UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN
CENTRO DE ESTUDIOS DE INFORMÁTICA
DPTO. INTELIGENCIA ARTIFICIAL

ESTUDIO DE LA CLASIFICACIÓN MULTIETIQUETA GRADUAL (*Graded MLC*)

AUTOR: GUILLERMO ABREU STINCER
TUTORES: DRA. YANET RODRÍGUEZ SARABIA

MSC. ADIS PERLA MARIÑO RIVERO

Santa Clara

2015

Declaración Jurada

El que suscribe, **Guillermo Abreu Stincer**, hago constar que el trabajo titulado Estudio de la Clasificación Multietiqueta Gradual fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Licenciatura en Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor Firma del Jefe del Laboratorio

Frase

"Se alcanza el éxito convirtiendo cada paso en una meta y cada meta en un paso."

C.C. Cortéz

DEDICATORIA

Dedico este trabajo con todo mi corazón: a mis padres, por ser la razón y fuerza que me permitió dar cada paso sin temor a caer, a mi hermano, por ser mi mejor amigo, mi consejero y mi guía, y a mi novia por ser mi inspiración en cada línea y mi luz en los momentos oscuros de la batalla.

AGRADECIMIENTOS

A toda mi familia por su apoyo, confianza y preocupación continua e ilimitada.

A mi novia Eileen por existir en mi vida y regalarme la inmensa felicidad de su compañía.

A mis tutoras Yanet y Adis por toda la ayuda incondicional y toda la paciencia y el tiempo dedicado.

A mis amigos Yordan, Pabel, Pablo y Pedro por estos cinco años compartidos en la lucha por ganar esta batalla.

RESUMEN

En los últimos años la Clasificación Multietiqueta Gradual (GMLC) ha ganado especial atención dentro del Aprendizaje Automatizado debido a la gran utilidad que ofrece para la solución de problemas actuales e importantes para el desarrollo de la vida humana. En la Clasificación Multietiqueta se busca encontrar dado un conjunto de etiquetas preestablecidas, a cuál de ellas pertenecen o no cada instancia de un conjunto dado. La Clasificación Multietiqueta Gradual generaliza esta metodología permitiendo especificar el grado de pertenencia en que se relacionan cada instancia y las etiquetas. Existen una gran cantidad de problemas que pueden ser solucionados utilizando este paradigma como la clasificación de películas por géneros o de libros por temáticas abordadas. Generalmente la Clasificación Multietiqueta Gradual está asociada a las preferencias y resulta mucho más entendible cuando es explicada a través de ranking de etiquetas. Sin embargo, al ser un tema tan reciente y poco documentado, su bibliografía se encuentra limitada a unos pocos artículos y todavía no muestra una amplia gama de soluciones. Por estas razones este trabajo se centra en la realización de un marco teórico que facilite el estudio de esta rama a través de la recopilación y resumen de diferentes artículos existentes en la literatura hasta la fecha. Además con la utilidad agregada se convirtió al marco de trabajo MULAN en una herramienta más potente debido a la incorporación de un paquete dedicado a la resolución de problemas de GMLC a través de la integración de cinco métodos que utilizan Comparación por pares y Ranking de Etiquetas Calibradas.

ABSTRACT

Over the last years the Graded Multilabel Classification (GMLC) has gained special attention in Machine Learning for the great utilities it offers in the solution of current and important problems for the development of the human life. The purpose of Multilabel Classification is given a group of preset labels, to find which of them are related or not to each instance of a given group. The Graded Multilabel Classification generalizes this methodology allowing to specify the membership deegres between the instances and the labels. There is a big amount of problems that can be solved using this paradigm, such as the classification of movies for genders or books for thematics. Generally the Graded Multilabel Classification is associated to preferences and it is much understandable when it is explained through of labels ranking. However, because it is such a recent topic, it´s bibliography is limited to some few articles and it doesn't still show a wide range of solutions. For these reasons this work is centered in the compilation of theories that facilitates the study of this branch of the Multilabel Classification through the summary of different existing articles in the literature so far. Beside, with the added utility MULAN becomes in a more powerful tool with the incorporation of a new package dedicated to the solving of GMLC problems containing five approaches which use Pairwise Comparisons and Calibrated Label Ranking.

ÍNDICE

Índice de figuras

Índice de cuadros

INTRODUCCIÓN

El almacenamiento de información es un proceso que desde hace varios años atrás comenzó a masificarse y crecer exponencialmente a lo largo de nuestro mundo sin muestras de freno alguno. Debido a este fenómeno, la cantidad de información almacenada duplica su volumen cada aproximadamente 20 meses y dado que en la actualidad es en extremo usual que dispositivos electrónicos almacenen nuestras decisiones en muchas esferas de la vida humana (de compras en un supermercado, en las búsquedas en internet, etc.) resulta muy útil y necesario contar con mecanismos que permitan almacenar, recuperar y procesar con eficiencia los datos, y sobre todo tener procedimientos que obtengan información relevante y conocimiento útil de estos datos en bruto[1].

Es usual encontrar bases de datos saturadas de información que en muchas ocasiones por la inmensa cantidad de datos que almacenan no han sido procesadas, acarreando consigo un aumento de la diferencia entre la rapidez con que crece el almacenamiento de dichos datos y la rapidez en que se les da utilización real, o sea, disminuye la proporción entre número de datos e información útil. Sin duda alguna dentro de todo ese cúmulo de datos existe mucho conocimiento útil que no ha sido descubierto debido a la interferencia proporcionada por la creciente cantidad de dichos datos. La Minería de Datos (*Data Mining*, DM) trata de resolver estos problemas mediante el análisis de toda esta información que ya está almacenada en bases de datos y, se puede definir como un proceso de búsqueda y descubrimiento de instancias en datos. El proceso idealmente debe ser automático y el objetivo principal es que las instancias descubiertas deben tener algún significado en el sentido de que puedan ser utilizados de alguna forma como conocimiento útil. Estas instancias, o dicho de manera más adecuada, estos conceptos o modelos nos permitirán hacer predicciones sobre nuevos datos que se almacenen. Resulta necesario que estos modelos se encuentren expresados en una forma inteligible para lograr además de hacer predicciones, llegar a una comprensión de por qué se estructuraron los datos de una manera específica y no de otra.

La disciplina de la minería de datos abarca un muy extenso rango de tareas, sin embargo, la mayoría de las técnicas de dicha disciplina pertenecen a lo que se denomina Aprendizaje Automático (*Machine Learning*, ML). Una de las principales áreas dentro del ML y que ha recibido mayor interés es la Clasificación que trata de hallar una función, denominada clasificador, que reciba como entrada los atributos de una determinada instancia y devuelva como salida una clase, de entre una serie de clases predefinidas en el problema, a la que se considera está asociada dicha instancia. Mediante la clasificación dada una extensa cantidad de datos, se puede predecir en un cierto grado

la clase a la que un determinado conjunto de atributos puede estar asociado. Sin embarago en ocasiones es difícil o imposible obtener resultados categóricos pues determinado conjunto de atributos puede estar presente en varias o todas las clases. Cuando la clasificación se realiza sobre instancias o patrones que pueden ser multietiquetados se denomina Clasificación Multietiqueta(*Multilabel Classification*, MLC). Este caso es ampliamente visto tanto en la computación como en múltiples ramas de la vida diaria como el diagnóstico médico donde un conjunto muy específico de síntomas pueden estar asociados al padecimiento de varias enfermedades.

En algunos casos un conjunto de características no están evidenciadas en su totalidad en las clases preestablecidas pero si se encuentran presentes en determinada cuantía, por lo que resulta muy complejo o imposible determinar si pertenecen o no en el sentido completo de la palabra. Este tipo de datos se denomina gradual y esta marcado por un grado, como su nombre sugiere, que explica el nivel de asociación o pertenencia de las etiquetas. Es muy utilizado en el ámbito de las preferencias aunque puede verse en muchas otras problemáticas. Existen muchas bases de datos constituidas por conjuntos de características o atributos de las que se extraen preferencias. En el campo de las preferencias es extremadamente difícil encontrar unisidad de criterio y exactitud en el momento de la elección, por tanto es generalmente deseable que los resultados obtenidos además de ser multietiqueta esten organizados en forma de ranking o muestren el grado en que cada elemento al que se asoció la instancia esta relacionado con él. En situaciones como las anteriores resulta necesario o, al menos, deseable saber en qué cuantía dichas instancias se relacionan con las clases en cuestión. Este último concepto trata acerca de la Clasificación Multietiqueta Gradual (*Graded Multilabel Classification*,GMLC), que ha ido incrementando la atención de la comunidad científica debido a su alta presencia en problemáticas de la actualidad, salidas del acelerado desarrollo de la técnología, la masificación de las redes sociales y la necesidad de automatización de la mayor cantidad posible de facetas en la vida humana. Al ser una rama de la MLC también es resuelta a través de lo métodos de transformación de problemas o con los métodos de adaptación de problemas. Los primeros son bastamente utilizados y ofrecen buenos resultados. Dentro de esta categoría se encuentra el método de comparación de pares (*Pairwise Comparison*, PC) que convierten le conjunto de datos multietiqueta en varios subconjuntos con etiqueta simple para luego aplicar una técnica clásica. En el presente caso se utiliza la técnica de ranking de etiquetas calibradas que no es más que la resolución de problemas de clasificación multietiqueta haciendo uso del ranking de etiquetas y más específicamente intenta estimar las etiquetas a ser clasificadas a través del aprendizaje de clasificadores binarios haciendo comparaciones por parejas de las etiquetas. Esto significa que para cada par de etiquetas uno o más clasificadores son entrenados para predecir cual de las dos etiquetas es más relevante. En los diferentes problemas de clasificación tales clasificadores por pares mostraron muy buenos resultados. Específicamente el ranking de etiquetas calibradas como un método individual perteneciente a la comparación de etiquetas demostró ser muy bueno ante el ajuste a clasificación multietiqueta. Esto hace que el enfoque por pares y, especialmente, el enfoque de ranking de etiquetas calibradas sea muy prometedor para el ajuste de la clasificación

multietiqueta gradual.

La clasificación multietiqueta gradual fue descrita y obtuvo una solución por primera vez con Weiwei Cheng, Krzysztof Dembczynski y Eyke Hüllermeier con su artículo "*Graded Multilabel Classification: The Ordinal Case*" en 2008 donde reducen la problemática de clasificación multietiqueta gradual a multietiqueta mediante el uso del método de IBLR-ML propuesto por ellos que combina aprendizaje basado en instancias con regresión logística.

Luego el Dr. Johannes Fürnkranz y Eneldo Loza Menzia con su artículo "*Graded multilabel classification by pairwise comparison*" [2] introducen una nueva solución utilizando comparación por pares mostrando mejores resultados que los obtenidos anteriormente por Cheng. Debido a lo antes expuesto el presente proyecto incorpora esta última solución al marco de trabajo MULAN para adicionarle el tratamiento de datos graduales.

Dada la importancia del tema será centro de estudio del actual proyecto de diploma que reunirá un compendio de información sobre el tema y propondrá un marco de trabajo para el tratamiento de problemas de este tipo.

Descripción del problema

Actualmente, el estudio y utilización de la Clasificación Multietiqueta como un área especialmente útil dentro del Aprendizaje Automático y como herramienta capaz de solucionar una vasta cantidad de problemas de la vida diaria y no tan diaria, ha recibido un aumento bastante acelerado de la atención científica. Dado que en muchas problemáticas una misma combinación de características constituyendo una instancia pueden pertenecer en cierto grado a las clases del problema en cuestión, se le ha concedido gran atención a una rama, contenida en el área expuesta anteriormente que ofrece soluciones a este tipo específico de problemas y se denomina Clasificación Multietiqueta Gradual. Sin embargo la información sobre el tema resulta muy escasa y el software dedicado al modelado y solución de problemas de tipo Clasificación Multietiqueta (MULAN) que se utiliza en la Universidad Central "Marta Abreu" de Las Villas no contiene ninguna implementación para el tratamiento de la Clasificación Multietiqueta Gradual por lo que resulta necesario el establecimiento de un Marco Teórico que ofrezca ayuda bibliográfica sobre el tema y el desarrollo de un Marco de Trabajo que permita a los usuarios trabajar con datos multietiqueta graduales.

Objetivo general.

- Establecer un marco de trabajo para la Clasificación Multietiqueta Gradual(*Graded Multilabel Classification*).

