TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Aplicación de Técnicas de Recuperación de Información y Aprendizaje Automático a la Minería de Opiniones

Estudante: Manuel Corujo Muíña

Dirección: Miguel Ángel Alonso Pardo

Jesús Vilares Ferro

A Coruña, septiembre de 2022.

A mi familia y amigos por su apoyo incondicional

Agradecimientos

En primer lugar me gustaría agradecer a los directores de este proyecto, Jesús y Miguel, ya que sin su ayuda no sería posible la realización de este trabajo. También agradecer a mis padres, a mi abuela y a mi hermano, que siempre han estado ahí para lo que necesitase. Por último, a mis amigos, gracias por todos los buenos ratos.

Resumen

La Minería de Opiniones, también conocida como Análisis de Sentimientos, se dedica al estudio de opiniones y sentimientos expresados en textos. Se encuentra enmarcada dentro del área de estudio del Procesamiento de Lenguaje Natural. Este proyecto en concreto consiste en identificar la polaridad (positiva o negativa) de un conjunto de textos extraídos de una red social, *Twitter*.

Las investigaciones en este campo han aumentado en los últimos años gracias a la mayor disponibilidad de recursos de evaluación con los que se puede trabajar. Así, actualmente resulta sencillo encontrar multitud de textos, independientemente de la temática. El hecho de poder filtrar los tuits en base a rangos de edad, geografía o hashtag (etiquetas señaladas con #), permite la realización de estudios poblacionales, de especial interés en el ámbito de, por ejemplo, la política y el marketing, al hacer posible conocer de forma automática la opinión que genera un producto entre la comunidad de usuarios.

Al contrario que en otras aproximaciones más clásicas, en este trabajo no se utilizarán las palabras de los textos como atributos, sino que tan solo se utilizarán un conjunto de atributos derivados del ranking producido por un motor de búsqueda en respuesta a una consulta, donde esta consulta es el texto cuya polaridad queremos conocer. De este modo, el funcionamiento del sistema es el siguiente: En primer lugar, se usa un motor de búsqueda para, a partir de un conjunto de textos (tuits) cuya polaridad es ya conocida, construir un índice. Con el mismo motor de búsqueda y utilizando el texto que se quiere clasificar como consulta, lanzamos esta contra el motor de búsqueda, lo que devolverá un ranking con los tuits del índice más similares a aquel a clasificar. A partir de este ranking se extraen una serie de atributos que serán los que posteriormente utilice el clasificador para determinar la polaridad del texto. Como clasificadores se han utilizado distintos algoritmos de aprendizaje supervisado, como Máquinas de Soporte Vectorial, árboles de decisión o Naïve-Bayes.

Abstract

Opinion Mining, also known as Sentiment Analysis, is devoted to the study of opinions and emotions expressed in texts. It is framed within the study area of the Natural Language Processing. This particular project consists of the identification of the polarity (positive or negative) of a sample of texts extracted from a social network, namely *Twitter*.

Research in this field has been increasing in recent years due to the growing number of texts that can be analysed as the use of social networks has expanded. Thus, it is currently easy to find a large number of texts, regardless of the subject matter. Due to the possibility of

being able to filter tweets based on age ranges, geography or hashtag (labels marked with #), among others, these investigations are highly useful for fields such as politics. Furthermore, said research can also prove to be very convenient in the business world, as they allow to automatically determine the opinion generated by a product among the public.

This paper will not consider the words in the texts as attributes, instead only 24 attributes derived from the ranking produced by a Search Engine in response to a consulta will be employed. This consulta is the text to be classified. As such, the system works as follows: A SE is used to build an index based on a set of texts. With the same SE and using the text to classify as a consulta, the index created is consulted, which will generate a ranking of the tweets in the index that are most similar to the consulta. From this ranking, 24 attributes are extracted, which will later be the ones that the classifier uses to determine the polarity of the text. Different supervised learning algorithms have been used as classifiers, such as Support Vector Machines, decision trees or Naïve-Bayes.

Palabras clave:

- Minería de Opiniones
- Recuperación de Información
- Aprendizaje Automático
- Procesamiento de Lenguaje Natural

Keywords:

- Opinion Mining
- Information Retrieval
- Machine Learning
- Natural Language Processing

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Desde la aparición de las primeras redes sociales a finales de los 90 y principios de los 2000, su popularidad no ha dejado de aumentar, como se ve en la Figura 1.1 (página 1).

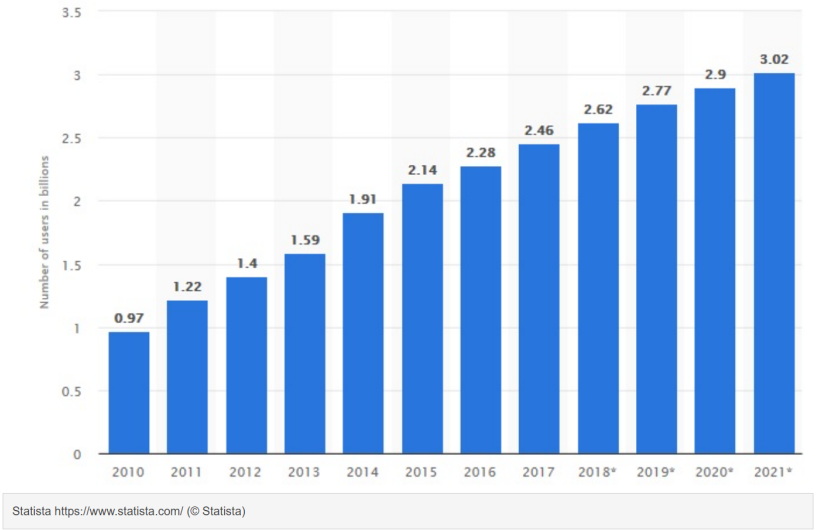

Figura 1.1: Número de usuarios de redes sociales en el mundo entre 2010 y 2021.

Desde entonces han surgido redes sociales de todo tipo: Generalistas, orientadas a conocer gente nueva, a la fotografía o de *microblogging*, por ejemplo. En este último segmento destaca desde 2006, su año de lanzamiento, *Twitter*. Esta es una red social de *microblogging* que, a pesar de nunca haber sido la red social más popular, lleva más de 15 años situándose entre las más utilizadas. En ella los usuarios pueden publicar pequeños textos, llamados *tuits*, de un máximo de 280 caracteres (aunque hasta 2017 tan solo eran 140). Por la naturaleza de sus publicaciones (textos cortos) y el gran número de ellas (como se explica en [1], en agosto de 2022 se publican 6.000 tuits por segundo, lo que equivale a 500 millones al día) Twitter es la red social objeto de mayor estudio. Uno de esos campos de estudio es la llamada Minería de

Opiniones (MO) que, como se verá posteriormente más en detalle en la Sección 2.1 (página 3), permite analizar qué tipo de opiniones, positivas o negativas, contiene un texto. Por ejemplo, en [2], se analizan tuits para determinar la opinión de los usuarios sobre 16 marcas.

1.1 Motivación

La intención en este trabajo es probar una aproximación a la Minería de Opiniones más eficiente que las habituales, y que de esta forma permita trabajar con un mayor volumen de entradas en tiempos razonables. Debido a la complejidad de las técnicas de Aprendizaje Automático (AA), si se quieren limitar tiempos, estas no se suelen aplicar a más que unos pocos miles de instancias, mientras que un motor de búsqueda es capaz de indexar millones de documentos en segundos. Esto es porque crear un índice tiene complejidad lineal [3], menor que la de entrenar un clasificador. Por ejemplo, en [4] explican que, por contra, entrenar un Máquinas de Soporte Vectorial (MSV) tiene una complejidad de $O(m^3)$, siendo m el tamaño del conjunto de entrenamiento.

En la aproximación aquí empleada no se prescinde de los sistemas clasificadores sino que, en lugar de usar las palabras de los tuits como atributos, como ocurre con las aproximaciones clásicas, en este caso tan solo se utilizan 24 atributos que se calculan a partir del ranking devuelto por el motor de búsqueda. Así se consigue reducir notablemente el tiempo de ejecución, y es razonable aplicar el sistema a un mayor número de instancias.

1.2 Objetivos

El objetivo principal de este proyecto es, inicialmente, construir un sistema similar al presentado en [5], donde se emplea la aproximación antes descrita. A partir de este sistema se pretenden comparar los resultados obtenidos utilizando los mismos corpus que en el paper original (en inglés) con los resultados obtenidos utilizando un corpus constituido por tuits en español.

Capítulo 2

Fundamentos

En este capítulo se introducirán términos fundamentales para el desarrollo de este proyecto, y por tanto también para la lectura de la memoria.

2.1 Minería de Opiniones

La Minería de Opiniones (MO), también referida a veces como Análisis de Sentimientos (AS), es un área de estudio que utiliza Procesamiento de Lenguaje Natural para estudiar las opiniones y emociones que las personas expresan en un texto hacia una entidad. De este modo, el objetivo de la MO es detectar opiniones, identificar el sentimiento que expresan y, así, clasificar su polaridad (positiva o negativa), tal y como se muestra en la Figura 2.1 (página 3). Como explica esa misma figura, la MO se puede considerar como un proceso de clasificación.

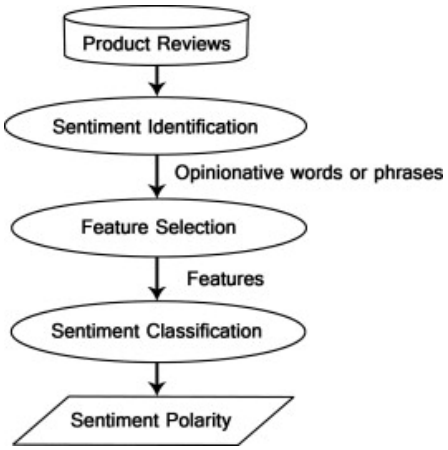

Figura 2.1: Proceso del Análisis de Sentimientos.

Los datasets utilizados en MO, NLP, RI, etc. se llaman corpus. Estos corpus suelen estar constituidos por reseñas de productos o servicios, ya que siempre expresan una opinión (bien

sea positiva o negativa). Sin embargo, también existen corpus formados por publicaciones en redes sociales, como es el caso de aquellos utilizados en este proyecto. Este tipo de corpus también son una buena fuente de información, porque las personas comparten y discuten sus opiniones sobre un tema determinado de forma libre.

2.2 Recuperación de Información

La Recuperación de Información (RI) es "el campo relacionado con la estructura, el análisis, la organización, el almacenamiento, la búsqueda y la recuperación de información.", [6]. Los documentos, sobre los que se realiza la búsqueda, pueden ser de diverso tipo, como páginas web, emails, documentos, multimedia, etc., aunque su aplicación más extendida, y también aquí empleada, se refiere a documentos que incluyan algún tipo de texto. La diferencia entre una base de datos y un conjunto de documentos es que la base de datos normalmente está formada por campos bien definidos [7]. Por ejemplo, una base de datos de un banco puede tener almacenados el nombre de sus clientes o cantidad de dinero en la cuenta. de esta forma resulta sencillo comparar los campos con consultas y encontrar coincidencias. Sin embargo, en RI se trabaja con contenidos de texto no estructurado, y comparar texto resulta más complicado: comparar el texto de una consulta con el texto de un documento y determinar cuál es una buena coincidencia es el tema central de la RI [7]. La coincidencia exacta de palabras no es suficiente, porque una de las características principales del lenguaje natural es su capacidad de representar el mismo mensaje de formas diferentes, y viceversa, lo que se denomina *variación lingüística* [8].

