MÁSTER EN SOFT COMPUTING E INTELIGENCIA COMPUTACIONAL

Estudio y definición de tareas de preprocesamiento para clasificadores multietiqueta

Universidad de Granada - Curso 2010/2011

Francisco Charte Ojeda

Tutor: Francisco Herrera Triguero

Trabajo de investigación presentado por Francisco Charte Ojeda en septiembre de 2011 como parte del Máster en SCSI de la UGR

Estudio y definición de tareas de preprocesamiento para clasificadores multi-etiqueta

Francisco Charte Ojeda
Tutor: Francisco Herrera Triguero

Dpto. Informática
Grupo SIMIDAT - TIC-207
Universidad de Jaén
fcharte@ujaen.es

Resumen. La clasificación multi-etiqueta es una generalización de problemas bien conocidos, como son la clasificación binaria y la clasificación multi-clase, de forma que a cada instancia procesada se le asocia no una clase (etiqueta) sino un subconjunto de éstas. En los últimos años han surgido técnicas que, mediante la transformación de los datos o la adaptación de algoritmos, persiguen dar una solución a esta modalidad de clasificación relativamente reciente. En el presente trabajo se analizan varias técnicas de preprocesamiento de los datos específicas para clasificación multi-etiqueta, se describe un trabajo de experimentación realizado con los métodos de transformación más populares y se desarrolla una técnica novedosa de transformación basada en el uso de reglas de asociación dirigida a la reducción del espacio de etiquetas para el tratamiento de este tipo de problemas.

1 Introducción

El cada vez mayor volumen de documentos de todo tipo en la web, especialmente textos e imágenes, y la necesidad de clasificarlos adecuadamente en categorías no excluyentes entre sí, así como la disponibilidad de grandes bases de datos en áreas como la genética y la necesidad de analizarlas para obtener conocimiento útil, ha hecho crecer en los últimos años el interés por resolver este problema, surgiendo los denominados *clasificadores multi-etiqueta*. Si bien es una técnica que se origina en el campo de la categorización de documentos (Zhang y Zhou 2005), también resulta aplicable a otras tareas de similar interés como el etiquetado de mapas (Zhu y Poon 1999), el diagnóstico médico (Karalič y Pirnat 1991) o la bioinformática (Clare y King 2001).

En las diferentes publicaciones especializadas se ha abordado la clasificación multi-etiqueta principalmente a través de dos vías: 1) la reducción del problema a un conjunto de clasificadores binarios o multi-clase, mediante técnicas de transformación de datos como son BR (*Binary Relevance*) o LP (*Label Powerset)*; y 2) la modificación de algoritmos existentes a fin de dotarles de capacidad para tratar con varias etiquetas/clases simultáneamente, como es el caso de MLkNN, BP-MLL o el C4.5 multi-etiqueta. En los apartados 2.2 y 2.3 del capítulo 2 se analizan las principales propuestas de ambos enfoques.

Además de estas dos familias de técnicas para abordar el problema, también se han definido en la bibliografía especializada nuevas medidas que permiten evaluar la bondad de los clasificadores teniendo en cuenta el hecho de que cada instancia puede pertenecer a más de una clase. Asimismo existen medidas específicas que permiten analizar las características multi-etiqueta de un cierto conjunto de datos, tales como el promedio de etiquetas por muestra. Tanto unas como otras serán discutidas en el apartado 2.1 del mismo capítulo 2.

El objetivo de este trabajo es analizar el problema de clasificación multi-etiqueta y, más concretamente, aportar estudios y desarrollos dentro del área de transformación del problema. Se comienza haciendo una revisión del problema de clasificación multi-etiqueta, describiéndose las dos formas clásicas de abordarlo (transformando el problema o adaptando los algoritmos) y se describen algunas propuestas dentro de cada una de ellas.

Dado que nos centramos en el enfoque de transformación del problema, hacemos un primer estudio de aproximación a éste y analizamos el comportamiento de diferentes métodos de transformación del problema con varios algoritmos de clasificación y, en particular, uno propio basado en redes neuronales de función de base radial (RBFNs), un tipo de red neuronal que será descrita con detalle en el segundo capítulo. Como se verá (a partir de la experimentación realizada) no hay ningún método de transformación del problema de los analizados que pueda ser calificado como mejor que el resto, por lo que consideramos hacer alguna propuesta en este campo.

Para ello nos planteamos una serie de potenciales alternativas de preprocesamiento y, en particular, presentamos una metodología de reducción de la dimensionalidad del conjunto de etiquetas, que implica un preprocesamiento y un postprocesamiento en torno al algoritmo de clasificación que utilicemos y el uso de reglas de asociación. Realizamos una experimentación con un algoritmo de extracción de reglas de asociación y con diferentes algoritmos de clasificación y mostramos los resultados y las conclusiones que arrojan los mismos. El propósito final es determinar la influencia, positiva o negativa, de dichas tareas de preprocesamiento a la hora de abordar un problema de clasificación multi-etiqueta.

El presente trabajo tiene la siguiente estructura: tras esta introducción, en el segundo capítulo se describe en qué consiste la clasificación multi-etiqueta y las técnicas y medidas que para este problema concreto pueden encontrarse actualmente en la bibliografía especializada. Al final de dicho capítulo se describe un trabajo de análisis experimental sobre métodos de transformación de datos en un tipo de algoritmos evolutivos de diseño de redes neuronales. A continuación, en el capítulo 3, se definen los conceptos fundamentales relativos al uso de reglas de asociación y la minería de éstas, describiendo con detalle el algoritmo que se empleará en la experimentación posterior. El cuarto capítulo se centra en analizar, en

principio desde una perspectiva teórica, diferentes métodos de preprocesamiento tradicionales y la forma en que podrían adaptarse con el objetivo de producir mejoras en los clasificadores multi-etiqueta. La técnica propia diseñada para reducir la cardinalidad mediante reglas de asociación, junto con los resultados obtenidos a partir de la experimentación, se encuentran en la parte final de dicho capítulo. Finalmente, en la sección 5, se describen las conclusiones y se proponen potenciales investigaciones futuras a partir de este trabajo.

2 Clasificación multi-etiqueta

En el campo del *Machine Learning*, concretamente en el área conocida como *aprendizaje supervisado*, una de las aplicaciones más importantes es la clasificación. Partiendo de un conjunto de muestras o ejemplos etiquetados (*dataset*) se procede, a través de un proceso denominado *entrenamiento*, a obtener un modelo que sea capaz de etiquetar nuevas muestras no vistas durante la fase de entrenamiento (*test*). Tradicionalmente los clasificadores se han entrenado con conjuntos de datos en los que cada muestra tiene asociada una y solo una clase o *etiqueta*, siendo ésta el valor objetivo a obtener una vez que se ha construido el modelo.

Siendo L el conjunto de etiquetas aplicables a las instancias de un dataset y X_l el conjunto de muestras pertenecientes a una clase determinada, han de cumplirse las siguientes premisas:

$$\boldsymbol{L = \{l_1, l_2, \dots, l_k\},\ \ |L| = k > 1} \tag{1}$$

$$\boldsymbol{X_{l_a} \cap X_{l_b} = \emptyset,\ \ \forall l_a, l_b\ \ a \neq b} \tag{2}$$

La premisa (1) indica que el conjunto L debe tener al menos dos clases para que exista el problema, ya que de no ser así todas las muestras tendrían asociada la misma clase. En (2) se establece que los subconjuntos de instancias son disjuntos respecto a su clase o, dicho de otra forma, que a cada muestra le corresponde una única clase.

Cuando $|L|=2$ se dice que el clasificador es binario y las clases suelen identificarse como `cierto` o `falso`. Un ejemplo claro lo encontramos en los clasificadores empleados para procesar el correo electrónico separándolo en dos categorías: *spam* y *no spam*. Si $|L|>2$ se precisa un clasificador multi-clase, un algoritmo que no se limitará a dar un positivo o negativo, como haría un clasificador binario, sino un valor que, tras la adecuada interpretación, permitirá determinar la clase. En cualquier caso cada instancia pertenece exclusivamente a una de esas clases.

Al abordar un problema de clasificación multi-etiqueta (Tsoumakas y Katakis 2007) la premisa (2), que establece que los subconjuntos de instancias han de ser disjuntos respecto a la clase a que pertenecen, no se cumple, es decir, una muestra puede pertenecer a más de una clase. Como se indica en (3) el clasificador, una vez entrenado, ha de facilitar un conjunto Y, subconjunto de L, con las etiquetas asociadas a cada instancia de prueba. La premisa (2) da paso a la expresión (4) que nos dice que los subconjuntos de instancias no son necesariamente disjuntos respecto a su clase.

$$\boldsymbol{Y = f(x_i),\ \ Y \subseteq L} \tag{3}$$

$$\boldsymbol{X_{l_a} \cap X_{l_b} \neq \emptyset,\ \ \forall l_a, l_b} \tag{4}$$

Mientras que en la clasificación clásica el objetivo es asociar a cada muestra una clase entre $|L|$ posibles, por lo que el rango de valores de salida posibles está limitado por el número de clases existentes, en un clasificador multi-etiqueta hay potencialmente $2^{|L|}$ valores distintos posibles como salida, dado que podría tomarse cualquier combinación de etiquetas en L.

Un algoritmo de clasificación multi-etiqueta, ya esté basado en un método de transformación o en una adaptación de otro algoritmo, puede generar esta salida fundamentalmente por dos vías distintas, tal y como se explica en (Tsoumakas, Katakis y Vlahavas 2010), que son:

- **Bipartición**: La función empleada por el clasificador únicamente predice si cada una de las etiquetas está o no asociada a la muestra procesada, por lo que la salida es un vector de valores booleanos que parte el conjunto de etiquetas en dos subconjuntos disjuntos.

- **Ranking**: En este caso la función del clasificador ofrece como salida para cada etiqueta un valor real, una medida que permite ordenar la predicción obteniendo un *ranking* de etiquetas. La aplicación de un valor que actúa como umbral permite generar, a partir de ese ranking, la bipartición para determinar qué etiquetas se asocian a la muestra procesada.

Los algoritmos de clasificación tradicionales, basados en árboles, redes neuronales, máquinas de soporte vectorial o basados en instancias, están diseñados para ofrecer como salida un único valor: la clase a la que se predice pertenecerá la muestra procesada. Estos algoritmos no pueden emplearse directamente, tal cual, para afrontar un problema de clasificación multi-etiqueta. En la literatura especializada (Tsoumakas, Katakis y Vlahavas 2010) se proponen dos vías diferentes para abordar dicho problema:

- Llevar a cabo un proceso de transformación previa de los datos (preprocesamiento) que permita aplicar los algoritmos de clasificación ya conocidos, por ejemplo entrenando un clasificador para cada etiqueta que permita saber si una muestra pertenece o no a la misma.

- Introducir cambios en los algoritmos de clasificación agregando la capacidad de tratar el hecho de que cada muestra puede tener asociadas varias etiquetas, en lugar de solamente una. Se tendrían clasificadores preparados para ofrecer como salida un conjunto de etiquetas (ya sea en forma de partición binaria, de ranking o ambos).

Lógicamente cada enfoque aporta beneficios pero también cuenta con desventajas que es necesario conocer para poder elegir la mejor opción. En (de Carvalho y Freitas 2009) se ofrece una taxonomía completa de métodos dirigidos a la resolución del problema de la clasificación multi-etiqueta que, básicamente, coincide con la que acaba de hacerse en los párrafos anteriores, si bien la denominación dada difiere algo. En ella se construye una jerarquía de cuya raíz cuelgan dos ramas principales: métodos independientes del algoritmo (que son los antes referidos como de transformación de datos o *transformación del problema*) y métodos dependientes del algoritmo, consistentes en la adecuación de algoritmos ya existentes o el diseño de otros nuevos específicos para este tipo de tarea.

Antes de abordar los diferentes métodos, en puntos posteriores de este mismo apartado, es preciso, no obstante, describir las medidas de análisis de datasets multi-etiqueta y evaluación de clasificadores multi-etiqueta, ya que se hará referencia a las mismas de manera reiterada en el resto de este trabajo.

2.1 Medidas de análisis y evaluación

Las peculiaridades del problema afrontado exigen el uso de nuevas medidas que, por una parte, permitan analizar el grado en que los conjuntos de datos empleados *son* multi-etiqueta y, por otro, faciliten la evaluación de las nuevas técnicas de clasificación. Del primer grupo forman parte la cardinalidad y la densidad de etiquetas.

La *cardinalidad de etiquetas* es el número promedio de etiquetas por muestra y se obtiene mediante la siguiente expresión, siendo D el dataset a analizar e Y_i el conjunto de etiquetas asociadas a cada muestra:

$$Card(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i| \qquad (5)$$

La cardinalidad guarda una cierta relación con el número total de etiquetas. Si éste es pequeño la cardinalidad también lo será, necesariamente. Sin embargo, un gran número de etiquetas distintas no conlleva una alta cardinalidad.

La *densidad de etiquetas* es el número medio de etiquetas por muestra dividido entre el número total de etiquetas, lo cual permite tener una medida que es independiente del número absoluto de etiquetas existente en cada dataset. Tomando como base la expresión (5) no hay más que dividir el sumando entre el número total de etiquetas:

$$Dens(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|} \qquad (6)$$

Tanto la cardinalidad como la densidad de etiquetas son métricas que aportan información útil sobre la estructura del dataset, ya que permiten saber el grado en que éste *es* multi-etiqueta. En los experimentos realizados (descritos en apartados posteriores) se facilitarán estas medidas para cada dataset empleado y se discutirá sobre su influencia en los resultados obtenidos.

Además de las medidas de análisis el problema afrontado también precisa de nuevas medidas de evaluación, ya que las empleadas tradicionalmente en clasificación binaria o multi-clase, como la exactitud (*accuracy*), precisión (*precision*), retorno (*recall*) y medida-F (Sebastiani 2002) no resultan útiles cuando cada muestra procesada tiene asociada no una clase sino un conjunto de ellas.

En múltiples publicaciones sobre el tema, como son (Elisseeff y Weston 2001) y (Zhang y Zhou 2007) entre otras, se hace una agrupación de las medidas de evaluación multi-etiqueta según el tipo de salida generada por el clasificador: una partición binaria o un ranking. En el primer caso el clasificador facilita como resultado un vector binario, en el cada uno de los elementos indica la presencia o ausencia de una cierta etiqueta, sin mayor información. La mayoría de los clasificadores, sin embargo, usan una función con valores de salida reales, lo cual permite generar un ranking ordenado de etiquetas y obtener el conjunto de las que se predicen aplicando un cierto umbral de corte.

Las medidas que se citan en esos trabajos, y que se describen a continuación, son en su mayor parte introducidas originalmente por (Schapire y Singer 2000) en BoosTexter, un sistema para el etiquetado de textos en múltiples categorías.

Siendo *D* el conjunto de muestras, *L* el conjunto total de etiquetas, Y_i el subconjunto real de etiquetas de la muestra i-ésima, Z_i el subconjunto de etiquetas predichas para la muestra i-ésima y $f(x_i, y_l)$ la función con valor de salida real para una muestra respecto a una cierta etiqueta, las medidas de evaluación más usuales son las indicadas a continuación:

$$HammingLoss = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \,\triangle\, Z_i|}{|L|} \tag{7}$$

El operador $\triangle$ halla la diferencia simétrica entre dos conjuntos, equivale a la operación XOR en lógica booleana. Lo que esta métrica evalúa es el número de veces en que se clasifican de manera errónea parejas instancia-etiqueta, ya sea porque se predice una etiqueta que no corresponde para una instancia (falso positivo) o porque se deja de predecir una que sí debería aparecer (falso negativo). Al tratarse de una medida que cuantifica el error el objetivo es minimizar esta función, alcanzándose la clasificación perfecta cuando $HammingLoss(D)=0$**.** Puede aplicarse con clasificadores que generan como salida únicamente una partición binaria de las etiquetas, no un ranking (sin asociar valores reales a cada predicción).

La siguiente métrica también cuantifica el error cometido, pero basándose en la salida con valores reales producida por el clasificador para cada instancia respecto de una etiqueta:

$$OneError = \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\!\left[\left[\arg\max_{y \in Z_i} f(x_i, y)\right] \notin Y_i\right]\!\right] \tag{8}$$

Para cada muestra se obtiene la etiqueta que está al frente del ranking predicho por *f()* y se comprueba si está en el subconjunto de etiquetas reales que corresponden a esa instancia, incrementándose la suma de errores en caso de que no fuese así.

La métrica (8) sirve para medir el rendimiento de un clasificador multi-etiqueta respecto a la etiqueta que aparece al frente del ranking en cada muestra, sin tomar en consideración el resto. En contraposición, la medida (9) evalúa cuánto hay que avanzar en dicho ranking de media para obtener todas las etiquetas asociadas a cada muestra:

$$Coverage = \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\max_{y \in Y_i} \operatorname{rank} f(x_i, y)\right] - 1 \tag{9}$$

El valor mínimo teórico para esta métrica sería 0, pero únicamente se alcanzará para un problema de clasificación mono-etiqueta en el que las predicciones sean todas correctas. El objetivo es minimizarlo y no tiene un límite superior, magnitud en la que influirá el número total de etiquetas y la cardinalidad.

Tanto (8) como (9) son medidas que permiten realizar un análisis parcial del rendimiento de un clasificador multi-etiqueta, al centrarse una solamente en la etiqueta más alta del ranking y la otra en la posición en dicho ranking de las etiquetas que corresponden a cada muestra. Dependiendo de las características del conjunto de datos procesado es totalmente factible que un clasificador tenga un *OneError* muy bajo y por el contrario el *Coverage* sea alto, y viceversa. Por ello en (Schapire y Singer 2000) se propone el uso de la métrica *AveragePrecision* (10), procedente del campo de la IR (*Information Retrieval*), como alternativa para evaluar la efectividad de un clasificador multi-etiqueta:

$$AvgPrecision = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|\mathcal{P}_i|}{\operatorname{rank} f(x_i, y)} \qquad (10)$$

$$\text{con } \mathcal{P}_i = \{y' \in Y_i \mid \operatorname{rank} f(x_i, y') \leq \operatorname{rank} f(x_i, y)\}$$

Lo que se hace es calcular la proporción media de etiquetas pertenecientes a la muestra procesada que quedan, en el ranking de la función de predicción, por encima de una cierta etiqueta *y*. Es una medida a maximizar. Si todas las etiquetas del subconjunto real asociado a la muestra se encuentran en el ranking (sin ningún falso positivo intermedio) el resultado será 1.

Todas las métricas mencionadas, a excepción de *HammingLoss*, están diseñadas para trabajar sobre algoritmos de clasificación que facilitan un ranking de etiquetas (asocian a cada etiqueta predicha para una muestra un valor real). Para los que entregan únicamente un partición binaria en (Godbole y Sarawagi 2004) y (S. J. Zhu 2005) se proponen las mencionadas a continuación:

$$ClassificationAccuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}\delta(Y_i = Z_i) \qquad (11)$$

También conocida como *SubsetAccuracy*, la métrica (11) se caracteriza por ser extremadamente estricta, al exigir una coincidencia absoluta entre el conjunto de etiquetas que predice el clasificador y el que realmente corresponde a cada muestra. La expresión $\delta(Y_i = Z_i)$ se evalúa como 1 si se cumple la igualdad o como 0 en caso contrario, por lo que la presencia de un único falso positivo o falso negativo en Y_i se consideraría como un fallo absoluto en la clasificación.

$$Accuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} = \frac{TP}{TP + FN + FP} \qquad (12)$$

Esta medida es una versión relajada de la anterior, agregando por cada muestra el porcentaje de las predicciones que han sido correctas. El cardinal de la intersección de los dos subconjuntos de etiquetas, el predicho y el real, equivale al número de verdaderos positivos producidos por el clasificador. La unión de esos mismos dos conjuntos agregaría también los falsos negativos y falsos positivos, por lo que el denominador siempre será igual o superior al numerador.

$$Precision = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Z_i|} = \frac{TP}{TP + FP} \quad (13)$$

La precisión del clasificador se calcula como el porcentaje de las predicciones positivas que han sido correctas y, puesto que no se toman en consideración para el cálculo los falsos negativos, siempre entregará resultados iguales o superiores que la métrica (12) para un mismo resultado generado por el clasificador.

$$Recall = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i|} = \frac{TP}{TP + FN} \quad (14)$$

Finalmente, esta medida indica el porcentaje de casos positivos que han sido recuperados respecto del total, lo cual permite evaluar la cantidad de predicciones incompletas del clasificador multi-etiqueta. Se trata de una medida habitual también en técnicas de recuperación de información (IR), principalmente documentos de texto.

Las medidas de evaluación anteriores se agrupan en una categoría que (Tsoumakas, Katakis y Vlahavas 2010) denominan *example-based* (basada en ejemplos), ya que se calculan de manera independiente para cada muestra de datos y después se promedia para todas las instancias de la partición empleada. En contraposición, en (Yang y Liu 1999) se definen las medidas *Macro-averaging* y *Micro-averaging* que, al ser aplicadas al problema de la clasificación multi-etiqueta, se calculan de manera independiente para cada etiqueta (no cada muestra) obteniéndose después el promedio para todas las etiquetas. Por ello (Tsoumakas, Katakis y Vlahavas 2010) llaman a dichas medidas *label-based* (basada en etiquetas).

Como puede apreciarse hay una gran cantidad de medidas de evaluación disponibles, con independencia de que la salida del clasificador sea una partición binaria o un ranking. A éstas podrían sumarse otras, definidas ad-hoc por algunos autores para evaluar sus propuestas, que habitualmente no son de aplicación general. En cualquier caso, tanto en la bibliografía revisada como en el software empleado en la experimentación: MULAN (Tsoumakas, Spyromitros-Xioufis, y otros 2011) dos de las medidas más usadas y tomadas como referencia son *HammingLoss* y *AveragePrecission*.

2.2 Transformación de datos (métodos independientes del algoritmo)

También conocidos como métodos de *transformación del problema*, el objetivo de los métodos de transformación de datos es obtener, a partir de los originales, nuevos conjuntos de datos tratables por los algoritmos de clasificación tradicionales, con una única etiqueta asociada a cada muestra. Para ello se ha de llevar a cabo un trabajo de preprocesamiento de los datasets, obteniendo como resultado un nuevo dataset o un conjunto de ellos, según los casos, siendo cada uno de ellos binario respecto a la clase (solamente se ha de predecir si cada muestra pertenece o no a la clase) o bien multi-clase (existen varias clases posibles, de las cuales cada muestra pertenece a una).

Algunas propuestas descritas en (Tsoumakas, Katakis y Vlahavas 2010) son:

- Descartar todas aquellas instancias de datos que tengan asociada más de una etiqueta, usando el resto para entrenar el clasificador. El resultado de esta transformación sería la obtención de un nuevo dataset de tipo multi-clase, con una sola etiqueta por muestra, y con igual o menor número de instancias que el original.

- Seleccionar aleatoriamente una de las etiquetas asociadas a cada muestra, ignorando el resto durante el entrenamiento. La aplicación de esta transformación produciría, aunque por distintos medios, el mismo resultado que la anterior: un dataset multi-clase, si bien en este caso el número de instancias respecto al original se mantendría siempre.