Objetivos específicos.

1. Establecer un marco teórico referencial sobre el tema.

2. Integrar 5 métodos de trabajo de clasificación multietiqueta gradual a la herramienta MULAN.

3. Verificar los resultados mediante la comparación de los resultados de los métodos independientes e integrados a MULAN obtenidos en varias corridas con varios conjuntos de datos estructurados gradualmente.

Preguntas de investigación.

1. ¿Existe algún repositorio de datos de tipo Graded MLC o el conjunto de datos se debe generar a partir de bases de datos de tipo gradual?

2. ¿Existe relación entre Graded MLC y las Técnicas de Ranking?

3. ¿Existen actualmente en la literatura métricas que permitan medir el comportamiento de la GMLC?

ESTRUCTURA DE LA TESIS

El documento se encuentra estructurado en una introducción, tres capítulos, conclusiones, recomendaciones y bibliografía. En el Capítulo 1 se encuentra descrito el planteamiento del problema, así como lo relativo a toda la base teórica necesaria para lograr una mejor comprensión posterior de las respuestas dadas al problema general expuestas en los siguientes capítulos. En el capítulo 2 se aborda todo lo relacionado con los métodos utilizados en la solución al problema y la forma en que se estructuró el trabajo; del mismo modo en el capítulo 3 se explica de forma detallada el funcionamiento del nuevo paquete y maneras de utilizarlo acoplado a la herramienta MULAN, los resultados obtenidos y la validación realizada, así como los pasos a seguir de manera general para insertar un nuevo clasificador y las ventajas que esta integración ofrece a MULAN.

Capítulo 1

CLASIFICACIÓN MULTIETIQUETA

En este capítulo se expone la problemática en cuestión y se detallan elementos teóricos necesarios para el entendimiento de la solución brindada al problema planteado incluyendo una breve explicación, notaciones y aplicabilidad de conceptos como:

- Aprendizaje multietiqueta
- Clasificación
- Clasificación multietiqueta
- Clasificación multietiqueta gradual

1.1. Aprendizaje Multietiqueta. Conceptos, notación y utilidad

El Aprendizaje Multietiqueta (*Multilabel Learning*, MLL) es un paradigma de aprendizaje supervisado que recientemente ha acaparado gran atención por su capacidad de mejorar el rendimiento y calidad de muchos temas actuales y de gran interés como la clasificación de multimedia, la predicción de la función de genes y proteínas, la dirección del mercado o la minería en las redes sociales. Todas estas aplicaciones tienen en común que generalmente son requeridas más de una salida. El MLL trabaja con más de una etiqueta por instancia, analiza las relaciones entre etiquetas y la existencia de etiquetas desbalanceadas, trata de minimizar el costo computacional de generar el modelo, corre sobre alta dimensionalidad de datos y/o espacio de salida [1].

En el aprendizaje multietiqueta dada $X = X_1 \times ... \times X_d$ un espacio de entrada d-dimensional de características numéricas o categóricas y un espacio de salida de q etiquetas, $\gamma = \{\lambda_1, \lambda_2, ..., \lambda_q\}$, una instancia multietiqueta puede ser definido como un par (x, Y) , donde $x = (x_1, ..., x_d) \in X$ y $Y \in \gamma$ es llamado conjunto de etiquetas[3]. Cuatro tareas diferentes están incluidas en MLL:

- Clasificación multi-etiqueta

- Ranking
- Clasificación jerárquica multietiqueta
- Clasificación multidimensional

En el cuadro 1.1 se puede apreciar un conjunto de datos multietiqueta con 4 instancias y 4 etiquetas. Se puede observar como la intancia 1 está asociada con las etiquetas λ_2 y λ_3, es decir, cada instancia puede pertenecer a más de una clase simultáneamente.

Instancia	Etiquetas				Etiquetas Asociadas a cada Instancia
	λ_1	λ_2	λ_3	λ_4	
1	0	1	0	1	$\{\lambda_2, \lambda_3\}$
2	1	0	0	0	$\{\lambda_1\}$
3	1	1	1	0	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	0	1	0	1	$\{\lambda_2, \lambda_4\}$

Cuadro 1.1: Ejemplo de conjunto de datos multietiqueta

1.2. Clasificación

En el mundo actual la información digital almacenada tanto en internet como en servidores locales crece de una manera aceleradamente alta, proceso que acrecenta la diferencia entre los índices de entrada de datos y la rapidez con que se procesan estos datos convirtiéndolos en información útil. La DM trata de resolver estos problemas mediante el análisis de toda esta información no procesada que ya está almacenada en bases de datos y se puede definir como un proceso de búsqueda y descubrimiento de instancias en datos. El proceso idealmente debe ser automático y el objetivo principal es que las instancias descubiertas deben tener algún significado en el sentido de que puedan ser utilizados de alguna forma como conocimiento útil y así permitan hacer predicciones sobre nuevos datos que se almacenan y guardan alguna similitud[4].

Esta disciplina (Minería de Datos), abarca un conjunto de tareas bastante amplio, sin embargo, la mayor parte de sus técnicas se encuentran ubicadas dentro de lo que se denomina Aprendizaje Automático y son específicamente utilizadas en la resolución de un gran número de tareas agrupadas en dos grandes categorías: descriptivas y predictivas.

Las tareas descriptivas son las que tratan de obtener información acerca de la estructura de los datos almacenados, abarcando las tareas de Clustering y Asociación. Por otro lado, se encuentran las tareas predictivas, que tratan de predecir un comportamiento a partir de información ya almacenada. Estas abarcan las tareas de Regresión y Clasificación.

Debido a su utilidad y efectividad la clasificación ha sido una de las tareas más ampliamente usadas en la resolución del problema antes expuesto y se caracteriza porque los ejemplos se presentan, como un conjunto de pares de elementos que representan las entradas y la salida esperada,

siendo el objetivo de la tarea aprender una función mediante aprendizaje supervisado. Esta función se denomina clasificador y almacena la correspondencia existente en los ejemplos y clases, de modo que para cada valor de entrada tenemos un único valor de salida.

1.2.1. Clasificación Binaria

El problema de clasificación binaria es la forma más simple de un problema de clasificación. Para algunos conjuntos de datos X con instancias x=$(x_1...x_n)$ con $x \in X$ tiene que decidirse si estas instancias x pertenecen a alguna clase λ o no. Por esta razón $x_1,...,x_n$ son los valores para los rasgos

$$A=\{a_1,...,a_n\} \tag{1.1}$$

del conjunto de datos X. Así, el clasificador de manera general sería: $H : X \to M, x \mapsto 1$ si x pertenece λ o 0 en otro caso con $M = 0,1$. Cuando se asigna 1 la instancia pertence a la clase λ y cuando es 0 no pertenece[2].

1.2.2. Clasificación Ordenada

Cuando en el problema descrito anteriormente se extiende con un grado de pertenencia a la clase entonces se tiene un problema de clasificación ordinal. El espacio del valor objetivo M del clasificador H es cambiado de un rango binario $M = 0,1$ a un espacio finito ordenado y discreto:

$$M=\{m,...,m_n\} \tag{1.2}$$

estructurado internamente de la siguiente manera:

$$m_1 \prec m_2 \prec ... \prec m_n. \tag{1.3}$$

Los valores en dicho espacio pueden ser por ejemplo un subespacio de números naturales M=$\{0,1,...,5\}$, donde '0' tiene exactamente el mismo significado que en el caso binario y '5' corresponde a '1'. Los valores interiores representan más o menos una clasificación parcial de λ correspondientes a las frases 'probablemente pertence' o 'casi no'. Un ejemplo real sería la puntuación dada por los clientes a los productos en un sitio de venta virtual otorgando hasta 5 estrellas de acuerdo a su nivel de satisfacción. Para esto se trata de construir problemas de clasificación binaria de una manera tal que por cada par de grados $(m_i, m_j) \in M$ con $i < j$ tiene que ser predicha a cual de las dos la instancia pertenece. A través de votaciones de los clasificadores se predice cual es el grado con mayor número de votos. Por ejemplo, si 5 clasificadores predicen que la instancia pertenece al grado m_1 y dos clasificadores predicen el grado m_2, resulta evidente que la instancia pertenece a m_1[2].

1.3. Clasificación multietiqueta y tareas relacionadas

En la actualidad toda la información multimedia, desde los textos, a los videos, pasando por la música o las películas, se etiqueta para facilitar a los usuarios la búsqueda de los contenidos que desean. Las etiquetas asignadas pertenecen a conjuntos previamente definidos y su utilidad es dar una descripción rápida del contenido del elemento. Existen numerosos problemas de creciente actualidad en los que el requisito de que cada instancia pertenezca a una y sólo una clase es difícil de cumplir. Por ejemplo en la clasificación de multimedia, una imagen puede contener elementos tan diferentes que sería deseable relacionarla a ambos como se muestra en la figura 1.1.

Figura 1.1: Imagen de la ciudad de Londres que podría clasificarse como ciudad o río

Para poder tratar adecuadamente este tipo de problemas surge el paradigma de aprendizaje denominado Aprendizaje Multietiqueta, que es la encargada, dentro del campo del aprendizaje, de producir modelos capaces de asignar automáticamente estos subconjuntos de etiquetas a nuevos objetos. Desde un punto de vista formal, su principal diferencia con respecto a la clasificación tradicional es que los objetos pueden pertenecer simultáneamente a más de una clase, lo que impide que se puedan aplicar directamente los clasificadores multiclase conocidos. Por ese motivo diversas técnicas de aprendizaje han sido adaptadas para tratar este nuevo problema, como los árboles de decisión, los algoritmos basados en instancias o las máquinas de vectores soporte. La mayoría de los problemas de clasificación asocian a cada instancia una única etiqueta, l_i, de un conjunto disjunto

de etiquetas posibles, L. Si dicho conjunto de etiquetas tiene dos posibles valores, que representan la pertenencia o no pertenencia a una clase, hablamos de clasificación binaria ($|L| = 2$), mientras que en el resto de los casos hablamos de clasificación multiclase ($|L| > 2$). No obstante, este no es el único escenario posible, ya que hay numerosos problemas de creciente interés en los que una instancia puede tener asociado un conjunto de etiquetas de clase[5]. En estos casos se habla de multietiqueta (Figura 1.2).

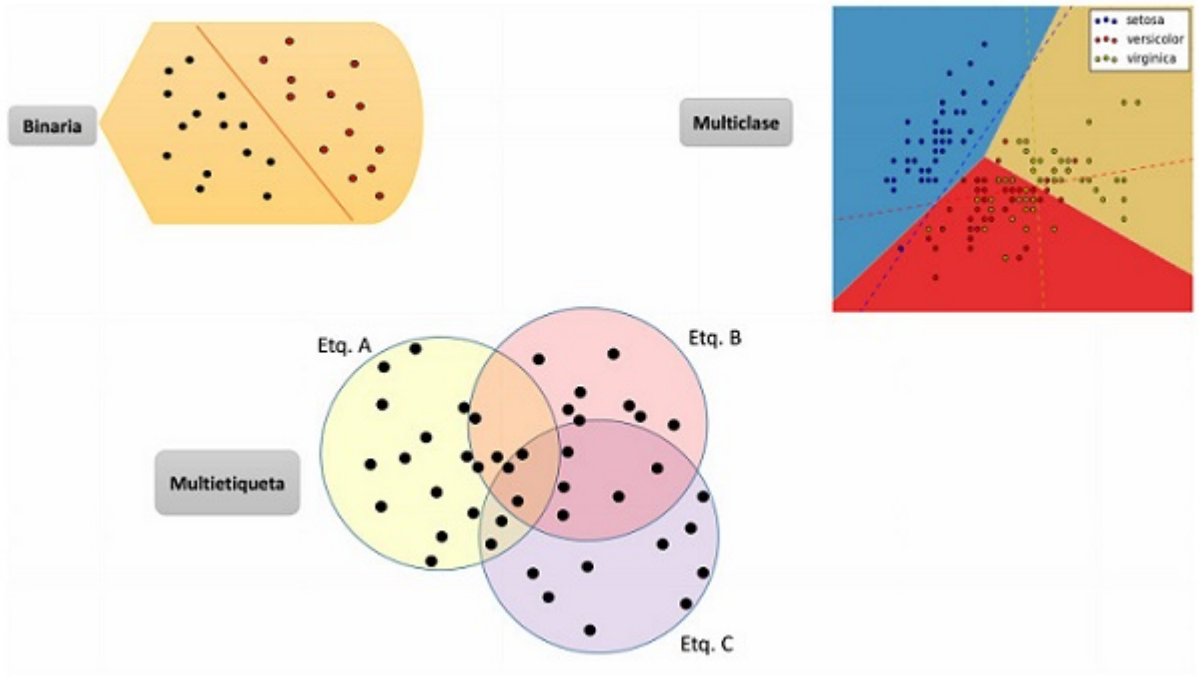

Figura 1.2: Diferencia conceptual entre los distintos tipos de clasificación.