Un concepto clave en RI es el de *relevancia*. Se dice que un documento es relevante cuando cubre la necesidad de información que tenía el usuario cuando introduce la consulta en el motor de búsqueda. También se debe definir cómo se realiza la evaluación: un conjunto de procedimientos y medidas experimentales para comparar la salida del sistema con las expectativas del usuario en base a un conjunto de test. Dos de las principales métricas de evaluación en RI son la *precisión* y el *Recall*, que se definirán en la Sección 2.3.1 (página 5).

Los motores comerciales de búsqueda web, como *Google*, son la aplicación más conocida de la RI, pero existen también, por ejemplo, los motores de búsqueda *open source* (como el que se usa en este trabajo, *Terrier*) ampliamente utilizados para la investigación. Los "grandes problema" en el diseño de motores de búsqueda incluyen los identificados para la Recuperación de Información: algoritmos de clasificación efectivos, evaluación e interacción con el usuario. Hay, sin embargo, una serie de características críticas adicionales de los motores de búsqueda que resultan de su implementación en entornos operativos a gran escala [7]. La principal de estas características es el rendimiento del motor de búsqueda en términos de medidas como el tiempo de respuesta, el rendimiento de las consultas o la velocidad de indexación. El tiempo

de respuesta es el retraso entre el envío de una consulta y la recepción de la lista de resultados, el rendimiento mide la cantidad de consultas que se pueden procesar en un tiempo determinado y la velocidad de indexación es la velocidad a la que los documentos de texto se pueden transformar en índices para la búsqueda.

2.3 Aprendizaje Automático

El Aprendizaje Automático (AA), quizás más conocido por sus siglas en inglés, Machine Learning (ML), consiste en la programación de ordenadores utilizando datos de ejemplo o experiencia pasada. Es realmente útil en aquellos casos donde no se puede escribir directamente un programa para solucionar un problema pero existen datos de ejemplo [9]. Para que los sistemas aprendan es necesario mostrarles ejemplos, cuya disponibilidad es ahora mucho mayor gracias a que la sociedad de la información genera una gran cantidad de datos. Esto, unido a la mayor potencia computacional de las máquinas actuales, ha impulsado notablemente el empleo de este tipo de soluciones. El AA se puede aplicar a problemas de prácticamente cualquier campo, como la sanidad, la domótica o la banca. Por ejemplo se pueden diseñar sistemas que aprendan cómo diagnosticar mejor una enfermedad, cómo limpiar el suelo o cuándo aceptar el cargo a una tarjeta. El aprendizaje consiste en, partiendo de un sistema que interactúa u observa un entorno, el proceso de modificación del comportamiento del sistema de modo que se obtenga una mejora del sistema de acuerdo a algún criterio de evaluación.

Se pueden definir tres tipos de aprendizaje [9]:

- **Aprendizaje supervisado.** El comportamiento que se desea obtener se representa mediante un conjunto de ejemplos. Estos ejemplos permiten definir un criterio para evaluar el comportamiento real del sistema.
- **Aprendizaje no supervisado.** No tiene ningún tipo de señal que indique un comportamiento deseado para el sistema. El criterio de evaluación se basa en la regularidad de los grupos de datos identificados.
- **Aprendizaje por refuerzo.** El comportamiento deseado no se representa mediante ejemplos sino mediante una evaluación sobre los resultados que genera el sistema en su entorno.

Dado que en este proyecto tan solo se ha utilizado aprendizaje supervisado, será el único del que se explicará en detalle su funcionamiento.

2.3.1 Aprendizaje Supervisado

El aprendizaje supervisado consta de dos fases: entrenamiento y utilización. En la primera de ellas se presentan los ejemplos (*conjunto de entrenamiento*) al sistema. Este aprende a partir

de los ejemplos, modificando sus parámetros hasta que la salida del sistema se ajuste a la salida deseada. Es necesaria una medida de rendimiento o precisión. En la segunda fase se presentan al sistema nuevos ejemplos *(conjunto de test*),esperando que este generalice. Esto es, que el sistema debe dar respuestas correctas para cualquier ejemplo, no solo memorizar las respuestas a los ejemplos de entrenamiento y limitarse a responder correctamente tan solo a estos ejemplos de entrenamiento. Que un modelo responda muy bien a los ejemplos de entrenamiento pero responda mal a los de test se conoce como *sobreajuste*. Una forma de controlar este sobreajuste es controlar la complejidad del modelo. Se prefiere el modelo más simple que sea capaz de explicar los datos. Por tanto, un buen modelo es aquel cuyo error en el test es bajo.

A este respecto, la realización de un diseño experimental consta de tres pasos:

1. **Preprocesado de los datos.** Existen dos tareas principales: transformación de datos y reducción de los mismos. Dentro de la transformación de datos existen la limpieza de datos, para eliminar datos incorrectos o con ruido, la normalización o la construcción de nuevos atributos. Por su parte, la reducción de datos está orientada a reducir el tamaño de los datos mediante la agregación o eliminación de características redundantes.

2. **Entrenamiento del modelo.** La capacidad de un modelo para resolver un problema está ligada a los ejemplos utilizados durante el aprendizaje. Por eso, el conjunto de entrenamiento debe ser, en primer lugar, significativo, es decir, que el número de ejemplos sea suficiente; y en segundo lugar, también debe ser representativo, es decir, que dichos ejemplos tienen que ser diversos, de forma que todas las regiones del espacio de estados deben estén suficientemente cubiertas.

   Aunque se pueden considerar problemas de predicción, ajuste de curvas, regresión y clasificación, nos centraremos en los de clasificación por ser la categoría a la que pertenece este proyecto. En los problemas de clasificación el sistema debe clasificar objetos en un número finito de clases de acuerdo a propiedades. A la hora de evaluar la bondad de un clasificador existen una serie de criterios y métricas de evaluación [10]:

   - **Matriz de confusión.** Esta muestra la distribución de los errores cometidos por un clasificador a lo largo de las distintas categorías del problema. La Tabla 2.1 (página 7) muestra la estructura de una matriz de confusión: VN representa el número de verdaderos negativos, los ejemplos negativos que se clasificaron como tal; FP es el número de falsos positivos, ejemplos negativos pero que se clasificaron como positivos; FN es la cifra de falsos negativos, ejemplos positivos clasificados como negativos, y finalmente VP son verdaderos positivos, ejemplos positivos que sí se clasificaron como tal. Esto nos permite luego calcular las métricas utilizadas

para evaluar el clasificador, como son la precisión, la tasa de error, la sensibilidad o la especificidad.

		Clase predicha	
		Negativo	**Positivo**
Clase real	**Negativo**	VN	FP
	Positivo	FN	VP

Tabla 2.1: Plantilla de matriz de confusión.

- **Curva ROC.** Es un espacio bidimensional acotado en los ejes X e Y entre 0 y 1, donde el eje X representa la tasa de falsos positivos y el eje X la tasa de verdaderos positivos, como se puede ver en el ejemplo de la Figura 2.2 (página 7). De este modo, el punto (0,0) representa al clasificador que etiqueta todo como negativo; el (0,1) al clasificador perfecto; el (1,1) al que predice todos los casos como positivos; y el (1,0) sería aquel que clasifica todos los casos incorrectamente. El clasificador se sitúa en un punto; pero si ese clasificador tiene parámetros que puedan variar, cada valor de los parámetros colocaría al clasificador en un punto diferente. La curva ROC representa todas las posibilidades de estos valores.

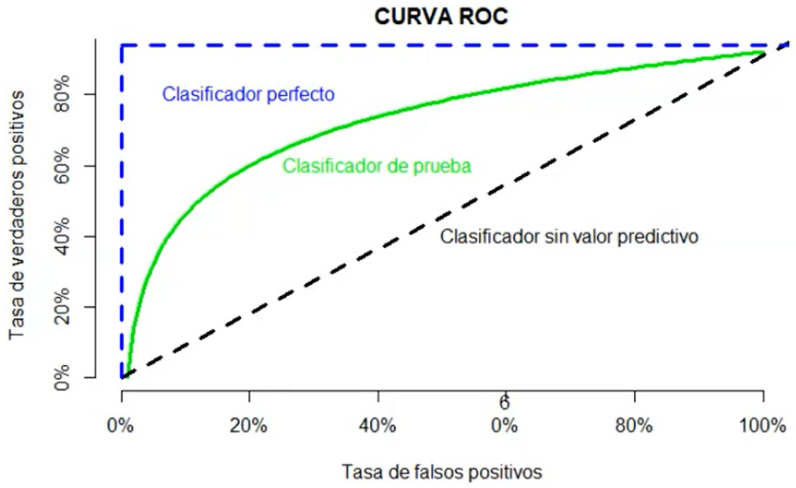

Figura 2.2: Ejemplo de curva ROC.

Asociado al concepto de curva ROC está el de Area Under the Curve, una medida de la bondad del clasificador, que representa el porcentaje de espacio comprendido debajo de la curva ROC, por lo que puede tomar valores en el rango [0,1]. Es una

medida que encapsula toda la información contenida en la matriz de confusión.

- **Índice Kappa.** Es un índice que permite determinar hasta qué punto la concordancia observada entre dos clasificadores es superior a la que se espera obtener por puro azar. Se calcula como[1]: $Kappa = \frac{p_o - p_e}{1 - p_e}$
- **Precisión.** Es un estadístico que se define como el porcentaje de verdaderos positivos, VP, entre todos los ejemplos que el clasificador ha etiquetado como positivos: $VP + FP$. Se calcula como: $Precision = \frac{VP}{VP+FP}$

  Dicho de otro modo, la precisión es la probabilidad de que el clasificador acierte cuando etiqueta un ejemplo como positivo.
- **Recall.** Es un estadístico que mide la probabilidad de que un ejemplo positivo resulte correctamente reconocido por el clasificador. El valor es obtenido dividiendo el número de verdaderos positivos, VP, entre el número de positivos en el dataset: $VP + FN$. Se calcula como: $Recall = \frac{VP}{VP+FN}$

  Se puede ver que las fórmulas de la precisión y el recall tan solo difieren en el denominador, lo que tiene sentido. Mientras que la precisión es la frecuencia de verdaderos positivos entre todos los ejemplos etiquetados como positivos por el clasificador, recall es la frecuencia de verdaderos positivos entre todos los ejemplos positivos del dataset.
- **F1.** El valor F1 se utiliza para combinar las medidas de *precisión* y *recall* en un único valor. Esto facilita poder comparar el rendimiento combinado de la precisión y el recall entre varias soluciones. F1 se calcula haciendo la media armónica entre la precisión y el recall: $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

3. **Estimación del error real del modelo.** Como ya se ha comentado, la bondad de un clasificador nunca se mide en el conjunto de entrenamiento. Esto es así porque el clasificador ha sido entrenado con esos ejemplos, por lo que siempre va a ofrecer buenos resultados. Es necesario, por tanto, contar con un conjunto de test. Este conjunto no debe participar en el entrenamiento del clasificador, y debe ser disjunto con el conjunto de entrenamiento. Para construir este conjunto existen dos métodos principales: *Holdout* y validación cruzada. El *Holdout*, el más simple de los dos, consiste en simplemente dividir el conjunto de casos en los dos grupos: entrenamiento y test. Normalmente se asignan $2/3$ de los ejemplos al conjunto de entrenamiento y $1/3$ al de test. La validación cruzada consiste en dividir el conjunto de casos en K subconjuntos del mismo tamaño. Se utilizan $K - 1$ subconjuntos como datos de entrenamiento y el subconjunto restante como datos de test. Se repite el proceso para los K subconjuntos y, finalmente, se calcula

[1] Siendo p_o la precisión observada y p_e la precisión esperada.

la media de los valores obtenidos. En este proyecto se ha utilizado un valor de $K = 10$, el más común.