- Considerar el subconjunto de etiquetas asociado a cada muestra como una etiqueta única, a la que se asignaría una nueva denominación que podría obtenerse de la combinación de las originales (Boutell, y otros 2004). Produce, como las dos anteriores, un único dataset de tipo multi-clase.

- Separar las etiquetas existentes en el dataset mediante técnicas de tipo OVA (*one-vs-all*) u OVO (*one-vs-one*) a fin de poder usar clasificadores binarios (Godbole y Sarawagi 2004) o generar un ranking a partir de clasificadores centrados en el aprendizaje de etiquetas individuales (Crammer y Singer 2003) y (Hüllermeier, y otros 2008). Este tipo de transformación producirá, a partir del dataset origen, tantos datasets como etiquetas haya en uso en las muestras de datos, lo que en el algunos casos extremos (véase el apartado 2.4) puede provocar que de un dataset se pase a tener 4.000 datasets diferentes y un clasificador para cada uno de ellos.

Las dos primeras técnicas generan una importante pérdida de información en el dataset, por lo que no es de esperar que arrojen buenos resultados. No hay constancia de pruebas experimentales de su aplicación en la bibliografía consultada.

2.2.1 Partición del dataset original en un conjunto de datasets por etiqueta

El uso de clasificadores binarios para reducir un problema de clasificación multi-etiqueta se basa en el uso de una técnica de tipo OVA u OVO, según el cual se entrenarían tantos clasificadores como etiquetas distintas existan en el dataset. Cada clasificador indicaría la pertenencia o no de una muestra a una clase concreta. El último paso de un clasificador de este tipo sería realizar una unión de los resultados de cada clasificador binario para obtener el subconjunto de etiquetas definitivo predicho para cada muestra.

En (Godbole y Sarawagi 2004) se introduce un método de este tipo, al que se denomina *ensemble* de clasificadores binarios de tipo *one-vs-others*. En (de Carvalho y Freitas 2009) denominan a este método como *transformación basada en etiquetas*, mientras que (Tsoumakas y Katakis 2007) hacen referencia a él como *Binary Relevance* (BR), siendo ésta la denominación empleada en el presente trabajo. Es un método que tiene las siguientes propiedades:

- Es una técnica simple que puede ser implementada rápidamente y permite usar cualquier algoritmo de clasificación tradicional como clasificador subyacente.

- Según pruebas documentadas en la bibliografía especializada ofrece un buen nivel de precisión.

- Desestima por completo la relación existente entre las etiquetas, ya que cada clasificador se entrena únicamente respecto a una etiqueta, sin tomar en consideración las demás. Esa relación aporta información que teóricamente permitiría mejorar los resultados.

- Es necesario procesar el dataset tantas veces como etiquetas existan, una con cada método de aprendizaje, lo que multiplica el tiempo de ejecución, es decir, se incrementa la complejidad computacional del algoritmo base de forma lineal al número de etiquetas.

En (Godbole y Sarawagi 2004) se apunta la posibilidad de emplear para clasificación multi-etiqueta el método propuesto en (Crammer y Singer 2003) con el objetivo de etiquetar textos. El problema original era la clasificación de documentos de texto en múltiples categorías, proponiéndose para el mismo una familia de algoritmos de gradación de temas (*topic-ranking*) con los que puede estar etiquetado cada documento. El clasificador propuesto funciona como un perceptrón, ajustando un peso durante el entrenamiento y ofreciendo un valor real como salida para cada etiqueta. La unión de los resultados producidos por el grupo de clasificadores, y su ordenación, facilita la obtención del ranking final.

Otra propuesta para la generación de rankings de etiquetas partiendo del aprendizaje de clasificadores independientes para cada etiqueta es el denominado RPC (*Ranking by Pairwise Comparison*) introducido en (Hüllermeier, y otros 2008), en el que se comparan las etiquetas por pares para obtener el ranking final. Es la misma idea de la que parten otros algoritmos similares, de los mismos autores, mencionados en el apartado 2.3.5. En ese mismo apartado se describen también varios algoritmos que, en cierta manera, transforman el problema de clasificación original de manera que puede abordado mediante el encadenamiento (*chains*) y agrupaciones (*ensembles*) de clasificadores binarios o multi-clase y que no se incluyen aquí para evitar su duplicación.

La principal desventaja de los métodos de transformación que descomponen el dataset multi-etiqueta en varios datasets binarios, como es el caso de BR, es que no aprovechan la potencial influencia que la presencia de una cierta etiqueta tiene sobre la aparición de otras. Al explotar esas dependencias entre etiquetas el grado de acierto podría mejorarse. También es destacable el incremento en el tiempo de cómputo, un factor que depende exclusivamente del número total de etiquetas, sin que influya la cardinalidad o la densidad de etiquetas.

2.2.2 Obtención de un dataset multi-clase a partir del dataset multi-etiqueta

El método de combinación de etiquetas es introducido en (Boutell, y otros 2004) con la denominación *MODEL-n*, designando la *n* final el hecho de cada combinación de etiquetas que aparezca en las muestras de datos se convertirá en una *nueva* clase. Este mismo método es llamado *Label-Powerset* (LP) por (Tsoumakas y Katakis 2007) y transformación basada en la creación de nuevas etiquetas en (de Carvalho y Freitas 2009).

Como el dataset original genera un nuevo y único dataset, en el que cada subconjunto de etiquetas que aparezca asociado a las muestras del dataset es anotado como si fuese una sola etiqueta, no precisa el entrenamiento de múltiples clasificadores como las técnicas del apartado previo. Por ello es una técnica que:

- Permite utilizar, sin más trabajo, cualquier algoritmo clasificador multi-clase (puede haber más de dos clases en total).

- Mantiene, en cierta forma, información sobre la relación existente entre las etiquetas al generar nuevas etiquetas que representan la combinación de las originales.

- Potencialmente pueden existir, como se indicó anteriormente, $2^{|L|}$ combinaciones posibles, por lo que el número de clases podría llegar a ser intratable.

- Cuanto mayor sea el número de combinaciones distintas, menor el número de muestras de cada una, lo cual afectaría al entrenamiento del clasificador al surgir un problema de desbalanceo.

En cuanto a la transformación que genera nuevas etiquetas que representan combinaciones de las etiquetas originales, generando un dataset de tipo multi-clase pero con una sola etiqueta por instancia, incluso en aquellos casos en los que la cardinalidad no es alta, si el número total de etiquetas sí lo es una transformación LP puede acabar produciendo un problema mayor aún que el que se tenía al principio. El número de potenciales combinaciones es tan alto que se puede acabar por tener un dataset en el prácticamente cada instancia tiene asignada una clase diferente, lo cual hará extremadamente difícil el trabajo del clasificador subyacente.

En ambos casos la posibilidad de reducir el número de etiquetas, como la propuesta que se presenta en el capítulo 4, resulta de gran interés, ya que ayudará a paliar en parte tanto el aumento de complejidad computacional como la explosión de combinaciones de etiquetas.

2.3 Adaptación de algoritmos (métodos dependientes del algoritmo)

Los métodos de transformación descritos en el punto previo hacen posible afrontar el problema de la clasificación multi-etiqueta empleando algoritmos que no están diseñados para las especificidades de dicha tarea. Frente a esa opción, el enfoque de la adaptación de algoritmos tiene como objetivo adecuar algoritmos existentes a fin de que puedan tratar directamente muestras con múltiples clases, sin precisar preprocesamiento alguno.

En los últimos años el número de propuestas publicadas en este sentido ha ido incrementándose de manera notable. En los siguientes puntos se mencionan las más destacables, agrupadas según el método base del que parten. Hay que decir que la experimentación aportada para muchas de ellas es bastante específica, como la clasificación de imágenes o de genes según su funcionalidad, y se reduce con frecuencia a pruebas sobre un único dataset y comparación respecto a un solo algoritmo ya existente tomado como referencia.

2.3.1 Métodos basados en árboles

Los árboles de decisión (DT, *Decision Trees*) son uno de los mecanismos más sencillos de clasificación, a pesar de lo cual ciertos algoritmos de este tipo, como es el caso del conocido C4.5 (Quinlan 1992), obtienen resultados que les permiten competir con otros clasificadores.

En (Clare y King 2001) los autores modifican el algoritmo C4.5 con el objetivo de clasificar genes de acuerdo a su función. Un mismo gen puede influir en varias funciones y, por tanto, estar asociado a más de una clase (etiqueta). Se trata, como puede apreciarse, de un problema de clasificación multi-etiqueta.

La adaptación tiene dos puntos clave: las hojas del árbol ahora pueden almacenar un conjunto de clases, en lugar de una sola, y la medida de entropía original se ha modificado para tomar en consideración la probabilidad (frecuencia relativa) de que la muestra procesada no pertenezca a una cierta clase. De esta forma el particionado de cada conjunto de muestras (a medida que va generándose el árbol) de acuerdo con un cierto atributo se calcula como suma ponderada de la entropía para cada subconjunto posible. En dicha ponderación si un cierto elemento aparece dos veces en un subconjunto, porque pertenezca a más de una clase, entonces se suma esas dos veces. El hecho de cada hoja se corresponda con un conjunto de clases influye tanto en el proceso de etiquetado de los nodos del árbol como el posterior proceso de poda propio del algoritmo C4.5.

Finalmente, una vez completado el árbol, los autores obtienen a partir de él una serie de reglas que aplican a la clasificación funcional de los genes. Dado que cada hoja se asocia a un conjunto de clases el proceso de generación de reglas también se ha adaptado, produciendo por cada hoja tantas reglas como clases contenga. La experimentación realizada se limitó al dataset `yeast` y el resultado fue la predicción de funciones en 83 genes cuya función se desconocía hasta ese momento, con una precisión estimada superior al 80%.

El enfoque seguido por (Comité, Gilleron y Tommasi 2003) parte de la elección como algoritmo base de un ADT (*Alternate Decision Tree*), una generalización de los árboles de decisión tradicionales introducida en (Freund y Mason 1999) y que propone una alternativa al uso de técnicas de *boosting* para mejorar la precisión de los clasificadores basados en árboles. La adaptación de ADT al problema de la clasificación multi-etiqueta se lleva a cabo mediante una descomposición con la técnica OVA, por lo que podría decirse que es un algoritmo que incorpora en sí mismo la transformación de los datos.

2.3.2 Métodos basados en instancias

A diferencia de otros algoritmos de aprendizaje los basados en instancias no construyen un modelo a través de un proceso previo de entrenamiento, sino que operan de manera *perezosa* seleccionando los *k* vecinos más cercanos (*kNN*) en el momento en que se ha de procesar una nueva muestra. Existen varias adaptaciones de esta técnica al campo de la clasificación multi-etiqueta.

En (Zhang y Zhou 2007) se describe el algoritmo ML-kNN (*Multi-Label k Nearest Neighborhoods*), de tipo kNN, en el que para cada instancia a clasificar se toman los *k* vecinos más cercanos, se genera un ranking de las etiquetas que tienen asignadas y se usa dicha información para calcular la probabilidad de aparición de cada etiqueta en la muestra a clasificar. Utilizan como medida de distancia entre vecinos la distancia euclídea, lo cual demuestra que ésta es una buena medida dados los resultados obtenidos. El cálculo de la pertenencia o no de una muestra a una etiqueta se realiza con técnicas bayesianas, calculando una probabilidad a priori, determinada por la frecuencia de cada etiqueta en toda la población de entrenamiento, y una probabilidad a posteriori, calculada con las etiquetas de la vecindad de la muestra a clasificar. La experimentación está documentada exhaustivamente y muestra que ML-kNN supera en general en todas las medidas a algoritmos multi-etiqueta genéricos como BoosTexter (Schapire y Singer 2000), AdtBoost.MH (Comité, Gilleron y Tommasi 2003) y Rank-SVM (Elisseeff y Weston 2001).

Basándose en ML-kNN, los autores de (Cheng y Hüllermeier 2009) proponen dos algoritmos similares que mejoran al anterior aplicando métodos bayesianos. El artículo parte de que el algoritmo ML-kNN funciona como un clasificador BR (*Binary Relevance*), creando un clasificador binario para cada etiqueta sin tener en cuenta las correlaciones entre éstas, a pesar de lo cual obtiene resultados muy buenos. El algoritmo que proponen considera las etiquetas asociadas a los vecinos más cercanos de la instancia a clasificar como características adicionales de dicha instancia. Con esta información calculan unas probabilidades a priori y obtienen una ecuación de regresión. A partir de ahí plantean dos variantes del algoritmo: IBLR-ML (*Instance-based Logistic Regression for Multi-label Classification*) e IBLR-ML+.

En la experimentación comparan su propuesta con los métodos de transformación BR y LP tomando como base el algoritmo de clasificación C4.5, así como con ML-kNN que era considerado el estado del arte en clasificadores multi-etiqueta basados en instancias en aquel momento, demostrando que IBLR-ML los supera en precisión en todos los casos. Para el correcto funcionamiento del algoritmo propuesto, sin embargo, es necesario ajustar previamente un parámetro α que determina el peso de esos atributos adicionales en el cálculo de la probabilidad a posteriori, mediante un proceso de adaptación que exige la exploración de todo el dataset para aplicar el método de estimación de parámetros estadísticos ML (*Maximum Likelihood*).

2.3.3 Métodos basados en redes neuronales

Las redes neuronales artificiales (RNA) en general, y las RBFNs (un tipo de RNA que se estudiará con mayor detalle en el punto 2.5 de este mismo capítulo) en particular, han demostrado su efectividad en problemas de clasificación, regresión y predicción de series temporales, por lo que su adaptación para abordar de manera satisfactoria la clasificación multi-etiqueta es uno de los temas que más publicaciones ha ocupado recientemente.

Partiendo de uno de los modelos de RNA más básicos, como es el perceptrón, y el algoritmo de aprendizaje más popular: *Back-Propagation*, en (Zhang y Zhou 2006) se describe el algoritmo BP-MLL, una adaptación para clasificación multi-etiqueta que está considerada la primera aplicación de las RNA a este campo. El aspecto clave de BP-MLL estriba en la introducción de una función propia para el cálculo del error cometido adaptada al hecho de que cada muestra tiene asociadas varias etiquetas, concretamente penalizando las predicciones en que se colocan en el ranking etiquetas que no corresponden con la muestra procesada. La capa de entrada tiene tantas neuronas como atributos de entrada y la de salida tantas como etiquetas existan. La capa oculta se compone de neuronas con función sigmoidal y su número está en función también del número de etiquetas.

A fin de determinar el conjunto de etiquetas predichas para la muestra procesada, el algoritmo BP-MLL usa un parámetro que actúa como umbral de corte para el ranking que se obtiene a la salida de la RNA. La dificultad que plantea el ajuste de dicho parámetro es resuelta en (Grodzicki, Mandziuk y Wang 2008), incorporándolo a la función de cálculo del error de forma que el umbral va adaptándose automáticamente durante el proceso de aprendizaje. En realidad se genera un umbral personalizado para cada una de las etiquetas, en lugar de recurrir a uno general como en BP-MLL. Este hecho contribuye a que la versión modificada mejore al algoritmo original en la experimentación realizada que, como en muchas otras publicaciones, se reduce al dataset `yeast`.

En el artículo (M.-l. Zhang 2009) se propone un algoritmo para el diseño de RBFNs, llamado ML-RBF, especializado para el tratamiento de datasets multi-etiqueta. Tomando las muestras asociadas a cada etiqueta, ejecuta el algoritmo K-medias tantas veces como etiquetas distintas haya para realizar un clustering que servirá para obtener los centros de las RBFs. Un parámetro α determina el número de clusters por etiqueta. El número de neuronas en la capa interna es igual o superior al de etiquetas en el dataset, dependiendo de dicho parámetro. El entrenamiento se completa ajustando los pesos hacia las neuronas de salida (una por etiqueta) mediante el método SVD (*Singular Value Decomposition*) minimizando una función de suma de los cuadrados del error cometido.

La activación de todas las neuronas está fijada a 1 y existe un sesgo (*bias*) por cada etiqueta que se aplica a todas las neuronas asociadas. El habitual parámetro σ es calculado en una ecuación en la que intervienen la distancia media entre cada par de vectores prototipo y un parámetro de escalado μ. El ajuste de este parámetro y de α afecta de manera fundamental al algoritmo como se muestra en unas gráficas del propio artículo. La batería de pruebas documentadas indica que ML-RBF es competitivo, en cuanto a precisión se refiere, respecto a otros algoritmos multi-etiqueta y que es más rápido que el BP-MLL (un perceptrón multicapa para este tipo de problema que supera al ML-RBF en precisión en algunos casos).

2.3.4 Métodos basados en SVM

Las máquinas de soporte vectorial o SVM han sido aplicadas tradicionalmente a problemas de clasificación binaria pero, como ha ocurrido con otras técnicas, a lo largo del tiempo se han incorporado a las mismas extensiones y mejoras que han permitido su uso en otras áreas, como es la de la clasificación multi-etiqueta.

El objetivo de los autores de (Boutell, y otros 2004) es el etiquetado de escenas naturales en las que pueden aparecer múltiples objetos y, en consecuencia, ser asociadas a más de una clase: `urbana`, `playa`, `campo`, `montaña`, `playa+campo`, `campo+montaña`, `playa+urbana`, etc. Para ello optan por una SVM porque está probado empíricamente, en la bibliografía especializada, que tienen mejor comportamiento en la clasificación de imágenes y textos que otras técnicas como podrían ser las RNA. Para adecuar la SVM al problema multi-etiqueta se define un modelo al que los autores denominan *MODEL-x*, consistente en usar varias veces los datos para entrenar un clasificador por etiqueta. El aspecto más interesante de la propuesta es precisamente el método de entrenamiento empleado, denominado *cross-training*, que radica básicamente en que las muestras que pertenecen a varias clases se tratan como positivas al entrenar cada modelo individual, no como negativas como ocurriría al considerar la combinación de sus etiquetas como una clase diferente. Según las pruebas experimentales realizadas, esta idea consigue mejorar la precisión cuando el entrenamiento se ha de llevar a cabo con pocas muestras disponibles por etiqueta (algo habitual al usar la transformación LP).

En uno de los apartados de (Elisseeff y Weston 2001) también se expone una técnica basada en SVM que los autores denominan *Rank-SVM*. Es la solución que proponen para aquellos casos en los que existen correlaciones entre las etiquetas asociadas a cada muestra, caso en el que, como apuntan, los clasificadores binarios no ofrecen los mejores resultados. Por ello definen una medida que les permite obtener un sistema ranking de etiquetas minimizando una métrica denominada AHL, una aproximación lineal de la *HammingLoss*, considerando un modelo SVM en el que una serie de hiperplanos separan unas clases de otras. Este sistema de ranking se complementa con una función predictora que ajusta el umbral o *threshold*, permitiendo así ofrecer un subconjunto de etiquetas para cada muestra procesada. En la experimentación, con un dataset médico relativo a tumores, la conclusión es que a pesar de que Rank-SVM toma en consideración la correlación entre etiquetas, sus resultados no son mejores que la del enfoque binario cuando los datos tienen unas ciertas características: un gran número de características y pocas instancias para entrenar. En las pruebas con el dataset `yeast`, por el contrario, Rank-SVM sí que obtiene resultados mucho mejores que la alternativa basada en clasificadores binarios.

2.3.5 Otros métodos

Además de las agrupadas en las categorías anteriores, en la bibliografía relativa a la clasificación multi-etiqueta es posible encontrar propuestas basadas en otras ideas como el encadenado de clasificadores (Read, y otros 2009) o el uso de técnicas estadísticas para aprovechar la correlación entre etiquetas (Ghamrawi y McCallum 2005), entre otras.

Tanto la clasificación multi-clase como multi-etiqueta pueden enfocarse como casos particulares del *label ranking*. En el primer caso se crea el ranking y se toma únicamente la primera etiqueta, la que está en lo alto del ranking, mientras que en el segundo se tomarían aquellas que superan un cierto umbral o *threshold*. Los algoritmos de ranking multi-etiqueta se limitan a crear una lista ordenada por una cierta medida de las posibles etiquetas que puede tener una muestra, pero sin facilitar un punto que indique a partir de cuál de ellas debería cortarse, es decir, qué etiquetas deberían tomarse y cuáles no. Dado que están ordenadas según la citada medida, basta con conocer el punto de corte o umbral.

El método propuesto en (Fürnkranz, y otros 2008) introduce una etiqueta ficticia que representa una medida nula o punto cero cuyo objetivo es facilitar la calibración del clasificador, de forma que las etiquetas por encima de ella serían las predichas finalmente y las que estén por debajo se descartan. El método, llamado RPC (*Ranking by Pairwise Comparison*), usa un modelo para cada par de etiquetas que determina si una está por encima de la otra o viceversa (en el ranking). El resultado de todos los modelos genera el ranking global tomando el de cada pareja como un voto. Aplican esta técnica sobre un algoritmo previo, denominado MLPC (*Pairwise Multilabel Perceptron*), obteniendo una adaptación llamada CMLPC (*Calibrated MLPC*) que se propone como mejora.

En (Ghamrawi y McCallum 2005) se destaca el hecho de que los métodos basados en clasificadores binarios no tienen en cuenta las correlaciones existentes entre las etiquetas (asumen que son independientes), es decir, entre los valores asignados a las instancias como resultado de la clasificación, proponiéndose un método que se apoya precisamente en esa información para mejorar la precisión de los resultados.

Para recoger las correlaciones entre etiquetas se usa un CRF (*Conditional Random Field*), un modelo probabilístico, usado habitualmente en segmentación de textos, de tipo discriminativo que asocia una probabilidad a una etiqueta dependiendo de las observaciones realizadas. Con él se representan las dependencias entre los valores de salida. Para cada par de etiquetas, respecto a una instancia dada, se contemplan cuatro posibles estados: no se le asocia ninguna, se le asocian las dos, se le asocia la primera o se le asocia la segunda.

Se analizan dos modelos distintos: CML (*Collective Multi-Label*) y CMLF (*Collective Multi-Label with Features*). El primero mantiene parámetros de correlación para cada pareja de etiquetas, mientras que el segundo examina ternas del tipo *atributo-etiqueta1-etiqueta2*. El modelo CMLF se basa en que la correlación entre dos etiquetas dadas viene determinada por la presencia de una cierta característica (atributo) en las muestras (en el artículo, la presencia de una cierta palabra en un texto a clasificar). El modelo CML mantiene un parámetro de correlación adicional entre las dos etiquetas, mientas que CMLF lleva dicho parámetro a cada terna *característica-pareja de etiquetas*.

El algoritmo que usan descarta aquellas palabras (que son los atributos que se usan para etiquetar los textos) cuya frecuencia de aparición es menor a un cierto límite inferior. Con el resto se crea el modelo CMLF para establecer la correlación entre la presencia de una palabra y la asignación de una pareja de etiquetas al documento. Los resultados experimentales sobre un corpus de texto de Reuters prueba que el método propuesto mejora la precisión del uso de clasificadores binarios por un margen muy importante.