En las tablas 1.2 y 1.3 se puede ver un conjunto de una sola etiqueta y multietiqueta respectivamente. De esta manera se observa que, mientras que el primero solo permite que una instancia este asociado a una sola clase, en el segundo cada instancia puede estar asociado a varias categorías simultáneamente.

Ejemplo	Atributos	Etiqueta
1	X_1	Religión
2	X_2	Historia
3	X_3	Filosofía
4	X_4	Ciencia

Cuadro 1.2: Conjunto de una sola etiqueta

Ejemplo	Atributo	Etiquetas			
		Religión	Historia	Filosofía	Ciencia
1	X_1	T	F	F	T
2	X_2	T	T	F	T
3	X_3	F	F	T	T
4	X_4	T	F	F	F

Cuadro 1.3: Conjunto multietiqueta

La clasificación multietiqueta se caracteriza porque los conjuntos de clases no son excluyentes entre sí, y por tanto puede haber instancias a los que se les asocie más de una etiqueta a la vez. En MLC las etiquetas asociadas a cada instancia son binarias, que indican si dicha instancia está asociada o no a la etiqueta de manera que dado un conjunto de atributos específico se obtendrá el conjunto de etiquetas con su correspondiente valor 0 o 1(no pertenencia o pertenencia respectivamente).

Esta rama de la clasificación ofrece soluciones a problemáticas muy abundantes y de gran interés en aspectos de la vida humana que enmarcan el desarrollo de la sociedad como la salud, la economía, las investigaciones y las relaciones sociales mostradas de manera resumida en la tabla 1.4.

Problema	Subproblemas
Cat. de textos	Noticias[6][7]
	Doc. jurídicos[8, 9]
	E-*mail[10][11]*
	Clasif. de sent.[12]
Bioinformática	Clasif. de proteínas[13]
	Clasif. de genes[14]
Imágenes y video	Clasif. de imágenes[15]
	Clasif. de videos[16]
	Anotado de imágenes[17]
	Anotado de video[18]
	Rec. de patrones[19]
Medicina	Diagnosis[20]
	Medicamentos[21]
	Textos médicos[22]
	Imágenes médicas[23]
Cat. de sonidos	Géneros[24]
	Instrumentos[25]
	Tagging[26]
	Emociones[27]
Otras	E-*learning[28]*
	Ingeniería[29]
	Economía[30]

Cuadro 1.4: Utilidades de la clasificación mutietiqueta

1.3.1. NOTACIÓN

A continuación se define una notación para el problema de la clasificación multietiqueta. Esta notación trata de sintetizar la que se viene utilizando en la mayor parte de los trabajos publicados. Para una formalización de la tarea de clasificación multietiqueta se considerará $L = \{\lambda_i : i = 1...n\}$ al conjunto de etiquetas presente en un problema determinado siendo n el número máximo de etiquetas que puede tener asociada una instancia.

Por otro lado, se denominará al conjunto $D = \{D_j : j = 1...m\}$ conjunto de datos multietiqueta. Cada elemento del conjunto de datos D, también denominado instancia multietiqueta, estará compuesta por un par de elementos $D_j = (X_j, Y_j)$ siendo X_j el vector de atributos condicionales, siendo $|X|$ su tamaño e $Y_j \subseteq L$ el subconjunto de etiquetas o atributos de decisión asociados a la instancia j, por tanto $|Y_j|$ será el número de etiquetas asociado a la instancia j.

MLC está asociado con el aprendizaje de un modelo predictivo que proporcione como salida una bipartición del conjunto de etiquetas en relevantes e irrelevantes respecto a una instancia de consulta.

Los problemas tradicionales de clasificación binaria y multiclase pueden ser considerados como

casos específicos de MLC, considerada esta última por su generalidad una tarea más compleja de resolver que las otras dos [31]. De esta manera se define la tarea de MLC como encontrar una función $h : X \rightarrow 2^L$ tal que h maximice o minimice un parámetro determinado[32].

1.3.2. MÉTODOS DESARROLLADOS PARA LA CLASIFICACIÓN MULTIETIQUETA

Los métodos basados en aprendizaje supervisado existentes se agrupan en dos categorías: métodos de transformación de problemas y métodos de adaptación de algoritmos.

Para resolver los problemas de clasificación multietiqueta se tienen básicamente dos métodos basados en aprendizaje supervisado, por un lado se ha tratado de modificar los datos de entrada multietiqueta para convertirlos en datos de una sola etiqueta, a los que aplicar una técnica de clasificación clásica, y por otro lado, se han tratado de crear nuevas técnicas que puedan trabajar directamente con los datos multietiqueta.

La primera vía contiene un grupo de métodos denominados métodos de transformación de problemas y solucionan la problemática mediante la transformación del problema multietiqueta para obtener uno o varios conjuntos de datos de una sola etiqueta. Esta perspectiva puede tener algunos inconvenientes porque en algunos casos puede implicar una pérdida de información, al despreciar algunas etiquetas, o sencillamente por el hecho de que se pierdan las correlaciones entre etiquetas al generarse nuevos conjuntos de datos de una sola etiqueta. También puede llegar a tener alta complejidad en ciertas situaciones donde resulta alto o complicado el coste de su transformación. Sin embargo, este enfoque permite la utilización de técnicas de clasificación clásica sobre estos datos, más ámpliamente probadas y aceptadas que sus equivalentes multietiqueta y gran flexibilidad, dado que permiten el uso de cualquier algoritmo de aprendizaje automático existente. Dentro de los métodos de transformación de problemas se encuentran:

- Ranking con aprendizaje de una sola etiqueta
- Métodos de combinación de etiquetas
- Métodos binarios(Relevancia Binaria....)
- Métodos de aprendizaje por pares(Ranking por Comparación de pares)

La segunda vía consiste en los denominados métodos de adaptación de algoritmos, pues adaptan técnicas de clasificación clásica a la clasificación multietiqueta. La mayor parte de las técnicas utilizadas en la clasificación clásica han sido adaptadas para tratar con datos multietiqueta directamente. La principal diferencia entre los métodos de transformación de problemas y los de adaptación de algoritmos es que los primeros son independientes del paradigma de clasificación que se utilice posteriormente, pero los segundos no, aunque algunos métodos, internamente, modifiquen los datos de entrada de manera similar a alguna técnica de transformación[33].

1.3.2.1. Métodos de transformación de problemas

Estos métodos de transformación de problemas se basan en la conversión de un problema multietiqueta para obtener uno o varios conjuntos de datos de una sola etiqueta. Estos métodos están compuestos por 5 familias que se muestran claramente en la figura 1.3:

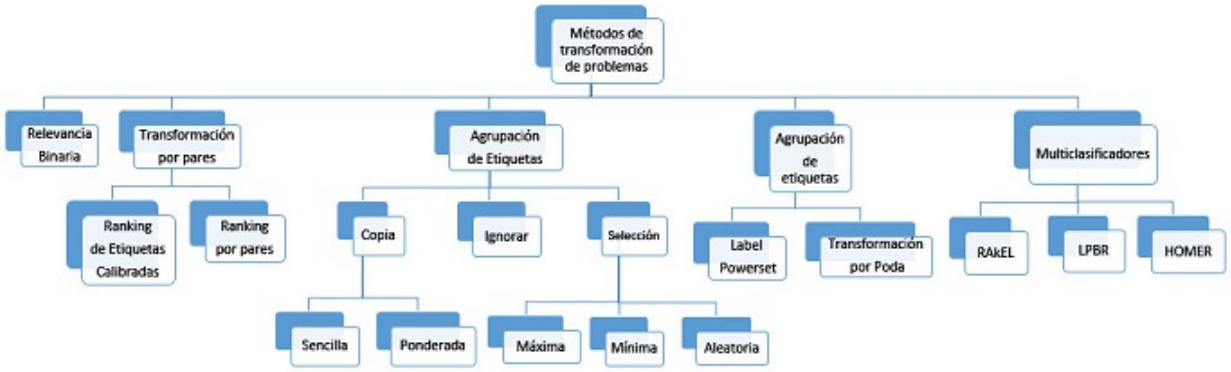

Figura 1.3: Taxonomía de los principales métodos de transformación de problemas

La primera familia esta basada en Relevancia Binaria(*Binary Relevance*, BR) [34] en este para cada etiqueta es entrenado un clasificador binario, es decir, se construye un nuevo conjunto de datos de etiqueta simple, donde cada ejemplo del conjunto original se fija como positivo si tiene asociado la etiqueta y se fija como negativo en caso contrario. Cuando se realiza la predicción, cada clasificador binario predice si su etiqueta asociada es relevante o no. Además puede llegar a ser poco práctico para determinados conjuntos de datos por el coste que supone su transformación. Como contrapartida, este enfoque permite utilizar técnicas de clasificación clásica sobre estos datos, más ampliamente probadas y aceptadas que sus equivalentes multietiqueta. Otro problema de este método es que asume independencia entre etiquetas, elemento que de ser aprovechado ayudaría a la obtención de resultados más reales y óptimos. Además puede conducir a fallos al predecir combinaciones de etiquetas o ranking de etiquetas. Sin embargo, es computacionalmente simple, hace escalas linealmente con el número de etiquetas y puede ser paralelizado. Se han desarrollado algunos otros métodos basados en BR pero tomando atención en la dependencia entre etiquetas aumentando así su complejidad. Por ejemplo:

- Clasificador de cadenas
- Meta-BR
- BRplus(BR+)

La segunda familia incluye los métodos de Comparación de Pares o *Pairwise Comparison* que será detallado más adelante por su importancia en el presente trabajo . El método de *ranking* por comparación de pares sigue la filosofía uno vs uno (OVO) y transforma un conjunto de datos con q etiquetas en $q(q-1)/2$ conjuntos de datos binarios, uno por cada par de etiquetas. Cada con-

junto de datos utiliza los ejemplos pertenecientes a una de las dos clases como positiva o negativa respectivamente(instancias pertenecientes a ambas etiquetas no son considerados). Entonces un clasificador binario es construido para cada conjunto resultante.

La tercera familia de técnicas son muy utilizadas y son conocidas como *Ranking* mediante una sola Etiqueta (*Ranking Via Single Label*). Estas técnicas en primer lugar aplican una transformación sobre el conjunto de datos de entrada, que genere un conjunto de datos de una sola etiqueta. Posteriormente se ha de utilizar una técnica de clasificación clásica, que devuelve un valor que indique el valor de pertenencia de una instancia a cada clase (por ejemplo la probabilidad de pertenencia), con el fin de obtener un ranking de etiquetas según la transformación utilizada. Se han propuesto varias formas de transformar el conjunto de datos, siendo estos métodos clasificados como métodos de copia, métodos de selección y el método de ignorar. Los métodos de copia transforman cada instancia multietiqueta en varias instancias de una sola etiqueta. Para ello copian varias veces la instancia con cada una de las etiquetas que contiene. Los métodos de selección transforman cada instancia multietiqueta en una instancia con una sola etiqueta, en contraposición con los de copia. El método debe seleccionar de algún modo la etiqueta que se le asigna a la instancia de entre todas las que contiene la instancia multietiqueta. El método de ignorar elimina del conjunto de datos las instancias multietiqueta, quedando dicho conjunto constituido solamente por las instancias no multietiqueta del conjunto de datos original. La idea es que el algoritmo aprenda varios clasificadores binarios, uno para cada etiqueta, y el clasificador final se construya en base a estos clasificadores, que devuelven si una instancia pertenece o no a cada una de las clases existentes.