Capítulo 3

Métodos utilizados

En este capítulo se explicarán en detalle los métodos, técnicas y algoritmos utilizados para la realización de este proyecto.

3.1 Recuperación de Información

En lo relativo a la Recuperación de Información hay dos decisiones que tomar: (1) qué motor de búsqueda (search engine en inglés) y (2) qué esquema de pesos utilizar. A continuación se describen las alternativas escogidas.

3.1.1 Motor de búsqueda

Hoy en día existen multitud de motores de búsqueda de código abierto susceptibles de uso tanto para desarrollo de software como para investigación. Por esa razón, es importante pararse a ver qué ofrece cada uno y cuál puede ser el más adecuado para las necesidades del proyecto. Tras barajar varias opciones, como Zettair (`http://www.seg.rmit.edu.au/zettair/`), Galago (`https://www.lemurproject.org/galago.php`) o la propia Lucene y sus derivados [11], finalmente se ha elegido Terrier [12]. Los aspectos que más han influido a su favor son:

- Tiene una documentación actualizada y detallada, con tutoriales cortos para aprender rápidamente un uso simple del motor pero con extensos capítulos que explican cada detalle de su funcionamiento. Los otros motores no tienen tanta documentación ni está tan bien detallada. La de Zettair, por ejemplo, es de 2009. Este es el principal motivo por el que se descartó este motor, a pesar de ser el que usaron en [5].
- Terrier fue concebido desde un principio para su empleo en investigación, razón por la cual es empleado a menudo por otros autores, como [13].

- A pesar de inicialmente haber sido desarrollado en Java, proporciona también un framework, PyTerrier [14], que permite su rápida integración en Python; el lenguaje utilizado en todos los scripts de este proyecto. Se ha elegido Python por tener una sintaxis sencilla, lo que provoca que el desarrollo sea más rápido, el gran número de bibliotecas que ofrecen soporte a este lenguaje (la propia PyTerrier, por ejemplo), y su cada vez mayor popularidad [15].

.

3.1.2 Esquema de Pesos

En el contexto de los sistemas de RI, el modelo de recuperación y el esquema de pesos son los encargados de construir el ranking de documentos relevantes respecto a una consulta, asignando para ello a los documentos del índice una puntuación (*score*) en base a su supuesta relevancia respecto a la consulta, para a continuación devolver como resultado la lista de documentos más relevantes (supuestamente). Por esto, es fundamental escoger un modelo y un esquema de pesos que ofrezcan buenos resultados.

Los modelos más sencillos son el booleano [7] y el vectorial [7], pero no se han tenido en cuenta y se ha optado por un modelo posterior, que ofrece mejores resultados: el modelo probabilístico [7]. Dentro del modelo se han escogido dos esquemas de peso para realizar los experimentos: Okapi BM25 [7] y PL2 [16]. El primero de ellos, el Okapi BM25, es de la familia de los IDF (*Inverse Document Frequency*) [16], y goza de gran popularidad por su buen rendimiento (por ejemplo en [5]). El segundo, el PL2, pertenece a una familia diferente, la de los DFR (*Divergence from Randomness*), y es también bastante popular y muestra buen rendimiento. Al pertenecer ambos a familias diferentes es de esperar que obtengan resultados diferentes.

3.2 Aprendizaje Automático

Existen gran cantidad de técnicas de clasificación, desde las más populares como las Redes de Neuronas Artificiales hasta modelos Bayesianos o árboles de decisión. En este proyecto se han utilizado 8 clasificadores diferentes[1], que son los siguientes [9]:

3.2.1 Árboles de Decisión

Un *árbol de decisión* es un modelo predictivo que divide el espacio muestral agrupando observaciones con valores similares para la variable respuesta. Para dividir el espacio muestral en subregiones es necesario aplicar una serie de reglas o decisiones, de forma que cada

[1] Todos ellos implementados en Weka (`https://www.cs.waikato.ac.nz/ml/weka/`)

subregión contenga la mayor proporción posible de individuos de una de las poblaciones. Si una subregión contiene datos de diferentes clases, se subdivide en regiones más pequeñas hasta fragmentar el espacio en subregiones menores que contienen datos de la misma clase. Los árboles de decisión están formados por nodos y su lectura se realiza de arriba hacia abajo. En un árbol de decisión existen tres tipos diferentes de nodos:

- **Nodo raíz**: En él se produce la primera división en función de la variable más importante.
- **Nodos internos**: Tras la primera división encontramos estos nodos, que vuelven a dividir el conjunto de datos en función de las variables.
- **Nodos terminales**: Se ubican en la parte inferior del esquema y su función es indicar la clasificación definitiva.

Las ramas de un nodo representan los diferentes valores que puede tomar el atributo respecto al que se pregunta en el nodo. En la Figura 3.1 (página 12) se puede ver gráficamente un ejemplo de árbol de decisión. Si aparece una nueva instancia, se recorre el árbol hasta llegar a una hoja.

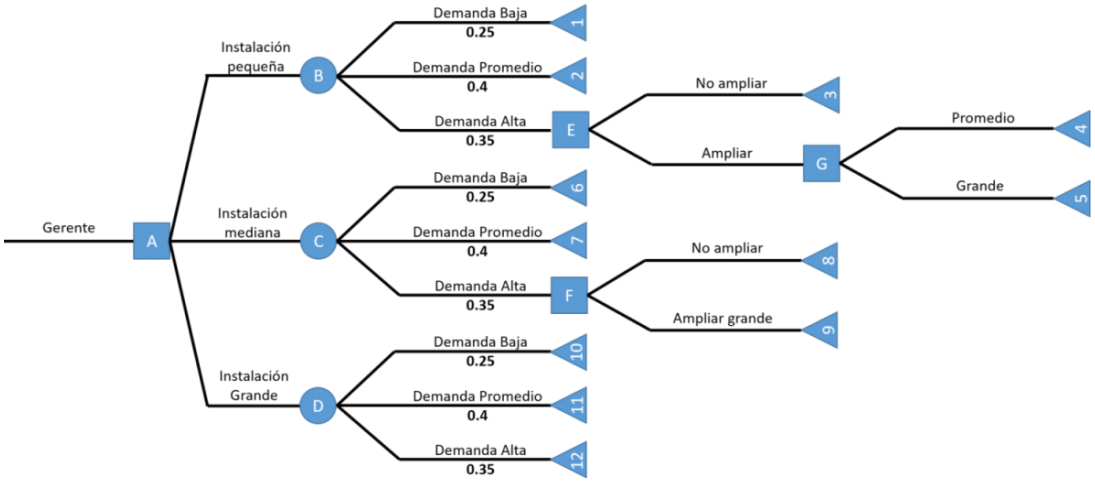

Figura 3.1: Ejemplo de árbol de decisión.

Existen diversos algoritmos de aprendizaje basados en árboles de decisión. En este trabajo se han elegido J48 y REPTree, de los cuales se habla más en detalle a continuación.

J48

J48 [17] es la implementación que hace Weka de $C4.5$ [17], un algoritmo para la creación de árboles de decisión. Es una extensión del anteriormente desarrollado algoritmo ID3 [17].

REPTree (Reduced Error Pruning Tree)

Árbol de decisión muy rápido. Este algoritmo construye un árbol de decisión utilizando la ganancia de la información y lo poda usando una poda de errores reducidos (con reajuste). Solo ordena los valores de los atributos numéricos una vez. Los valores ausentes se tratan dividiendo las instancias correspondientes en partes (es decir, como en C4.5).

3.2.2 Máquinas de Soporte Vectorial

Las Máquinas de Soporte Vectorial (MSV) son una técnica de clasificación relativamente reciente que ha recibido mucho interés porque: demuestran una precisión muy alta en la clasificación, tienen una base matemática muy firme y en algunos casos han demostrado superar a las Redes de Neuronas Artificiales, los clasificadores más exitosos, además de requerir un tiempo de entrenamiento menor, tal y como se explica en [18].

Una MSV es un modelo que representa a los puntos de muestra en el espacio, separando las clases en dos espacios lo más amplios posibles mediante un hiperplano de separación, definido este último como el vector entre los dos puntos, de las dos clases, más cercanos al que se llama *vector soporte*. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, se clasifican en una u otra clase en función del espacio al que pertenezcan. Por lo tanto se trata, originalmente, de un clasificador binario, aunque se puede generalizar a multiclase. Como se puede ver en la Figura 3.2 (página 14), pueden existir varios hiperplanos que separen perfectamente los datos en las dos categorías deseadas, por lo que es necesario definir un criterio para elegir qué hiperplano utilizar. Dicho criterio es que el hiperplano se debe construir de tal forma que la separación entre las dos clases sea máxima.

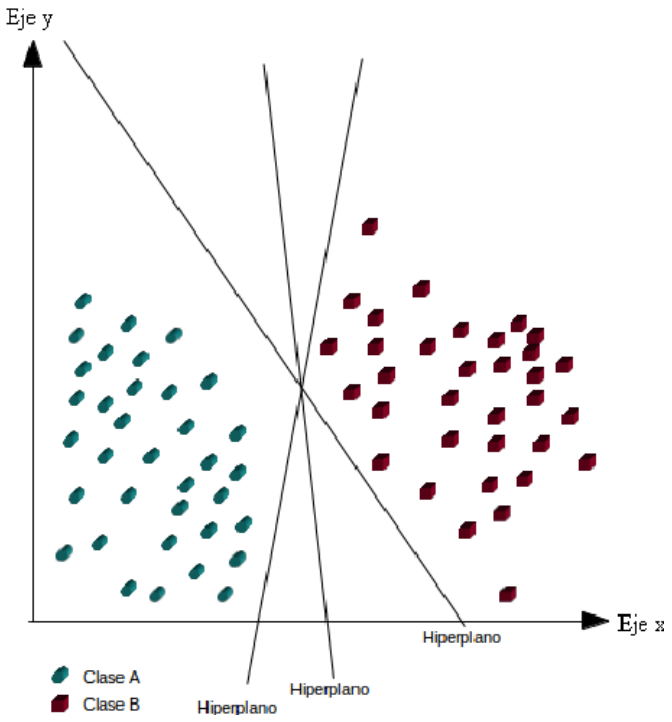

Figura 3.2: Ejemplo de MSV.

3.2.3 Naive Bayes

Un clasificador Naive Bayes es un clasificador probabilístico basado en el teorema de Bayes:

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

Se conoce como *naive* (ingenuo) debido a que, con el fin de simplificar, se asume que las variables predictoras son independientes entre sí. Es una manera fácil de construir modelos con un comportamiento muy bueno debido a su simplicidad. Se consigue proporcionando una forma de calcular la probabilidad 'posterior' de que ocurra un cierto evento A, dadas las probabilidades de eventos B 'anteriores'.

Sus principales ventajas son que es una manera fácil y rápida de predecir clases para problemas de clasificación binarios y multiclase; y que en los casos en que sea apropiada una presunción de independencia, el algoritmo se comporta mejor que otros modelos de clasificación, incluso con menos datos de entrenamiento.