También en (Sun, Ji y Ye 2008) persiguen el objetivo de capturar en el modelo las relaciones existentes entre las etiquetas, para lo cual recurren a un algoritmo basado en *hipergrafos*, en el que cada etiqueta se representa por una *hiperarista* y que da lugar a un problema computacionalmente difícil de resolver pero que, bajo ciertas premisas, puede simplificarse y ser abordado con algoritmos pensados para el problema de los mínimos cuadrados. En la experimentación que aportan no comparan con otros algoritmos y apuntan a estudios posteriores sobre la formulación de este algoritmo, por lo que puede ser considerado como un trabajo en curso por parte de los autores.

Los *ensembles* o agrupaciones de clasificadores son la base de la propuesta hecha en (Tsoumakas y Vlahavas 2007) con el algoritmo RAkEL (*Random k-Labelsets*). El método consiste en utilizar un conjunto de clasificadores de tipo LP (*Label Powerset*), entrenando cada uno de ellos con un pequeño subconjunto de las etiquetas existentes. De esta manera se evitan los problemas que plantea la generación de un único clasificador con todas las combinaciones posibles de etiquetas y, al tiempo, se tiene en cuenta la información de correlación entre etiquetas. Se ejecutan m iteraciones tomando en cada una, y de manera aleatoria, un subconjunto de k etiquetas de L, sin reemplazamiento y entrenando un clasificador. Si k=1 y m=$|L|$ se tiene el modelo binario BR. Si k=$|L|$ y m=*1* tenemos el modelo LP. BR y LP, por tanto, podrían verse como casos específicos del algoritmo RAkEL descrito en este artículo, por lo que puede decirse que los autores lo que han hecho es generalizar y parametrizar los dos métodos de transformación más usuales.

A la hora de clasificar una instancia, ésta es procesada por los distintos clasificadores obteniendo para cada etiqueta una serie de predicciones que son promediadas. Si el valor final es superior a un parámetro *threshold* dado, en el rango [0,1], entonces se asigna la etiqueta a la muestra. Las pruebas experimentales muestran gráficamente cómo se comporta el algoritmo según varían los citados parámetros m y k. Se prueba con tres datasets y distintas medidas de evaluación, pero se echa en falta alguna comparación con otros algoritmos.

Apoyándose en métodos de transformación anteriormente descritos, en (Read, y otros 2009) se proponen dos métodos diferentes, pero relacionados, para abordar el problema de la clasificación multi-etiqueta aprovechando las características de la transformación BR, es decir, la creación de un clasificador binario para cada etiqueta existente en el dataset.

El primer método se denomina modelo CC (*Classifier Chain*) y es conceptualmente muy simple: si L es el conjunto de etiquetas, se crean $|L|$ clasificadores de forma que el primero predice usando los atributos de $\boldsymbol{x_i}$, el segundo agrega a ese vector de atributos uno adicional que será 0 ó 1, según la predicción del primero para su etiqueta, y así sucesivamente hasta que el clasificador $|L|$ se entrene con los atributos de la muestra y todas las predicciones previas. La adición de la predicción de clasificadores previos en la cadena como información adicional permite usar la información de correlación entre etiquetas, mejorando así la precisión de los resultados. Este encadenamiento, sin embargo, impide el funcionamiento en paralelo de los clasificadores.

Es fácil darse cuenta de que el funcionamiento de esta cadena de clasificadores binarios se verá influido por el orden en que se tomen las etiquetas, ya que para el entrenamiento del primero no se dispone de información de los demás, mientras que el último tiene información de todos los demás. Por ello el segundo método, denominado ECC (*Ensemble Classifier Chain*), genera un conjunto de cadenas de clasificadores tomando aleatoriamente el orden de las etiquetas y un subconjunto del dataset. Cada cadena del *ensemble* facilitará unos resultados que son tomados como votos para obtener el conjunto de etiquetas asociado a la muestra a clasificar. Los datos experimentales que documenta el artículo son exhaustivos, con múltiples datasets y comparando CC con métodos de tipo BR y ECC con otros algoritmos de tipo *ensemble* como EPS (*Ensembles of Pruned Sets*) o RAkEL (*RAndom K labEL subsets*). CC obtiene claramente mejores resultados que los demás, mientras que ECC no muestra tanta ventaja respecto a los algoritmos con los que compite.

Uno de los problemas que plantea la clasificación multi-etiqueta es que el número de subconjuntos distintos de etiquetas (cada muestra tiene un subconjunto que puede ser igual o no al de otras muestras) es potencialmente de $2^{|L|}$, generando un problema secundario aún más importante: que el número de muestras que comparten un mismo subconjunto (una misma clase *global*) puede ser muy reducido. Es lo que en (Veloso, y otros 2007) denominan *disjuncts* que, de ser ignorados dada su poca representatividad, pueden inducir un modelo poco preciso ya que en un dataset de gran tamaño puede haber múltiples casos así.

Los autores de (Veloso, y otros 2007) proponen un algoritmo de clasificación *perezoso*, que en el momento en que llega una instancia a clasificar toma como referencia los atributos de esa muestra y se filtra el dataset descartando atributos e instancias sin relación. El resultado es la reducción del dataset de entrenamiento y, en consecuencia, que los *disjuncts* adquieran más relevancia. Además el clasificador propuesto, denominado CLAC (*Correlated Lazy Associative Classifier*), mantiene información sobre la correlación entre etiquetas usando un mecanismo similar al del modelo CC de (Read, y otros 2009). De manera iterativa, con un algoritmo tipo *greedy*, se va incorporando a las muestras información sobre las etiquetas que tienen asociadas. En las pruebas experimentales se usan datasets ya etiquetados procedentes de la ACM, así como el dataset `yeast` con información sobre el genoma. Las mejoras de los algoritmos propuestos, y de CLAC en particular, respecto a otros ya existentes son muy considerables y llegan en algunos casos al 24% de ganancia en precisión.

Más por su originalidad que por su relevancia en cuanto a la resolución del problema, también merece mención la propuesta de (Chan y Freitas 2006). Los algoritmos basados en colonias de hormigas (Dorigo, Caro y Gambardella 1999) suelen aplicarse a problemas de optimización (ACO, *Ant Colony Optimization*), pero existe un método denominado Ant-Miner (Parpinelli, Lopes y Freitas 2002) de extracción de reglas que permite usarlo en problemas de clasificación. En (Chan y Freitas 2006), basándose en Ant-Miner, crean un algoritmo llamado MuLAM (Multi-Label Ant Miner) capaz de trabajar con problemas de clasificación multi-etiqueta. Solamente lo prueban con datasets en los que existen dos clases posibles y, además, comparan los resultados únicamente contra algoritmos no multi-etiqueta, como el C5.0 o el propio Ant-Miner. Éstos se ejecutan dos veces: una por clase a predecir. Los resultados que obtienen tienen mayor precisión que C5.0, pero no que Ant-Miner.

Tanto en (Yang, y otros 2009) como en (Esuli y Sebastiani 2009) se proponen técnicas de aprendizaje activo, partiendo de un conjunto de muestras en las que solamente una pequeña parte están etiquetadas. En el primer caso se entrena una SVM de manera iterativa: se van asignando etiquetas a las muestras no etiquetadas y se utiliza esa información para mejorar el entrenamiento del clasificador. El proceso se repite hasta alcanzar la precisión deseada. El punto clave es la estrategia de selección de las muestras a procesar. En el segundo se usa como algoritmo subyacente MP-BOOST y se obtiene un ranking de los ejemplos que más información aportan. Éstos se agregan al lote de instancias etiquetas y se reentrena el clasificador iterativamente. En ambos casos se trata de mecanismos muy asociados al problema del etiquetado de textos, más que a la clasificación muti-etiqueta en general.

Algunos trabajos, como es el caso de (Fan y Lin 2007), concentran sus esfuerzos en encontrar métodos de optimización de parámetros, más que adaptar o definir nuevos algoritmos de clasificación. Lo que proponen sus autores son diversos métodos para realizar ajustes en parámetros de umbral (*threshold*) que controlan el funcionamiento de los algoritmos de tipo BR con el objetivo de mejorar los resultados. Básicamente se trata de problemas de optimización dentro del problema de clasificación. Elegida la medida de evaluación que se quiere optimizar, por ejemplo la precisión, se va modificando el *threshold* para maximizarla. También describe una heurística, llamada *fbr* (la denominación procede de un parámetro de entrada al que los autores dieron dicho nombre y que almacena el límite inferior de una medida), que permite mejorar la precisión en aquellos casos en que se cuenta con muy pocas muestras positivas para el entrenamiento. Lo más destacable es la importancia que tiene la elección adecuada de los parámetros de funcionamiento del algoritmo y su ajuste durante la ejecución del mismo.

2.4 Características de los dataset multi-etiqueta

En la mayor parte de las publicaciones referenciadas en los puntos previos la experimentación se lleva a cabo con unos pocos datasets, entre los que destacan `yeast` y `scene` por la frecuencia con que se utilizan. Los creadores de la herramienta MULAN (Tsoumakas, Spyromitros-Xioufis, y otros 2011) mantienen un repositorio de datasets con diferentes características: pocas clases y muestras, muchas muestras, muchas clases, más atributos que muestras, etc. Parte de ellos serán utilizados en este capítulo para el estudio experimental sobre el algoritmo CO^2RBFN y en el capítulo 4 de este trabajo para completar la experimentación propia de la propuesta original realizada.

Las diferentes características de estos datasets representan un tipo de problema diferenciado, a saber:

- **Alta dimensionalidad**: Normalmente se habla de alta dimensionalidad cuando los datasets cuentan con un gran número de atributos o características de entrada. En el caso multi-etiqueta debe tenerse en cuenta no solamente ese factor, sino también la existencia de un gran número de etiquetas que, por regla general, suele conllevar una alta cardinalidad. Ante la existencia de una gran cantidad de etiquetas ciertos métodos de transformación, como el LP, son impracticables.

- **Gran dispersión**: Hay datasets con un número de muestras inferior al número de atributos o incluso el de clases, por lo que puede no contarse con suficientes muestras representativas de cada tipo como para completar el entrenamiento del clasificador.

- **Desbalanceo**: En los casos en que hay un gran número de etiquetas distintas pero la cardinalidad (promedio de etiquetas por muestra) es pequeña, hay que afrontar un problema de distribución no balanceada de clases.

En el citado repositorio pueden encontrarse algunos casos extremos, como es el de `EUR-Lex` (*eurovoc descriptors*), con las siguientes características:

- 5.000 atributos de entrada y casi 4.000 etiquetas posibles.

- Solamente 19.348 instancias de las cuales se usarán para entrenar el modelo una porción.

- De esas muestras 16.467 son diferentes entre sí, debido al gran número de atributos y clases, por lo que el entrenamiento del clasificador resultará muy difícil.

Otros casos especiales son `rcv1v2`: con 47.236 atributos de entrada para un total de 30.000 muestras entre los cinco subconjuntos; `tmc2007`: con 49.060 atributos para solamente 28.596 instancias, o `genbase`: con el doble de atributos que de instancias.

El análisis de estos datos lleva a concluir que resulta imprescindible, por lo tanto, aplicar técnicas de selección de atributos de entrada y buscar métodos de reducción del número de etiquetas, combinándolos para tratar la alta dimensionalidad, si se quiere mejorar tanto el rendimiento como la efectividad de los clasificadores multi-etiqueta.

2.5 Experimentación con métodos de transformación

La primera prueba experimental llevada a cabo ha consistido en aplicar los métodos de transformación BR y LP empleando como base el algoritmo de clasificación CO^2RBFN (Pérez-Godoy, y otros 2010). El objetivo ha sido comprobar cuál es el comportamiento de un algoritmo no diseñado específicamente para el problema de la clasificación multi-etiqueta, comparándolo con algunos de los que incorpora el software MULAN. Este trabajo de experimentación fue presentado en el pasado IWANN (*International Work-Conference on Artificial Neural Networks*) de junio de 2011 (Rivera, Charte y otros 2011).

Las RBFNs (*Radial Basis Function Networks*) son, en el campo de los métodos de aprendizaje, uno de los paradigmas más importantes de red neuronal artificial (RNA). Una RBFN es una RNA de tipo *feed-forward* con una sola capa oculta compuesta de unidades llamadas RBFs (*Radial Basis Functions*) (Broomhead y Lowe 1988). En la Figura 1 puede apreciarse la estructura básica de este tipo de red neuronal, con la única capa oculta compuesta de *m* RBFs.

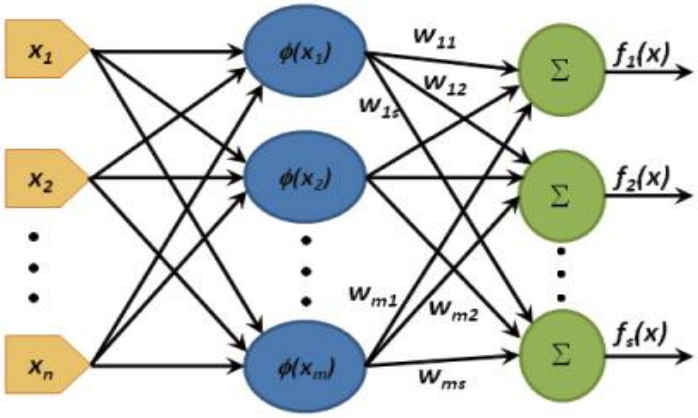

Figura 1. Estructura típica de una RBFN.

Cada una de las RBF que componen la capa oculta recibe todas las características a evaluar directamente de la capa de entrada. La RBF se sitúa en el espacio n-dimensional mediante un centro y tiene asociado un radio. Éste puede ser común a todas las RBF o no, según el diseño que se haya hecho. Mediante la aplicación de una función de tipo gaussiano cada RBF genera para la muestra procesada una salida que estará en función de su distancia al centro y el radio de la RBF (el valor de salida va disminuyendo con la distancia de la muestra al centro la neurona). Las neuronas de salida toman los resultados de las RBFs y efectúan una suma ponderada, obteniéndose así el valor que se interpreta como predicción. La eficiencia de las RBFNs ha sido demostrada en múltiples áreas (Buchtala, Klimek y Sick 2005) como son la clasificación de patrones, aproximación de funciones y predicción de series temporales.

Un paradigma importante a la hora de diseñar RBFNs es la Computación Evolutiva (Goldberg 1989). Existen diferentes propuestas en esta área con variados esquemas de representación: Pittsburgh (Harpham, Dawson y Brown 2004), en el que cada individuo es una RBFN completa, y cooperativo-competitivo (Whitehead y Choate 1996), donde cada individuo representa una sola RBF y la población completa forma la RBFN.

En (Pérez-Godoy, y otros 2010) se desarrolla un algoritmo para el diseño cooperativo-competitivo de Redes Neuronales con Funciones de Base Radial, llamado CO^{2}RBFN, aplicándolo con éxito a problemas de clasificación clásica y desbalanceada. Éste mismo algoritmo se ha tomado en el presente trabajo como base para procesar una serie de datasets, a los que se ha aplicado a modo de preprocesamiento las transformaciones *Binary Relevance* y *Label Powerset*.

En el punto siguiente de este apartado se explica con algo más de detalle el algoritmo CO^{2}RBFN, describiéndose la experimentación y resultados obtenidos en el punto inmediatamente posterior.

2.5.1 CO^{2}RBFN: un algoritmo evolutivo híbrido cooperativo-competitivo para el diseño de RBFN

CO^{2}RBFN (Pérez-Godoy, y otros 2010) es un algoritmo evolutivo híbrido de tipo cooperativo-competitivo para el diseño de RBFNs. En este algoritmo cada individuo de la población constituye, con una representación interna de tipo real, una RBF y la población completa es responsable de la solución final.

Los individuos colaboran para alcanzar la solución definitiva, pero también deben competir entre sí para sobrevivir. En este entorno, en el que la solución depende del comportamiento de muchos componentes, el *fitness* de cada individuo es conocido como *asignación de crédito*. Con el propósito de cuantificar la asignación de crédito de un individuo se han propuesto tres factores: la contribución de cada RBF a la salida global de la red, el error cometido en el radio de la función base y el grado de solapamiento entre las RBF.

La aplicación de los operadores viene determinada por un sistema basado en reglas difusas. Las entradas de este sistema son los tres parámetros usados para la asignación de crédito y las salidas son las probabilidades de aplicación de los operadores.

Los pasos fundamentales de CO^{2}RBFN, explicados a continuación, son los siguientes:

1. Inicializar la RBFN
2. Entrenar la RBFN
3. Evaluar las RBFs
4. Aplicar operadores a las RBFs
5. Sustituir las RBFs eliminadas
6. Seleccionar las mejores RBFs
7. Volver al paso 2 si no se alcanza la condición de parada

Inicializar la RBFN. Para definir la red inicial se parte de un número m especificado de neuronas (el tamaño de la población). El centro de cada RBF se asigna aleatoriamente asociado a un patrón diferente del conjunto de entrenamiento. El radio de las RBFs, d_i, se fija inicialmente a partir de un cálculo sencillo: la mitad de la distancia promedio entre los centros. Finalmente se ponen a cero los pesos de la RBF: w_{ij}.

Entrenar la RBFN. Se emplea el algoritmo LMS (*Least Mean Square*) (Widrow y Lehr 1990) para calcular los pesos de las RBFs.

Evaluar las RBFs. A fin de evaluar el papel de cada RBF ϕ_i en el entorno cooperativo-competitivo se precisa un mecanismo de asignación de crédito. Para cada RBF se definen tres parámetros: a_i, e_i, o_i:

- La contribución a_i de la RBF ϕ_i se determina tomando en consideración el peso w_i y el número de patrones del conjunto de entrenamiento que quedan dentro de su anchura: pi_i:

$$a_i = \begin{cases} |w_i| & si\ pi_i > q \\ |w_i| * \left(\frac{pi_i}{q}\right) & en\ caso\ contrario \end{cases}$$

Siendo q el promedio de los valores pi_i menos la desviación estándar de esos mismos valores.

- La medida de error e_i para cada RBF ϕ_i se obtiene mediante un conteo de los patrones que hay dentro de su radio mal clasificados:

$$e_i = \frac{pibc_i}{pi_i}$$

Donde $pibc_i$ y pi_i son el número de patrones clasificados erróneamente y el número total de patrones dentro de la RBF, respectivamente.

- El solapamiento de la RBF ϕ_i con otras RBFs se cuantifica mediante el parámetro o_i. Éste se calcula mediante la metodología de *finess* compartido (Goldberg 1989), cuyo objetivo es mantener la diversidad en la población.

Aplicar operadores a las RBFs. En CO^2RBFN se han definido cuatro operadores aplicables a las RBFs:

- Operador Remove: elimina una RBF.
- Operador Random Mutation: modifica el centro y anchura de una RBF en una magnitud aleatoria.
- Operador Biased Mutation: modifica la RBF, usando información local, para intentar colocarla en el centro del cluster que representa la clase.
- Operador Null: en este caso se mantienen todos los parámetros de la RBF.

Los operadores se aplican a toda la población de RBFs. La probabilidad de tomar un cierto operador viene determinada por un sistema de reglas difusas de tipo Mandani (Mandani y Assilian 1975) que representan el conocimiento experto sobre la aplicación de operadores con el objetivo de obtener una RBFN simple y precisa. Las entradas de este sistema son los parámetros a_i, e_i y o_i usados para la definición de la asignación de crédito de la RBF ϕ_i. Dichas entradas se consideran como las variables lingüísticas *vai*, ve_i y vo_i. Las salidas: p_{remove}, p_{rm}, p_{bm} y p_{null}, representan la probabilidad de que se apliques los operadores Remove, Random Mutation, Biased Mutation y Null, respectivamente. La Tabla 1 (página siguiente) muestra la base de reglas empleada para relacionar los antecedentes y consecuentes descritos.

	Antecedentes			Consecuentes			
	v_a	v_e	v_o	p_{remove}	p_{rm}	p_{bm}	p_{null}
R1	L			M-H	M-H	L	L
R2	M			M-L	M-H	M-L	M-L
R3	H			L	M-H	M-H	M-H
R4		L		L	M-H	M-H	M-H
R5		M		M-L	M-H	M-L	M-L
R6		H		M-H	M-H	L	L
R7			L	L	M-H	M-H	M-H
R8			M	M-L	M-H	M-L	M-L
R9			H	M-H	M-H	L	L

Tabla 1. Reglas difusas que representan el conocimiento experto en diseño de RBFNs.

Introducir nuevas RBFs. En este paso se sustituyen las RBFs eliminadas por otras nuevas. Las nuevas RBFs se colocan, con una probabilidad de 0.5, en en el centro del área que tiene error máximo o en el centro de un patrón elegido aleatoriamente.

Estrategia de reemplazo. El esquema de reemplazo determinar que RBFs (antes de la mutación) serán incluidas en la nueva población. Para ello se compara el papel que juega la RBF mutada con la original para quedarnos con aquella que tiene mejor comportamiento a fin de incluirla en la población.

2.5.2 Experimentación y resultados

El objetivo de este trabajo de experimentación ha sido analizar el comportamiento del algoritmo que acaba de describirse en el campo de la clasificación multi-etiqueta, comparándolo con otros algoritmos de clasificación clásica (binaria o multi-clase, pero no multi-etiqueta), recurriendo para ello a dos métodos de transformación de datos.

Se han seleccionado dos datasets multi-etiqueta:

- **Emotions**: El objetivo es clasificar una pieza musical en más de una categoría. Cuenta con 593 instancias, 72 atributos numéricos y 6 etiquetas.

- **Scene**: En este caso se pretende etiquetar una imagen que puede corresponder a varias clases semánticas. Tiene 2407 instancias, 294 atributos numéricos y 6 etiquetas.

Lo primero que hay que destacar es la alta dimensionalidad de los datasets multi-etiqueta, algo que es necesario tener en consideración a la hora de extraer conclusiones.

Además de con CO^2RBFN los datasets han sido procesados con los siguientes algoritmos: C4.5, KNN, Naive Bayes, MLP, PART, RBFN y SVM, todo ellos métodos de aprendizaje cuya implementación y referencia pueden encontrarse en Weka (Hall, y otros 2009). En todos los casos se han empleado los pa-

rámetros recomendados por sus autores. Para CO^{2}RBFN se han usado parámetros seleccionados de forma heurística, utilizándose 100 iteraciones del bucle principal y el número de neuronas se ha establecido en el rango 10 a 20.

Para poder ejecutar tanto CO^{2}RBFN como los demás algoritmos citados sobre los datasets antes indicados se ha recurrido a la metodología de transformación de datos, concretamente a las dos técnicas más populares: *Binary Relevance* y *Label Powerset*. De esta forma los datasets originales Emotions y Scene se han transformado con BR y LP, empleando para ello el software MULAN, disponible en `http://mulan.sourceforge.net` y descrito en (Tsoumakas, Spyromitros-Xioufis, y otros 2011).

