NEG	0,807	0,95
IDFT	a b <-- classified as 190 6 | a = POS 50 146 | b = NEG	0,822	0,934

Tabla 35: Matrices de confusión para el clasificador NB Multinomial y evaluado sobre el conjunto de entrenamiento en español.

Para el caso de los trigramas falla casi el 50% de las veces que la clase es negativa. En este caso es exagerado, pero en general clasifica mejor la positiva.

Vemos ahora aplicando el evaluador de *Percerntage Split 90%.*

	Percentage Split	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 12 2 | a = POS 8 17 | b = NEG	0,6	0,895
Unigramas	a b <-- classified as 13 1 | a = POS 8 17 | b = NEG	0,619	0,944
N-gramas (3,1)	a b <-- classified as 13 1 | a = POS 5 20 | b = NEG	0,722	0,952
Trigramas	a b <-- classified as 14 0 | a = POS 14 11 | b = NEG	0,5	1
TFT	a b <-- classified as 13 1 | a = POS 9 16 | b = NEG	0,591	0,941
IDFT	a b <-- classified as 13 1 | a = POS 9 16 | b = NEG	0,667	0,905

Tabla 36: Matrices de confusión para el clasificador NB Multinomial y evaluado sobre *Percentage Split* en español.

Este caso como en el conjunto de entrenamiento, para Trigramas no clasifica las instancias negativas bien. Y en general mejor los positivos.

Estudiamos ahora el caso de la validación cruzada:

	Cross Validation 10 folds	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 175 21 | a = POS 35 161 | b = NEG	0,833	0,88
Unigramas	a b <-- classified as 175 21 | a = POS 35 161 | b = NEG	0,833	0,885
N-gramas	a b <-- classified as	0,947	0,912

(3,1)	178 18 | a = POS 10 186 | b = NEG		
Trigramas	a b <-- classified as 166 30 | a = POS 58 138 | b = NEG	0,741	0,821
TFT	a b <-- classified as 176 20 | a = POS 38 158 | b = NEG	0,822	0,888
IDFT	a b <-- classified as 181 15 | a = POS 38 158 | b = NEG	0,865	0,88

Tabla 37: Matrices de confusión para el clasificador NB Multinomial y evaluado sobre el validación cruzada de 10 partes en español.

Un poco más suave la diferencia en *NGrams(3,3)* pero los positivos mejor en general.

8.2.3. Naïve Bayes Multinomial Updateable

Estudiamos ahora las matrices de confusión para ver el comportamiento por clase.

Primero sobre el conjunto de entrenamiento.

	Training Set	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 187 9 | a = POS 43 153 | b = NEG	0,813	0,944
Unigramas	a b <-- classified as 188 8 | a = POS 43 153 | b = NEG	0,814	0,95
N-gramas (3,1)	a b <-- classified as 189 7 | a = POS 16 180 | b = NEG	0,922	0,963
Trigramas	a b <-- classified as 196 0 | a = POS 95 101 | b = NEG	0,674	1
TFT	a b <-- classified as 196 0 | a = POS 111 85 | b = NEG	0,638	1
IDFT	a b <-- classified as 190 6 | a = POS 50 146 | b = NEG	0,792	0,961

Tabla 38: Matrices de confusión para el clasificador NB Multinomial Updateable y evaluado sobre el conjunto de entrenamiento en español.

En este caso el parámetro TFT consigue clasificar perfectamente la clase positiva, pero se equivoca más en la negativa, teniendo más instancias en falso negativo que en negativas.

Vemos ahora aplicando el evaluador de *Percerntage Split 90%.*

	Percentage Split	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 12 2 | a = POS	0,6	0,895

	8 17 | b = NEG		
Unigramas	a b <-- classified as 13 1 | a = POS 8 17 | b = NEG	0,619	0,944
N-gramas (3,1)	a b <-- classified as 13 1 | a = POS 5 20 | b = NEG	0,722	0,952
Trigramas	a b <-- classified as 14 0 | a = POS 14 11 | b = NEG	0,5	1
TFT	a b <-- classified as 14 0 | a = POS 19 6 | b = NEG	0,424	1
IDFT	a b <-- classified as 13 1 | a = POS 9 16 | b = NEG	0,591	0,941

Tabla 39: Matrices de confusión para el clasificador NB Multinomial Updateable y evaluado sobre *Percentage Split* en español.