Naive Bayes Multinomial

Como se explica en [19], existen dos modelos probabilísticos diferentes para la clasificación basados en la suposición ingenua de Bayes. Algunos utilizan un modelo de Bernoulli, es decir, una red bayesiana sin dependencias entre palabras y características de palabras binarias,

como el que se acaba de explicar. Sin embargo, otros usan un modelo multinomial, es decir, un modelo de lenguaje de unigramas con recuento de palabras. En [19] se explica que el modelo de Bernoulli se desempeña bien con tamaños de vocabulario pequeños, pero que el multinomial ofrece mejores resultados en tamaños de vocabulario más grandes, proporcionando en promedio una reducción del 27% en el error sobre el modelo de Bernoulli en cualquier tamaño de vocabulario.

3.2.4 Regla Conjuntiva

Una regla consta de antecedentes combinados mediante el operador $\wedge$ y un consecuente para la clasificación. Es un método similar al de los árboles de decisión (sección 3.2.1), de forma que cada camino equivaldría a una regla, y el árbol equivaldría a la disyunción de todas las reglas. Se establece un punto de corte mínimo para cada atributo antecedente. Para realizar la clasificación se evalúa la regla de forma que si todos los atributos superan el umbral la muestra se clasifica en una clase, y en caso contrario en la otra.

3.2.5 Regresión Logística

La regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica en función de las variables independientes o predictoras. Es útil para modelar la probabilidad de un evento en función de otros factores. Las probabilidades que describen el posible resultado de un único ensayo se modelan como una función de variables explicativas, utilizando para ello una función logística.

Regresión Logística Simple

La regresión logística simple puede usarse para tratar de correlacionar la probabilidad de una variable cualitativa binaria (valores “positivo” y “negativo”, por ejemplo) con una variable escalar (llamada variable predictora). La idea es que la regresión logística aproxime la probabilidad de obtener “positivo” o “negativo” con el valor de las variables explicativas.

Regresión Logística Multinomial

Este es un método de regresión logística que permite clasificar en más de una clase, aunque en nuestros experimentos tan solo se realiza la clasificación en dos clases: positivo y negativo.

Capítulo 4

Modelos: Implementación, Descripción y Resultados

En este capítulo se expondrán los diferentes experimentos realizados durante el desarrollo del proyecto, así como los corpus utilizados.

4.1 Diseño de experimentos

Para la realización de este proyecto se han utilizado cuatro corpus en inglés, formados por tuits reales y utilizados en otros experimentos, como [5], [20] o [21]; y uno en español, de la edición de 2014 del TASS [22]. Estos son:

- **Health-Care Reform Dataset (HCR)**. Este primer corpus contiene tuits publicados en 2010 con el hashtag "#hcr". Estos tuits hacen referencia a una reforma sanitaria que tuvo lugar en Estados Unidos en ese mismo año y fue muy polémica. Los tuits fueron etiquetados manualmente como *positivo*, *negativo* o *neutro*; pero para estos experimentos se han desechado los tuits con polaridad neutra. El corpus está dividido en tres: dev, test y train; pero en los experimentos realizados en este proyecto no se ha utilizado el conjunto dev.

- **Obama-McCain Debate (OMD)**. El segundo de los datasets está formado por tuits publicados en 2008 durante el primer debate presidencial de EEUU entre los candidatos Obama y McCain. Estos tuits se etiquetaron manualmente por ocho jueces con la ayuda de Amazon Mechanical Turk[1] como *positivo*, *negativo*, *mixto* u *otro*. Como cada tuit puede tener más de una etiqueta (cada juez lo etiqueta particularmente), para este proyecto tan solo se han tenido en cuenta aquellos tuits etiquetados como positivos o negativos por al menos 2/3 de los jueces.

[1] http://www.mturk.com

- **Stanford-Twitter Sentiment corpus (STD)**. Este corpus se construyó usando la API de Twitter para obtener tuits de diferentes temas y que contuviesen nombres de productos, marcas y personas. El dataset está divido en dos: *STD-Train* y *STD-Test*. STD-Train está formado por 1.600.000 tuits etiquetados automáticamente. Este método de etiquetado, conocido como *noisy labelling* y descrito en [23], consiste en asignar automáticamente al tuit una polaridad basándose en los emoticonos presentes en el tuit. De este modo, tuits que contienen :), :-), :), :D o =) se clasifican como positivos; mientras que tuits que contienen :(:'(o :-(se clasifican como negativos. De esta forma se ahorra mucho tiempo y esfuerzo en etiquetar manualmente los tuits, si bien el corpus resultante contiene ruido (de ahí su nombre), debido a que estos emoticonos a veces se pueden utilizar con sarcasmo o ironía, de modo que expresen todo lo contrario a lo que en un principio se podría pensar. La otra parte del corpus, STD-Test, es mucho más pequeña, y está formada por 359 tuits etiquetados a mano.

- **Stanford Sentiment Gold Standard (STS-Gold)**. La construcción de este dataset se detalla en [24]. En ese *paper* se explica que estudian y comparan ocho corpus diferentes formados por tuits y etiquetados manualmente, y posteriormente construyen el STS-Gold teniendo en cuenta las conclusiones obtenidas tras analizar los otros corpus. Está formado por tuits manualmente etiquetados y en los que tres revisores están de acuerdo en su polaridad. En este caso, los tuits tan solo tienen dos etiquetas posibles: *positivo* y *negativo*.

- **TASS 2014**. Este es el único corpus utilizado cuyos tuits están escritos en español. Su construcción está descrita en [22], donde se explica que el dataset está divido en tres partes: un conjunto de test compuesto por 1.000 tuits etiquetados manualmente, otro conjunto de test compuesto por más de 68.000 tuits, esta vez etiquetados de forma automática, y un conjunto de entrenamiento de 7.219 tuits, etiquetados manualmente. En este proyecto tan solo se utilizaron aquellos de etiquetado manual. Aunque los tuits fueron originalmente etiquetados como *positivos*, *negativos* o *neutros*; pero en este proyecto tan solo se han tenido en cuenta los tuits con polaridad positiva y negativa, descartando los de polaridad neutra.

Llegados a este punto, es importante señalar que la heterogeneidad en cuanto a estructura y etiquetado de los datasets empleados en nuestros experimentos responde a la necesidad de poder comparar dichos resultados con los de experimentos previos, como ya ocurría en el trabajo original [5]. En cuanto al contenido de los tuits, este ha sido preprocesado de la siguiente manera: sustituir todos los enlaces a páginas web por la etiqueta común "[LINK]", eliminar todos los caracteres no alfabéticos (incluyendo números y signos de puntuación), pasar el texto a minúsculas, eliminar los caracteres repetidos más de dos veces (por ejemplo

"goooooool" se sustituye por "gool"). Mediante esto último se consigue que, por un lado, se mantenga la expresividad que el autor trataba de transmitir con este tipo de repeticiones, a la vez que aquellas palabras que originalmente contienen dos letras seguidas iguales se mantienen (algo muy común en inglés, el caso de "soon" por ejemplo).

4.2 Baselines

Se han implementado dos *baselines* de tipo clásico, es decir, en las que las palabras de los tuits (unigramas) son utilizadas como atributos de los clasificadores. Previamente, el contenido de los tuits se ha preprocesado de la forma anteriormente descrita. Como ejemplo de método sencillo para el primer *baseline* se ha utilizado Naive Bayes, mientras que como segundo *baseline* se ha empleado un método sofisticado y difícil de batir, Máquinas de Soporte Vectorial. Los resultados obtenidos se pueden ver en la Tabla 4.1 (página 38). Las métricas utilizadas para la evaluación (tanto en las *baselines* como en las próximas secciones) son: índice Kappa, precisión, recall, F1 y AUC. Como se explica en [25], la implementación de Weka deriva Kappa de la matriz de confusión. Aún así, puede dar resultados ligeramente diferentes a los que ofrece una aplicación trivial de su fórmula, ya explicada en la Sección 2.3.1 (página 5). Además, en ocasiones se puede apreciar que el F1 no es un valor medio entre la precisión y el recall. Esto ocurre porque, salvo el índice Kappa, todas las métricas son una media de clases. Es decir, se calcula la métrica para la clase *positivos*, se calcula para la clase *negativos*, y el valor que se muestra en las tablas es la media entre esos dos valores. De este modo, el valor de F1 no siempre se encuentra entre el de recall y el de precisión. Se puede apreciar que en esta tabla algunas casillas están marcadas con una "X". Esto es porque en esas celdas es imposible calcular el valor de la métrica, normalmente debido a que el algoritmo clasificó o bien todas las instancias como positivas o todas como negativas, y el estadístico implicaría realizar una división entre 0. Esto se puede ver muy bien en la matriz de confusión de alguno de estos clasificadores (Tabla 4.2, página 19), donde se ve que está clasificando todas las instancias como negativas, y por tanto haciendo imposible el cálculo de la precisión y del F1. Por otro lado, en la Tabla 4.3 se puede ver la matriz de confusión de un clasificador que no clasifica todo como negativo y, por tanto, puede calcular todas las métricas. Sin embargo, de un total de 1.920 instancias tan solo está clasificando seis como positivas (un 0.31%), cuando en realidad son 704 positivas.

		Clase predicha	
		Negativo	**Positivo**
Clase real	**Negativo**	1401	0
	Positivo	632	0

Tabla 4.2: Matriz de confusión de Naive Bayes con el corpus STS-Gold.

		Clase predicha	
		Negativo	**Positivo**
Clase real	**Negativo**	1216	0
	Positivo	698	6

Tabla 4.3: Matriz de confusión de SVM con el corpus OMD.

4.3 Serie de experimentos Noisy

Este segundo conjunto de experimentos se llama así por el corpus en inglés utilizado en todos ellos, el *STD-Train*, creado empleando clasificación automática y ,consecuentemente, con presencia de ruido, como se explicó al comienzo de este capítulo. De esta forma, se puede hablar de 4 subconjuntos de experimentos, usando en cada uno un corpus en inglés diferente. Además, dentro de cada uno se han probado dos esquemas de pesos diferentes (BM25 y PL2) y ocho algoritmos de Aprendizaje Automático.

En este punto se plantea la cuestión de si se podría haber construido un corpus masivo de tuits en español etiquetado automáticamente, del mismo modo que el STD-Train, y así probar este tipo de aproximación también con el corpus en español. Sin embargo, esta opción se descartó porque para que tenga sentido, ese hipotético corpus debería tener unas dimensiones por lo menos similares a las de su homólogo en inglés, y debido a las restricciones que impone la API gratuita de Twitter, descargar todos esos tuits implicaría una demora significativa en la planificación del proyecto. Además, como se podrá comprobar más adelante, al menos en inglés no se obtienen mejores resultados con la versión Noisy que con la Manual.

Seguidamente analizaremos los resultados obtenidos para cada uno de los datasets considerados.

4.3.1 HCR

La estructura de este corpus es muy diferente del resto de corpus en inglés, pues es el único con conjuntos separados de entrenamiento y test, y por tanto no es necesario hacer validación cruzada. En el caso de STD, como se verá más adelante, tan solo se utiliza STD-Test.

En la Tabla 4.4 (página 20) se pueden ver los resultados de los diferentes algoritmos tras ser entrenados utilizando este corpus y el esquema de pesos BM25, mientras que los resultados con el esquema de pesos PL2 se pueden ver en la Tabla 4.5 (página 20).