Los parámetros generales del experimento con que se ha configurado MULAN son validación cruzada de tipo 10-cv (90% para el conjunto de entrenamiento y 10% para test) y 3 repeticiones de las que se han obtenido valores promedio de los resultados. Las métricas utilizadas para los resultados son las facilitadas por MULAN y son descritas con detalle en (Tsoumakas, Katakis y Vlahavas, Mining Multi-label Data 2010).

En la Tabla 2 se muestran los resultados promedio de test para la transformación BR sobre los dos datasets. La Tabla 3 refleja los resultados para la transformación LP. En cuanto a las medidas, para la métrica *HammingLoss* cuanto menor sea el valor mejor es el resultado (refleja los errores cometidos al clasificar) y para las demás cuanto mayor sea el valor mejor es el resultado.

Dataset Emotions								
	C4.5	CO2RBFN	KNN	MLP	Naïve Bayes	PART	RBFN	SVM
Hamming Loss	0.247	**0.204**	0.235	0.215	0.252	0.257	0.229	0.244
Subset Accuracy	0.184	**0.270**	0.268	**0.270**	0.206	0.157	0.213	0.180
Example-Based Recall	0.599	0.612	0.626	0.646	**0.773**	0.614	0.630	0.441
Example-Based Accuracy	0.462	0.514	0.514	0.525	**0.529**	0.456	0.494	0.391
Dataset Scene								
	C4.5	CO2RBFN	KNN	MLP	Naïve Bayes	PART	RBFN	SVM
Hamming Loss	0.137	0.141	0.111	**0.100**	0.242	0.119	0.139	0.126
Subset Accuracy	0.427	0.365	**0.629**	0.566	0.169	0.477	0.369	0.306
Example-Based Recall	0.634	0.457	0.693	0.706	**0.858**	0.668	0.484	0.325
Example-Based Accuracy	0.535	0.419	**0.674**	0.647	0.453	0.578	0.437	0.323

Tabla 2. Resultados promedio en test con la transformación Binary Relevance.

Dataset Emotions								
	C4.5	CO2RBFN	KNN	MLP	Naïve Bayes	PART	RBFN	SVM
Hamming Loss	0.277	0.243	0.235	0.234	0.233	0.293	**0.217**	0.281
Subset Accuracy	0.207	**0.301**	0.268	0.278	0.268	0.209	0.298	0.271
Example-Based Recall	0.541	**0.653**	0.626	0.630	0.630	0.526	0.647	0.595
Example-Based Accuracy	0.438	0.522	0.514	0.518	0.512	0.424	**0.542**	0.473
Dataset Scene								
	C4.5	CO2RBFN	KNN	MLP	Naïve Bayes	PART	RBFN	SVM
Hamming Loss	0.144	0.186	0.111	0.114	0.137	0.139	0.116	**0.095**
Subset Accuracy	0.547	0.427	0.629	0.641	0.537	0.563	0.621	**0.688**
Example-Based Recall	0.609	0.454	0.693	0.701	0.678	0.626	0.677	**0.720**
Example-Based Accuracy	0.589	0.454	0.674	0.684	0.615	0.605	0.662	**0.720**

Tabla 3. Resultados promedio en test con la transformación Power Labelset.

Como puede observarse en las tablas de resultados, no hay un método que destaque sobre los demás, ni para la transformación BR ni para la LP. El algoritmo CO^{2}RBFN obtiene sus mejores resultados con el dataset `Emotions` (con independencia del método de transformación aplicado), batiendo a los demás método en cuatro de las medidas de evaluación. Para la transformación BR con el dataset `Scene` CO^{2}RBFN consigue resultados similares a los otros métodos. El peor resultado se obtiene con la transformación LP sobre el dataset `Scene`. Es destacable la buena precisión alcanzada por el otro método RBFN y, por tanto, el buen comportamiento de los modelos RBFN en tareas de clasificación multi-etiqueta. En cualquier caso, CO^{2}RBFN es el método con mayor número de mejores resultados (en negrita) en medidas individuales, seguido por SVM.

En resumen, cuando se aplican transformaciones a los datasets multi-etiqueta con el objetivo de resolver el problema de clasificación afrontado, no hay un algoritmo que sea el mejor en todos los casos.

3 Reglas de asociación

Como se ha mencionado, en este trabajo se propone una metodología basada en la transformación de datos para problema multi-etiqueta que utiliza reglas de asociación, por lo que en este capítulo se introduce este tipo de herramienta de conocimiento, la tarea de minería asociada y los algoritmos más relevantes.

Las reglas de asociación (Hipp, Güntzer y Nakhaeizadeh 2000) describen coocurrencias entre elementos que aportan conocimiento sobre el sistema subyacente a un conjunto de datos y pueden ser interpretadas como implicaciones, de forma que la aparición de unos ciertos elementos predice la ocurrencia de otro. El ejemplo de uso más habitual es el del supermercado que quiere analizar la cesta de la compra para determinar qué productos son adquiridos juntos por sus clientes: el caso típico, mencionado de manera reiterada en la bibliografía sobre este tema, son los pañales y la cerveza, ya que las personas con hijos pequeños no suelen poder salir durante el fin de semana y, en consecuencia, se proveen anticipadamente de ambos productos. A partir de esa información el comerciante decide colocar cerca del primer producto el segundo, para inducir a su compra y evitar que los padres se olviden de la cerveza cuando van a por los pañales o viceversa.

Al proceso de extracción de reglas de asociación a partir de una base de datos o conjunto de transacciones es a lo que se conoce como *minería de reglas de asociación*, existiendo múltiples algoritmos para llevarlo a cabo. En (Hipp, Güntzer y Nakhaeizadeh 2000) se hace una revisión de los más comunes, muchos de ellos basados en el más conocido: el algoritmo Apriori introducido por (Agrawal y Srikant 1994).

Aunque las reglas de asociación suelen emplearse como una herramienta descriptiva, que permite obtener conocimiento sobre unos hechos, también han sido empleadas con otros fines como son la clasificación (Liu, Hsu y Ma 1998) mediante los denominados sistemas CARs (*Classification Association Rules*). La minería de reglas para clasificación se basa en la obtención de un conjunto reducido de reglas que tienen un cierto valor mínimo de *soporte* y *confianza*, de manera que su evaluación puede servir para determinar la clase de las muestras de datos a partir del valor de sus atributos.

Nuestro interés en las reglas de asociación estriba, tal y como se explicará en el apartado *4.4 PROPUESTA: Reducción de la cardinalidad con reglas de asociación* del capítulo 4 de este trabajo, en su uso como herramienta para ocultar al clasificador multi-etiqueta aquellas etiquetas que puedan ser deducidas a partir de la aparición de otras con un cierto nivel de confianza. En el siguiente apartado se introducen los conceptos generales sobre minería de reglas de asociación y en el posterior se detalla el algoritmo que elegido para su extracción.

3.1 Conceptos generales y medidas

Sea $I = \{x_1, \dots, x_n\}$ un conjunto de objetos o elementos distintos, a los que llamaremos genéricamente *ítems*. En el ejemplo del supermercado cada *ítem* sería un producto a la venta. Se denomina *k-itemset*, o sencillamente *itemset*, a un subconjunto $X \subseteq I$ con $k = |X|$. *k* puede tomar valores entre 1 y *n*, el número de ítems existentes, y para un cierto valor de *k* existirán múltiples combinaciones diferentes, por lo que existirá un determinado número de itemsets distintos pero del mismo tamaño, salvo para *k*=*n* caso en el que solamente habría uno posible.

Sea $D = \{T_1, \dots, T_n\}$ un conjunto de subconjuntos de *I*, de forma que cada T_i es un itemset de un cierto tamaño al que se llama *transacción*. En el ejemplo del supermercado *D* sería la base de datos en la que se registran las ventas y cada transacción la venta realizada a un cliente, compuesta de varios ítems.

Tomando un itemset $X \subseteq I$ se dice que una cierta transacción *T* da soporte a dicho itemset si se cumple que $X \subseteq T$, es decir, si todos los *ítems* contenidos en *X* están también en *T*. A la fracción de transacciones en *D* que dan soporte a un cierto itemset *X* es a lo que se denomina *soporte o cobertura de X*, expresado como:

$$supp(X) = \frac{|\{T \in D \mid X \subseteq T\}|}{|D|}$$

A los itemsets cuyo soporte supera un cierto valor umbral, normalmente fijado de antemano a la ejecución del algoritmo de minería de reglas de asociación, se les llama *itemsets frecuentes*. El primer paso de todo algoritmo de extracción de reglas será la obtención de una lista con los itemsets frecuentes que existan en la base de datos.

Una regla de asociación se define como una implicación del tipo $X \Rightarrow Y$ en la que tanto *X* (antecedente) como *Y* (consecuente) son itemsets: $X, Y \subseteq I$ y se cumple que $X \cap Y = \emptyset$. El soporte de una regla se obtiene a partir del soporte de antecedente y consecuente:

$$supp(X \Rightarrow Y) = supp(X \cup Y)$$

Lo habitual es buscar reglas que tengan un soporte relativamente alto, ya que aquellas con un soporte bajo pueden haber aparecido por simple casualidad y no ser representativas de un comportamiento colectivo. Es un aspecto en el que también influirá el tamaño del conjunto de transacciones con que se trabaje.

El soporte de una regla indica la frecuencia con que aparece en la base de transacciones la unión de los itemsets que forman antecedente y consecuente, pero no nos dice nada sobre la precisión de la implicación, es decir, la frecuencia con que al presentarse el antecedente lo hace también el consecuente. Para medir este factor se recurre a la *confianza*, medida como:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

Lo que ofrece esta medida es una estimación de la probabilidad condicional *P(Y/X)* y el objetivo es que ésta sea lo más alta posible. Una regla con una alta confianza indica que existe una correlación entre antecedente y consecuente. Si $conf(X \Rightarrow Y) = 1$ ello indicará que en todas las transacciones en que aparece *X* también lo hace *Y* y que, por lo tanto, podría asumirse que la primera implica la presencia de la segunda. Por el contrario, una confianza baja puede indicar que no existe relación alguna entre esos itemsets más allá de la casual.

Ha de tenerse en cuenta que las reglas de asociación no son entidades reversibles: $X \Rightarrow Y \neq Y \Rightarrow X$. En ambos casos el soporte sería el mismo, pero no así la confianza:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \neq \frac{supp(X \cup Y)}{supp(Y)} = conf(Y \Rightarrow X)$$

Si bien la confianza es la medida más empleada a la hora de seleccionar reglas de asociación generadas por un algoritmo de minería de reglas, existen medidas alternativas como puede ser el interés de una regla (también conocida como *lift*) introducida en (Brin, y otros 1997) y definida como:

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

Al calcular la confianza de una regla puede darse el caso de que $supp(X \cup Y)/supp(X) = supp(Y)$ en caso de que *X* e *Y* sean itemsets independientes (desde un punto de vista estadístico). El interés, al tomar en consideración tanto el soporte de *X* como el de *Y* en el denominador, asume esa independencia pero, tal y como indican los autores, es una medida que indica grado de coocurrencia, pero no de implicación, ya que es completamente simétrica y se cumple que $lift(X \Rightarrow Y) = lift(Y \Rightarrow X)$.

3.2 Minería de reglas de asociación

La extracción de reglas de asociación consiste en encontrar todas aquellas reglas que tienen un soporte y confianza mínimos a partir de una base de transacciones. El problema es que el espacio de búsqueda crece exponencialmente respecto a $|I|$ (número de ítems distintos), por lo que puede llegar a ser inabordable. El proceso se divide en dos partes diferenciadas tal y como se indica en (Agrawal y Srikant 1994):

1. Obtener el conjunto *F* formado por todos los itemsets frecuentes, asociando a cada uno de ellos el soporte que le corresponda.

2. Para cada uno de los itemsets en *F* se calcula la confianza de todas las regla del tipo $X \backslash Y \Rightarrow Y, Y \subseteq X, X \neq Y \neq \emptyset$ y se conservan aquellas que alcanzan la confianza mínima preestablecida (o el valor mínimo para la medida alternativa, si no se emplea la confianza).

La generación de itemsets que tienen un mínimo soporte (frecuentes), fijado de antemano, es un problema combinatorio. Partiendo de los *1-itemset*, conjuntos formados solamente por un ítem, hay que calcular el soporte de cada uno y, a continuación, formar todas las combinaciones posibles de *2-itemset*, repitiendo el proceso hasta generar todos los *k-itemset* posibles.

El algoritmo más conocido para minería de reglas de asociación a partir de una base de datos o dataset es Apriori (basado en el principio del mismo nombre) introducido en (Agrawal y Srikant 1994), que reduce en cierta medida el problema combinatorio usando la técnica de generación de itemsets candidatos. Ésta se pasa en una sencilla premisa: si un cierto *k-itemset* no es frecuente (no alcanza el soporte mínimo), tampoco lo serán aquellos k_{+1}*-itemsets* en los que esté contenido. A pesar de ello su uso no es apropiado con grandes bases de datos o itemsets compuestos de muchos elementos dada su complejidad computacional y el hecho de que recorre la colección de transacciones múltiples veces.

De los algoritmos alternativos descritos en la bibliografía especializada para nuestra experimentación hemos optado, por su rendimiento, por el algoritmo FP-Growth (Han, Pei y Yin 2000), descrito en el siguiente apartado de esta sección.

3.3 El algoritmo FP-Growth

Frente al algoritmo Apriori clásico, FP-Growth es un método en el que se prescinde de la generación de itemsets candidatos, es decir, se elimina la parte más costosa de la extracción de reglas. Los patrones o itemsets frecuentes se producen a partir de una estructura de datos: el FP-Tree. Dicha estructura de datos, y el propio algoritmo en sí, agregan algo de complejidad a la implementación (respecto a Apriori) pero el resultado es mucho más eficiente.

La primera fase del algoritmo es la encargada de construir la estructura de datos, denominada FP-Tree, para lo cual da únicamente dos pasadas sobre el dataset o base de datos que aloja las transacciones:

1. Durante la primera pasada se obtiene un recuento de la aparición de cada uno de los valores (itemsets de longitud 1) existentes en la BDD de transacciones, al final de la cual se eliminan aquellos que no llegan al soporte mínimo (fijado de antemano) y se genera una lista ordenada por soporte de mayor a menor.

2. Apoyándose en la citada lista ordenada, una segunda pasada sobre la BDD procede a generar el FP-Tree paso a paso con cada transacción procesada. Partiendo de un nodo raíz null, por cada transacción leída se hace lo siguiente:

    - Se toma el elemento con mayor soporte de los que almacena la transacción y, partiendo de la raíz, se busca en el primer nivel del árbol. Si no está se añade y se inicializa su contador a 1, si está se incrementa el contador asociado.

    - Asumiendo que el nodo del elemento anterior es la nueva raíz, se toma el siguiente elemento con mayor soporte en la transacción y se repite la operación. De esta forma se va creando una ruta descendente entre los nodos de los diferentes niveles.

    - Procesados todos los elementos de la transacción se lee la siguiente y se reinicia el proceso.

Concluida la segunda pasada se tiene en memoria el FP-Tree con todos los enlaces entre nodos padres e hijos, formando un árbol n-ario que no estará completo hasta que se agreguen enlaces entre apariciones de un mismo elemento en diferentes ramas del árbol. Dichos vínculos facilitarán, como se verá después, la navegación por el árbol. Lo que se hace es recorrer éste agregando a una tabla cada elemento y la referencia a los nodos en que aparece, después se recorre la tabla y por cada entrada con más de un nodo se vinculan éstos entre sí.

El FP-Tree es una estructura compacta, normalmente mucho más reducida que los datos originales, y servirá para extraer los itemsets frecuentes que, a la postre, es lo que perseguimos. Lo interesante es que ese proceso ya no requerirá recorrer la BDD de transacciones ni generar candidatos (como en Apriori).

El procedimiento de extracción de patrones frecuentes a partir del FP-Tree es realmente el cometido del algoritmo FP-Growth. Basándose en la técnica del divide y vencerás, este algoritmo parte de las hojas del FP-Tree y va generando subárboles para cada conjunto de hojas que compartan el mismo elemento, sirviéndose de los enlaces padre-hijo y también de los vínculos (añadidos previamente) entre elementos que aparecen en distintas ramas.

Cada subárbol representa un conjunto de rutas o caminos que comparten un mismo elemento en la hoja pero con distintos prefijos (si hay más de un camino). En un proceso recursivo, se corta el nivel inferior de esos subárboles y se repite la operación anterior obteniendo todos los árboles posibles para cada elemento hasta llegar a la raíz. A partir de los prefijos, y tras un recuento interno que permite ordenar los elementos según su frecuencia, se obtienen árboles condicionales para el elemento que se había cortado.

El último paso del algoritmo es la obtención a partir de cada subárbol de los itemsets frecuentes, usando para ello los enlaces padre-hijo entre los nodos y los contadores asociados. Básicamente se toma el elemento de la hoja como itemset frecuente de longitud 1, a continuación se le pone como prefijo las combinaciones de elementos de los nodos padre que tenga para los itemset de longitud 2 y así sucesivamente.

3.3.1 Ejemplo de uso

Con el objeto de facilitar la comprensión de cómo funciona el algoritmo FP-Growth lo pondremos en práctica con un ejemplo extremadamente sencillo, partiendo de una supuesta base de datos (Tabla 4) en la que cada transacción corresponde a una persona y se compone de una serie de entradas que indican los lenguajes informáticos que utiliza habitualmente.

Usuario	Lenguajes
1	C++, JavaScript, HTML, Java
2	HTML, JavaScript, Python, C++, Java
3	Ensamblador, Haskell
4	Python, JavaScript, Java, C++
5	Java, JavaScript, HTML
6	Haskell
7	HTML
8	HTML, Java
9	JavaScript, HTML, C++
10	Java, Python

Tabla 4. Base de datos de ejemplo.

Se realiza la primera pasada sobre la base de datos y se obtiene, para cada uno de los ítems (lenguaje), el recuento del número de veces que aparece a partir del que se calculará el soporte. La lista obtenida, ordenada de mayor a menor, sería la mostrada en la Tabla 5.

Lenguaje	Recuento	Soporte
Java	6	0.6
HTML	6	0.6
JavaScript	5	0.5
C++	4	0.5
Python	3	0.3
Haskell	2	0.2
Ensamblador	1	0.1

Tabla 5. Recuento y soporte para 1-itemsets.

Supongamos que hemos fijado el soporte mínimo en un 0.25, por lo que las dos últimas filas quedarían ya descartadas al no poder formar parte esos lenguajes de itemsets frecuentes. Nos queda, por tanto, una lista de 5 lenguajes ordenados por soporte.

Llega el momento de comenzar a generar el FP-Tree, inicializándolo con el nodo raíz `null`. Se lee la primera transacción de la BDD y se ordenan los elementos según el orden indicado en la Tabla 5, por lo que quedaría como: `1 - Java, HTML, JavaScript, C++`. Dado que el árbol está vació, se crea la primera rama partiendo de `null` con la ruta `Java -> HTML -> JavaScript -> C++` quedando todos los contadores a 1.

Leemos la siguiente transacción, ordenamos sus elementos y obtenemos la ruta `Java -> HTML -> JavaScript -> C++ -> Python`. Partiendo de la raíz vamos pasando por los nodos ya existentes e incrementando su contador hasta llegar al elemento `Python` que, al no existir, se agrega como un nuevo nodo colgando de esa ruta y con su contador a 1.

En las imágenes de la página siguiente se muestra el árbol tras leer la primera transacción (Figura 2) y tras la segunda (Figura 3). La tercera transacción se descarta y la cuarta, una vez ordenada, da lugar a la ruta `Java -> JavaScript -> C++ -> Python`, por lo que el contador del nodo `Java` ya existente se incrementa y, a continuación, se crea una nueva rama partiendo de ese nodo con la sub-ruta `JavaScript -> C++ -> Python` y sus contadores a 1. Puesto que en esta rama hay nodos que ya están en otras ramas del árbol se agregan los enlaces necesarios. El árbol queda como puede apreciarse en la Figura 4.

Al leer la siguiente transacción nos encontramos con una ruta que ya existe: `Java -> HTML -> JavaScript`, por lo que no hay más que incrementar los respectivos contadores de esos nodos, no sufriendo alteración alguna la estructura del árbol.

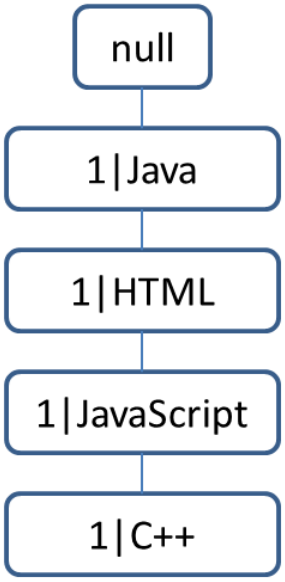

Figura 2. Árbol tras primera transacción.

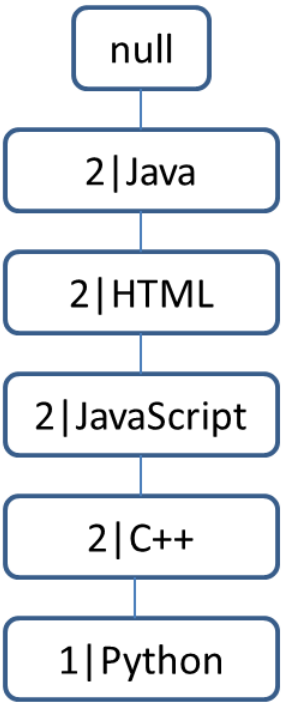

Figura 3. Árbol tras segunda transacción.

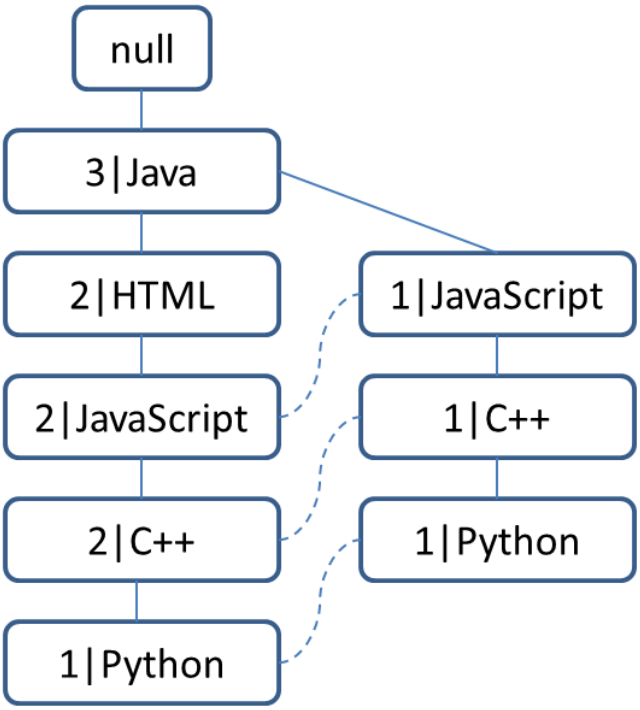

Figura 4. Árbol tras la cuarta transacción.