La cuarta familia denominada Métodos de Agrupación de Etiquetas transforma los conjuntos multietiqueta agrupando las etiquetas entre sí, convirtiendo posteriormente cada grupo de etiquetas en una sola clase que pueda ser manejada como una tarea de clasificación de una sola etiqueta. El método *Label Powerset* (LP) es uno de ellos y revisa el conjunto de datos original y detecta todas las combinaciones de etiquetas que se dan en cada una de las instancias. Posteriormente genera un solo conjunto de datos multiclase, con tantas clases como combinaciones haya en el conjunto original, estando cada instancia asociada con la clase que representa a la combinación que tenía originalmente. Es un método simple y efectivo que además tiene en cuenta las relaciones entre etiquetas.

La última familia de métodos de transformación son los denominados multiclasificadores (*ensemble methods*), los cuales combinan las respuestas de varios clasificadores, desarrollados mediante la misma o distinta técnica, para obtener una respuesta en general más adecuada que la de cada uno de los clasificadores por separado. El algoritmo *Random k Labelset* (RAkEL) aprende una serie de clasificadores de tipo LP cada uno de ellos especializado en clasificar subconjuntos de etiquetas presentes en el problema. Para ello genera aleatoriamente una serie de subconjuntos de *k* etiquetas (*k-labelset*), entrenando un clasificador multietiqueta para cada uno de los subconjuntos[35].

1.3.2.2. Métodos de Adaptación de problemas

El segundo método es el de adaptación de algoritmos, ya que adaptan técnicas de clasificación clásica a la clasificación multietiqueta.

La mayor parte de las técnicas utilizadas en clasificación clásica han sido adaptadas para tratar con datos multietiqueta directamente. La principal diferencia entre los métodos de transformación de problemas y los de adaptación de algoritmos es que los primeros son independientes del paradigma de clasificación que se utilice posteriormente, pero los segundos no, aunque algunos métodos, internamente, modifiquen los datos de entrada de manera similar a alguna técnica de transformación.

Los métodos de adaptación emplean una técnica de clasificación clásica adaptada para trabajar directamente con datos multietiqueta. Casi todas las técnicas de clasificación se han tratado de adaptar para solventar problemas multietiqueta entre las que podemos mencionar las Máquinas de Soporte Vectorial [36][37], Árboles de Decisión [38][39], Redes Neuronales [40][41], Clasificación Asociativa [42], Métodos Probabilísticos [43], Algoritmos Bioinspirados [44], y el k vecino más cercano.

1.4. Clasificación Multietiqueta Gradual

Existen situaciones donde las etiquetas o clases son relevantes a las instancias en un cierto grado, siendo esta una utilidad que permite una mejor comprensión y utilización de la información. Esta extensión de la MLC se denomina Clasificación Multietiqueta Gradual.

La relevancia de cada etiqueta esta representada por un conjunto borroso en lugar de un conjunto binario (0/1) que denota la relación de pertenencia. De esta manera, el conjunto de grados de relevancia o pertenencia está generalizado del conjunto $\{0,1\}$ a un conjunto finito y ordenado, M, típicamente representado por un subconjunto de enteros contiguos que pueden ser leídos como variables lingüísticas. Adicionalmente, GMLC puede ser vista como un conjunto de clasificaciones ordinales. Para cada etiqueta o clase, en lugar de una clasificación binaria, GMLC define un ranking de instancias en el conjunto M de grados de pertenencia. En realidad, el objetivo de la clasificación ordinal (algunas veces llamada regresión ordinal) consiste en encontrar hipótesis capaces de predecir clases o rankings que pertenezcan a un conjunto ordenado finito, como el conjunto M de grados de pertenencia.[2] Podemos ver en el cuadro 1.5 como fue adaptado a trabajar. con GMLC.

Ejemplo	Atributo	Etiquetas			
		Religión	Historia	Filosofía	Ciencia
1	X_1	0.6	0.1	0.2	0.5
2	X_2	0.6	0.5	0.2	0.5
3	X_3	0.3	0.1	0.7	0.6
4	X_4	0.8	0.1	0.2	0.5

Cuadro 1.5: Conjunto multietiqueta gradual

Existen una amplia cantidad de ejemplos que evidencian la aplicabilidad de esta clasificación. Una película, por ejemplo, puede catalogarse como suspenso, crimen y acción al mismo tiempo, sin embargo al permitir que las instancias pertenezcan a una clase en un cierto grado de pertenencia se podría saber en que cuantía el filme contiene las clasificaciones propuestas. Por tanto en lugar de obtener sí o no $(1/0)$ se obtendrían respuestas de una escala finita y ordenada como:

M= {nada, algo, casi, completamente}

Donde cada elemento del conjunto es el significado de un valor numérico correspondiente al nivel o grado de pertenencia.

1.4.1. Notación para GMLC

Generalizando la notación utilizada para multietiqueta, se asume que cada instancia $x \in X$ puede pertenecer a cada clase $\lambda \in L$ en un cierto grado. En otras palabras, el conjunto L_x de etiquetas relevantes para una instancia x es ahora un subconjunto borroso de L. Este conjunto borroso está caracterizado por una función de pertenencia denominada mapeo $L \to M$, donde M es el conjunto de grados de pertenencia gradual. Por simplicidad notacional no se distinguirá entre el conjunto borroso L_x y su función de pertenencia y se denotara por $L_x(\lambda)$ al grado de pertenencia de la etiqueta $\lambda \in L$ en el conjunto borroso L_x .

Se utilizará una escala ordinal de grados de pertenencia, que es, un conjunto finito ordenado de grados de pertenencia. Más generalmente, se asume que $M = m_0, m_1, \ldots..m_k$, donde $m_0 < m_1 < , \ldots.. < m_k$ (donde $m_0 = 0$ y $m_1 = 1$). En el contexto de MLC, una escala ordinal de pertenencia es discutiblemente más conveniente desde un punto de vista práctico, especialmente con lo relacionado a la obtención de datos, pues las personas mayormente prefieren dar un ranking en una escala ordinal en vez de escoger un número preciso en una escala cardinal.

El objetivo es entrenar un mapa $H : X \to F(L)$, donde $F(L)$ es la clase de subconjuntos borrosos de L (con grados de pertenencia en M). Siguiendo la idea general de Reducción se trata de hacer que los problemas GMLC sean resueltos con métodos convencionales multietiqueta a través de transformaciones de un tipo a otro. Para ello se tienen tres métodos de reducción básicos: a través de la descripción "Vertical" de un subconjunto borroso F de un conjunto U (a través de la función de pertenencia, especificando el grado de pertenencia $F(u)$ para cada elemento $u \in U$) y a través

de la descripción "Horizontal" (vía cortes de nivel $[F]^{\alpha} = u \in U|F(u) \geq \alpha$), y un tercero que se compone por la unión de estos dos anteriores[2].

1.4.1.1. Reducción Vertical

En la Relevancia Binaria[34] de la MLC convencional donde para cada etiqueta $\lambda_i \in L$ un clasificador binario por separado H_i es entrenado para predecir si una etiqueta es relevante ($H_i(x) = 1$) o no ($H_i(x) = 0$) para una instancia en cola $x \in X$, generalizando este procedimiento a GMLC, la idea es introducir un clasificador $H_i : X \to M$ para cada etiqueta λ_i. Para cada instancia $x \in X$, este clasificador debe predecir el grado de pertenencia de λ_i en el conjunto borroso de etiquetas L_x. En lugar de problemas de clasificación binaria como en MLC, cada clasificador H_i, está ahora resolviendo un problema multiclase. En otras palabras, la reducción vertical de un problema GMLC eventualmente lleva a resolver un conjunto de m (no independiente) problemas de clasificación ordinal (figura 1.4)[45].

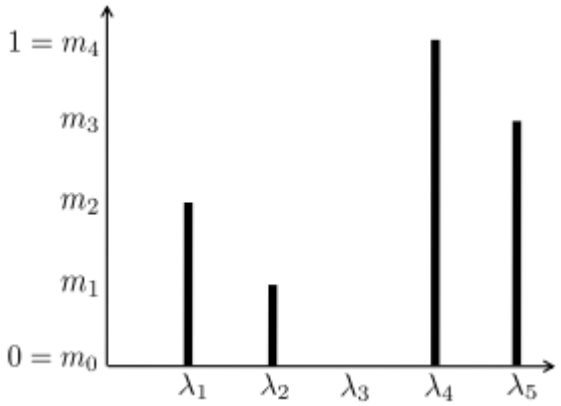

Figura 1.4: Reducción Vertical, predicción de grados de pertenencia (ordenada) por cada etiqueta (abscisa)

1.4.1.2. Reducción Horizontal

Desde la teoría de conjuntos borrosos es bien sabido que un conjunto difuso F puede ser representado horizontalmente en términos de sus cortes de nivel. Esta representación es también una descomposición razonable de un problema GMLC: Por cada nivel $\alpha \in m_1, m_2, ..., m_k$ se entrena un mapa $H^{(\alpha)} : X \to 2^M, x \to [L_x]^{\alpha}$.

Obviamente, cada uno de estos problemas es a su vez un problema de la MLC estándar, pues los cortes de nivel $[L_x]^{\alpha}$ son subconjuntos estándar del conjunto de etiquetas L. De esta manera, la reducción horizontal permite resolver k problemas de MLC estándar (1.5)[45].

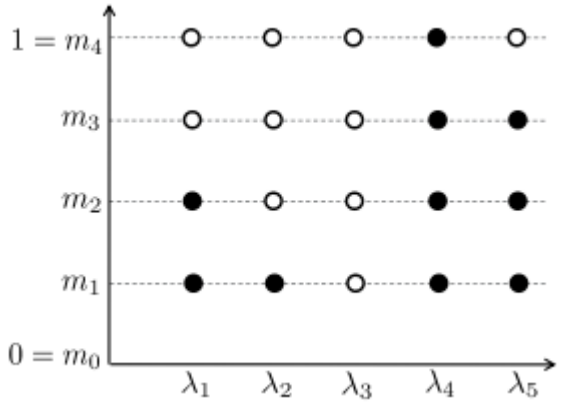

Figura 1.5: Reducción Horizontal, predicción de un subconjunto de etiquetas (indicadas por los puntos negros) en cada nivel

Como los cortes de nivel están relacionados en el sentido de que $[F]^{\alpha} \subset [F]^{\beta}$ para $\beta < \alpha$, los k problemas MLC no son independientes entre sí. Por tanto, cuando una etiqueta es predicha como perteneciente al corte m_j del conjunto de etiquetas borrosas L_x asociado con x, estará también en los cortes de nivel inferiores. Una vez que los k clasificadores multietiqueta $H^{(m1)}, ..., H^{(mk)}$ se encuentren entrenados, las predicciones pueden obtenerse a través de la siguiente ecuación: $H(x)^{(\lambda)} = max\left\{m_i \in M \,|\lambda \in H^{(mi)}(x\right\})$. Así pues, el grado de pertenencia de una etiqueta $\lambda \in L$ en un conjunto borroso predicho asociado con x está dado por el máximo grado $m_i \in M$ para el cual λ está aún en el corte predicho m_i de este conjunto.

1.4.1.3. Reducción combinada o completa

Como se mencionó anteriormente, la Relevancia Binaria es una metatécnica estándar para la resolución de problemas MLC. Consecuentemente, puede ser aplicada también para cada problema obtenido de la reducción horizontal.

Así puede obtenerse una combinación de ambos métodos utilizando primero reducción horizontal y luego reducción vertical. De cualquier manera los dos tipos de reducción pueden ser combinados en el orden inverso. Esto puede hacerse resolviendo los problemas de clasificación ordinal obtenidas en la reducción vertical a través de la descomposición horizontal, la cual constituye una metatécnica propuesta por Frank & Hall en 2001 donde: dado un conjunto ordenado de etiquetas de clase $M = \{m_0, m_1, ...m_k\}$ se entrenan k clasificadores binarios. El i-ésimo clasificador considera las instancias con etiqueta $m_0, ..., m_{i-1}$ como positivos y las que tengan etiqueta $m_i, ..., m_k$ como negativas[45].

1.4.1.4. *Pairwise Comparison*

Los métodos de trasformación por pares (*pairwise methods*) [46] siguen la filosofía "uno contra uno" y convierten un conjunto multietiqueta en varios conjuntos de datos binarios, uno por cada par de etiquetas. Están contenidos en los métodos de transformación de problemas como se observa en la figura 1.6.