Tabla 4.4: Resultados de Noisy usando HCR como consultas y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0	X	0.768	X	0.500
REPTree	0.041	0.664	0.733	0.684	0.504
SVM	0	X	0.768	X	0.500
NB	0.027	0.653	0.640	0.646	0.478
NBM	-0.030	0.642	0.518	0.556	0.491
RC	0	X	0.768	X	0.5
RLS	0	X	0.768	X	0.507
RLM	-0.056	0.615	0.711	0.652	0.467

Tabla 4.5: Resultados de Noisy usando HCR como consultas y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0	X	0.768	X	0.500
REPTree	-0.058	0.636	0.755	0.669	0.506
SVM	0	X	0.768	X	0.500
NB	0.008	0.647	0.632	0.639	0.474
NBM	-0.027	0.632	0.504	0.543	0.488

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.5 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
RC	0	X	0.768	X	0.5
RLS	0	X	0.768	X	0.507
RLM	-0.031	0.628	0.719	0.661	0.473

Comparando los resultados de ambas tablas se aprecia que, como era de esperar, las diferencias entre ambos esquemas de pesos son mínimas: algunos clasificadores funcionan mejor con un esquema de pesos y otros con otro, pero no se puede concluir que uno de los dos esquemas sea preferible porque, como ya se ha dicho, las diferencias son mínimas. También son pequeñas las diferencias entre clasificadores, sin haber ninguno que se desmarque claramente del resto. Otra conclusión que se puede obtener a partir de estos resultados es que este es un corpus muy difícil de clasificar, lo que confirma lo ya indicado por [5]. El índice Kappa toma siempre valores muy próximos a 0, llegando incluso a tomar valores negativos, lo que indica que la concordancia de los resultados es muy mala. Además, los valores de precisión también son bastante bajos, sin llegar a 0.7 en ninguno de los casos. Por último, el Area Under the Curve (AUC) toma siempre valores muy próximos a 0.5, por lo que estos clasificadores no están obteniendo mejores resultados que una clasificación azar. En general, son resultados similares a los obtenidos en los *baselines* (Sección 4.2).

4.3.2 OMD

En este caso, y a pesar de no tener un conjunto de entrenamiento y otro de test, los resultados que se obtienen son mejores que con el anterior corpus, como se puede comprobar en las Tablas 4.6 y 4.7.

Tabla 4.6: Resultados de Noisy usando OMD como consultas y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.266	0.659	0.655	0.657	0.630
REPTree	0.193	0.632	0.649	0.632	0.621
SVM	0.011	0.603	0.635	0.503	0.504

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.6 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
NB	0.238	0.661	0.611	0.617	0.677
NBM	0.222	0.649	0.607	0.614	0.643
RC	0.035	0.593	0.635	0.529	0.574
RLS	0.179	0.640	0.660	0.624	0.678
RLM	0.197	0.644	0.663	0.633	0.664

Tabla 4.7: Resultados de Noisy usando OMD como consultas y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.217	0.638	0.647	0.641	0.636
REPTree	0.201	0.632	0.645	0.635	0.637
SVM	0.009	0.595	0.634	0.502	0.504
NB	0.238	0.662	0.609	0.616	0.677
NBM	0.222	0.650	0.606	0.613	0.646
RC	0.084	0.604	0.638	0.621	0.567
RLS	0.178	0.640	0.660	0.623	0.677
RLM	0.200	0.642	0.661	0.635	0.671

Igual que ya ocurría con el corpus HCR, no hay ningún clasificador que se desmarque del resto, aunque es cierto que tanto Máquinas de Soporte Vectorial (MSV) como la Regla Conjuntiva obtienen un índice Kappa claramente inferior al resto. Sin embargo, lo que sí se aprecia es que el comportamiento de los clasificadores para este corpus es mejor que en la Sección 4.3.1: usando este corpus ningún algoritmo clasifica todas las instancias como negativas o positivas, y por tanto se pueden calcular todas las métricas. Pero además, los valores del índice Kappa son bastante superiores. En HCR este estadístico tomaba siempre valores próximos al 0, mientras que en OMD toma valores en torno a 0.2 (salvo utilizando Máquinas de Soporte Vectorial y RC, como ya se dijo). A pesar de esta mejora, 0.2 sigue sin ser un gran valor para

el índice Kappa.

Los resultados para la AUC, como se puede ver en la Figura 4.1 (página 23), corroboran lo ya dicho: Ninguno de los clasificadores sobresale claramente respecto al resto excepto, si acaso, las Reglas Conjuntivas, y en sentido negativo. Asimismo, se puede afirmar que los resultados son mejores que los obtenidos en los *baselines*.

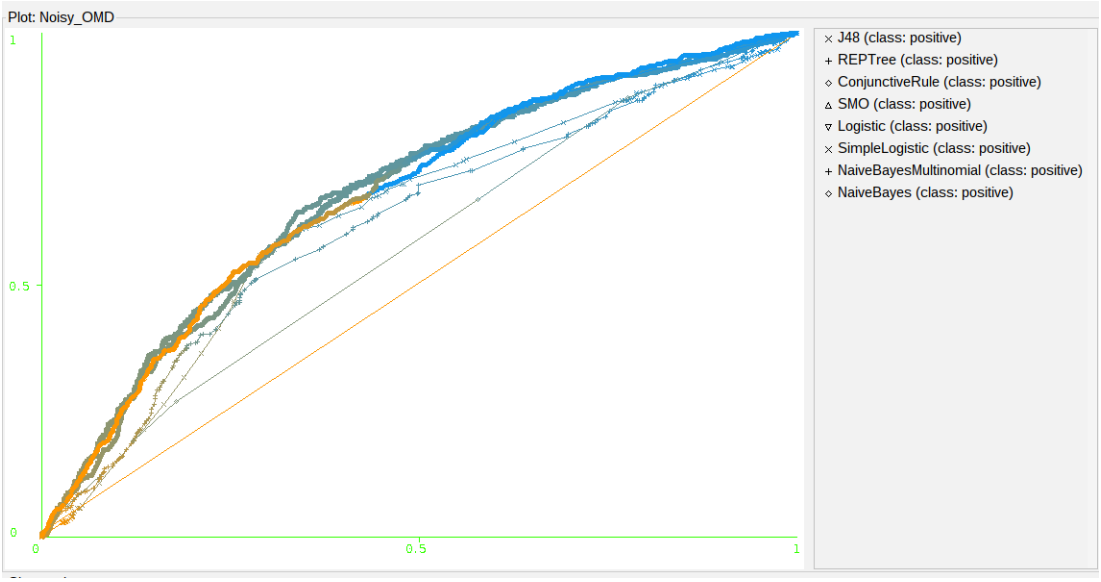

Figura 4.1: Curva ROC con el esquema de pesos BM25 y los tuits de OMD como consultas.

4.3.3 STD

En el caso de STD, como el índice se construyó con los tuits presentes en STD-Train, en estos experimentos tan solo se han utilizado como consultas los tuits de STD-Test. Además, el conjunto de test fue etiquetado manualmente (igual que los corpus del resto de experimentos), mientras que, como ya se ha dicho, el de entrenamiento fue etiquetado mediante *noisy labelling*, con los problemas que esto conlleva. Los resultados obtenidos se pueden ver en las Tablas 4.8 (página 23) y 4.9 (página 24).

Tabla 4.8: Resultados de Noisy usando STD como consultas y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.202	0.604	0.680	0.599	0.614
REPTree	0.308	0.657	0.655	0.653	0.661

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.8 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
SVM	0.314	0.658	0.657	0.657	0.657
NB	0.281	0.641	0.641	0.641	0.697
NBM	0.270	0.635	0.635	0.635	0.654
RC	0.253	0.627	0.627	0.626	0.649
RLS	0.303	0.652	0.652	0.651	0.705
RLM	0.225	0.613	0.613	0.613	0.660

Tabla 4.9: Resultados de Noisy usando STD como consultas y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.203	0.602	0.602	0.601	0.624
REPTree	0.226	0.613	0.613	0.613	0.635
SVM	0.297	0.650	0.649	0.648	0.649
NB	0.293	0.646	0.646	0.646	0.700
NBM	0.270	0.635	0.635	0.635	0.654
RC	0.185	0.598	0.593	0.587	0.618
RLS	0.298	0.649	0.649	0.649	0.697
RLM	0.220	0.610	0.610	0.609	0.657

De la misma forma que ocurría en los datasets anteriores, las diferencias entre usar un esquema de pesos u otro son mínimas. Los resultados son similares a los obtenidos utilizando el corpus OMD y mejores que los obtenidos en los *baselines*, con la mayoría de valores de índice Kappa entre 0.2 y 0.3, y con un AUC en torno a 0.6. Sin embargo, cambian los mejores clasificadores: por ejemplo, en el caso de corpus OMD el clasificador MSV era de los peores, con un índice Kappa de 0.011, un $F1$ de 0.617 y un AUC de 0.504. Sin embargo, en el corpus actual es de los mejores clasificadores, sino el mejor: utilizando el esquema de pesos BM25

obtiene un índice Kappa de 0.314 (el más alto de entre todos los clasificadores), un $F1$ de 0.657 y un AUC de 0.657.

4.3.4 STS-Gold

Los resultados obtenidos usando este corpus se ven en las Tablas 4.10 (página 25) y 4.11 (página 25).

Tabla 4.10: Resultados de Noisy usando STS-Gold como consultas y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.421	0.754	0.762	0.756	0.699
REPTree	0.427	0.758	0.767	0.760	0.745
SVM	0.397	0.759	0.768	0.750	0.678
NB	0.432	0.764	0.736	0.744	0.781
NBM	0.406	0.757	0.719	0.728	0.736
RC	0.455	0.766	0.765	0.765	0.732
RLS	0.395	0.754	0.765	0.749	0.782
RLM	0.402	0.755	0.766	0.752	0.781

Tabla 4.11: Resultados de Noisy usando STS-Gold como consultas y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.433	0.761	0.770	0.762	0.727
REPTree	0.409	0.749	0.758	0.752	0.750
SVM	0.382	0.754	0.764	0.744	0.671
NB	0.434	0.765	0.737	0.745	0.782

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.11 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
NBM	0.410	0.759	0.721	0.730	0.744
RC	0.450	0.765	0.760	0.762	0.738
RLS	0.410	0.761	0.770	0.755	0.785
RLM	0.396	0.753	0.764	0.749	0.782

En las tablas con los resultados se puede apreciar claramente una mejoría en los resultados respecto al resto de corpus, y también respecto a los *baselines*. Los valores del índice Kappa se encuentran en todos los casos en torno a 0.4, lo que ya constituye un valor aceptable. Además, los valores de AUC se encuentran entre 0.7 y 0.8, cifras mucho mejores que las de HCR (con valores próximos a 0.5, el clasificador aleatorio) e incluso que OMD y STD (con valores en torno a 0.6). Igual que pasaba en el resto de corpus, apenas se encuentran diferencias entre los dos esquemas de pesos probados. En la Figura 4.2 (página 26) ya se puede ver una curva ROC con una forma más parecida a una función logarítmica que la de los experimentos con OMD. Esto resulta coherente con los valores que encontramos de AUC, mucho más altos que en OMD y que vienen dados porque la curva ROC se aproxima mucho más al punto (0,1) que en el otro caso, que toma una forma más lineal.