La sexta transacción queda descartada, ya que contiene un único elemento no frecuente, y la séptima se compone de un único elemento: `HTML`. Al ser frecuente hay que agregarlo al árbol y, puesto que aún no existe ninguna ruta desde la raíz hacia `HTML` se procede a crearla e inicializar su contador. El nuevo estado del árbol sería el mostrado en la Figura 5.

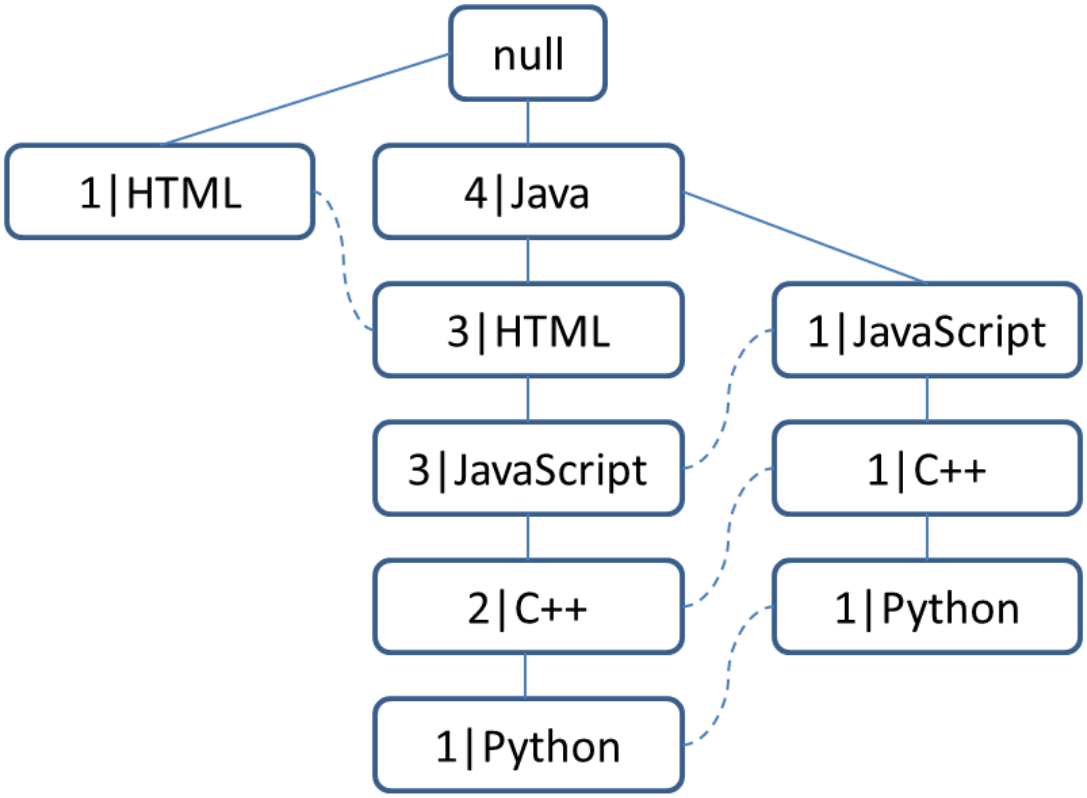

Figura 5. Árbol tras la séptima transacción.

La octava transacción incrementa los contadores de la ruta `Java -> HTML`. La siguiente genera la ruta `HTML -> JavaScript -> C++`, lo que lleva a incrementar el contador del nodo `null->HTML` y agregar los nodos `JavaScript` y `C++` bajo el mismo tal y como queda reflejado en la Figura 6 (página siguiente).

La ultima transacción, tras ordenar sus dos elementos según la lista, da lugar a la ruta `Java -> Python`. Se incrementa el contador del nodo de primer nivel `Java` y se agrega como hijo un nuevo nodo `Python`. El FP-Tree queda finalmente como puede verse en la Figura 7.

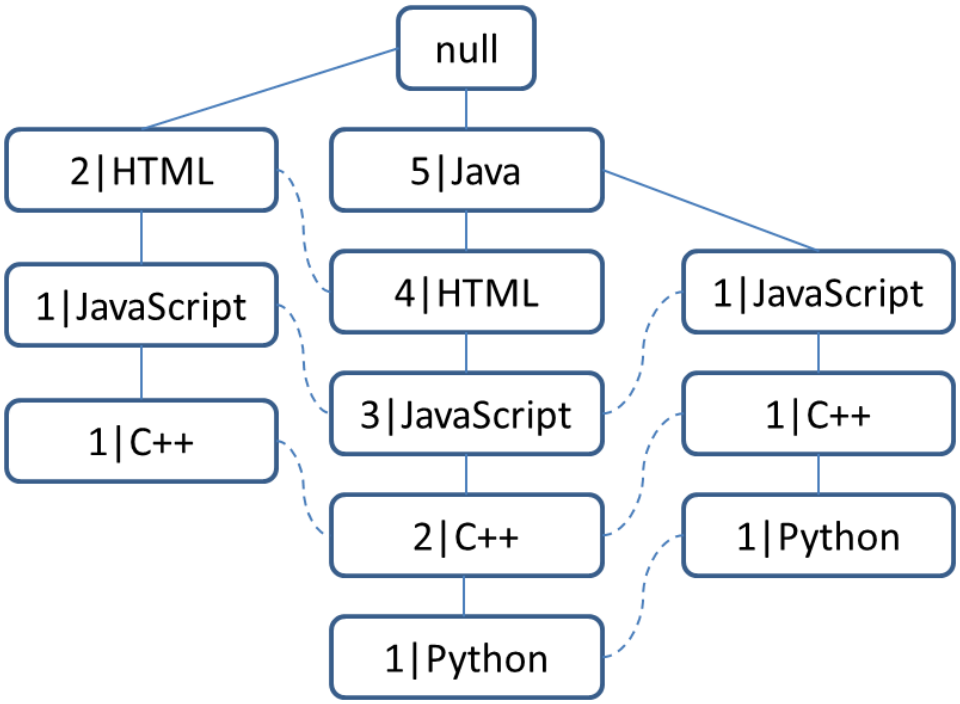

Figura 6. Árbol tras la novena transacción.

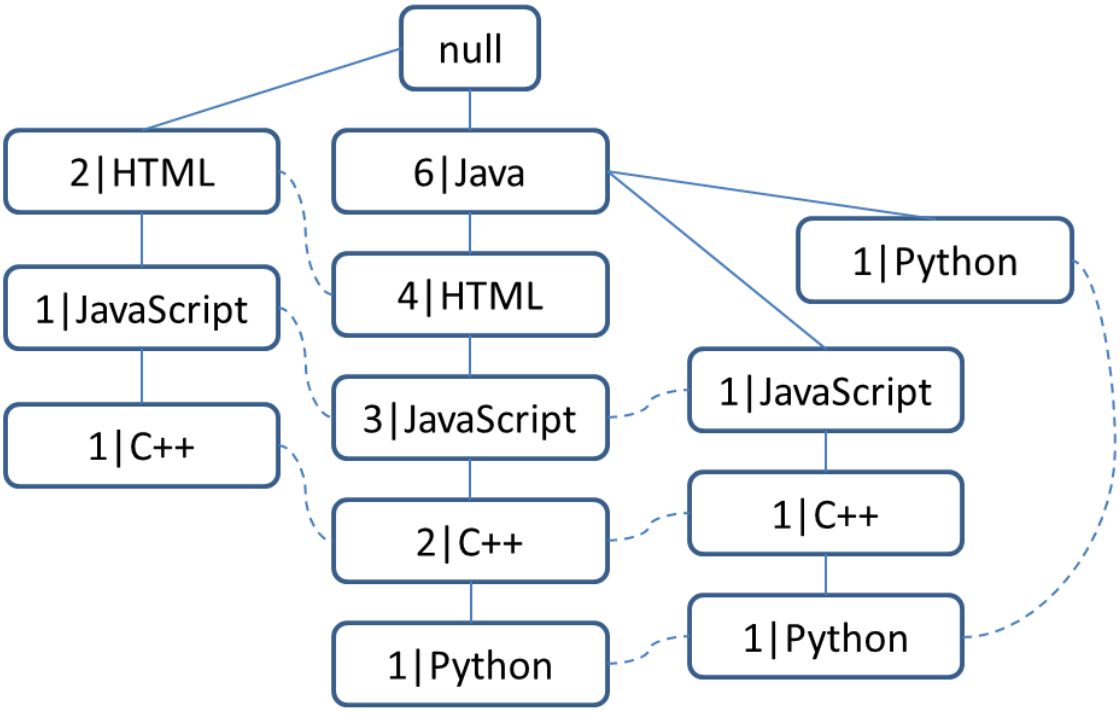

Figura 7. Árbol tras la última transacción.

Con el FP-Tree terminado de construir pasaríamos a la fase siguiente: la generación de los itemsets frecuentes. Como se indicaba anteriormente se trata de un algoritmo recursivo, tomando uno de los elementos que aparecen en las hojas y quedándonos con su subárbol, dividiéndolo en cada iteración podando la parte inferior. Comencemos por ejemplo con el elemento `Python`, que aparece en tres de las hojas del árbol anterior y cuyo subárbol sería el de la Figura 8.

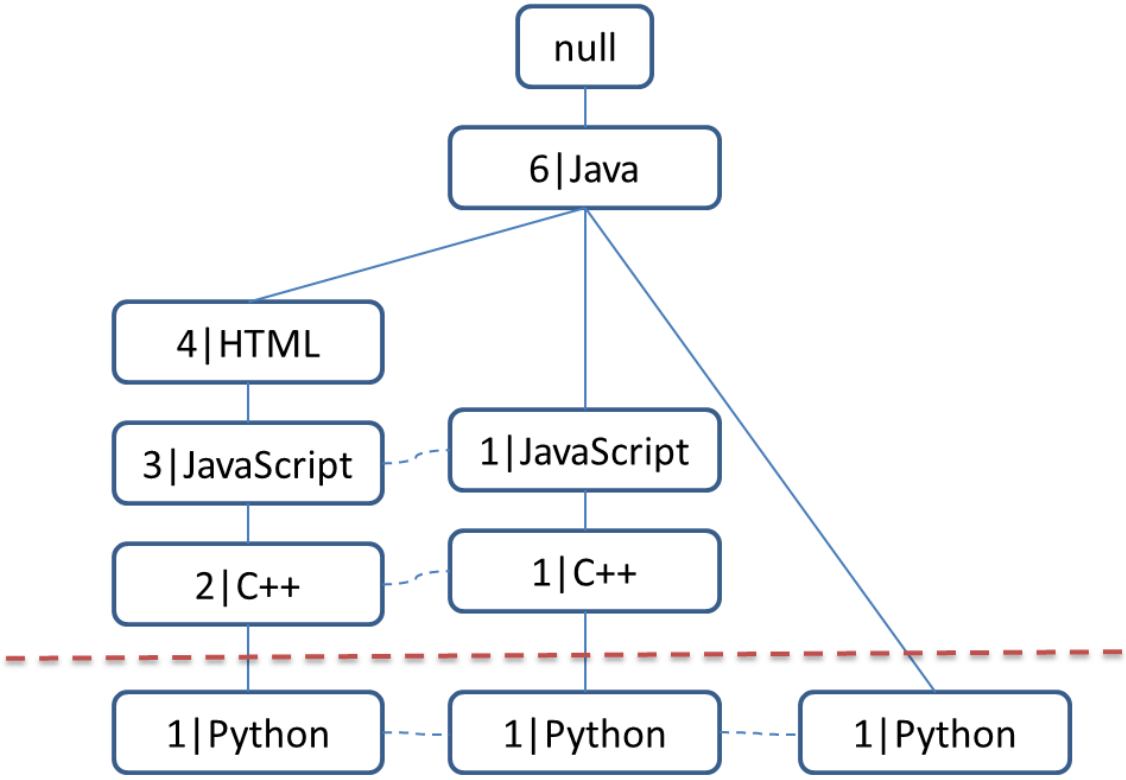

Figura 8. Subárbol para el elemento Python.

Está claro que `Python` es un itemset frecuente de longitud 1, los vínculos entre las hojas nos permiten realizar el recuento (en cualquier caso ya lo teníamos en la lista que se generó en la primera pasada sobre la BDD).

Los tres caminos posibles hasta `Python` son `Java>HTML>JavaScript>C++`, `Java>JavaScript>C++` y `Java` y cada uno de ellos tendría asociado el contador 1 como prefijos de `Python`, ya que en cada caso el elemento Python aparece una sola vez. Por lo tanto ninguno de esos itemsets seria frecuente (se fijó el soporte en 0.25). Haciendo un recuento de las veces que aparece en esas rutas cada elemento comprobamos que `C++` y `JavaScript` están dos veces, `HTML` una y `Java` tres. Es este último el único que supera el soporte mínimo, por lo que los patrones frecuentes para `Python` como elemento final serían:

`{3|Python}` y `{3|Java,Python}`

Cortando del árbol original completo las hojas que quedan debajo de la línea roja de la Figura 8 nos queda un árbol podado, como el mostrado en la Figura 9, en el que todas las hojas tienen el elemento C++.

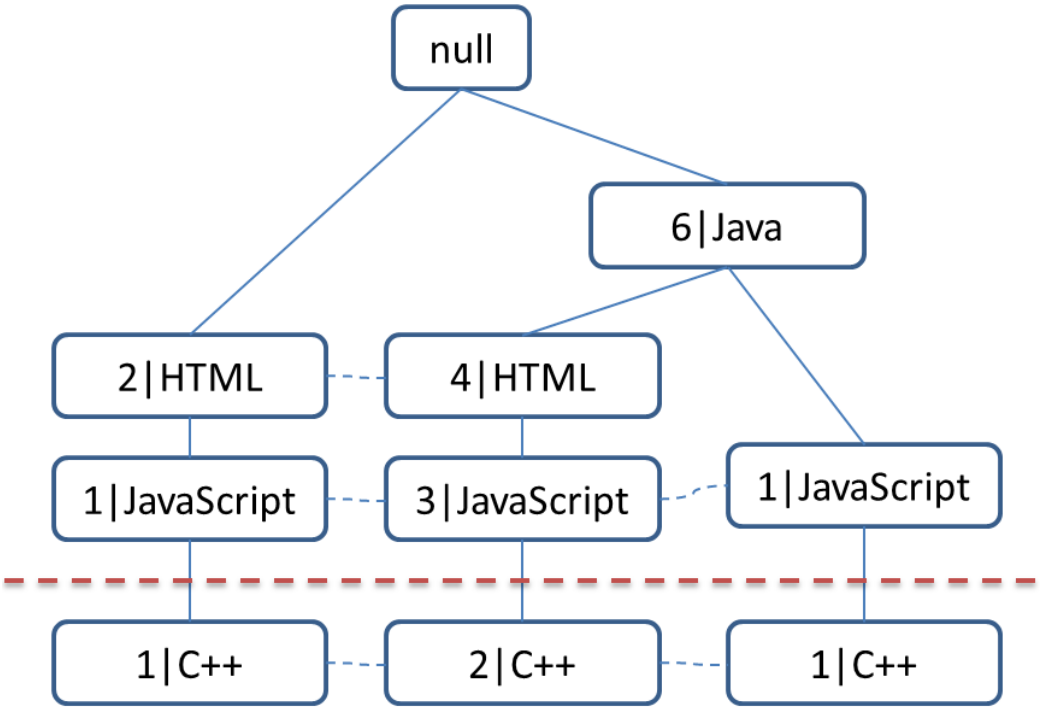

Figura 9. Árbol tras la primera poda.

El recuento del itemset de longitud 1 `C++` es 4 y los prefijos para llegar hasta él son `HTML>JavaScript` con valor 1, `Java>HTML>JavaScript` con valor 2 y `Java>JavaScript` con valor 1. El recuento para cada elemento, teniendo en cuenta que la segunda ruta multiplica por 2, es la siguiente: `JavaScript` 4 veces, `HTML` 3 y `Java` 3. Reordenamos las rutas de los prefijos de acorde a esta cuenta:

Ruta original	Ruta reordenada
`HTML>JavaScript`	`JavaScript>HTML`
`Java>HTML>JavaScript`	`JavaScript>Java>HTML`
`Java>JavaScript`	`JavaScript>Java`

A partir de las rutas reordenadas obtenemos el árbol condicional para el elemento C++ (hoja que aparece al final de cada ruta) que sería el mostrado en la Figura 10.

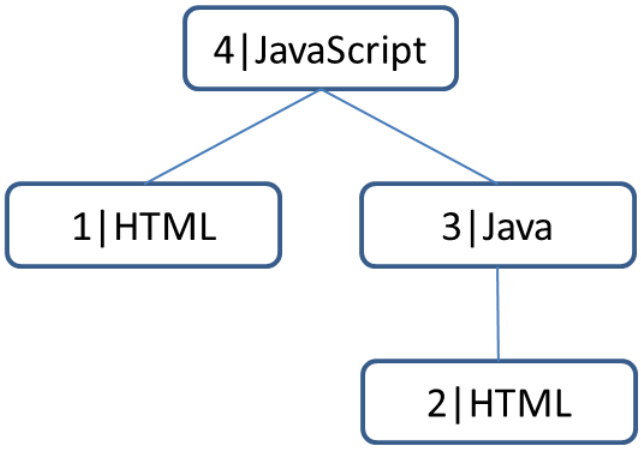

Figura 10. Árbol condicional para el elemento C++.

Las 3 apariciones de `HTML` se dividen en dos ramas (se deduce de las rutas y el contador asociado, en el paso previo). Procesando recursivamente este árbol, con el mismo procedimiento de los pasos anteriores, llegamos a obtener los itemsets frecuentes:

```
{4|C++}, {4|JavaScript, C++}, {3|Java, C++},
{3|HTML, C++}, {3|JavaScript, Java, C++} y
{3|JavaScript, HTML, C++}
```

Finalizado el procesamiento de este subárbol se produce la vuelta atrás de la recursividad, hacia el árbol en que se cortaban las ramas con el elemento C++. Se tiene un árbol en el que las hojas tienen todas el elemento `JavaScript` (véase la Figura 11), que serán las que se corten entrando en una nueva llamada recursiva:

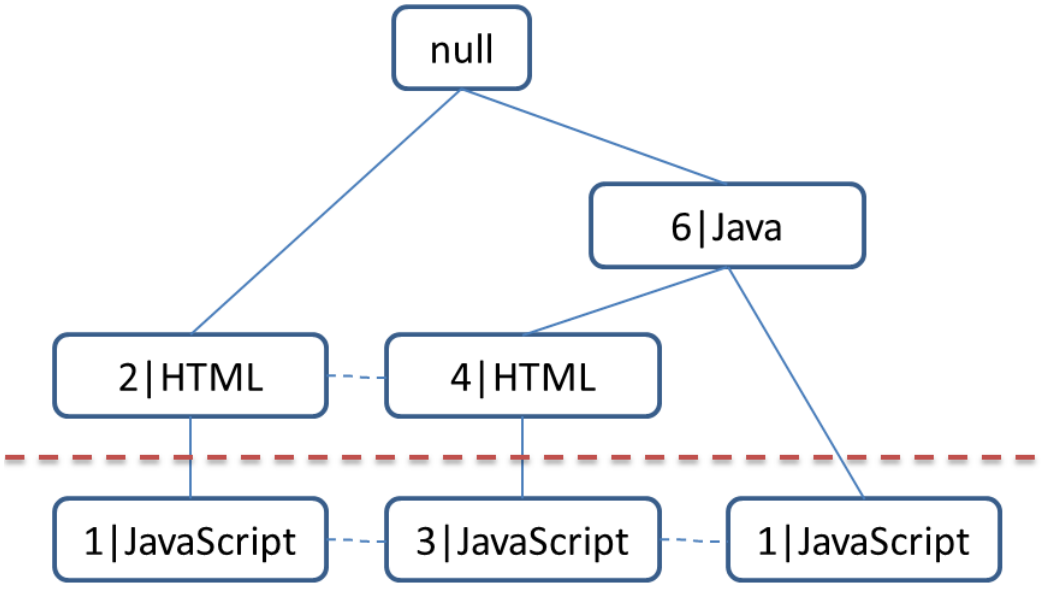

Figura 11. Estado del árbol al entrar en la nueva llamada.

Repitiendo una vez más el algoritmo obtenemos las rutas que llevan a `JavaScript` como hoja: `HTML` con valor 1, `Java>HTML` con valor 3 y `Java` con valor 1. El recuento de repeticiones de cada elemento seria 5 para `JavaScript`, 4 para `HTML` y 4 para `Java`. Se reordenan las rutas según los contadores para obtener el árbol condicional para el elemento `JavaScript` y se obtienen los itemsets frecuentes siguientes:

```
{5|JavaScript}, {4|JavaScript, Java},
{4|JavaScript, HTML} y
{3|JavaScript, Java, HTML}
```

En la siguiente iteración del proceso recursivo tenemos un árbol con el elemento `HTML` en las hojas, se cortará ese nivel inferior y, repitiendo todo el proceso, tenemos los siguientes itemsets frecuentes:

```
{6|HTML}, {6|Java} y {4|HTML,Java}
```

La ultima iteración nos lleva a quedarnos como raíz con el elemento `null`, por lo que el proceso ha terminado. Ya tenemos la lista completa de patrones frecuentes que existen en la BDD de transacciones con un nivel de soporte mínimo de 0.25.

A pesar de que en el desarrollo de este ejemplo de juguete el procedimiento puede parecer muy tedioso, hay que tener en cuenta que se trata de un proceso recursivo y que, por tanto, la implementación en un lenguaje de programación resultaría relativamente sencilla. En cualquier caso es más complejo que el algoritmo Apriori, como ya se apuntaba al principio de este apartado, pero es esa complejidad añadida la que permite extraer todos los itemsets frecuentes sin necesidad de generar patrones candidatos de manera exponencial.

Otro aspecto a tener en cuenta es que en este ejemplo la BDD de transacciones tiene muy pocas entradas y el árbol parece, proporcionalmente hablando, más grande. En una BDD real, con muchas más transacciones, la posibilidad de que existan raíces comunes se incrementa y, por tanto, habría una mayor concentración de información en el FP-Tree. Éste es un hecho que los autores del algoritmo analizan en su artículo y es una necesidad, ya que el FP-Tree es la estructura que se emplea recursivamente para obtener los itemsets frecuentes y, por tanto, debe caber completo en memoria.

4 Preprocesamiento de datos para clasificación multi-etiqueta

Este trabajo está centrado en el objetivo de mejorar la clasificación multi-etiqueta mediante métodos de transformación del problema y, en particular, intentando reducir la dimensionalidad. Aunque la propuesta presentada en esta sección se centra en un tipo particular de reducción de la dimensionalidad en problemas multi-etiqueta, en la primera parte de esta sección se analizan diferentes formas de reducción y algunas ideas aplicables a clasificación multi-etiqueta que se plantean como posibles líneas de trabajo futuro.

Habitualmente esta tarea se lleva a cabo a través de dos vías: la selección de instancias y la selección de características. Para este tipo de problema en particular también se podría preprocesar seleccionando etiquetas, es decir, extrayendo relaciones entre las etiquetas de manera que podamos eliminarlas al inicio e inferirlas al final. A continuación se realiza un análisis breve sobre cada una de las opciones posibles y se analizan las posibilidades que quedan abiertas al estudio.