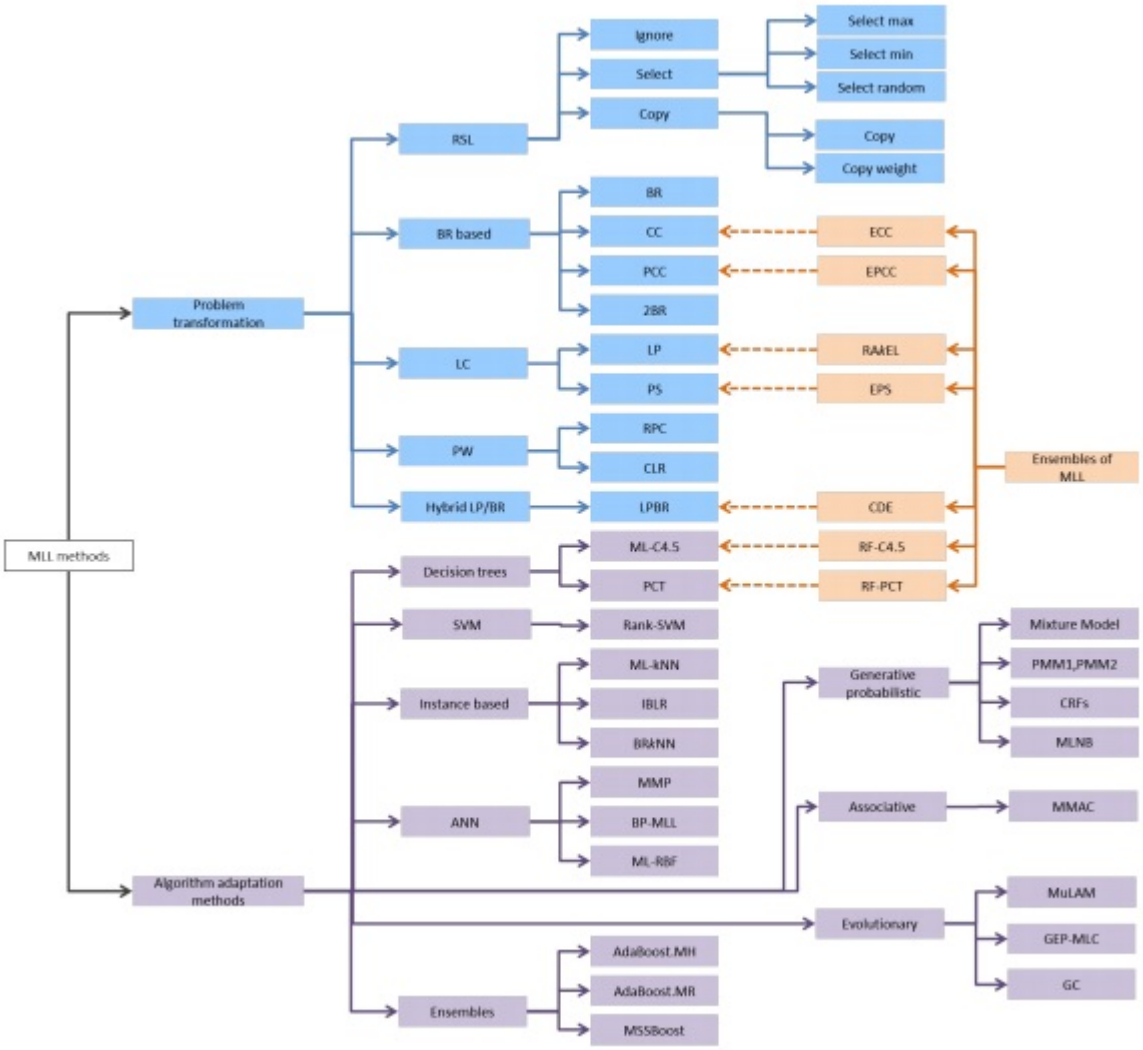

Figura 1.6: Taxonomía de los métodos MLL

Específicamente, un conjunto de datos con q clases es transformado en $q(q-1)/2$ conjuntos de datos binarios, uno por cada par de etiquetas. Posteriormente por cada uno de ellos se aprenderá un clasificador binario, utilizando cualquier técnica de clasificación binaria clásica. Cada uno de los conjuntos de datos resultantes, λ_i vs λ_j, contiene la instancia etiquetada con al menos una de

las dos etiquetas, pero no ambas, siendo verdadero si λ_i es verdadero y falso en otro caso. Así dada una nueva instancia, son invocados todos los modelos obtenidos y a través de un conteo de votos es obtenido un ranking para la instancia.

El método Ranking por Comparación de pares (RPC) [47] entrena cada uno de los modelos utilizando las instancias que están etiquetadas con alguna de las dos etiquetas del par, pero omitiendo las instancias que estan asociadas con ambas etiquetas. Las instancias de una de las etiquetas actúan como instancias positivas y los de la segunda como instancias negativos. Cada uno de estos conjuntos se utilizan para entrenar un clasificador binario, estando el modelo resultante constituido por todos los clasificadores entrenados. Las nuevas instancias son ordenadas mediante un ranking obtenido tras llamar a cada uno de estos clasificadores, y contando los votos que recibe cada etiqueta por cada clasificador.

Dado el conjunto de datos ejemplo en la sección referente a Clasificación Multietiqueta en el primer capítulo a continuación en los cuadros 1.6 y 1.7 la transformación obtenida por el RPC.

Instancia.	Etiqueta	Instancia	Etiqueta	Instancia	Etiqueta
1	$\lambda_{\neg 1,2}$	1	$\lambda_{\neg 1,3}$	2	$\lambda_{1,\neg 4}$
2	$\lambda_{1,\neg 2}$	2	$\lambda_{1\neg 3}$	3	$\lambda_{1,\neg 4}$
4	$\lambda_{\neg 1,2}$	4	$\lambda_{\neg 1,\neg 3}$	4	$\lambda_{\neg 1,4}$

Cuadro 1.6: Tres primeras tablas obtenidas de la transformación por RPC

Instancia	Etiqueta
4	$\lambda_{2,\neg 3}$

Instancia	Etiqueta	Instancia	Etiqueta
1	$\lambda_{2,\neg 4}$	3	$\lambda_{3,\neg 4}$
3	$\lambda_{2,\neg 4}$	4	$\lambda_{\neg 3,4}$

Cuadro 1.7: Tres últimas tablas obtenidas de la transformación por RPC

Calibrated Label Ranking

El método *Calibrated Label Ranking* (CLR) [2] extiende el método anterior. La dinámica es similar a éste, pero añade una etiqueta virtual que se utiliza como punto de corte entre las etiquetas que se consideraran positivas y las que se considerarán negativas. A la hora de construir los conjuntos con la etiqueta virtual, las instancias que tienen asociada una determinada etiqueta se consideran positivas para esta y negativas para la virtual, y por el contrario los que no estén asociados a la etiqueta se consideran negativas para esta y positivas para la virtual. Luego, puede ser aplicado un clasificador de relevancia binaria para discriminar entre las etiquetas calibradas y el resto de las etiquetas. Finalmente un *ranking* de etiquetas calibradas quedaría de la siguiente manera:

$$\lambda_{i1} \succ \lambda_{i2} \succ \ldots\ldots\ldots \succ \lambda_{ij} \succ \lambda_0 \succ \lambda_{ij+1} \succ \ldots\ldots \succ \lambda_{ik} \tag{1.4}$$

El CLR es un método que utiliza el label ranking para dar solución al problema multietiqueta.

En CLR el problema multietiqueta original es reinterpretado como un problema de ranking de etiquetas. La mayor dificultad de este método es la búsqueda de un umbral t apropiado para reducir la diferencia entre el conjunto de etiquetas predichas $\hat{P}_x$ y las etiquetas correctas P_x. Por eso este método trata de predecir el ranking aprendiendo comparación por pares entre etiquetas. Esto significa que por cada par de etiquetas (λ_i, λ_j) con $i \neq j$ es aprendido un clasificador binario $H_{(\lambda_i,\lambda_j)}$ para predecir cual es más probable.

Es entrenado con las instancias $x \in X_{train}$ etiquetadas con λ_i o λ_j pero no con ambas. Para inferir el ranking la función de puntuación es obtenida por la expresión 1.5.

$$f(x,\lambda_i) = \sum_{\lambda_j}^{\zeta} H_{(\lambda_i,\lambda_j)} \tag{1.5}$$

La puntuación es obtenida sumando los votos de cada uno de los clasificadores binarios. En la figura 1.7 se muestra en (a) la comparación por pares entre las etiquetas de un problema de clasificación multietiqueta, en (b) las comparaciones adicionales por la etiqueta virtual introducida por CLR al problema de clasificación multietiqueta y en (c) todas las comparaciones por pares realizadas por CLR.

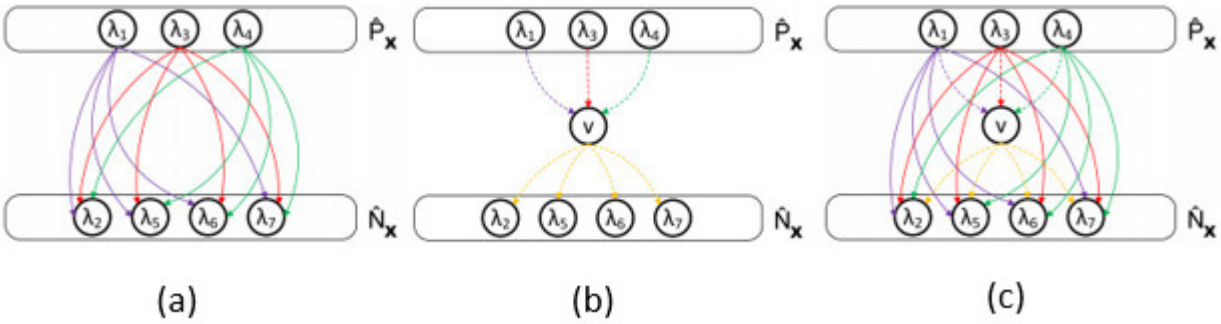

Figura 1.7: Comparaciones del método de comparación por pares en MLC en CLR con una etiqueta virtual.

En la figura 1.8 se puede observar el ranking predicho por CLR a través de la etiqueta virtual que separa las etiquetas irrelevantes de las relevantes.

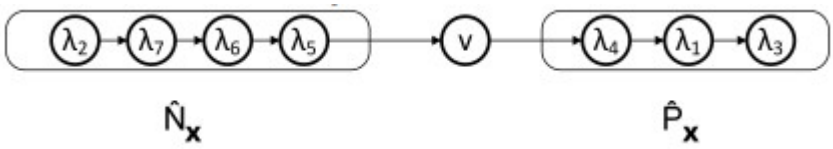

Figura 1.8: Ranking predicho de las etiquetas de un problema de clasificación mutietiqueta.

Preferencias

El tema de las preferencias ha atraído recientemente considerable atención en investigaciones de la Inteligencia Artificial (*Artificial Inteligence*, AI), sobre todo en campos tales como agentes, el razonamiento no monótono, satisfacción de restricciones, planificación y teoría de la decisión cualitativa. Las preferencias proporcionan un medio para especificar los deseos de una manera declarativa, que es un punto de importancia crítica para el paradigma de la IA de un agente que actúa racionalmente: El comportamiento de un agente tal tiene que ser impulsado por un modelo de preferencia subyacente, y un agente de recomendar decisiones o actuar en nombre de un usuario debe reflejar claramente preferencias de dicho usuario. No resulta sorprendente que los métodos automáticos de aprendizaje y predicción de preferencias se encuentran entre los muy recientes temas de investigación en disciplinas como las máquinas de aprendizaje, descubrimiento de conocimiento, y los sistemas de recomendación. En la figura 1.9 se muestra los tipos de preferencias existentes ubicándose la gradual dentro de las preferencias absolutas y más detalladamente en el cuadro 1.8 se puede observar cuando el entrenamiento y la predicción son gradual entonces nos encontramos en la clasificación multietiqueta gradual.