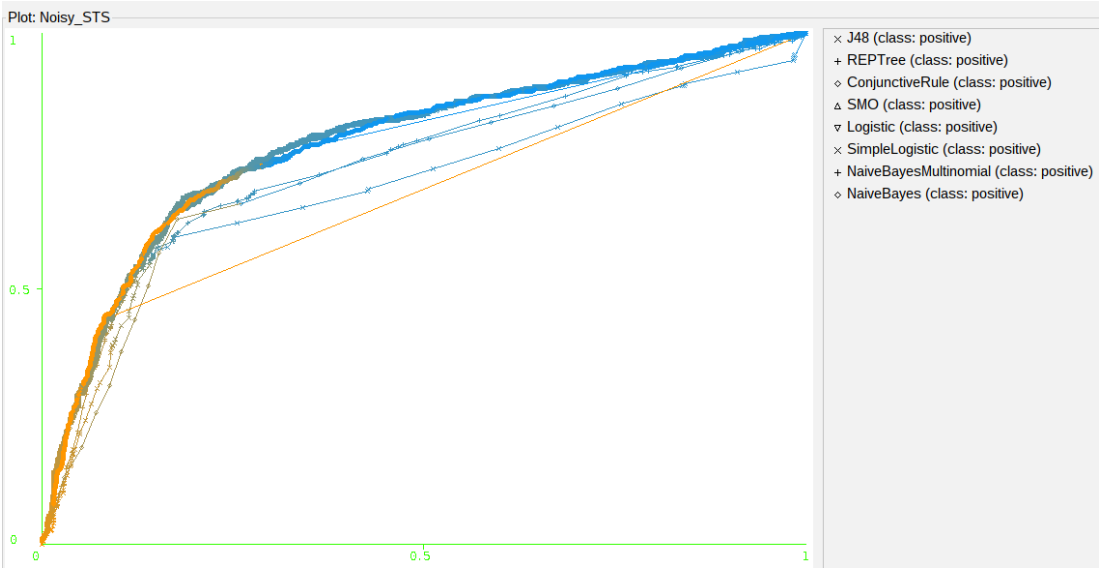

Figura 4.2: Curva ROC con el esquema de pesos BM25 y los tuits de STS-Gold como consultas.

4.4 Serie de experimentos Manual

En este segundo grupo de experimentos se ha prescindido del conjunto STD-Train. De este modo, ahora la forma de operar es la siguiente: para los datasets (HCR y TASS-2014) que incluyen de forma separada dos conjuntos, uno de entrenamiento, (E), y uno de test, (T), la colección de documentos indexada por del motor de búsqueda es E, y lo que se hace es emplear los tuits de T como consultas. Así, se obtiene el ranking de tuits en E, a partir del cual se obtienen los atributos con los que trabaja el clasificador. Para el resto de corpus, que no distinguen entre conjunto de entrenamiento y conjunto de test, se utiliza validación cruzada. Esto consiste en que se divide el dataset en 10 partes, que a su vez generan 10 experimentos (donde para cada uno de ellos se emplea una de esas partes como conjunto de test y las otras nueve como conjunto de entrenamiento). Los resultados finales a devolver se calculan como la media de los 10 experimentos.

Del mismo modo que en los experimentos Noisy (Sección 4.3, página 19), se han probado dos esquemas de pesos diferentes y ocho sistemas de clasificación. Además, las métricas de evaluación también son las mismas que en Noisy.

Esta sección consta en esta ocasión de cinco subsecciones, las correspondientes a los cuatro mismos corpus que en Noisy, más el corpus en español. Esta vez sí se puede utilizar el corpus en español porque al ser el mismo dataset el que se utiliza para crear el índice que del que se extraen las consultas, el índice está compuesto por tuits en español y se supone que parte en igualdad de condiciones frente a los corpus en inglés. [2]

4.4.1 HCR

Este es uno de los dos datasets que separan el conjunto de entrenamiento del de test, por lo que no fue necesario realizar validación cruzada. Se utilizó HCR-train para construir el índice y de HCR-test se extrajeron los tuits que se usaron como consultas. Este proceso tiene como resultado el archivo con los atributos con los que se entrenarán los clasificadores. Sus resultados utilizando los esquemas de pesos BM25 y PL2 se pueden ver en las Tablas 4.12 (página 28) y 4.13 (página 28), respectivamente. Como ocurría hasta ahora, los resultados no varían mucho al cambiar el esquema de pesos.

[2] Exceptuando detalles menores inevitablemente derivados del cambio de idioma, como el tokenizer utilizado influyen en el ranking. Por ejemplo, para los corpus en inglés se ha utilizado el tokenizer de PyTerrier *English-Tokenizer*, un tokenizer optimizado para textos en inglés; mientras que para el corpus en español se ha utilizado *UTFtokenizer*, un tokenizer genérico, ya que PyTerrier no incluye ningún tokenizer para el español.

Tabla 4.12: Resultados de Manual usando HCR como corpus y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.017	0.662	0.756	0.677	0.476
REPTree	-0.015	0.589	0.761	0.664	0.504
SVM	0	X	0.768	X	0.500
NB	0.013	0.650	0.531	0.567	0.492
NBM	0.057	0.681	0.474	0.506	0.642
RC	0	X	0.768	X	0.491
RLS	0	X	0.768	X	0.462
RLM	-0.011	0.618	0.759	0.666	0.538

Tabla 4.13: Resultados de Manual usando HCR como corpus y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	-0.004	0.636	0.759	0.669	0.486
REPTree	-0.004	0.636	0.759	0.669	0.467
SVM	0	X	0.768	X	0.500
NB	0.026	0.662	0.448	0.480	0.502
NBM	-0.026	0.632	0.519	0.556	0.465
RC	0	X	0.768	X	0.486
RLS	0	X	0.768	X	0.470
RLM	0.002	0.649	0.762	0.670	0.434

Se puede observar que en cualquier configuración, tanto de Manual como de Noisy, los resultados de HCR son muy malos. Conociendo solo los resultados de Noisy se podría pensar que es debido a que su temática es muy particular (una reforma sanitaria de EEUU), y dado que los tuits del índice son de temática general eso podría ser el factor que provoca los malos re-

sultados. Sin embargo, una vez vistos los resultados de Manual, donde se indexa con el mismo dataset del que se extraen las consultas, se puede concluir, igual que en [5], que simplemente este es un corpus cuyos tuits son muy difíciles de clasificar. Por otra parte, los resultados son, además, ligeramente peores que los obtenidos en los *baselines*. Dadas las características del corpus anterior, el HCR, era complicado sacar conclusiones del funcionamiento del sistema. Sin embargo, ya con el corpus OMD, ya se puede observar que Manual obtiene mejores resultados que Noisy. Con ambos esquemas de pesos el índice Kappa toma valores entre 0.341 y 0.473 (valores aceptables), mientras que en la versión Noisy este mismo estadístico toma valores en torno a 0.2, con algunos clasificadores muy próximos a 0.

4.4.2 OMD

En las Tablas 4.14 (página 29) y 4.15 (página 29) se pueden ver los resultados de este experimento.

Tabla 4.14: Resultados de Manual usando los tuits de OMD y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.403	0.731	0.735	0.727	0.722
REPTree	0.391	0.720	0.725	0.720	0.738
SVM	0.452	0.754	0.761	0.750	0.714
NB	0.386	0.714	0.713	0.713	0.767
NBM	0.348	0.698	0.688	0.691	0.726
RC	0.341	0.694	0.697	0.695	0.682
RLS	0.473	0.762	0.766	0.758	0.820
RLM	0.462	0.755	0.760	0.753	0.819

Tabla 4.15: Resultados de Manual usando los tuits de OMD y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.397	0.722	0.728	0.723	0.734

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.15 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
REPTree	0.398	0.722	0.729	0.724	0.744
SVM	0.431	0.754	0.754	0.739	0.699
NB	0.367	0.706	0.711	0.708	0.760
NBM	0.326	0.687	0.683	0.685	0.714
RC	0.416	0.734	0.740	0.731	0.693
RLS	0.446	0.753	0.756	0.747	0.815
RLM	0.451	0.752	0.757	0.749	0.815

Como ya ocurría previamente, apenas se aprecian diferencias entre los esquemas de pesos, ni tampoco hay ningún clasificador que sobresalga en ninguna de las métricas, ni por arriba ni por abajo. Como se ha dicho, los valores del índice Kappa se encuentran entre 0.341 y 0.473, los de precisión, *recall* y *F1* en torno a 0.7 y los de AUC entre 0.682 y 0.820; todos valores muy aceptables. Si se compara la curva ROC de esta aproximación (Figura 4.3, página 30) con la curva ROC del mismo corpus para Noisy (Figura 4.1, página 23), se puede apreciar claramente una forma muy distinta, con una curva más próxima al punto (0,1). De ahí que en este caso los valores de AUC resulten más elevados. Los resultados son superiores a los obtenidos en los *baselines* (Sección 4.2).

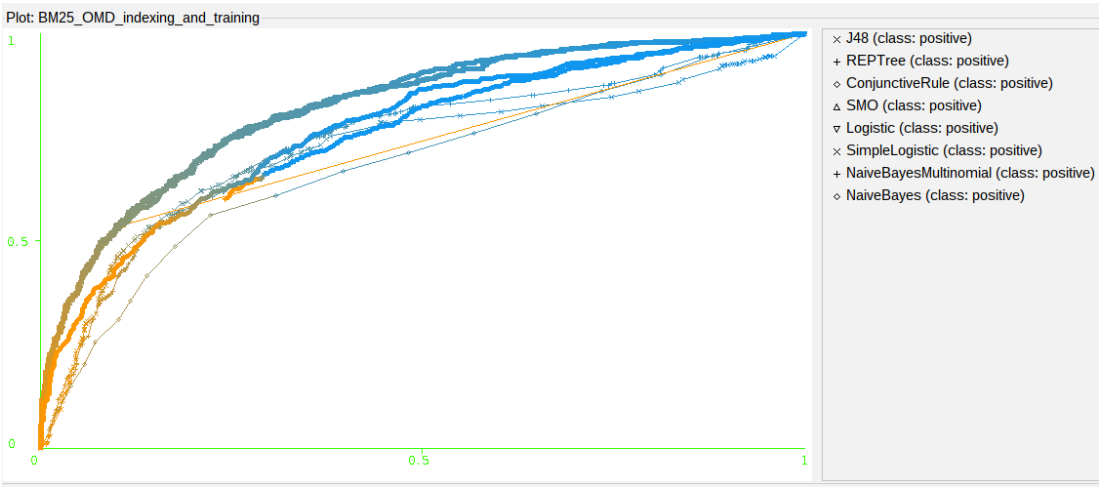

Figura 4.3: Curva ROC con el esquema de pesos BM25 y los tuits de OMD como índice y consultas.

4.4.3 STD

Originalmente, este dataset también está dividido en un conjunto de entrenamiento y un conjunto de test. Sin embargo, dado que STD-Train está etiquetado automáticamente,con los problemas que se pueden derivar de ello, se ha obviado ese conjunto y se ha usado tan solo STD-Test. De esta forma, el modo de proceder ha sido el mismo que con el resto de datasets que no diferencian en dichos conjuntos.