La extensa bibliografía revisada para la realización de este trabajo recoge los esfuerzos de los investigadores por resolver el problema de la clasificación multi-etiqueta principalmente adaptando algoritmos ya existentes y, en mucha menor medida, a través del preprocesamiento de los datos que, hasta el momento, se fundamenta básicamente en las dos transformaciones descritas con anterioridad: *Binary Relevance* y *Power Labelset*.

Hasta donde sabemos, no se han definido métodos de preprocesamiento específicos para datasets multi-etiqueta, ya sean adaptaciones de técnicas existentes (como la selección de características) u otros diseñados a medida para este problema y que puedan ser aplicados a cualquiera de los algoritmos de clasificación multi-etiqueta con el objetivo de mejorar su funcionamiento.

El propósito del preprocesamiento de datos puede ir dirigido a obtener unos mejores resultados en las métricas de evaluación, es quizá el objetivo más obvio, pero también es interesante su aplicación para optimizar el rendimiento del clasificador, un aspecto que cobra mayor importancia cuando se emplean varios clasificadores (según los modelos BR y LP) y el número de etiquetas es grande. El tiempo de ejecución necesario para procesar un conjunto de datos relativamente grande con ciertos algoritmos, como los basados en cadenas y *ensembles* de clasificadores binarios (descritos en el segundo capítulo), puede provocar que la tarea sea inviable.

Dependiendo de la parte del dataset sobre la que actúe, el preprocesamiento puede aplicarse en tres espacios diferenciados o bien en una combinación de éstos. Denominando n al número de instancias y c al de atributos, dichos espacios serían:

- **Espacio de atributos**: El conjunto de datos se interpreta como un grupo de c columnas (las características) cada una de ellas con n valores, de las que quedarían en principio excluidas las correspondientes a las etiquetas.

- **Espacio de instancias**: El conjunto de datos se interpreta como un grupo de *n* filas (las muestras de datos) cada una de ellas con *c* valores en los que pueden estar incluidos o no los atributos correspondientes a las etiquetas.

- **Espacio de etiquetas**: Solamente se toman del conjunto de datos las columnas correspondientes a las etiquetas y se trabaja exclusivamente con los valores que toman éstas en cada una de las muestras.

Los problemas a los que puede aplicarse el trabajo de preprocesamiento en cada uno de estos espacios son lógicamente distintos. En los dos primeros, el de atributos e instancias, el enfoque más obvio pasa por la adaptación de algoritmos ya existentes para la selección de características e instancias. El tercero representa un espacio nuevo al considerar únicamente las etiquetas y sólo tiene sentido cuando la cardinalidad del dataset es superior a 1, es decir, en problemas multi-etiqueta, no binarios o multi-clase.

4.1 Preprocesamiento en el espacio de atributos

La alta dimensionalidad de los datasets, el hecho de que exista un gran número de características por muestra, representa un problema en sí misma para los clasificadores, sean o no multi-etiqueta. En muchas ocasiones, dependiendo de los algoritmos empleados, un incremento lineal en el número de atributos se traduce en un crecimiento exponencial de la complejidad del modelo obtenido, lo cual afecta de manera muy importante tanto al rendimiento como a la eficacia (medida como precisión o cualquier otra métrica de evaluación).

El grado de complejidad de ciertos modelos, como son los árboles, los sistemas basados en reglas o las redes neuronales, guardan una estrecha relación con la cantidad de atributos que es necesario tomar en consideración para construirlos. En una RNA el número de neuronas en la capa de entrada se fija en función del de atributos, mientras que los algoritmos basados en árboles de decisión a medida que generan las ramas tienen que evaluar qué atributo es el más óptimo en cada caso. El resultado son no solamente modelos más grandes y complejos, sino también más tiempo empleado en su obtención. Estos ejemplos podrían extrapolarse a los demás métodos de clasificación. Una reducción (selección) de características, por tanto, contribuirá a tener modelos más simples, más interpretables, más eficientes y, por regla general, con menos sobreajuste a los datos de entrenamiento. La selección de características es una de las tres técnicas, junto con la derivación de nuevos atributos y el clustering, que suelen aplicarse con el objetivo de reducir la dimensionalidad en el espacio de entrada. Aquí nos centraremos en la primera de dichas técnicas.

Los diferentes métodos de selección de características (Jain y Zongker 1997) se han agrupado tradicionalmente en dos categorías conocidas como *filters* (filtros) y *wrappers* (envoltorios). Los de la primera categoría realizan su trabajo tomando en consideración únicamente los valores de los atributos, mientras que los de la segunda se apoyan en un clasificador para evaluar cómo afecta al resultado la operación realizada.

Un filtro puede eliminar aquellos atributos que no resulten relevantes, por ejemplo al tomar el mismo valor prácticamente en todos los casos lo cual le hará inútil para diferenciar unas muestras de otras. También puede, mediante análisis a través de técnicas estadísticas, detectar atributos muy correlacionados y que resultan redundantes. Son métodos que siguen teniendo validez cuando se aplican a clasificadores multi-etiqueta, ya que son independientes de la clase o clases asociadas a cada instancia de datos.

No puede decirse lo mismo de los métodos de selección de características que analizan la correlación entre los atributos y la clase, ya sea con técnicas estadísticas o ejecutando un algoritmo de aprendizaje (*wrapper*), para decidir cuáles de ellos son los que más información aportan y descartar el resto. En el caso multi-etiqueta no hay una sola clase que pueda tomarse como referencia, existiendo una cierta probabilidad, más o menos alta dependiendo del conjunto de datos usado, de que unos atributos tengan gran influencia sobre la aparición de un subconjunto de etiquetas pero ninguna sobre otras, y viceversa.

Una propuesta lógica, tomando como base el uso de clasificadores binarios para procesar datasets multi-etiqueta, podría ser la aplicación del método de selección de características varias veces, una por cada etiqueta. De esta forma se obtendrían tantos subconjuntos de atributos como etiquetas, formado cada uno de ellos por las características que más influyen en cada etiqueta. Para obtener el conjunto final de atributos podría recurrirse a un sistema de voto por mayoría. Otro enfoque sería la elaboración, por parte del algoritmo de selección de características, de un ranking de todos los atributos para cada etiqueta. De esos rankings se tomarían los *i* atributos que estén al frente, produciendo el conjunto final de características que se emplearían para realizar el entrenamiento del clasificador. Una problemática común a estas propuestas estriba en la limitación que implica el considerar las etiquetas de manera individual, sin tomar en cuenta las relaciones existentes entre ellas, un aspecto que es importante solventar.

En un apartado previo se apuntaba que una de las características habituales de los datasets multi-etiqueta es su alta dimensionalidad en cuanto a atributos de entrada se refiere. Tal y como argumentan los autores en (Hua, Tembe y Dougherty 2009), los métodos tradicionales de selección de características han sido pensados para operar con unas pocas decenas de atributos, pero no con cientos o miles de ellos. En esos casos no es esperable que ofrezcan un subconjunto óptimo de características, por lo que la combinación de alta dimensionalidad y multi-etiqueta se convierte en un problema más, que será necesario abordar en el futuro.

4.2 Preprocesamiento en el espacio muestral

Los procedimientos más habituales para preprocesar las muestras de un conjunto de datos son la selección de instancias, la eliminación de inconsistencias y la detección de *outliers* o muestras anómalas, entendiendo como tales aquellas muy diferentes al resto.

El problema de la alta dimensionalidad afecta no solamente a la cantidad de características por muestra, sino también al número de muestras que componen el conjunto de datos que, en ocasiones, puede ascender a decenas o cientos de miles. Procesar tal número de instancias afecta tanto al tiempo empleado por los algoritmos como a la cantidad de memoria necesaria para su tratamiento, por lo que cualquier método que permita reducirlo es bienvenido.

Determinados algoritmos de clasificación incrementan su conocimiento acerca del conjunto de datos sobre el que trabajan únicamente al procesar muestras diferentes, las que son iguales no aportan nada o casi nada al modelo que se construye. Un ejemplo de ello son los árboles o el perceptrón. Para el resto, la aparición de muestras iguales va reduciendo paulatinamente la tasa de error cometida, es decir, disminuye la incertidumbre sobre la clase que corresponde a un cierto conjunto de valores en los atributos cuando éstos se repiten, siendo factible el cálculo de cuántas instancias se necesitan como máximo dependiendo de la tasa de error que quiera alcanzarse. El objetivo sería, por tanto, quedarse con aquellas instancias con mayor representatividad respecto a cada una de las clases existentes, aplicando para ello un proceso de selección de instancias que, como se explica en (Michalski 1975), puede llevarse a cabo de diferentes maneras: definiendo una medida de distancia que pueda calcularse para cada muestra (fila de una tabla en el artículo) y seleccionando aquellas instancias que están más distantes entre sí; quedándose con aquellas muestras que están cerca de la línea que separa a unas clases de otras o bien aplicando una técnica de clustering y eligiendo los centroides como representantes de los grupos de instancias.

Los citados métodos de selección de instancias, y otros que surgieron con posterioridad, han de ser rediseñados antes de poder aplicarse a datasets multi-etiqueta, tomando en consideración las siguientes especificidades:

- **Equilibrio entre número de etiquetas y de instancias**: Existen datasets muti-etiqueta en los que el número de etiquetas está cercano, o incluso por encima, al número de muestras de datos. Si la cardinalidad es baja, esto implicará que la mayor parte de esas muestras pertenecerán a clases distintas y, en consecuencia, resulta difícil extraer un modelo válido para la clasificación de muestras no vistas con anterioridad.

- **Desbalanceo en el número de muestras por etiqueta**: El tratamiento de datasets desbalanceados, en los que existen muchas muestras asociadas a unas clases y muy pocas a otras, es uno de los desafíos afrontados en el campo de la minería de datos durante los últimos años, tal y como se destaca en (Yang y Wu 2006). Este problema puede verse agravado al trabajar con datasets multi-etiqueta, ya que al hecho de que haya muchas muestras representativas de unas etiquetas y pocas de otras se suma su aparición conjunta en grupos de etiquetas para cada instancia, resultando en una representatividad aún menor de la etiqueta con menor frecuencia de aparición. Es una problemática que podría abordarse con técnicas similares a las de la transformación BR, separando las muestras según las etiquetas que tienen asociadas.

- **Dispersión de los datos**: Cuando la cardinalidad es relativamente alta en datasets con un importante número de etiquetas, y en especial si se aplican transformaciones como LP que generan una clase única por cada combinación distinta de etiquetas, habrá que hacer frente también a un problema de dispersión en las muestras, pudiendo llegarse al extremo de que todas las instancias sean diferentes y, en consecuencia, no pueda llevarse a cabo una selección de las más representativas.

El tratamiento de inconsistencias en un dataset se basa en la detección de la existencia de muestras de datos idénticas, con los mismos valores en todos sus atributos, pero que tienen asociada una clase distinta. Normalmente dichas muestras se eliminan a fin de facilitar el proceso de aprendizaje del clasificador, asumiendo que a idénticas características de entrada debería corresponder un idéntico resultado de salida: la misma clase. Cuando no existe un solo valor de salida sino varios, como ocurre en la clasificación multi-etiqueta, eliminar instancias con idénticas características por el hecho de que en una aparezca una clase y en otra no, implicaría pérdida de información potencialmente valiosa: los datos sobre el resto de etiquetas asociadas a dichas instancias. Una posible adaptación, para aplicar esta tarea sobre datasets multi-etiqueta, podría consistir en separar las instancias en subconjuntos disjuntos respecto a las etiquetas que tienen asociadas. Las muestras de un subconjunto tendrían una o más etiquetas comunes entre sí, pero ninguna con las muestras de los demás subconjuntos. Aquellas instancias que sean idénticas en cuanto a los valores de sus características pero residan en subconjuntos distintos podrían considerarse inconsistentes.

Una tercera vertiente del preprocesamiento de instancias es el relativo al tratamiento del ruido y las excepciones: muestras de datos con características atípicas respecto a la mayor parte de las que componen el dataset. Cuando se entrena un clasificador a partir de datos con ruido el resultado es un sobreajuste (*overfitting*) del modelo, algo no deseable ya que afecta a la capacidad de generalización y, por tanto, la habilidad para tratar con muestras de datos no conocidas. Por ello es importante aplicar al dataset filtros que eliminen el ruido tal y como se demuestra en (Brodley y Friedl 1996).

La diferencia entre el ruido y las excepciones (*outliers*) a veces es sutil, pero ha de ser tenida muy en cuenta. Una excepción es una muestra que no se ajusta al mismo modelo que el resto de los datos, pero no se trata de ruido (datos con errores), por lo que no ha de ser eliminado sin más. Es más, en ocasiones son precisamente esas excepciones los que aportan información de mayor interés a la hora de descubrir nuevo conocimiento, por ejemplo en campos como la investigación de dolencias poco comunes, el fraude en uso de tarjetas, etc. Existen métodos que, como el propuesto en (Knorr y Ng 1997), pueden aplicarse de forma general incluso a datasets de gran tamaño, pero ninguno está preparado para abordar datasets multi-etiqueta. Las características de éstos, en especial la dispersión a la que se hacía referencia con anterioridad al aplicar transformaciones como LP, tiene como efecto la aparición de un gran número de muestras que serían consideradas excepciones con los algoritmos de preprocesamiento tradicionales.

4.3 Preprocesamiento en el espacio de etiquetas

Determinadas técnicas de preprocesamiento, como la selección de características o de instancias, emplean la clase asociada a cada muestra para analizar las correlaciones existentes y, por ejemplo, determinar qué atributos son los que aportan mayor información útil para el modelo a construir. Más allá de esta aplicación, ciñéndonos al preprocesamiento para problemas de clasificación binaria o multi-clase, el valor del atributo que establece la clase de cada muestra no suele ser de mucha más utilidad. En la clasificación multi-etiqueta, por el contrario, el conjunto de etiquetas de cada muestra puede llegar a ser incluso mayor que el de características de entrada.

Algunos de los problemas que pueden presentarse en el espacio de etiquetas son, en cierta manera, análogos a los afrontados en el espacio de atributos: alta dimensionalidad (muchas etiquetas diferentes), redundancia, etc. Otros, por el contrario, son más específicos, como puede ser una cardinalidad elevada que dé lugar a multitud de combinaciones distintas. Se abre, por tanto, un nuevo espacio para las tareas de preprocesamiento que no existía hasta el momento: el que actúa exclusivamente sobre las etiquetas.

La cantidad total de etiquetas y la cardinalidad son aspectos que influyen directamente en los métodos de transformación (BR y LP) y también afectan, en mayor o menor medida, a los de adaptación. Cuando se recurre a la combinación *Binary Relevance+clasificador binario*, por ejemplo, es preciso entrenar tantos clasificadores como etiquetas distintas haya en cada dataset. Reducir el número de etiquetas mejoraría la eficiencia del clasificador y también afectará a su precisión. Cuando la cardinalidad es alta, la combinación *Label Powerset+clasificador multi-clase* se encuentra con el obstáculo del gran número de combinaciones posibles que, como se apuntaba anteriormente, provoca dispersión y desbalanceo. La reducción de la cardinalidad relajaría la explosión combinatoria y, teóricamente, mejoraría el clasificador.

A diferencia de lo que ocurre con los métodos de selección de características, que pueden derivar en la eliminación de uno o más atributos del dataset antes de que éste sea entregado al algoritmo que entrena el clasificador, una hipotética selección de etiquetas no puede hacer exactamente lo mismo. Las etiquetas que se eliminen del dataset original han de recuperarse con posterioridad, ya que de lo contrario sencillamente desaparecerían de los resultados de la clasificación. Se trataría, en consecuencia, de una *ocultación* temporal de ciertas etiquetas, de forma que el clasificador se entrene con un conjunto reducido de ellas obteniendo así un modelo más simple (más fácil de interpretar), más eficiente (menor complejidad computacional) y, probablemente, más eficaz (mayor precisión). Finalizado el entrenamiento, el clasificador ofrecería para cada muestra de test una predicción de etiquetas entre las que no estarían visibles las que se habían ocultado. La presencia o no de éstas en el resultado final quedaría en manos de un proceso de postprocesamiento.

La metodología que se propone aquí está basada en estas ideas a través del uso de reglas de asociación (descritas en el capítulo 3) y se divide en dos fases: preprocesamiento y postprocesamiento. La primera se iniciará con una selección de etiquetas que, básicamente, se guiará por los pasos siguientes:

- Aplicar sobre las etiquetas del dataset un algoritmo de obtención de reglas de asociación adecuado para grandes conjuntos de datos, como FP-Growth, en previsión de la necesidad de tratar datasets con gran número de etiquetas e instancias.

- Ordenar las reglas obtenidas según su confianza y quedarse con aquellas que estén por encima de un cierto umbral, habitualmente λ=0.5.

- Para cada una de las reglas existentes en la lista (que es necesario conservar hasta el final) recorrer el consecuente, formado por una o más etiquetas que es posible deducir a partir del consecuente, eliminando del conjunto de etiquetas original aquellas que forman parte del consecuente.

Terminada esta fase se tiene un dataset con un conjunto de etiquetas reducido, por una parte, y una serie de reglas de asociación, por otra. Ese dataset reducido se entrega al algoritmo de clasificación multi-etiqueta, puede ser alguno de los descritos en el capítulo 2 o cualquier otro, para llevar a cabo el entrenamiento.

El postprocesamiento se llevará a cabo tras la predicción hecha por el clasificador para cada muestra de test. En ese instante se habrán de aplicar las reglas de asociación, guardadas tras el preprocesamiento, con el objetivo de recuperar las etiquetas que es posible deducir de las reglas. De esta forma se completará la predicción del conjunto de etiquetas que corresponde a cada muestra procesada.

4.4 PROPUESTA: Reducción de la cardinalidad con reglas de asociación extraídas con el algoritmo FP-Growth

El propósito fundamental de este trabajo es la propuesta de un método original que, mediante la aplicación de un preprocesamiento y postprocesamiento y apoyándose en la extracción previa de reglas de asociación en el espacio de etiquetas, permite reducir la cardinalidad de los datasets multi-etiqueta. La cardinalidad, promedio de etiquetas por instancias del dataset, en un dataset multi-etiqueta es un factor que influye de manera importante en el rendimiento de los clasificadores, afectando tanto a la eficiencia como a su eficacia. Al emplear técnicas de transformación como las antes mencionadas, con BR a mayor cardinalidad más clasificadores binarios habrá que emplear, por lo que la complejidad computacional se incrementa, y con LP el número de combinaciones de etiquetas también aumenta, lo cual se traduce generalmente en una reducción de la precisión al clasificar. Parece lógico por tanto pensar que si se consigue reducir la cardinalidad, especialmente cuando se da un problema de alta dimensionalidad en el espacio de etiquetas, el funcionamiento del clasificador podría mejorar globalmente.

Operando exclusivamente en el mencionado espacio de etiquetas (ignorando el resto de atributos), asumiremos que cada una de ellas es un *ítem* y que las etiquetas asignadas a cada instancia del dataset forman una transacción. El objetivo es obtener, mediante el algoritmo FP-Growth descrito en el tercer capítulo de este trabajo, reglas que permitan inferir unas etiquetas a partir de la presencia de otras con un cierto nivel de confianza, lo cual permitiría *ocultarlas* al clasificador multi-etiqueta. Las etiquetas ocultadas se agregan a cada muestra tras la predicción hecha por el clasificador, en una fase de postprocesamiento, sencillamente aplicando las reglas obtenidas con anterioridad.

La cantidad de reglas obtenidas a partir de un dataset multi-etiqueta es un parámetro que dependerá no del número de etiquetas diferentes que contenga, sino de la cardinalidad, es decir, el promedio de etiquetas por instancia del dataset. Si dicha medida es pequeña, cercana a 1 como ocurre con algunos de los datasets usados para la experimentación (véase página 51), las transacciones estarán formadas en su mayor parte por 1 ó 2 items y será difícil que puedan obtenerse reglas. De ciertos datasets, como es el caso de `Scene` (usado anteriormente en la experimentación de los métodos de transformación, apartado 2.5), no resulta posible obtener regla alguna con un soporte y confianza mínimos, lo cual no es de extrañar ya que su cardinalidad es de sólo 1,074. Por esta razón dicho dataset, y otros similares (todos ellos obtenidos del repositorio de MULAN), no serán incluidos en el proceso experimental.

Análogamente, cuando al aplicar una de las reglas obtenidas se eliminan una o más etiquetas de un dataset debe tenerse en cuenta que, aparte de disminuir el número total de etiquetas en una unidad, el parámetro que se verá más afectado será precisamente la cardinalidad, reducida de manera importante. En teoría esto implicará que los métodos tradicionales, no diseñados expresamente para este tipo de clasificación, puedan realizar un mejor tratamiento de los datos, al acercar la cardinalidad al valor unidad característico de los problemas de clasificación que no son multi-etiqueta.

El método propuesto, incluyendo preprocesamiento y postprocesamiento es el descrito en el siguiente seudocódigo:

```
1.  X = Dataset a procesar
2.  NR = Número de reglas a aplicar (0=todas las obtenidas)
3.  T = Ø  # Conjunto de etiquetas de las instancias (transacciones)
4.  Para cada instancia Xi en X
5.    Li = etiquetas asociadas a Xi
6.    T = T ∪ Li
7.  R = FPGrowth(T)     # Conjunto de reglas ordenado por confianza
8.  MR = SubC(R, NR)    # NR mejores reglas en R
9.  C = EtiquetasEnConsecuente(MR)
10. X = X - C    # Eliminar del dataset original las etiquetas en C
11. DTra = ParticionEntrenamiento(X)
12. DTst = ParticionPrueba(X)
13. Clas = EntrenaClasificador(DTra)
14. Para cada instancia Xi en DTst
15.   Pi = Clas(Xi) # Obtener la i-ésima predicción del clasificador
16.   Pi = Pi ∪ AplicaReglas(MR) # Inferir etiquetas con las reglas
17. Evaluación(P)
```

Dado que el algoritmo de minería de reglas de asociación puede obtener múltiples reglas (línea 6), cada una de ellas con una determinada confianza, el método cuenta con un parámetro de entrada llamado `NR` que permite especificar cuántas reglas quieren aplicarse como máximo. Si `NR=0` se usarán todas las reglas obtenidas, al igual que si `NR > |R|`. En cualquier otro caso se seleccionará el número de reglas indicado siempre de mayor a menor confianza. Es posible plantearse métodos alternativos para la selección del número de reglas a aplicar, en sustitución del mencionado parámetro, como podrían ser:

- **Confianza mínima**: Tomar como parámetro de entrada un nivel mínimo de confianza, aplicando aquellas reglas que lo alcancen, modificando para ello la selección que se hace en la línea 8 del seudocódigo y que en la implementación actual usa una constante λ=0.5.

- **Ajuste heurístico**: Definir una heurística que a partir de las características del dataset: número total de etiquetas, cardinalidad, densidad, etc., ajuste el número de reglas que sería óptimo aplicar, prescindiendo así de parámetro de entrada alguno. Éste es un trabajo que queda pendiente de estudio.