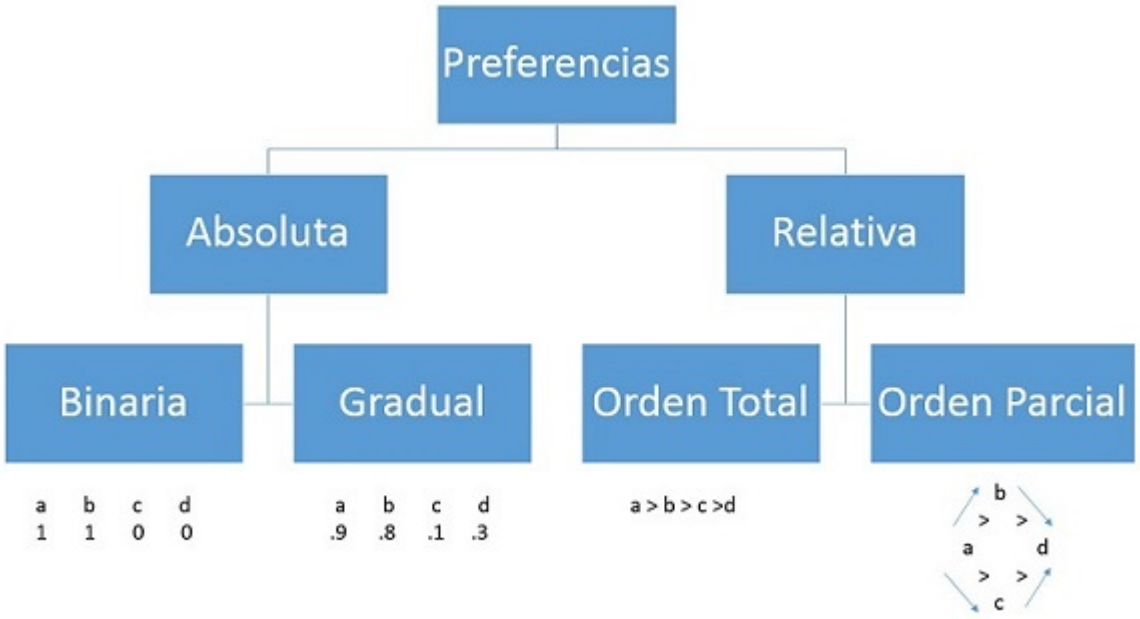

Figura 1.9: Gráfico que muestra el aprendizaje por preferencias

Entrenamiento	Predicción	
binaria	binaria	clasificación multietiqueta
binaria	orden total	clasificación multietiqueta(*ranking*)
graded	graded	clasificación multietiqueta gradual
orden parcial	orden total	*ranking* de etiquetas
orden parcial	orden parcial	*ranking* con abstención

Cuadro 1.8: Tabla que muestra de acuerdo al entrenamiento y la predicción el tipo de problema a tratar con aprendizaje por preferencias.

Capítulo 2

Extensión de MULAN a Clasificación Multietiqueta Gradual

En el presente capítulo se explican cada uno de los métodos de solución existentes en la bibliografía, desde clásicos readaptados al trabajo con clasificación multietiqueta gradual hasta los más recientes, que introducen nuevos métodos de transformación de problemas que logran mejores resultados. Además, se muestra la estructura seguida por el nuevo paquete haciendo especial énfasis en las clases más importantes.

2.1. Métodos de solución a la clasificación multietiqueta gradual existentes en la literatura

Desde que la clasificación de instancias multietiquetadas con datos de tipo gradual comenzó a ser de interés para la comunidad científica también comenzaron a serlo sus posibles vías de solución y la búsqueda de lograr en ellas mayor rapidez y optimización. A continuación se realiza una revisión y breve reseña de las tres respuestas creadas para esta problemática hasta la actualidad. Primeramente se discuten las reducciones vertical y horizontal de los problemas de clasificación multietiqueta gradual. Estas metatécnicas de reducción son utilizadas por los dos primeros métodos existentes en la literatura. El primero de ellos es precisamente la combinación de éstas dos reducciones, denominado reducción completa. El segundo fue llamado IBLR-ML por sus creadores. El tercero es una familia de tres métodos que utilizan comparación por pares y ranking de etiquetas calibradas denominados Horizontal CLR, Full CLR y Joined CLR.

Reducción Vertical

Debido a la alta complejidad de los problemas de clasificación multietiqueta gradual resulta muy razonable y necesaria la reducción de los mismos a problemas de clasificación menos com-

plejos para reducir radio de error y tiempo de cómputo de los algoritmos. Un método que permite esto es la reducción vertical. Este reduce la GMLC de la misma manera que la Relevancia Binaria en la clasificación multietiqueta. Para cada etiqueta $\lambda_1, ..., \lambda_n$ de la clasificación multietiqueta la relevancia binaria aprende un clasificador binario para predecir la relevancia de la etiqueta. Así pues, los problemas de clasificación multietiqueta son reducidos a n problemas de clasificación binarios individuales. Ya en la reducción vertical, los problemas de clasificación multietiqueta gradual son reducidos a n problemas de clasificación ordinal u ordenada.

$H : X \times \zeta \to M$ es transformado a $[H]_{\lambda_1}, ..., [H]_{\lambda_n}$

con $[H]_{\lambda_i} : X \to M$ para $i = 1, ..., n$

El conjunto de entrenamiento de los clasificadores ordinales individuales $[H]_{\lambda_i}$consisten en todas las instancias del conjunto de entrenamiento del problemas de clasificación multietiqueta gradual H pero su clasificación es reducida al grado $m \in M$ de la etiqueta λ_i en la clasificación original. Como en la relevancia binaria toda la información acerca de la correlación e interdependencias entre las etiquetas individuales se pierde no puede ser usada para optimizar el proceso de clasificación[2].

Reducción Horizontal

De igual manera que la reducción a problemas de clasificación ordenada, los problemas de clasificación multietiqueta gradual pueden ser reducidos a varios problemas binarios de clasificación multietiqueta. Este proceso de reducción fue denominado reducción horizontal. Inspirado por la teoría de conjuntos borrosos, el espacio objetivo del clasificador H es dividido por cada grado $m_i \in M$ en niveles. Específicamente, por cada grado $m_i \in M$ es aprendido un clasificador para predecir si una etiqueta debería ser clasificada o no. La función de clasificación es reemplazada: $H : X \times \zeta \to M$ es transformada en $[H]_{m_0}, ..., [H]_{m_n}$.

Análogamente a la reducción vertical los clasificadores multietiquetas individuales son entrenados con el conjunto de entrenamiento completo. Sin embargo, a diferencia de la vertical en esta reducción la clasificación de las instancias individuales de entrenamiento es reducida a la información de si el grado de una etiqueta distinta es mayor que el grado dependiendo al nivel o no. Si el grado es mayor o igual que el grado del nivel la instancia es tratada como un ejemplo positivo y en otro caso como uno negativo. Los clasificadores $[H]_{mi}$no son independientes entre ellos. Si alguna etiqueta es relevante a algún grado m_i, es también relevante a todos los grados $m_j \prec m_i$.

El tamaño de los conjuntos crece monótonamente con grados incrementales. Esto obviamente lleva al problema no trivial de asegurar esta monotonía durante la agregación de las predicciones de los $n = |M|$ clasificadores[2].

2.1.1. Reducción Completa

La combinación de las dos reducciones anteriores constituyen el primer método de solución en la bibliografía sobre el tema. Esta metodología reduce el problema multietiqueta gradual a un conjunto de problemas de clasificación binarios constituidos por un clasificador binario por cada posible combinación de salida etiqueta-grado. Existen métodos específicos para cada unos de las reducciones anteriores, tanto para la reducción vertical como la horizontal que pueden ser aplicados al resultante problema de clasificación ordenado o multietiqueta. Para el primer tipo de problema esta el método Frank & Hall y para el segundo es la reducción binaria.

Si ambas reducciones son aplicadas a los problemas de clasificación multietiqueta gradual se obtiene un conjunto de $|M \times \zeta|$ problemas de clasificación binarios. Uno por cada combinación grado-etiqueta $(\lambda_i, m_j) \in M \times \zeta$. Los clasificadores sencillos son:

$[H]_{(\lambda_i, m_j)} : X \to \{0,1\}$

La predicción del clasificador $H]_{(\lambda_i, m_j)}$ asume que para una instancia x la etiqueta λ_i es relevante con un grado $m_k \geq m_j$. El orden de aplicación de las reducciones horizontal y vertical es intercambiable mientras se mantenga la condición de monotonía especificada anteriormente.

2.1.2. IBLR-ML

El segundo método para resolver clasificación multietiqueta gradual utiliza la reducción horizontal para obtener un conjunto de $k = |\zeta|$ problemas independientes de clasificación multietiqueta. Estos problemas son individualmente resueltos por el método IBLR-ML. Este combina la regresión logística con el aprendizaje basado en instancias para tomar ventaja teniendo en cuenta las interdependencias y correlación entre las diferentes etiquetas de un problema de clasificación multietiqueta. La predicción del método esta basada en la probabilidad asumida posteriormente $\pi_0^{(i)}$ de una etiqueta λ_i siendo relevante para alguna instancia x_0. El clasificador sería:

$$[H]_{m_j}(x_0) = \{\lambda_i \in \zeta | \pi_0^{(i)} > \frac{1}{2}\} \tag{2.1}$$

Los autores sugirieron otra configuración para el método. Esta utiliza la reducción vertical en lugar de la horizontal y cambia la regresión logística binaria por la regresión logística ordinal[45].

2.2. Implementación en MULAN

MULAN es una biblioteca de código abierto que trabaja con conjuntos de datos multietiqueta los cuales constan de ejemplos de entrenamiento de una función objetivo que tiene varias variables de destino binarios. Esto significa que cada elemento de un conjunto de datos multietiqueta puede ser miembro de varias categorías o anotado por muchas etiquetas (clases). Esto es en realidad la naturaleza de muchos problemas del mundo real, tales como la anotación semántica de imágenes y

de vídeo, página web categorización, el marketing directo, la genómica funcional y la categorización de la música en géneros y emociones. Una introducción a la minería de datos multietiqueta se proporciona en [48]

Actualmente, la biblioteca incluye una variedad de algoritmos del estado del arte para realizar las siguientes tareas principales de aprendizaje multietiqueta:

- Clasificación. Esta tarea se refiere a la salida de una bipartición de las etiquetas en los más relevantes e irrelevantes para una instancia determinada entrada.
- *Ranking*. Esta tarea se refiere a la salida de una ordenación de las etiquetas, de acuerdo con su importancia para un elemento de datos dado.
- Clasificación y ranking. Una combinación de las dos tareas mencionadas anteriormente.

Además, la biblioteca ofrece las siguientes características:

- La selección de características. Métodos de referencia simples son soportados actualmente.
- Evaluación. Las clases que calculan una gran variedad de medidas de evaluación a través de la evaluación de retención y validación cruzada.

Como ya se mencionó, MULAN es una biblioteca. Como tal, ofrece sólo API programático a los usuarios de la biblioteca. No hay ninguna interfaz gráfica de usuario (GUI) disponibles. La posibilidad de utilizar la biblioteca a través de línea de comandos, tampoco esta disponible en la actualidad.

2.2.1. Códigos integrados (Métodos clásicos)

Para el desarrollo del marco de trabajo que abarca la clasificación multietiqueta gradual se incorporan a MULAN cinco algoritmos que abordan este tema desde diferentes perspectivas. Dos de los algoritmos, constituyen clásicos en la literatura.

- Binary Relevance
- Frank & Hall

A continuación se describe como funcionan cada uno de los algoritmos utilizados:

Binary Relevance

El método de Relevancia Binaria es uno de los métodos más famosos dentro de los métodos de transformación y sigue la filosofía de uno contra todos, de esta manera construye un conjunto de datos binarios por cada etiqueta procesada. Las instancias que predicen la etiqueta son considerados

positivos mientras que el resto son considerados negativos. Una vez una instancia desconocida es introducida en el modelo, la salida será el conjunto de clases predichas positivas. Más formalmente, este método crea k conjuntos de datos ($k = |\zeta|$, número total de clases), cada uno para una etiqueta clase y entrena un clasificador para cada uno de estos conjuntos de datos. Cada conjunto contiene exactamente el mismo número de instancias que el conjunto original, sin embargo, cada conjunto $D_{\lambda_j}, 1 \leq j \leq k$ etiqueta positivamente las instancias que pertenecen a la clase λ_j y negativamente en otro caso. El principal problema de este método es que supone la independencia entre las etiquetas ignorando la relación entre ellas. Sin embargo, es computacionalmente simple y puede ser paralelizado.

Dado el conjunto de datos ejemplo en la sección referente a Clasificación Multietiqueta en el primer capítulo[2].