Tabla 4.16: Resultados de Manual usando los tuits de STD y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.422	0.726	0.712	0.707	0.699
REPTree	0.412	0.729	0.707	0.699	0.696
SVM	0.435	0.725	0.718	0.715	0.717
NB	0.385	0.711	0.693	0.685	0.738
NBM	0.463	0.733	0.731	0.731	0.763
RC	0.390	0.736	0.695	0.681	0.706
RLS	0.441	0.724	0.720	0.719	0.774
RLM	0.380	0.697	0.690	0.687	0.751

Tabla 4.17: Resultados de Manual usando los tuits de STD y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.418	0.723	0.709	0.704	0.736
REPTree	0.402	0.709	0.700	0.697	0.691
SVM	0.424	0.721	0.711	0.709	0.711
NB	0.401	0.720	0.701	0.694	0.737
NBM	0.453	0.728	0.726	0.725	0.764

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.17 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
RC	0.390	0.720	0.695	0.686	0.695
RLS	0.436	0.722	0.718	0.716	0.777
RLM	0.446	0.736	0.723	0.719	0.749

Se pueden comprobar los resultados obtenidos en las Tablas 4.16 (página 31) y 4.17 (página 31). De la misma forma que ya ocurría en Noisy, se trata de unos resultados similares a los de OMD y bastante mejores que los de HCR, y por tanto también mejores que los de los *baselines* (Sección 4.2). Además, igual que ocurría con OMD, también mejora los resultados obtenidos en Noisy utilizando el mismo corpus.

4.4.4 STS-Gold

En las Tablas 4.18 (página 32) y 4.19 (página 33) se pueden ver los resultados de estos experimentos. Siendo este el corpus que mejores resultados ofreció en los experimentos Noisy, se podría esperar que este volviera a ser el caso. Sin embargo, no solo no ocurre eso, sino que además es el único corpus con el que los resultados para Manual son peores que los obtenidos para Noisy. Respecto a Noisy no empeora de forma brusca, pero sí resulta sorprendente teniendo en cuenta que el resto de corpus ofrecen mucho mejores resultados en la versión Manual. Estos resultados son mejores que los obtenidos con HCR, pero ligeramente peores que los obtenidos con OMD y STD. A pesar de eso, los resultados son mejores que los de los *baselines* (Sección 4.2), sobre todo teniendo en cuenta que en este corpus en los *baselines* los dos clasificadores etiquetaron todos los tuits como negativos, y por tanto varias métricas no se pudieron calcular.

De nuevo, no se aprecian diferencias significativas entre los esquemas de pesos, y las diferencias entre los clasificadores son mínimas.

Tabla 4.18: Resultados de Manual usando los tuits de STS-Gold y BM25 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.328	0.717	0.732	0.719	0.695

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.18 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
REPTree	0.295	0.708	0.727	0.706	0.697
SVM	0.205	0.714	0.723	0.668	0.583
NB	0.384	0.736	0.740	0.737	0.744
NBM	0.361	0.728	0.711	0.718	0.732
RC	0.124	0.659	0.696	0.635	0.616
RLS	0.318	0.728	0.742	0.718	0.752
RLM	0.321	0.725	0.740	0.718	0.745

Tabla 4.19: Resultados de Manual usando los tuits de STS-Gold y PL2 como esquema de pesos.

	Kappa	Precisión	Recall	F1	AUC
J48	0.328	0.725	0.740	0.722	0.718
REPTree	0.317	0.712	0.728	0.715	0.703
SVM	0.206	0.715	0.724	0.669	0.583
NB	0.375	0.732	0.736	0.734	0.743
NBM	0.354	0.725	0.709	0.715	0.732
RC	0.040	0.644	0.690	0.588	0.528
RLS	0.314	0.727	0.741	0.715	0.748
RLM	0.305	0.719	0.736	0.714	0.744

4.4.5 TASS-2014

Este es el primero de nuestros experimentos en español. Sus resultados se pueden ver en las Tablas 4.20 (página 34) y 4.21 (página 34). Los resultados obtenidos son inferiores a los de los corpus anteriores, en inglés, independientemente de si el esquema de pesos es BM25 o PL2. Sin importar si se habla de la configuración con esquema de pesos BM25 o con PL2, los

resultados son bajos. Los clasificadores tienden a clasificar todas las instancias como positivas, como se ve en la Tabla 4.22 (página 35), distorsionando los resultados. Los valores del índice Kappa siempre están en torno a 0 (muy malos), llegando incluso a tomar valores negativos. Igual que ocurría con HCR, los resultados son ligeramente peores que los obtenidos en los *baselines* (Sección 4.2).

No se ha podido concluir si estos resultados son debidos al uso de otro *tokenizer*; a si este es un corpus difícil de clasificar, igual que ocurre con HCR; o si simplemente son cuestiones inherentes al idioma español, de forma que resulte más difícil de clasificar que el inglés.

Tabla 4.20: Resultados de Manual usando los tuits de TASS-2014 y BM25 como esquema de pesos.

	Kappa	**Precisión**	**Recall**	**F1**	**AUC**
J48	0.017	0.662	0.756	0.677	0.476
REPTree	-0.015	0.589	0.761	0.664	0.504
SVM	0	X	0.768	X	0.500
NB	0.013	0.650	0.531	0.567	0.492
NBM	0.056	0.681	0.474	0.506	0.518
RC	0	X	0.768	X	0.491
RLS	0	X	0.768	X	0.462
RLM	-0.011	0.618	0.759	0.666	0.538

Tabla 4.21: Resultados de Manual usando los tuits de TASS-2014 y PL2 como esquema de pesos.

	Kappa	**Precisión**	**Recall**	**F1**	**AUC**
J48	0	X	0.622	X	0.495
REPTree	-0.011	0.514	0.603	0.498	0.495
SVM	0	X	0.622	X	0.500
NB	-0.064	0.499	0.499	0.499	0.440

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.21 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
NBM	0.001	0.530	0.491	0.498	0.484
RC	-0.005	0.517	0.612	0.490	0.494
RLS	-0.003	0.386	0.620	0.476	0.497
RLM	-0.021	0.503	0.597	0.496	0.518

		Clase predicha	
		Negativo	**Positivo**
Clase real	**Negativo**	0	308
	Positivo	0	506

Tabla 4.22: Matriz de confusión de SVM con el corpus TASS-2014.

4.5 Experimentos multilingües

La última serie de experimentos corresponde al caso multilingüe. Para ello, se combinaron los conjuntos de entrenamiento de dos datasets, uno en inglés (HCR) y otro en español (TASS-2014) para construir la base documental a indexar, para luego lanzar sobre ella las consultas de los conjuntos de test de dichos datasets. Las métricas y los clasificadores son los mismos que hemos venido empleando hasta ahora. Los resultados pueden verse en las Tablas 4.23 (página 35) y 4.24 (página 36).

Tabla 4.23: Resultados del experimento multilingüe con esquema de pesos BM25.

	Kappa	Precisión	Recall	F1	AUC
J48	0.29	0.655	0.642	0.642	0.633
REPTree	0.261	0.635	0.636	0.635	0.631
SVM	0.333	0.671	0.667	0.668	0.668

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 4.23 – *(viene de la página anterior)*

	Kappa	Precisión	Recall	F1	AUC
NB	0.276	0.642	0.643	0.643	0.674
NBM	0.302	0.655	0.653	0.654	0.672
RC	0.299	0.665	0.643	0.641	0.653
RLS	0.319	0.664	0.661	0.662	0.685
RLM	0.301	0.655	0.654	0.654	0.681

Tabla 4.24: Resultados del experimento multilingüe con esquema de pesos PL2.

	Kappa	Precisión	Recall	F1	AUC
J48	0.285	0.573	0.735	0.644	0.631
REPTree	0.270	0.640	0.637	0.638	0.643
SVM	0.342	0.677	0.671	0.672	0.673
NB	0.323	0.666	0.663	0.664	0.681
NBM	0.287	0.647	0.646	0.647	0.674
RC	0.294	0.667	0.639	0.636	0.660
RLS	0.317	0.663	0.660	0.661	0.686
RLM	0.329	0.668	0.667	0.667	0.683

Lo primero que llama la atención es que, al contrario de lo que intuitivamente se podría pensar, los resultados son positivos. Sin duda mucho mejores que los de HCR o TASS en la configuración Manual. Mientras que en aquellos experimentos monolingües iniciales el índice Kappa siempre se encontraba en torno a 0, esta vez toma valores en torno a 0.3, una clara mejora. Asimismo, los valores de AUC en ningún caso bajan ahora de 0.631, mientras que en los experimentos monolingües para Manual se situaba en torno a 0.5 (clasificador aleatorio), en ocasiones incluso con valores ligeramente por debajo. Es difícil comparar estos resultados con los de los *baselines*, ya que para estos últimos no se realizó ningún experimento multilingüe. Lo único que se puede afirmar es que los resultados son mejores que los obtenidos para cualquier

corpus en los *baselines*.

		Kappa	Precisión	Recall	F1	AUC
HCR	**SVM**	0.012	0.775	0.656	0.524	0.505
	NB	0.012	0.775	0.656	0.524	0.505
OMD	**SVM**	0.011	0.769	0.636	0.498	0.504
	NB	0.053	0.577	0.624	0.556	0.536
STD	**SVM**	-0.067	0.455	0.469	0.433	0.466
	NB	-0.027	0.485	0.489	0.471	0.455
STS-Gold	**SVM**	0	X	0.689	X	0.500
	NB	0	X	0.689	X	0.500
TASS-2014	**SVM**	0	X	0.622	X	0.500
	NB	0	X	0.622	X	0.500

Tabla 4.1: Resultados de referencia para los *baselines*.

Capítulo 5

Metodología, planificación y costes

La metodología escogida para este proyecto es la metodología iterativa-incremental. Se profundiza en ella en la Sección 5.1. Acerca de su planificación y costes se puede leer la Sección 5.2 (página 40).

5.1 Metodología iterativa-incremental

Para el desarrollo de este proyecto se ha optado por el uso de una metodología iterativa-incremental. En este tipo de metodología, las tareas a realizar se agrupan en una serie de etapas que se repiten en ciclos, denominados incrementos. Se puede entender cada iteración como pequeños proyectos donde en cada una de ellas se repite un proceso de trabajo similar para proporcionar un resultado completo sobre el producto final.

En cada iteración el producto evoluciona a raíz de los resultados obtenidos en iteraciones anteriores, y se añaden nuevos requisitos o se mejoran los que ya fueron completados. Se puede ver un ejemplo ilustrativo en la Figura 5.1 (página 40).

Esta metodología surge para dar respuesta a las debilidades de la metodología en cascada[1]. Se ha escogido esta metodología porque encaja muy bien con la naturaleza del proyecto: Incremental porque en primer lugar se realiza el preprocesado de los textos, en segundo lugar se trabaja con el SE y por último con el clasificador basado en AA. Iterativa porque se comienza por un sistema básico y datasets tan solo en inglés, pero a medida que avanza el proyecto se añaden nuevas funcionalidades y características, tales como el empleo de textos en otros idiomas.

[1] Se puede leer una explicación de estas metodologías en [26].