Estudiamos ahora el caso de la validación cruzada:

	Cross Validation 10 folds	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 175 21 | a = POS 35 161 | b = NEG	0,833	0,885
Unigramas	a b <-- classified as 175 21 | a = POS 35 161 | b = NEG	0,833	0,885
N-gramas (3,1)	a b <-- classified as 178 18 | a = POS 10 186 | b = NEG	0,947	0,912
Trigramas	a b <-- classified as 166 30 | a = POS 58 138 | b = NEG	0,741	0,821
TFT	a b <-- classified as 187 9 | a = POS 114 82 | b = NEG	0,621	0,901
IDFT	a b <-- classified as 181 15 | a = POS 38 158 | b = NEG	0,826	0,913

Tabla 40: Matrices de confusión para el clasificador NB Multinomial Updateable y evaluado sobre el validación cruzada de 10 partes en español.

En todos los evaluadores, el TFT produce un mayor número de confusiones de la clase negativa.

8.2.4. SMO

Estudiamos ahora las matrices de confusión para ver el comportamiento por clase:

	Training Set	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 164 32 | a = POS	0,982	0,858

	3 193 | b = NEG		
Unigramas	a b <-- classified as 165 31 | a = POS 3 193 | b = NEG	0,982	0,862
N-gramas (3,1)	a b <-- classified as 178 18 | a = POS 0 196 | b = NEG	1	0,916
Trigramas	a b <-- classified as 139 57 | a = POS 0 196 | b = NEG	1	0,775
TFT	a b <-- classified as 165 31 | a = POS 3 193 | b = NEG	0,982	0,862
IDFT	a b <-- classified as 165 31 | a = POS 3 193 | b = NEG	0,982	0,962

Tabla 41: Matrices de confusión para el clasificador SMO y evaluado sobre el conjunto de entrenamiento en español.

En contra de los casos anteriores, el SMO clasifica mejor las instancias de la clase negativa.

Vemos ahora aplicando el evaluador de *Percentage Split 90%.*

	Percentage Split	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 7 7 | a = POS 2 23 | b = NEG	0,778	0,767
Unigramas	a b <-- classified as 7 7 | a = POS 1 24 | b = NEG	0,875	0,774
N-gramas (3,1)	a b <-- classified as 8 6 | a = POS 1 24 | b = NEG	0,889	0,8
Trigramas	a b <-- classified as 7 7 | a = POS 0 25 | b = NEG	1	0,781
TFT	a b <-- classified as 7 7 | a = POS 2 23 | b = NEG	0,778	0,767
IDFT	a b <-- classified as 7 7 | a = POS 2 23 | b = NEG	0,778	0,767

Tabla 42: Matrices de confusión para el clasificador SMO y evaluado sobre *Percentage Split* en español.

Estudiamos ahora el caso de la validación cruzada:

	Cross Validation 10 folds	**Precisión NEG**	**Precisión POS**
Wordtokenizer	a b <-- classified as 124 72 | a = POS 22 174 | b = NEG	0,849	0,707
Unigramas	a b <-- classified as	0,863	0,732

	132 64 | a = POS 21 175 | b = NEG		
N-gramas (3,1)	a b <-- classified as 131 65 | a = POS 16 180 | b = NEG	0,891	0,735
Trigramas	a b <-- classified as 103 93 | a = POS 1 195 | b = NEG	0,99	0,667
TFT	a b <-- classified as 124 72 | a = POS 22 174 | b = NEG	0,849	0,707
IDFT	a b <-- classified as 124 72 | a = POS 22 174 | b = NEG	0,849	0,707

Tabla 43: Matrices de confusión para el clasificador SMO y evaluado sobre el validación cruzada de 10 partes en español.

En este caso, se clasifican mejor las instancias negativas.

9. Índice de figuras

10. Glosario de acrónimos

ARFF: Attribute Relation File Format

AS: Attribute Selection

CSV: Comma-Separated Values

CV: Cross Validation

CV-10: Cross Validation 10 folds

DAG SVM: Directed Acyclic Graph SVM

IDFT: Inverse Document Frequency Transform

IG: Information Gain

JST: Joint Sentiment-Topic

KFF: Kernel farthest first

KKT: Karush–Kuhn–Tucker

LS-SVM: Least Squares Support Vector Machine

MAP: Máxima A Posteriori

ME: Máxima Entropía

MNB: Multinomial Naïve Bayes

NB: Naïve Bayes

NB M: Naïve Bayes Multinomial

NB MU: Naïve Bayes Multinomial Updateable

POS: Parts of Speech

PSP: Positive Sentence Percentage

QP: Quadratic Programming

RS: Random Sampling

SC: Simple Classifier

SMO: Sequential Minimal Optimization

STWV: String To Word Vector

SVM: Support Vector Machines

SVM-OVA: SVM - one versus all

SVM-OVM: SVM - one versus many

SVR: Support Vector Regression

TFT: Term Frequency Transform

US: Uncertainty Samples