Una vez que se ha entrenado el clasificador `Clas`, al procesar la partición de test entra en escena el proceso de postprocesamiento que, como puede verse en la línea 16, se limita a aplicar sobre la predicción del clasificador para cada muestra de test las reglas con las que se realizó el entrenamiento, infiriendo las etiquetas que fuese necesario agregar. Finalmente, teniendo en `P` el conjunto de predicciones total para las muestras de test, se lleva a cabo el cálculo de las medidas de evaluación de calidad.

El método descrito ha sido implementado en Java e integrado con el software MULAN, de manera que el proceso de particionado del dataset, entrenamiento del clasificador y obtención de resultados de test recae por completo en MULAN. El preprocesamiento actúa creando un nuevo dataset multi-etiqueta del que se han eliminado las etiquetas de las reglas elegidas, dataset que se entrega a MULAN como entrada y que no difiere en nada respecto a cualquier otro. Para integrar el postprocesamiento se ha derivado una clase especializada a partir de la clase `Evaluator` de MULAN, agregando las etiquetas inferidas de las reglas tras haber obtenido la predicción del clasificador pero antes de que se calculen las medidas de evaluación.

MULAN almacena por cada muestra procesada la bipartición de etiquetas asociada y también la medida que, asociada a cada una de ellas, servirá para realizar el ranking de etiquetas. Si al aplicar las reglas de asociación, durante el postprocesamiento, solamente se actualizase la bipartición no sería posible obtener las medidas basadas en ranking. Por ello también se entrega a MULAN, como medida de ranking, la confianza correspondiente a la regla de asociación.

4.5 Configuración de experimentación

Con el objeto de comprobar de forma empírica el método propuesto se seleccionó un conjunto de datasets del repositorio de MULAN en el que se mezclasen características dispares: muchas vs pocas etiquetas, alta vs baja cardinalidad, etc. En la Tabla 6 se indica el nombre de cada dataset y sus características fundamentales en el espacio de instancias, atributos y etiquetas.

Datasets	Instancias		Atributos		Etiquetas		
Nombre	Número	Distintas	Nominales	Numéricos	Etiquetas	Cardinalidad	Densidad
emotions	593	27	0	72	6	1,869	0,311
CAL500	502	502	0	68	174	26,044	0,150
Corel5k	5000	3175	599	0	374	3,522	0,009
enron	1702	753	1001	0	53	3,378	0,064
genbase	662	32	1186	0	27	1,252	0,046
medical	978	94	1449	0	45	1,245	0,028
yeast	2417	198	0	103	14	4,237	0,303

Tabla 6. Datasets empleados en la experimentación y sus características.

La ejecución del algoritmo FP-Growth sobre las etiquetas de cada dataset, a fin de extraer las reglas de asociación resultantes, se ha hecho con los siguientes parámetros:

- **Soporte**: El rango habitual suele fijarse entre 0.01 y 0.1, fundamentalmente dependiendo del tamaño de la base de transacciones a procesar. En nuestro caso se ha establecido a 0.025.

- **Confianza**: El objetivo es obtener siempre las reglas con mayor confianza posible, de ahí que el valor máximo para esta medida se haya fijado en 1.0 y el mínimo en 0.5.

- **Nº de reglas**: Se ha fijado el número máximo de reglas a obtener en 10, de forma que el algoritmo FP-Growth partirá intentando obtener reglas con el máximo nivel de confianza (1.0 en este caso), reduciéndolo paulatinamente hasta alcanzar el nivel mínimo o haber recuperado el máximo de reglas establecido. El nivel de confianza se reduce a cada paso en una constante *delta* que se ha fijado a 0.05.

En cuanto a los algoritmos de clasificación multi-etiqueta usados para esta experimentación, se ha querido tener una representación de métodos basados tanto en la transformación de datos como en la adaptación de métodos. Debe tenerse en cuenta que los de transformación hacen su trabajo después de que se haya completado el preprocesamiento descrito anteriormente, operando sobre el dataset reducido del que ya han sido eliminadas las etiquetas que sea posible inferir de las reglas. Los métodos elegidos, todos ellos implementados en MULAN, han sido:

- **BR-J48**: Lleva a cabo una transformación de tipo *Binary Relevance* y usa como algoritmo base para la clasificación binaria el algoritmo C4.5, denominado J48 en Weka/MULAN.

- **LP-J48**: Lleva a cabo una transformación de tipo *Label Powerset* y usa como algoritmo base para la clasificación multi-clase el algoritmo C4.5, como en el caso anterior.

- **MLkNN**: Es el algoritmo propuesto por (Zhang y Zhou 2007) y que se describió en el apartado *2.3.2 Métodos basados en instancias*.

- **IBLR-ML**: Es el algoritmo propuesto por (Cheng y Hüllermeier 2009) y que se describió en el apartado *2.3.2 Métodos basados en instancias*.

- **BP-MLL**: Es el algoritmo propuesto por (Zhang y Zhou 2006) y que se describió en el apartado *2.3.3 Métodos basados en redes neuronales*.

Cada ejecución del método propuesto completo, una vez fijado el número de reglas a aplicar, preprocesado el dataset y fijado el algoritmo de clasificación multi-etiqueta a usar, se repite 5 veces distribuyendo en cada una de ellas aleatoriamente el orden de las muestras que componen el dataset. En cada ejecución se emplea validación cruzada con la habitual configuración en 10 particiones, por lo que se tiene un total de 5 repeticiones x 10 particiones = 50 ejecuciones del algoritmo base. Los resultados obtenidos son promedios de esas 50 ejecuciones, obteniéndose para cada combinación de dataset-algoritmo dos medidas: *HammingLoss* (HL) y *AvgeragePrecision* (AP), descritas ambas en el apartado 2.1.

En principio el experimento se configuró de forma que se aplicasen todas las reglas obtenidas (con la confianza mínima establecida) sobre cada uno de los datasets. El análisis de los resultados (véase el apartado siguiente) obtenidos nos indujo a realizar una segunda tanda de pruebas fijando el número de reglas a aplicar en 1: solamente la regla de mayor confianza.

4.6 Resultados obtenidos y análisis

La primera fase del experimento ejecutó cada algoritmo base para procesar los datasets tras aplicar todas las reglas extraídas. En las tablas 7, 8, 9, 10 y 11 se recogen los resultados obtenidos. Los mejores valores para cada dataset/medida se han destacado en negrita.

BR-J48	Hamming Loss		Avg Precision	
Dataset	Base	Propuesto	Base	Propuesto
emotions	**0,2474 ± 0,0248**	0,2793 ± 0,0277	**0,7014 ± 0,0316**	0,6547 ± 0,0330
CAL500	0,1615 ± 0,0049	**0,1608 ± 0,0047**	0,3513 ± 0,0141	**0,3536 ± 0,0143**
Corel5k	0,0098 ± 0,0001	**0,0097 ± 0,0001**	**0,2494 ± 0,0093**	0,2352 ± 0,0077
enron	**0,0508 ± 0,0022**	0,0581 ± 0,0025	**0,5929 ± 0,0213**	0,5606 ± 0,0202
genbase	**0,0011 ± 0,0010**	0,0049 ± 0,0015	**0,9927 ± 0,0042**	0,8931 ± 0,0364
medical	**0,0103 ± 0,0014**	0,0144 ± 0,0017	0,8341 ± 0,0278	**0,8342 ± 0,0283**
yeast	**0,2454 ± 0,0091**	0,2508 ± 0,0100	**0,6216 ± 0,0151**	0,5807 ± 0,0170

Tabla 7. Resultados BR-J48 aplicando todas las reglas.

LP-J48	Hamming Loss		Avg Precision	
Dataset	Base	Propuesto	Base	Propuesto
emotions	0,2777 ± 0,0258	**0,2771 ± 0,0244**	0,6608 ± 0,0388	**0,6647 ± 0,0288**
CAL500	0,1996 ± 0,0049	**0,1991 ± 0,0054**	0,1161 ± 0,0052	**0,1376 ± 0,0043**
Corel5k	0,0168 ± 0,0002	**0,0161 ± 0,0002**	0,0212 ± 0,0031	**0,0476 ± 0,0063**
enron	**0,0716 ± 0,0028**	0,0761 ± 0,0027	0,2245 ± 0,0168	**0,3118 ± 0,0245**
genbase	**0,0019 ± 0,0019**	0,0055 ± 0,0019	**0,9871 ± 0,0095**	0,8859 ± 0,0346
medical	**0,0135 ± 0,0016**	0,0162 ± 0,0016	0,7503 ± 0,0333	**0,7857 ± 0,0329**
yeast	**0,2779 ± 0,0168**	0,2809 ± 0,0101	**0,5723 ± 0,0282**	0,5322 ± 0,0157

Tabla 8. Resultados LP-J48 aplicando todas las reglas.

MLkNN	Hamming Loss		Avg Precision	
Dataset	Base	Propuesto	Base	Propuesto
emotions	**0,1951 ± 0,0243**	0,2299 ± 0,0203	**0,7965 ± 0,0406**	0,7226 ± 0,0345
CAL500	0,1388 ± 0,0050	0,1388 ± 0,0047	**0,4943 ± 0,0168**	0,4912 ± 0,0147
Corel5k	0,0094 ± 0,0001	0,0094 ± 0,0001	**0,2458 ± 0,0075**	0,2304 ± 0,0074
enron	**0,0524 ± 0,0020**	0,0573 ± 0,0023	**0,6310 ± 0,0148**	0,5610 ± 0,0183
genbase	**0,0050 ± 0,0027**	0,0081 ± 0,0022	**0,9875 ± 0,0103**	0,8887 ± 0,0349
medical	**0,0151 ± 0,0020**	0,0189 ± 0,0017	**0,8134 ± 0,0264**	0,7653 ± 0,0316
yeast	0,1933 ± 0,0123	0,1933 ± 0,0091	**0,7658 ± 0,0208**	0,6896 ± 0,0114

Tabla 9. Resultados MLkNN aplicando todas las reglas.

IBLR	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	**0,1883**	**± 0,0239**	0,2206	± 0,0159	**0,8126**	**± 0,0349**	0,7333	± 0,0348
CAL500	0,2307	± 0,0054	**0,2300**	**± 0,0049**	**0,2713**	**± 0,0118**	0,2613	± 0,0097
Corel5k	0,0225	± 0,0009	**0,0219**	**± 0,0010**	0,1444	± 0,0067	**0,1469**	**± 0,0081**
enron	**0,0557**	**± 0,0023**	0,0611	± 0,0026	**0,6186**	**± 0,0228**	0,5397	± 0,0193
genbase	**0,0027**	**± 0,0019**	0,0059	± 0,0019	**0,9872**	**± 0,0077**	0,8914	± 0,0330
medical	**0,0197**	**± 0,0021**	0,0221	± 0,0022	**0,7478**	**± 0,0211**	0,7345	± 0,0314
yeast	0,1934	± 0,0118	**0,1926**	**± 0,0092**	**0,7687**	**± 0,0200**	0,6950	± 0,0136

Tabla 10. Resultados IBLR-ML aplicando todas las reglas.

BPMLL	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	**0,2061**	**± 0,0228**	0,2633	± 0,0255	**0,8094**	**± 0,0323**	0,6746	± 0,0378
CAL500	**0,2478**	**± 0,0094**	0,2510	± 0,0087	**0,5008**	**± 0,0144**	0,4778	± 0,0142
Corel5k	0,8315	± 0,2246	**0,7736**	**± 0,3587**	**0,0134**	**± 0,0011**	0,0126	± 0,0010
enron	**0,3831**	**± 0,1344**	0,6424	± 0,0684	**0,2150**	**± 0,0766**	0,1226	± 0,0139
genbase	**0,5702**	**± 0,2260**	0,6160	± 0,1892	0,1086	± 0,0193	**0,1111**	**± 0,0278**
medical	**0,4881**	**± 0,2669**	0,5256	± 0,2363	**0,1353**	**± 0,0268**	0,1273	± 0,0161
yeast	**0,2258**	**± 0,0099**	0,2264	± 0,0087	**0,7496**	**± 0,0184**	0,6905	± 0,0131

Tabla 11. Resulados BP-MLL aplicando todas las reglas.

De la observación de estos resultados lo primero que se deduce es que el método propuesto incide claramente de forma distinta cuando se aplica a un algoritmo basado en transformación de datos, como son BR-J48 y LP-J48, y a un algoritmo adaptado como son el resto. En los dos primeros casos las mejores medidas se obtienen unas veces con el algoritmo base y otras con el propuesto, dándose prácticamente un empate que en ocasiones se decanta por la segunda opción, como ocurre con LP-J48 (Tabla 8) para la precisión media. Cuando el método se aplica sobre un algoritmo adaptado para el problema de clasificación multi-etiqueta, como son MLkNN, IBLR-ML y BP-MLL, se consigue mejorar en muy pocos casos, por lo que todo parece indicar que la técnica descrita resulta aplicable a métodos de transformación del problema más que a algoritmos diseñados para afrontar directamente la clasificación multi-etiqueta.

Apreciamos, no obstante, que para los datasets de mayor tamaño: `CAL500` y `Corel5k`, la reducción de etiquetas aplicando todas las reglas sí que se traduce en una mejora en el caso de MLkNN e IBLR-ML, teniendo menor incidencia en BP-MLL. Asimismo son los datasets en los que mejor funciona el método propuesto cuando el algoritmo empleado es de transformación: BR-J48 y LP-J48. En la Tabla 6 (página 51) puede comprobarse que son los datasets con mayor número de etiquetas: 174 para `CAL500` y 374 para `Corel5k`, correspondiéndoles también las cardinalidades más altas. Parece lógico concluir, por tanto, que la eliminación del máximo número de etiquetas (aplicación de todas las reglas) influye de manera positiva cuando la cardinalidad y cantidad total de etiquetas es grande.

Tras el análisis previo nos planteamos cuáles serían los resultados si solamente se aplicase la regla de asociación de mayor confianza en cada caso, por lo que repetimos toda la batería de pruebas modificando dicho parámetro. Los resultados obtenidos son los que pueden verse en las tablas 12, 13, 14, 15 y 16. Como en las anteriores, los mejores valores se han destacado en negrita.

BR-J48	Hamming Loss		Avg Precision	
Dataset	Base	Propuesto	Base	Propuesto
emotions	**0,2474 ± 0,0248**	0,2585 ± 0,0243	0,7014 ± 0,0316	**0,7015 ± 0,0289**
CAL500	0,1615 ± 0,0049	**0,1618 ± 0,0047**	0,3513 ± 0,0141	**0,3534 ± 0,0171**
Corel5k	0,0098 ± 0,0001	0,0098 ± 0,0001	**0,2494 ± 0,0093**	0,2468 ± 0,0099
enron	**0,0508 ± 0,0022**	0,0533 ± 0,0028	**0,5929 ± 0,0213**	0,5853 ± 0,0221
genbase	**0,0011 ± 0,0010**	0,0031 ± 0,0011	**0,9927 ± 0,0042**	**0,9403 ± 0,0257**
medical	**0,0103 ± 0,0014**	0,0144 ± 0,0017	0,8341 ± 0,0278	**0,8342 ± 0,0283**
yeast	**0,2454 ± 0,0091**	0,2468 ± 0,0097	0,6216 ± 0,0151	**0,6403 ± 0,0157**

Tabla 12. Resultados BR-J48 aplicando únicamente la regla de mayor confianza.

LP-J48	Hamming Loss		Avg Precision	
Dataset	Base	Propuesto	Base	Propuesto
emotions	0,2777 ± 0,0258	**0,2476 ± 0,0291**	0,6608 ± 0,0388	**0,7017 ± 0,0313**
CAL500	**0,1996 ± 0,0049**	0,2000 ± 0,0060	0,1161 ± 0,0052	**0,1401 ± 0,0043**
Corel5k	0,0168 ± 0,0002	**0,0167 ± 0,0002**	0,0212 ± 0,0031	**0,0262 ± 0,0050**
enron	0,0716 ± 0,0028	**0,0715 ± 0,0031**	0,2245 ± 0,0168	**0,2898 ± 0,0189**
genbase	**0,0019 ± 0,0019**	0,0040 ± 0,0018	**0,9871 ± 0,0095**	0,9337 ± 0,0211
medical	**0,0135 ± 0,0016**	0,0162 ± 0,0016	0,7503 ± 0,0333	**0,7857 ± 0,0329**
yeast	0,2779 ± 0,0168	0,2780 ± 0,0111	0,5723 ± 0,0282	**0,5811 ± 0,0166**

Tabla 13. Resultados LP-J48 aplicando únicamente la regla de mayor confianza.

MLkNN	Hamming Loss		Avg Precision	
Dataset	Base	Propuesto	Base	Propuesto
emotions	0,1951 ± 0,0243	**0,1904 ± 0,0204**	0,7965 ± 0,0406	**0,7978 ± 0,0322**
CAL500	**0,1388 ± 0,0050**	0,1390 ± 0,0048	**0,4943 ± 0,0168**	0,4914 ± 0,0137
Corel5k	0,0094 ± 0,0001	0,0094 ± 0,0001	0,2458 ± 0,0075	0,2458 ± 0,0072
enron	**0,0524 ± 0,0020**	0,0543 ± 0,0024	**0,6310 ± 0,0148**	0,6084 ± 0,0194
genbase	**0,0050 ± 0,0027**	0,0066 ± 0,0021	**0,9875 ± 0,0103**	0,9350 ± 0,0278
medical	**0,0151 ± 0,0020**	0,0189 ± 0,0017	**0,8134 ± 0,0264**	0,7653 ± 0,0316
yeast	0,1933 ± 0,0123	**0,1931 ± 0,0091**	**0,7658 ± 0,0208**	0,7563 ± 0,0144

Tabla 14. Resulados MLkNN aplicando únicamente la regla de mayor confianza.

IBLR	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	0,1883	± 0,0239	**0,1872**	**± 0,0179**	**0,8126**	**± 0,0349**	0,8066	± 0,0314
CAL500	**0,2307**	**± 0,0054**	0,2315	± 0,0063	**0,2713**	**± 0,0118**	0,2624	± 0,0096
Corel5k	**0,0225**	**± 0,0009**	0,0229	± 0,0010	**0,1444**	**± 0,0067**	0,1386	± 0,0061
enron	**0,0557**	**± 0,0023**	0,0573	± 0,0026	**0,6186**	**± 0,0228**	0,5929	± 0,0185
genbase	**0,0027**	**± 0,0019**	0,0040	± 0,0013	**0,9872**	**± 0,0077**	0,9388	± 0,0232
medical	**0,0197**	**± 0,0021**	0,0221	± 0,0022	**0,7478**	**± 0,0211**	0,7345	± 0,0314
yeast	0,1934	± 0,0118	**0,1929**	**± 0,0086**	**0,7687**	**± 0,0200**	0,7581	± 0,0167

Tabla 15. Resultados IBLR-ML aplicando únicamente la regla de mayor confianza.

BPMLL	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	0,2061	± 0,0228	**0,2000**	**± 0,0184**	**0,8094**	**± 0,0323**	0,7976	± 0,0283
CAL500	**0,2478**	**± 0,0094**	0,2506	± 0,0088	**0,5008**	**± 0,0144**	0,4809	± 0,0156
Corel5k	0,8315	± 0,2246	**0,7632**	**± 0,3603**	0,0134	± 0,0011	**0,0135**	**± 0,0011**
enron	**0,3831**	**± 0,1344**	0,5443	± 0,0788	**0,2150**	**± 0,0766**	0,1497	± 0,0237
genbase	0,5702	± 0,2260	**0,5326**	**± 0,2281**	0,1086	± 0,0193	**0,1192**	**± 0,0325**
medical	**0,4881**	**± 0,2669**	0,5256	± 0,2363	**0,1353**	**± 0,0268**	0,1273	± 0,0161
yeast	0,2258	± 0,0099	**0,2244**	**± 0,0087**	**0,7496**	**± 0,0184**	0,7435	± 0,0173

Tabla 16. Resultados BP-MLL aplicando únicamente la regla de mayor confianza.

Examinando estos nuevos resultados puede verse que, si bien las medidas para `CAL500` y `Corel5k` han empeorado, en casi todos los casos las correspondientes a los datasets `emotions` y `yeast` han mejorado bastante. Son los datasets que tienen un menor número de etiquetas, como cabría esperar, lo que reafirma la presunción de que el número de reglas a aplicar en cada caso debería establecerse según las características del dataset con el que se vaya a trabajar.

En general, los dos datasets sobre los que peores resultados se obtienen, con independencia del número de reglas usadas, son `genbase` y `medical`. Ambos se caracterizan por tener una cardinalidad muy baja, de poco más de 1, por lo que no es posible reducirla en más de una unidad y esto afecta negativamente a los resultados casi en todos los casos, salvo excepciones como la del algoritmo BPMLL con el dataset `genbase`. La conclusión que puede extraerse de este hecho es que el método propuesto debería aplicarse únicamente en aquellos casos en que la cardinalidad alcanza un cierto valor mínimo que es necesario estudiar en conjunto con el resto de características en el espacio de etiquetas: número total de etiquetas y densidad.

En las cinco tablas de la página siguiente se han combinado los resultados que, para cada algoritmo, se obtienen aplicando todas las reglas a `CAL500` y `Corel5k` y únicamente la regla de mayor confianza para el resto, eliminando `genbase` y `medical` por la razón que acaba de explicarse: la muy baja cardinalidad que no les hace apropiados para este tipo de preprocesamiento.

BR-J48	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	**0,2474**	**± 0,0248**	0,2585	± 0,0243	0,7014	± 0,0316	**0,7015**	**± 0,0289**
CAL500	0,1615	± 0,0049	**0,1608**	**± 0,0047**	0,3513	± 0,0141	**0,3536**	**± 0,0143**
Corel5k	0,0098	± 0,0001	**0,0097**	**± 0,0001**	**0,2494**	**± 0,0093**	0,2352	± 0,0077
enron	**0,0508**	**± 0,0022**	0,0533	± 0,0028	**0,5929**	**± 0,0213**	0,5853	± 0,0221
yeast	**0,2454**	**± 0,0091**	0,2468	± 0,0097	0,6216	± 0,0151	**0,6403**	**± 0,0157**

Tabla 17. Resultados combinados para BR-J48.

LP-J48	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	0,2777	± 0,0258	**0,2476**	**± 0,0291**	0,6608	± 0,0388	**0,7017**	**± 0,0313**
CAL500	0,1996	± 0,0049	**0,1991**	**± 0,0054**	0,1161	± 0,0052	**0,1376**	**± 0,0043**
Corel5k	0,0168	± 0,0002	**0,0161**	**± 0,0002**	0,0212	± 0,0031	**0,0476**	**± 0,0063**
enron	0,0716	± 0,0028	**0,0715**	**± 0,0031**	0,2245	± 0,0168	**0,2898**	**± 0,0189**
yeast	0,2779	± 0,0168	0,2780	± 0,0111	0,5723	± 0,0282	**0,5811**	**± 0,0166**

Tabla 18. Resultados combinados para LP-J48.