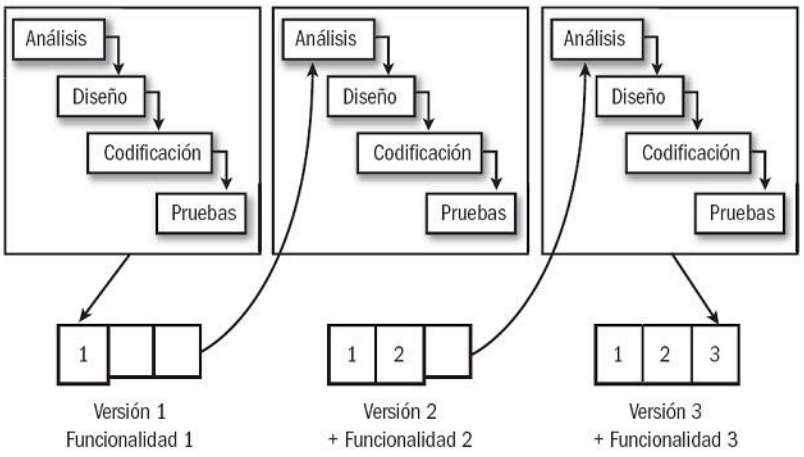

Figura 5.1: Ilustración del modelo de desarrollo iterativo-incremental.

5.1.1 Iteraciones

En esta sección se detallan las 3 iteraciones principales del proyecto.

- **Datasets.** Tareas relacionadas con los datasets. Elección, descarga y preprocesado.
- **Recuperación de Información.** Tareas relacionadas con el uso del Search Engine. Desde su elección hasta elección de esquemas de pesos y aprendizaje del framework.
- **Aprendizaje Automático.** Tareas relacionadas con el entrenamiento y test de los clasificadores de Aprendizaje Automático. Elección de la plataforma, aprendizaje de la misma y elección de los clasificadores y de las métricas de evaluación.

5.2 Planificación y costes

Se ha realizado un cálculo de los costes del proyecto. Para ello, se han calculado de forma separada los costes de los recursos técnicos y los de los recursos humanos, detallados en las Tablas5.1 y 5.2.

5.2.1 Recursos técnicos

Para la realización de este proyecto se han utilizado dos ordenadores, uno de sobremesa y el otro portátil, cuyas especificaciones y costes estimados se detallan en la Tabla 5.1.

CPU	GPU	Memoria principal	Coste estimado
Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz	NVIDIA GeForce GTX TITAN X	32 GB	1.600 €
Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz	NVIDIA GeForce MX150	8 GB	900 €

Tabla 5.1: Coste estimado de los dos ordenadores utilizados para el desarrollo del proyecto.

5.2.2 Recursos humanos

En este proyecto solamente se cuenta con un contribuyente, cuyos costes se detallan en la Tabla 5.2. Hay que tener en cuenta que es una simulación, ya que esta parte únicamente se desarrolla entre el estudiante y los directores del proyecto.

Recurso	Coste por hora	Número de horas	Coste total
Desarrollador	20 €	450	9.000 €

Tabla 5.2: Coste estimado de los recursos humanos partícipes en el proyecto.

De este modo, sumando los costes de los recursos técnicos y los recursos humanos, el coste total del proyecto asciende a 11.500 €.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

Como se ha expuesto a lo largo de este documento, se ha logrado cumplir el objetivo principal del proyecto: construir un sistema similar al presentado en [5], y a partir de ahí extenderlo para probar experimentos con textos en español y también experimentos multilingües (utilizando textos tanto en inglés como en español).

Tal y como se ha comentado a lo largo del Capítulo 4, los resultados obtenidos para los corpus en inglés concuerdan con los que se obtuvieron en [5], sin mostrar tampoco diferencias significativas entre los dos esquemas de pesos probados. En el caso del español, para el que los experimentos fueron más limitados al contar con un único dataset, el rendimiento del sistema fue inferior. No se pudo concluir si fue debido a las propias características de dicho corpus, si fue debido a las diferencias en el proceso de *tokenización*, o si simplemente son cuestiones inherentes al idioma español, que resulta más difícil de clasificar que el inglés. Los resultados más sorprendentes se obtuvieron en el caso de la serie de experimentos multilingües. Para la realización de esta serie de experimentos se utilizaron los dos datasets que peores resultados ofrecieron tanto en la versión Noisy como en la Manual, y sin embargo, los resultados obtenidos en el caso multilingüe superaron claramente a los resultados monolingües por separado.

El trabajo está claramente relacionado con la mención de Computación mediante el uso de Aprendizaje Automático y Recuperación de Información, dos campos muy importantes en la mención (especialmente el primero). Personalmente me ha servido para ampliar la experiencia en estos dos campos, además de conocer nuevos aspectos que no se detallan durante el grado.

6.2 Trabajo Futuro

Existen diversas líneas de actuación de cara a extender el trabajo desarrollado en esta primera aproximación. En vista de los resultados obtenidos, sería interesante probar otros

corpus en español para comprobar si los pobres resultados obtenidos utilizando el de TASS-2014 son debidos a que dicho corpus en particular es difícil de clasificar o si hay otras causas. De modo similar, podrían ampliarse los experimentos a más idiomas. También se podrían utilizar textos de otro tipo (reseñas de productos o servicios, por ejemplo), para así comprobar si el sistema es también multidominio.

Otro campo que se podría explorar es el de la clasificación no binaria, por ejemplo, ya que no todos los tuits presentan claramente una opinión positiva o negativa. De este modo, se podría crear una tercera clase *mixta* para estos tuits.

En la mayoría de los casos, la versión Noisy, en la que se utilizaron tuits automáticamente etiquetados, ha sido superada por la versión Manual, en la que tan solo se utilizaron tuits etiquetados a mano. Una posible extensión podría ser la aplicación de métodos para filtrar aquellos tuits mal clasificados automáticamente (por ironía o sarcasmo, entre otros motivos), esperando así aumentar la precisión a la hora de clasificar.

Finalmente, también se podría probar el sistema con un corpus en gallego. Esto es complicado ya que no existen (o por lo menos no he encontrado) corpus con tuits en gallego, por lo que sería necesario crear uno. Descargar los tuits y etiquetarlos de forma *noisy* es relativamente sencillo utilizando la API de Twitter, pero ya se vio que este tipo de etiquetado ofrece peores resultados que uno manual. Por ello, sería necesario incorporar expertos que se encargasen de etiquetar manualmente los tuits.

Apéndices

Lista de acrónimos

AA Aprendizaje Automático. 2, 5, 19, 39, 40, 42

AS Análisis de Sentimientos. iii, 3

AUC Area Under the Curve. 7, 18, 21, 23–26, 30, 36

ML Machine Learning. 5

MO Minería de Opiniones. 1–3

MSV Máquinas de Soporte Vectorial. iii, 1, 2, 13, 14, 18, 22, 24

NBC Naïve-Bayes. 1, 2

NLP Natural Language Processing. 1, 3

PLN Procesamiento de Lenguaje Natural. 1, 3

RI Recuperación de Información. 3, 4, 42

SE Search Engine. 2, 39, 40

SVM Support Vector Machines. 2

Glosario

baseline Valor conocido o inicial a partir del cual pueden compararse valores posteriores de lo que se está midiendo. 18

consulta Solicitud de datos o información que se realiza a un motor de búsqueda. 1, 2, 4

corpus Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación. 2, 16, 17

framework conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. 40

hashtag Término asociado a asuntos o discusiones que desean ser indexadas en redes sociales, insertando el símbolo de numeral (#) antes de la palabra, frase o expresión. Cuando la combinación es publicada, se transforma en un enlace que lleva a una página con otras publicaciones relacionadas con el mismo tema. 1, 2, 16

motor de búsqueda Aplicación práctica de técnicas de Recuperación de Información a colecciones de texto a gran escala. 1, 2, 4, 10, 27

tokenizer Primer elemento de un compilador o un motor de búsqueda, consistente en un programa que recibe como entrada el código fuente o texto y produce una salida compuesta de tokens o símbolos. 27

índice Estructura de datos que mejora la velocidad de búsqueda. 5

Bibliografía

[1] D. Sayce. (2022) The number of tweets per day in 2022. [En línea]. Disponible en: https://www.dsayce.com/social-media/tweets-day/

[2] M. M. Mostafa, "More than words: Social networks' text mining for consumer brand sentiments," *Expert Systems with Applications*, vol. 40, no. 10, pp. 4241–4251, 2013. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0957417413000328

[3] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[4] I. W. Tsang, J. T. Kwok, P.-M. Cheung, and N. Cristianini, "Core vector machines: Fast svm training on very large data sets." *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[5] A. U. Kauer and V. P. Moreira, "Using information retrieval for sentiment polarity prediction," *Expert Systems with Applications*, vol. 61, pp. 282–289, 2016. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0957417416302627

[6] G. Salton, "Automatic content analysis in information retrieval," *Cornell University*, 1968. [En línea]. Disponible en: https://hdl.handle.net/1813/5882

[7] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010, vol. 520.

[8] P. Trudgill and J. Chambers, "La dialectología," 1994.

[9] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[10] M. Kubat and Kubat, *An introduction to machine learning*. Springer, 2017, vol. 2.

[11] P. Yang, H. Fang, and J. Lin, "Anserini: Enabling the use of lucene for information retrieval research," in *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, 2017, pp. 1253–1256.

[12] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson, "Terrier information retrieval platform," in *European Conference on Information Retrieval*. Springer, 2005, pp. 517–519.

[13] A. B. Câmara and C. MacDonald, "Dockerising terrier for the open-source ir replicability challenge," *The Open-Source IR Replicability Challenge (OSIRRC 2019 (SIGIR'19 Workshop))*, 2019. [En línea]. Disponible en: http://ceur-ws.org/Vol-2409/docker02.pdf

[14] C. Macdonald and N. Tonellotto, "Declarative experimentation in information retrieval using pyterrier," in *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, 2020, pp. 161–168.

[15] S. Cass, "The 2018 top programming languages," *IEEE Spectrum*, vol. 31, p. 1, 2018.

[16] C. Macdonald and I. Ounis, "Voting for candidates: adapting data fusion techniques for an expert search task," in *Proceedings of the 15th ACM international conference on Information and knowledge management*, 2006, pp. 387–396.

[17] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[18] R. Moraes, J. F. Valiati, and W. P. G. Neto, "Document-level sentiment classification: An empirical comparison between svm and ann," *Expert Systems with Applications*, vol. 40, no. 2, pp. 621–633, 2013.

[19] A. Mccallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI-98 Workshop on 'Learning for Text Categorization'*, 1998.

[20] H. Saif, Y. He, M. Fernandez, and H. Alani, "Contextual semantics for sentiment analysis of twitter," *Information Processing & Management*, vol. 52, no. 1, pp. 5–19, 2016.

[21] N. F. Da Silva, E. R. Hruschka, and E. R. Hruschka Jr, "Tweet sentiment analysis with classifier ensembles," *Decision support systems*, vol. 66, pp. 170–179, 2014.

[22] J. V. Román, J. G. Morera, C. de Pablo Sánchez, M. A. G. Cumbreras, E. M. Cámara, A. U. López, and M. T. M. Valdivia, "Tass 2014-workshop on sentiment analysis at sepln-overview," in *TASS 2014-Workshop on Sentiment Analysis at SEPLN: Workshop proceedings:*

XXX Congreso de la Sociedad Espanola de Procesamiento de Lenguaje Natural SEPLN 2014, 2014, p. 1.

[23] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[24] H. Saif, M. Fernandez, Y. He, and H. Alani, "Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold," 2013.

[25] D. E. Parson. Data analytics, kappa statistic. [En línea]. Disponible en: https://faculty.kutztown.edu/parson/fall2019/Fall2019Kappa.html

[26] C. I. Rivas, V. P. Corona, J. F. Gutiérrez, and L. Hernández, "Metodologías actuales de desarrollo de software," *Revista de Tecnología e Innovación*, vol. 2, no. 5, pp. 980–986, 2015.