Instancia	Etiqueta	Instancia	Etiqueta
1	$\neg\lambda_1$	1	λ_2
2	λ_1	2	$\neg\lambda_2$
3	λ_1	3	λ_2
4	$\neg\lambda_1$	4	λ_2

(a) y (b)

Instancia	Etiqueta	Instancia	Etiqueta
1	λ_3	1	$\neg\lambda_4$
2	$\neg\lambda_3$	2	$\neg\lambda_4$
3	λ_3	3	$\neg\lambda_4$
4	$\neg\lambda_3$	4	λ_4

(c) y (d)

Figura 2.1: Conjuntos transformados por Relevancia Binaria.

Frank & Hall

Frank & Hall tiene gran parecido al anteriormente analizado ya que mientras el método de Relevancia Binaria utilizaba las descomposiciones horizontal y vertical en ese preciso orden, Frank & Hall combina estas dos descomposiciones pero en orden inverso, primero vertical y luego horizontal. De esta manera se resuelven los k problemas obtenidos por una reducción vertical a través de la descomposición horizontal. Esta metatécnica fue propuesta por Frank & Hall en el 2001. Formalmente, dado un conjunto ordenado de etiquetas clase $M = \{m_0, m_1 ... m_k\}$, la idea es entrenar k clasificadores binarios. El i-ésimo clasificador considera las instancias con etiquetas $m_0, ..., m_{i-1}$como positivas y aquellos con $m_i, ..., m_k$como negativas. En general un único problema binario es resuelto por cada combinación etiqueta/nivel $(\lambda_i, m_j) \in \zeta \times M$ decidiendo luego si $L_x(\lambda_i) \leq m_j$ o $L_x(\lambda_i) > m_j$.[2]

2.2.2. Métodos basados en Ranking de Etiqueta Calibradas

Los 3 restantes son algoritmos desarrollados y propuestos por el grupo de desarrollo "*Knowledge Engeineering Group*" de la Universidad Técnica de *Darmstadt* en Alemania que utiliza las técnicas de descomposición por comparación de pares y el Ranking de Etiquetas Calibradas (CLR) por sus siglas en inglés:

- Horizontal CLR
- Full CLR
- Joined CLR

Antes de explicar el funcionamiento básico de los métodos antes mencionados se realizará una breve reseña sobre la descomposición por pares y su estrecha relación con el *Ranking* de Etiquetas Calibradas(CLR), conceptos imprescindibles para la comprensión de dichos algoritmos dado que sobre la base de CLR se construyó una estrategia de comparación por pares para la resolución de problemas multietiquetas y estos métodos son precisamente una generalización de las técnicas CLR al caso de Clasificación Multietiqueta Gradual.

- Descomposición por pares

El aprendizaje mediante la descomposición por pares esta basado en la idea de modelar preferencias entre etiquetas. Estas preferencias son, o bien derivadas de la estructura de la etiqueta o dadas a mano por las instancias de entrenamiento en una forma total o parcial. La descomposición por pares toma de manera implícita las dependencias entre etiquetas dentro de una relación. Formalmente hablando, la descomposición por pares de problemas multietiquetas interpreta los datos de entrenamiento dados como un *ranking* bipartito $N_x \prec P_x$ del que se pueden deducir explícitamente sentencias de preferencias $\lambda_u \prec \lambda_v$ para todo $\lambda_u \in N_x, \lambda_{v \in} P_x$. Estas preferencias son aprendidas mediante clasificadores entrenados $H_{uv} : x \to \{0,1\}$ para cada uno de los posibles pares de etiquetas, $1 \leq u \leq v \leq n$. De esta manera el problema es descompuesto en $\frac{n(n-1)}{2}$ subproblemas binarios más pequeños. Para cada par de etiquetas (λ_u, λ_v) solo los ejemplos pertenecientes a λ_u o λ_v son usados para entrenar el correspondiente clasificador $H_{u,v}$. El resto de los ejemplos son ignorados. Más precisamente, asumiendo $u < v$, un ejemplo es agregado al conjunto de entrenamiento para el clasificador $H_{u,v}$ si λ_u es una etiqueta relevante y λ_v es irrelevante o viceversa. Así, ejemplos de entrenamiento pertenecientes a la primera etiqueta recibirán una señalización de entrenamiento de 1, mientras que ejemplos con la segunda etiqueta serán clasificados con 0.

Durante la clasificación, las predicciones de los $\frac{n(n-1)}{2}$ clasificadores base $H_{u,v}$ son interpretadas como sentencias de preferencias que predicen para un ejemplo dado cual de las dos etiquetas λ_u o λ_v es la preferida. Con el fin de convertir estas preferencias binarias en *ranking* de etiquetas, se usa un técnica de votación simple que interpreta cada preferencia binaria como un voto (sin peso)

completo (0 o 1) para la clase preferida. Las etiquetas son entonces organizadas en un *ranking* de acuerdo con el número de votos recibidos después de la evaluación de todos los clasificadores.

Para convertir el *ranking* de etiquetas resultante en una predicción multietiqueta se utiliza la técnica de CLR. Esta técnica evita la necesidad de construir una función de umbral para separar las etiquetas relevantes de las irrelevantes, que en la mayoría de los casos es ejecutada como una fase postprocesamiento después de computar un ranking de todas las posibles clases. La idea básica es introducir una etiqueta de calibración artificial $v = \lambda_0$, que representa el punto de corte entre las etiquetas relevantes e irrelevantes. Así, se asume que v será preferida sobre todas las etiquetas irrelevantes, pero todas las etiquetas relevantes serán preferidas sobre v.

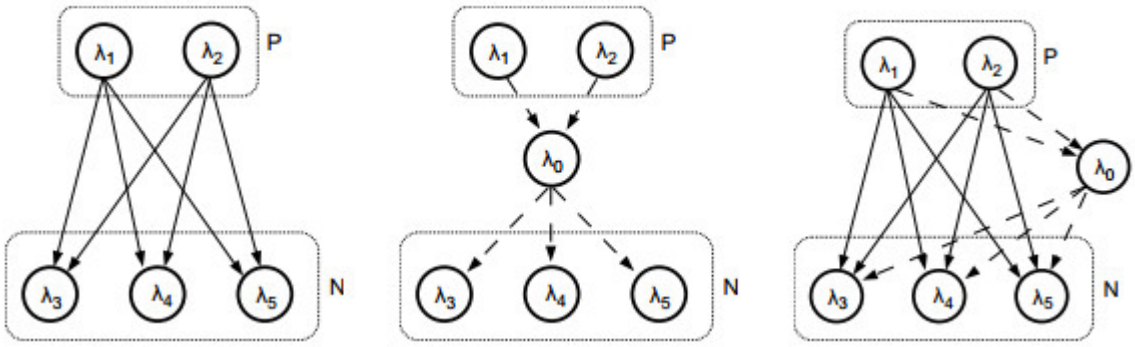

Figura 2.2: Preferencias en *ranking* de etiquetas calibradas: a la izquierda, se observan todas las preferencias entre las etiquetas relevantes y las irrelevantes. En el centro, se muestra la posición de la etiqueta virtual v y a la derecha todas las preferencias generadas (la unión de los dos gráficos anteriores)

Durante la predicción, la etiqueta virtual es insertada naturalmente en el ranking de etiquetas y es tratada como cualquier otra etiqueta. La posición de la etiqueta virtual en el *ranking* predicho entonces denota un punto de corte natural para dividir el *ranking* en dos conjuntos.

El método de Descomposición por Pares es a menudo considerado como superior al de relevancia binaria porque se obtienen resultados de límites de decisión más simples. La razón está en que cada unos de los clasificadores de la descomposición por pares contiene menos ejemplos.

Comparativamente, cada ejemplo de entrenamiento original es usado por cada uno de los n clasificadores aprendidos en Relevancia Binaria mientras que solo ocurre en $|P_x|(n-|P_x|)$ del número cuadrático de clasificadores obtenidos en la descomposición por pares, con $|P_x|$ usualmente pequeña (< 5). El hecho de que estos ejemplos son distribuidos sobre un número más grande de clasificadores hacen a este método particularmente atractivo para clasificadores costosos. Este algoritmo resulta práctico para problemas con miles de etiquetas y la fase de predicción puede ser considerablemente acelerada en casos donde solo se necesita un pequeño número de etiquetas relevantes.

Este método CLR generalizado a GMLC, al igual que el original, se encuentra dividido en dos fases principales. La primera es la fase de entrenamiento, donde se reduce el problema a problemas binarios y se realiza el entrenamiento de los correspondientes clasificadores binarios. La segunda fase realiza la clasificación de una instancia desconocida $x \in X$. Los problemas clásicos de clasificación multietiqueta pueden ser vistos como un caso simplificado de los problemas de GMLC con $|M| = 2$ donde si se usara CLR se necesitaria un valor umbral para separar estos dos grados para el ranking creado por esta técnica. Sin embargo, en GMLC al ser más de dos grados son necesarios $n-1$ valores de umbral para separar los $n = |M|$ diferentes grados en el ranking, por lo que para este caso en lugar de una etiqueta virtual es necesaria una por cada par de grados m_i y m_{i+1}. Luego clasificadores por pares son aprendidos por cada par de $\zeta \times \zeta$ como en el método normal de CLR. Para esto, la función de proyección tiene que ser cambiada para generalizar la decisión binaria $(\lambda_i \in P_x \wedge \lambda_j \notin P_x) \vee (\lambda_i \notin P_x \wedge \lambda_j \in P_x) \vee)$ a la comparación de los grados individuales $(L_x(\lambda_i) \prec L_x(\lambda_j)) \vee (L_x(\lambda_j) \prec L_x(\lambda_i))$. Así que en este caso se determina si una etiqueta es más relevante que otra, no si son relevantes o no. Las etiquetas virtuales deben cumplir varias reglas para mantener la restricción de monotonía del ranking, por ejemplo, una etiqueta virtual v_i separando dos etiquetas λ_i con grado m y λ_j con grado n donde $m < n$ entonces $v_i$es preferida sobre λ_i pero menos preferida que λ_j. Además, las etiquetas virtuales serán usadas para mantener un orden entre los diferentes niveles de grados y facilitar las comparaciones, no pueden ser asignadas a ninguna instancia. Al igual que en el método normal de CLR aquí las etiquetas normales además de compararse entre sí, son comparadas con cada una de la etiquetas virtuales, con la diferencia de que aquí el número de etiquetas virtuales asciende a la par del número de grados(figura 2.3).