MLkNN	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	0,1951	± 0,0243	**0,1904**	**± 0,0204**	0,7965	± 0,0406	**0,7978**	**± 0,0322**
CAL500	0,1388	± 0,0050	0,1388	± 0,0047	**0,4943**	**± 0,0168**	0,4912	± 0,0147
Corel5k	0,0094	± 0,0001	0,0094	± 0,0001	**0,2458**	**± 0,0075**	0,2304	± 0,0074
enron	**0,0524**	**± 0,0020**	0,0543	± 0,0024	**0,6310**	**± 0,0148**	0,6084	± 0,0194
yeast	0,1933	± 0,0123	**0,1931**	**± 0,0091**	**0,7658**	**± 0,0208**	0,7563	± 0,0144

Tabla 19. Resultados combinados para MLkNN.

IBLR	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	0,1883	± 0,0239	**0,1872**	**± 0,0179**	**0,8126**	**± 0,0349**	0,8066	± 0,0314
CAL500	0,2307	± 0,0054	**0,2300**	**± 0,0049**	**0,2713**	**± 0,0118**	0,2613	± 0,0097
Corel5k	0,0225	± 0,0009	**0,0219**	**± 0,0010**	0,1444	± 0,0067	**0,1469**	**± 0,0081**
enron	**0,0557**	**± 0,0023**	0,0573	± 0,0026	**0,6186**	**± 0,0228**	0,5929	± 0,0185
yeast	0,1934	± 0,0118	**0,1929**	**± 0,0086**	**0,7687**	**± 0,0200**	0,7581	± 0,0167

Tabla 20. Resultados combinados para IBLR-ML.

BPMLL	Hamming Loss				Avg Precision			
Dataset	Base		Propuesto		Base		Propuesto	
emotions	0,2061	± 0,0228	**0,2000**	**± 0,0184**	**0,8094**	**± 0,0323**	0,7976	± 0,0283
CAL500	**0,2478**	**± 0,0094**	0,2510	± 0,0087	**0,5008**	**± 0,0144**	0,4778	± 0,0142
Corel5k	0,8315	± 0,2246	**0,7736**	**± 0,3587**	**0,0134**	**± 0,0011**	0,0126	± 0,0010
enron	**0,3831**	**± 0,1344**	0,5443	± 0,0788	**0,2150**	**± 0,0766**	0,1497	± 0,0237
yeast	0,2258	± 0,0099	**0,2244**	**± 0,0087**	**0,7496**	**± 0,0184**	0,7435	± 0,0173

Tabla 21. Resultados combinados para BP-MLL.

Concentrándonos en las tablas 17 y 18, que corresponden a los métodos de transformación BR y LP respectivamente, en HL (*HammingLoss*) la propuesta obtiene el mejor valor en 6 ocasiones por 3 en que no lo hace, mientras que en AP (*AveragePrecision*) el mejor valor se obtiene en 8 casos frente a 2 del algoritmo base sin preprocesamiento. Esto nos lleva a concluir que la técnica propuesta es capaz de mejorar los resultados obtenidos en una importante parte de los casos, si bien para ello será necesario definir una heurística que establezca el número de reglas apropiado para cada dataset. En la experimentación realizada únicamente se ha probado con todas las etiquetas o solamente la que tiene mayor confianza, pero una heurística apropiada podría sugerir mejores valores que llevasen incluso a mejorar los resultados de las tablas 17 y 18, en especial para aquellos datasets en los que no se ha conseguido batir al algoritmo base si preprocesamiento.

En cuanto a los resultados obtenidos por el método propuesto cuando el algoritmo base está adaptado para trabajar con datasets multi-etiqueta, en las tablas 19, 20 y 21 puede comprobarse que para la medida HL en 9 de los casos se obtienen las mejores medidas, por 4 del algoritmo base sin preprocesamiento. Al considerar la medida AP, por el contrario, solamente se consigue el mejor resultado en 2 casos.

En la medida HL globalmente el método propuesto alcanza 15 mejores valores por 7 del algoritmo base, pero estos buenos resultados no se reflejan en la medida AP. La justificación hay que buscarla en cómo se calculan estas medidas y la forma en que lleva a cabo el postprocesamiento, agregando las etiquetas inferidas de las reglas.

Tal y como queda reflejado en la ecuación 7 (véase el apartado 2.1), la medida HL se calcula sobre la bipartición generada por el clasificador, es decir, solamente analiza si el conjunto de etiquetas predicho coincide con el real, penalizando tanto los falsos positivos como los falsos negativos. En este aspecto el preprocesamiento realizado en el método propuesto tiene, en general, un buen comportamiento y agrega a esa bipartición las etiquetas correctas a juzgar por la mejora en HL que se obtiene en la mayoría de los casos.

En la ecuación 10, que corresponde al cálculo de la medida AP, puede verse que su cálculo está basado no en la bipartición sino en el ranking de etiquetas entregado por el clasificador, con un valor real asociado a cada etiqueta que permite establecer un orden. En la fase de postprocesamiento del método propuesto, descrita en el punto previo, se agregan a la bipartición generada por el clasificador las etiquetas inferidas de las reglas, por una parte, y por otra se asigna a cada una de las etiquetas añadidas un valor real que no es otro que la confianza de la propia regla evaluada. Dicho valor puede, según los casos, ser inferior al umbral de corte establecido para dividir el ranking en dos subconjuntos de etiquetas, tomando las que están en la parte alta como predicción. Podría decirse que el postprocesamiento predice acertadamente la presencia de las etiquetas inferidas, tal y como se refleja en la medida HL, pero la asignación de un valor no ajustado a los presentes en el ranking penaliza la medida AP. Sería necesario, en consecuencia, estudiar un método de ajuste que permita obtener un valor de ranking adecuado a partir de la confianza de la regla inferida en cada caso, algo que seguramente contribuiría a que los buenos resultados en HL quedasen también reflejados en AP.

Un aspecto que no queda reflejado en las tablas de resultados previas es la mejora experimentada en el tiempo de ejecución de los algoritmos al reducirse el número de etiquetas respecto al dataset original. No es necesario realizar una medición del tiempo cronológico que, por otra parte, sería siempre un factor dependiente de aspectos como la velocidad de la CPU. Es suficiente con estudiar la complejidad computacional de los algoritmos y analizar cómo influye en la misma el número de etiquetas del dataset y la cardinalidad.

Tomando como ejemplo el algoritmo BR-J48 usado durante la experimentación, la reducción del número de etiquetas por muestra afectaría de la siguiente forma: dado que la transformación BR implica el entrenamiento de un clasificador independiente para cada etiqueta, al disminuir el número de etiquetas del dataset se obtiene una mejora lineal en el tiempo de ejecución. En los casos en que se aplican todas las reglas esto se traduce en una importante reducción del tiempo empleado en el procesamiento de los datos.

En algoritmos adaptados para el problema multi-etiqueta, por ejemplo los basados en árboles o redes neuronales, la reducción de la cardinalidad contribuirá a que el modelo generado sea más pequeño, menos complejo, lo cual suele traducirse en un menor tiempo de entrenamiento y una mayor interpretabilidad de su estructura.

5 Conclusiones y trabajo futuro

El objetivo de este trabajo era estudiar tareas de preprocesamiento enfocadas a mejorar el funcionamiento de los clasificadores multi-etiqueta. Partiendo de una revisión sobre este problema, y centrándonos en las técnicas de transformación de datos, se ha realizado un estudio con el fin de analizar el comportamiento de varios de los métodos descritos sobre algunos algoritmos de clasificación, incluyendo pruebas experimentales (Rivera, Charte y otros 2011) tal y como se explica en el apartado 2.5. También se ha propuesto un método original basado en un mecanismo de preprocesamiento y postprocesamiento usando reglas de asociación que, operando en el espacio de etiquetas, permite reducir la cardinalidad de los datasets multi-etiqueta. Las fases esenciales de dicho método son las siguientes:

- Preprocesamiento, dividido en los pasos:

    1. Aplicación de un algoritmo de minería de reglas de asociación, concretamente FP-Growth, sobre las etiquetas del dataset a procesar.

    2. Selección de un subconjunto con las mejores reglas obtenidas a partir de un parámetro de entrada que establece el número a tomar.

    3. Eliminación del dataset original de aquellas etiquetas que aparecen en el consecuente de las reglas seleccionadas, obteniendo un dataset con cardinalidad reducida.

- Entrenamiento del clasificador base con el dataset reducido.

- Postprocesamiento, compuesto de estos pasos:

    1. Obtención para cada muestra de test de la predicción hecha por el clasificador.

    2. Adición al resultado de las etiquetas que pueden inferirse a partir de las reglas seleccionadas durante el preprocesamiento.

Este método se ha implementado e integrado con el software MULAN y ha sido probado con siete datasets de diferentes características y con cinco algoritmos diferentes. La experimentación y resultados obtenidos, descritos en el capítulo previo, nos llevan a concluir que se trata de un enfoque especialmente útil en combinación con métodos de transformación de datos, precisando un análisis más profundo su aplicación con algoritmos adaptados al problema de la clasificación multi-etiqueta.

Tanto la definición del método propuesto como la experimentación realizada representan un primer acercamiento a la reducción de la dimensionalidad en el espacio de etiquetas, una alternativa poco estudiada en la bibliografía especializada. Los resultados obtenidos nos animan a continuar trabajando en esta vía y, en particular, en los siguientes aspectos:

- Es preciso analizar la influencia que las características del dataset: número total de etiquetas, cardinalidad y densidad, tienen en el número de reglas que es óptimo aplicar tomando también en consideración la confianza de éstas. El objetivo sería automatizar la selección del número adecuado de reglas a cada caso concreto, algo que con toda probabilidad mejoraría los resultados del clasificador.

- Es necesario evaluar otras medidas potencialmente útiles para el fin que se persigue, aparte de las ya citadas: número de etiquetas, cardinalidad y densidad. Se echa en falta, por ejemplo, una medida adicional que permita conocer la varianza respecto a la cardinalidad, un factor que puede influir considerablemente en los resultados y cambiar la manera en que se afronta el problema. Para una cardinalidad alta con poca varianza, por ejemplo, sería fiable aplicar un mayor número de reglas que para la misma cardinalidad pero con mucha varianza.

- A fin de que el método no se vea penalizado en las medidas de evaluación basadas en ranking, como es el caso de *AveragePrecision*, hay que establecer una vía para fijar el valor real asociado a cada etiqueta que esté en consonancia con el ranking generado por el clasificador base, en lugar de usar sin más la confianza de la regla de la que se infiere la etiqueta.

- Dado el buen comportamiento del método propuesto cuando se aplica sobre algoritmos basados en técnicas de transformación, es necesario extender la experimentación a otros algoritmos de este tipo, como pueden ser RAkEL (Tsoumakas y Vlahavas 2007) o RPC (Fürnkranz, y otros 2008) descritos en el apartado 2.3.5. Algoritmos que, como RAkEL, se apoyan en una transformación de tipo LP pueden beneficiarse del método propuesto ya que éste arroja los mejores resultados precisamente en esa transformación como puede apreciarse en la tabla 18.

El objetivo es perfeccionar en el futuro el método propuesto, principalmente explorando las vías que acaban de enumerarse, a fin de que pueda representar una alternativa real a otras técnicas de clasificación multi-etiqueta.

Bibliografía

Agrawal, R, y R Srikant. «Fast Algorithms for Mining Association Rules.» *20th Very Large Data Bases Conference*. 1994. 487-499.

Boutell, Matthew,R., Jiebo Luo, Xipeng Shen, y Christopher,M. Brown. «Learning multi-label scene classification.» *Pattern Recognition, 37*, 2004: 1757-1771.

Brin, Sergey, Rajeev Motwani, Jeffrey,D. Ullman, y Shalom Tsur. «Dynamic itemset counting and implication rules for market basket data.» *Sigmod Record, 26*, 1997: 255-264.

Brodley, Carla,E., y Mark,A. Friedl. «Identifying and Eliminating Mislabeled Training Instances.» *National Conference on Artificial Intelligence*. 1996. 799-805.

Broomhead, D, y D Lowe. «Multivariable functional interpolation and adaptive.» *Complex Systems*, 1988: 321-355.

Buchtala, O, M Klimek, y B Sick. «Evolutionary optimization of radial basis function classifiers for data mining applications.» *IEEE Transactions on System, Man, and Cybernetics, B, 35(5)*, 2005: 928-97.

Chan, Allen, y Alex,Alves Freitas. «A new ant colony algorithm for multi-label classification with applications in bioinfomatics.» *Genetic and Evolutionary Computation Conference*. 2006. 27-34.

Cheng, Weiwei, y Eyke Hüllermeier. «Combining Instance-Based Learning and Logistic Regression for Multilabel Classification.» *Machine Learning, 76*, 2009: 211-225.

Clare, A.J., y R.D King. «Knowledge discovery in multi-label phenotype data.» En *LNCS (LNAI), vol. 2168*, de A Sieves, & L De Raedt, 42. Springer, Heidelberg, 2001.

Clare, Amanda, y Ross,D. King. «Knowledge Discovery in Multi-label Phenotype Data.» *Principles of Data Mining and Knowledge Discovery*. 2001. 42-53.

Comité, Francesco,De, Rémi Gilleron, y Marc Tommasi. «Learning Multi-label Alternating Decision Trees from Texts and Data.» *Machine Learning and Data Mining in Pattern Recognition*. 2003. 35-49.

Crammer, Koby, y Yoram Singer. «A Family of Additive Online Algorithms for Category Ranking.» *Journal of Machine Learning Research, 3*, 2003: 1025-1058.

de Carvalho, A, y A Freitas. «A Tutorial on Multi-label Classification Techniques.» En *Foundations of Computational Intelligence Volume 5*, de A Abraham, A Hassanien, & V Snášel, 177-195. Springer Berlin / Heidelberg, 2009.

Dorigo, Marco, Gianni,Di Caro, y Luca,Maria Gambardella. «Ant Algorithms for Discrete Optimization.» *Artificial Life*, 1999: 137-172.

Elisseeff, A., y J. Weston. «A kernel method for multi-labelled classification.» *Neural Information Processing Systems.* 2001. 681-687.

Elisseeff, Andre, y Jason Weston. «Kernel methods for Multi-labelled classification and Categorical regression problems.» *Advances in Neural Information Processing Systems 14.* MIT Press, 2001. 681-687.

Esuli, Andrea, y Fabrizio Sebastiani. «Active Learning Strategies for Multi-Label Text Classification.» En *Lecture Notes in Computer Science, 2009, Volume 5478*, de Mohand Boughanem, Catherine Berrut, Josiane Mothe, & Chantal Soule-Dupuy, 102-113. Springer Berlin / Heidelberg, 2009.

Fan, Rong-En, y Chih-Jen Lin. *A Study on Threshold Selection for Multi-label Classication.* 2007.

Freund, Yoav, y Llew Mason. «The Alternating Decision Tree Learning Algorithm.» *International Conference on Machine Learning.* 1999. 124-133.

Fürnkranz, Johannes, Eyke Hüllermeier, Eneldo,Loza Mencía, y Klaus Brinker. «Multilabel classification via calibrated label ranking.» *Machine Learning, 73*, 2008: 133-153.

Ghamrawi, Nadia, y Andrew McCallum. «Collective multi-label classification.» *ACM Fourteenth Conference on Information and Knowledge Management (CIKM).* Bremen, 2005. 195-200.

Godbole, S., y S. Sarawagi. «Discriminative Methods for Multi-labeled Classification.» *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* 2004. 22-30.

Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning.* MA: Addison-Wesley, 1989.

Grodzicki, Rafal, Jacek Mandziuk, y Lipo Wang. «Improved Multilabel Classification with Neural Networks.» *Parallel Problem Solving from Nature, LNCS 5199.* Springer-Verlag Berlin Heidelberg, 2008. 409-416.

Hall, M, E Frank, G Holmes, B Pfahringer, P Reutemann, y I.H Witten. «The weka data mining software. An update.» *SIGKDD Explorations*, 2009.

Han, J, J Pei, y Y Yin. «Mining frequent patterns without candidate generation.» *SIGMOD Rec. Vol. 29 Issue 2 May 2000*, 2000: 1-12.

Harpham, C, C.W. Dawson, y M.R. Brown. «A review of genetic algorithms applied to training radial basis function networks.» *Neural Computing and Applications, 13*, 2004: 193-201.

Hipp, Jochen, Ulrich Güntzer, y Gholamreza Nakhaeizadeh. «Algorithms for association rule mining — a general survey and comparison.» *SIGKDD Explor. Newsl. Volume 2 Issue 1, June, 2000*, 2000: 58-64.

Hua, Jianping, Waibhav Tembe, y Edward,R. Dougherty. «Performance of feature-selection methods in the classification of high-dimension data.» *Pattern Recognition, 42*, 2009: 409-424.

Hüllermeier, Eyke, Johannes Fürnkranz, Weiwei Cheng, y Klaus Brinker. «Label ranking by learning pairwise preferences.» *Artificial Intelligence, 172*, 2008: 1897-1916.

Jain, Anil,K., y Douglas,E. Zongker. «Feature Selection: Evaluation, Application, and Small Sample Performance.» *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, 1997: 153-158.

Karalič, A, y V Pirnat. «Significance level based multiple tree classification.» *Informatica 15*, 1991: 12.

Karalič, A., Pirnat, V. «Significance level based multiple tree classification.» *Informatica 15*, 1991: 12.

Knorr, Edwin,M., y Raymond,T. Ng. «A unified approach for mining outliers.» *Conference of the Centre for Advanced Studies on Collaborative Research.* IBM Press, 1997. 11-.

Liu, Bing, Wynne Hsu, y Yiming Ma. «Integrating Classification and Association Rule Mining.» *Knowledge Discovery and Data Mining.* 1998. 80-86.

Mandani, E., y S. Assilian. «An experiment in linguistic synthesis with a fuzzy logic controller.» *International Journal of Man-Machine Studies, 7(1)*, 1975: 1-13.

Michalski, R.,S. *On the Selection of Representative Samples from Large Relational Tables for Inductive Inference.* 1975.

Parpinelli, R.S., H.S. Lopes, y A.A. Freitas. «Data mining with an ant colony optimization algorithm.» *IEEE Transactions on Evolutionary Computation*, 2002: 321-332.

Pérez-Godoy, M.D., A.J Rivera, M.J. del Jesús, y F.J. Berlanga. «CO2RBFN: An evolutionary cooperative-competitive RBFN design algorithm for classification problems.» *Soft Computing, 14(9)*, 2010: 953-971.

Quinlan, J.,R. *C4.5: Programming For Machine Learning.* 1992.

Read, Jesse, Bernhard Pfahringer, Geoffrey Holmes, y Eibe Frank. «Classifier Chains for Multi-label Classification.» *Principles of Data Mining and Knowledge Discovery.* 2009. 254-269.

Rivera, A.J., F. Charte, M.D. Pérez-Godoy, y M.J. del Jesús. «Multi-label Testing for CO2RBFN: A First Approach to the Problem Transformation Methodology for Multi-label Classification.» *IWANN 2011, Part I, LNCS 6691.* Málaga: Springer-Verlag Berlin Heidelberg, 2011. 41-48.

Schapire, R., y Y. Singer. «BoosTexter: a boosting-based system for text categorization.» *Machine Learning, 39*, 2000: 135-168.

Sebastiani, F. «Machine learning in automated text categorization.» *ACM Computing Surveys 34 (1)*, 2002: 1-47.

Sun, Liang, Shuiwang Ji, y Jieping Ye. «Hypergraph spectral learning for multi-label classification.» *Knowledge Discovery and Data Mining.* 2008. 668-676.

Tsoumakas, G, y I Katakis. «Multi-Label Classification: An Overview.» *International Journal of Data Warehousing and Mining 3*, 2007: 1-13.

Tsoumakas, G., E Spyromitros-Xioufis, J Vilcek, y I Vlahavas. «Mulan: A Java Library for Multi-Label Learning.» *Journal of Machine Learning Research*, 2011.

Tsoumakas, G., I. Katakis, y I. Vlahavas. «Mining Multi-label Data.» En *Data Mining and Knowledge Discovery Handbook*, de O Maimon, & L Rokach. Springer, 2010.

Tsoumakas, Grigorios, y Ioannis,P. Vlahavas. «Random k -Labelsets: An Ensemble Method for Multilabel Classification.» *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.* 2007. 406-417.

Veloso, Adriano, Wagner,Meira Jr., Marcos,André Gonçalves, y Mohammed,Javeed Zaki. «Multi-label Lazy Associative Classification.» *Principles of Data Mining and Knowledge Discovery.* 2007. 605-612.

Whitehead, B, y T. Choate. «Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction.» *IEEE Transactions on Neural Networks, 7(4)*, 1996: 869-880.

Widrow, B., y M.A. Lehr. «30 years of adaptive neural networks: perceptron, madaline and backpropagation.» *Proceedings of the IEEE, 78(9)*, 1990: 1415-1442.

Yang, Bishan, Jian-tao Sun, Tengjiao Wang, y Zheng Chen. «Effective multi-label active learning for text classification.» *Knowledge Discovery and Data Mining.* 2009. 917-926.

Yang, Qiang, y Xindong Wu. «10 Challenging Problems in Data Mining Research.» *International Journal of Information Technology and Decision Making*, 2006: 597-604.

Yang, Yiming, y Xin Liu. «A re-examination of text categorization methods.» *JOURNAL OF INFORMATION RETRIEVAL*, 1999: 42-49.

Zhang, M., y Z. Zhou. «ML-KNN: A lazy learning approach to multi-label learning.» *Pattern Recognition 40*, 2007: 2038-2048.

Zhang, M., y Z. Zhou. «Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization.» *IEEE Transactions on Knowledge and Data Engineering, 18*, 2006: 1338-1351.

Zhang, M.-L, y Z.-H Zhou. «A k-nearest neighbor based algorithm for multi-label.» *Proceedings of the 1st IEEE International Conference on Granular.* Beijing, 2005. 718–721.

Zhang, M.-L., Zhou, Z.-H. «A k-nearest neighbor based algorithm for multi-label.» *Proceedings of the 1st IEEE International Conference on Granular.* Beijing, 2005. 718–721.

Zhang, Min-ling. «Ml-rbf : RBF Neural Networks for Multi-Label Learning.» *Neural Processing Letters, 29*, 2009: 61-74.

Zhu, B, y C.K Poon. «Efficient Approximation Algorithms for Multi-label Map Labeling.» En *LNCS, vol. 1741*, de A.K., Pandu Rangan, C. (eds.) Aggarwal, 143–152. Springer, Heidelberg, 1999.

Zhu, B., Poon, C.K. «Efficient Approximation Algorithms for Multi-label Map Labeling.» En *LNCS, vol. 1741*, de A.K., Pandu Rangan, C. (eds.) Aggarwal, 143–152. Springer, Heidelberg, 1999.

Zhu, S., Ji, X., Xu, W., Gong, Y. «Multi-labelled classification using maximum entropy method.» *28th annual international ACMSIGIR conference on Research and development in Information Retrieval.* 2005. 274-281.