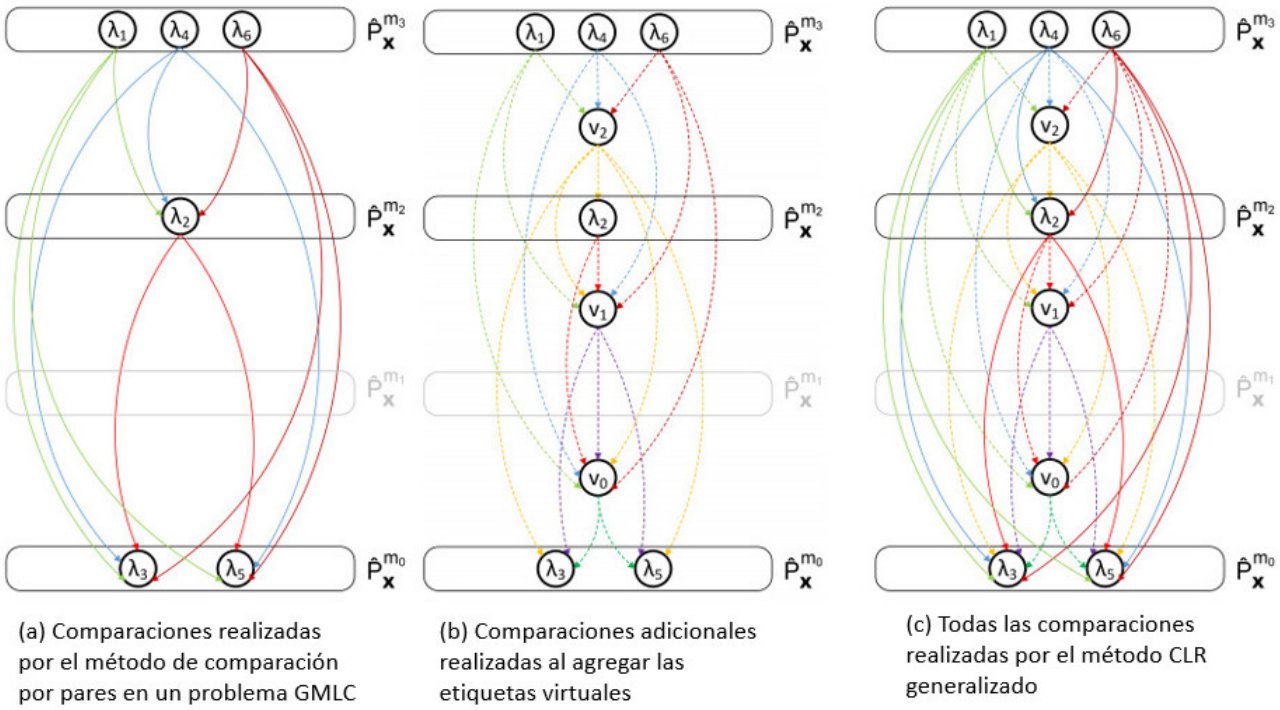

Figura 2.3: Un ejemplo de la predicción del CLR generalizado al caso GMLC

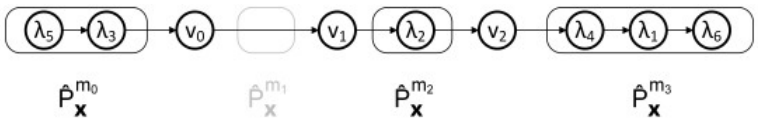

Figura 2.4: Particionamiento del ranking del ejemplo anterior

- Etiquetas Calibradas Múltiples

La idea general del método explicado anteriormente sobre multietiqueta gradual es la generalización de CLR al caso de múltiples etiquetas calibradas $V = v_1, ..., v_{m-1}$, donde cada etiqueta representa un grado intermedio entre las grados originales μ_i y μ_{i+1}. Así se obtiene $M_v = M \frown V$ con la siguiente estructura interna: $\mu_1 \prec v_1 \prec \mu_2 \prec v_2 \prec \mu_3 \prec ... \prec v_{m-1} \prec \mu_m$. Como consecuencia se obtiene un conjunto extendido de etiquetas $L \smile V$. A continuación se describen a grandes rasgos los tres métodos integrados al MULAN que siguen las técnicas explicadas anteriormente.[2]

Horizontal Calibrated Label Ranking

Este método para generalizar CLR al caso gradual utiliza la decomposición horizontal explicada en el capítulo 1 y resuelve cada uno de los problemas multietiquetas resultantes con CLR. Así, con el objetivo de aprender cada $[H]^i$, se escoge a v_i como el punto de corte. Traducido a CLR v_i se convierte en la etiqueta calibradadora y los conjuntos en que divide a las instancias, los conjuntos de etiquetas positivas y negativas respectivamente.

Más precisamente, se entrena cada clasificador $H^i_{\lambda_u,\lambda_v}, \lambda_u \neq \lambda_v, \lambda_u, \lambda_v \in L \smile \{v_i\}$ usando ejemplos de entrenamiento $(x, [p]^i_{\lambda_u,\lambda_v}(x))$ dados por:

$$[p]^i_{\lambda_u,\lambda_v}(\mathbf{x}) = \begin{cases} 1 & \text{if } [L]^i_{\mathbf{x}}(\lambda_v) \prec [L]^i_{\mathbf{x}}(\lambda_u) \\ 0 & \text{if } [L]^i_{\mathbf{x}}(\lambda_u) \prec [L]^i_{\mathbf{x}}(\lambda_v) \\ \varnothing & \text{if } [L]^i_{\mathbf{x}}(\lambda_u) = [L]^i_{\mathbf{x}}(\lambda_v) \end{cases}$$

$$[L]^i(\lambda_u) = \begin{cases} \mu_i & \text{if } \lambda_u \prec v_u \\ \mu_{i+1} & \text{if } v_u \prec \lambda_u \end{cases}$$

Figura 2.5: Clasificador en HorizontalCLR

Para hacer una predicción para una instancia de prueba x, los votos $h_x(\lambda_u) = \sum_{\lambda_u \neq \lambda_v} [H]^i_{\lambda_u,\lambda_v}(x)$ son sumados para cada etiqueta $\lambda_u \in L \smile \{v_i\}$y $\lambda_u$es predicha como relevante si $h_x(\lambda_u) > h_x(\lambda_{v_i})$. La predicción gradual final es obtenida a través del uso de la puntuación predicha máxima.[2]

Full Calibrated Label Ranking

La idea del método de Ranking de etiquetas calibradas completo es considerar los objetivos en los problemas GMLC como un ranking multipartito. Luego se transforma y en el ranking multipartito $P_x^{'1} \prec P_x^{'2} ... \prec P_x^{'m}$. Enriquecido con una etiqueta virtual se obtendría el siguiente ranking:

$$P_x^{'1} \prec \{v_1\} \prec P_x^{'2} ... \prec \{v_{m-1}\} \prec P_x^{'m} \tag{2.2}$$

La función de proyección para los clasificadores bases $[H]_{\lambda_u,\lambda_v}, \lambda_u \neq \lambda_v, \lambda_u, \lambda_v \in L \smile V$ solo ofrece ligeros cambios al de HorizontalCLR (figura 2.6):

$$[p]_{\lambda_u,\lambda_v}(\mathbf{x}) = \begin{cases} 1 & \text{if } L(\lambda_v) \prec L(\lambda_u) \\ 0 & \text{if } L(\lambda_u) \prec L(\lambda_v) \\ \varnothing & \text{if } L(\lambda_u) = L(\lambda_v) \end{cases}$$

Figura 2.6: Clasificador en FCLR

En contraste con la descomposición horizontal en la figura 2.6 se puede apreciar la posibilidad de sumar los votos por grados, obteniendo un ranking global sobre todos los grados y etiquetas. Luego de poner en cola los $(n+m-1)(n+m-2)/2$ clasificadores base, entonces se predice $\hat{y^j}$=$arg\,max_{\mu_i} h_x(\lambda_j) > h_x(\lambda_{v_i})$ para λ_j.

Una posible desventaja de este método es que el algoritmo tiende a producir muchos empates en el ranking debido a que $n+m-1$ etiquetas tienen que ser ordenadas en una escala de 0 a $n+m-2$ votos obtensibles. Esto puede ser remediado utilizando diferentes funciones de votación como la función de votación por pesos. La función de votación 0-1 es tambien un ejemplo bastante robusto y

hace la menor cantidad de suposiciones en los clasificadores bases. Otro problema con este método es que no considera la intensidad de las preferencias, o sea, ignora la diferencia entre los grados de dos etiquetas comparadas. El siguiente método soluciona estos problemas[2].

Joined Calibrated Label Ranking

El método Full CLR no es capaz de capturar los diferentes grados de intensidad de preferencia debido a que la preferencia entre dos etiquetas λ_u y λ_v, es obtenida de manera binaria. En el caso del Horizontal CLR se aprendía cada clasificador discriminante $[H]^i_{\lambda_u,\lambda_v}$ exactamente $m-1$ veces, una por cada transición entre grados. De hecho el número de clasificadores λ_u vs. λ_v que utilizan una instancia de entrenamiento x depende de la diferencia entre los grados de la etiqueta. Por ende, la diferencia en el número de votos entre ambas etiquetas correlaciona con la diferencia entre los grados reales. Una solución es computar un ranking compartido común entre grados y etiquetas. Desafortunadamente no se puede obtener un buen *ranking* sobre las etiquetas virtuales, pues cada una de las etiquetas virtuales aparece solo en un sub-problema horizontal y solo puede obtener como máximo n votos cuando las etiquetas reales pueden llegar a obtener hasta $n(m-1)$ votos.

Joined CLR resuelve este problema generalizando la descomposición horizontal introducida, así, todas las etiquetas virtuales son siempre usadas en todos los sub-problemas horizontales. Más precisamente descompone el problema inicial en $m-1$ problemas de *ranking* bipartitos con una etiqueta calibradora principal v_i en cada transición de grado. En lo que resta Joined CLR es equivalente al Horizontal CLR y todos los clasificadores bases por pares aprendidos en el Horizontal CLR son aprendidos también exactamente de la misma manera en Joined CLR. Sin embargo Joined CLR también adiciona todas las etiquetas virtuales restantes $v_j \neq v_i$en problemas de *ranking* bipartitos permitiéndoles acumular el monto necesario de votos. Formalmente hablando, se aprenden clasificadores $[H]^i_{\lambda_u,\lambda_v}$usando $[p]$ y $[L]$ de la figura 2.5, pero en este caso por cada $\lambda_u \neq \lambda_v, \lambda_u, \lambda_v \in L \smile V$. Por lo tanto se asigna $[H]^i_{v_u,v_v}(x)$=0 si $v_u \prec v_v$ y 1 en otro caso. Durante la predicción, se agregan los votos para cada etiqueta por todas las transiciones de grado.[2]

Comparación entre los métodos

Los cinco métodos utilizados arrojaron resultados perfectamente comparables entre sí que permitieron determinar las mejores opciones bajo determinados factores. El método que utiliza reducción completa mediante cortes horizontales y verticales, BR, generalmente fue superado por los métodos de comparación por pares. Sin embargo BR se mantuvo con mejores resultados que F&H incluso cuando ambos clasificadores son entrenados de la misma manera. Los resultados de los diferentes métodos de CLR muestran una alta correspondencia con sus respectivas estructuras internas. El método FullCLR presenta los más altos valores en las métricas *Hamming Loss* y 0-1 *Loss* de todos los métodos. Al analizar los resultados de las métricas *Optimistic Hamming Loss*, *C-Index* y *One Error* se puede deducir claramente que se obtendrá un mejor resultado en la medida

que mejor se posicionen las etiquetas virtuales evitando la cantidad de empates en el ranking. El Joined CLR mostró un comportamiento similar. En particular obtuvo los mejores resultados en las métricas de ranking excepto con el conjunto de datos *Medical*. Los bajos resultados en este conjunto se deben a que este método puede tener problemas con conjuntos de datos con muchas etiquetas asignadas a grados extremadamente bajos o altos. Finalmente el método más potente obteniendo los mejores resultados en las métricas *Hamming* y 0-1 vertical fue el Horizontal CLR. Esto es entendible debido al modo más fácil en el método posiciona la etiqueta calibrada. Sin embargo mostró no muy buenos resultados en la calidad de predicción de *ranking*, siendo el peor de los tres métodos usando comparación de pares obteniendo más bajos valores en la métricas C-*Index* y *One Error*. Resulta obvio entonces, que la estrategia de seleccionar el grado más alto para cada etiqueta usado también en BR no ofrece ninguna ventaja a la calidad del *ranking*.

A manera de resumen, los métodos utilizando comparación por pares generalmente obtuvieron mejores resultados que el resto de acuerdo a las métricas utilizadas. Sobre todo el Full y Joined CLR mostraron una ventaja clara cuando se necesitan un buen ranking de etiquetas. Finalmente el Horizontal CLR fue el de mayor integralidad y mejores resultados de manera general.

2.2.3. Integración al marco de trabajo MULAN

Luego de discutir los resultados internos y las comparaciones entre los diferente métodos utilizados en este proyecto, se procede a detallar la estructura del paquete, la funcionalidad de sus componentes y la manera en la que se introdujo en el marco de trabajo MULAN.

El nuevo paquete se denomina gradedMLC por la traducción al inglés de Clasificación Multietiqueta Gradual manteniendo el idioma del marco de trabajo MULAN. Cada uno de los métodos utilizados conservaron el nombre dado por sus respectivos autores siendo entonces, Binary Relevance, FrankHall, FullCLR, HorizontalCLR y JoinedCLR las cinco metodologías adiconadas.

Estructura interna del paquete y sus componentes

El nuevo paquete gradedMLC contiene siete directorios bases como se aprecia en la figura 2.7. El nombre de los primeros cinco coinciden con cada uno de los métodos, conteniendo los archivos y clases no comunes entre ellos. El sexto directorio contiene las clases que permiten evaluar los conjuntos de datos con cada uno de los métodos. La séptima carpeta denominada *tools* contiene los elementos comunes entre ellos, como las estructuras de datos para almacenar las instancias y realizar operaciones con ellas, las métricas replanteadas específicamente para ellos, los clasificadores, etc.

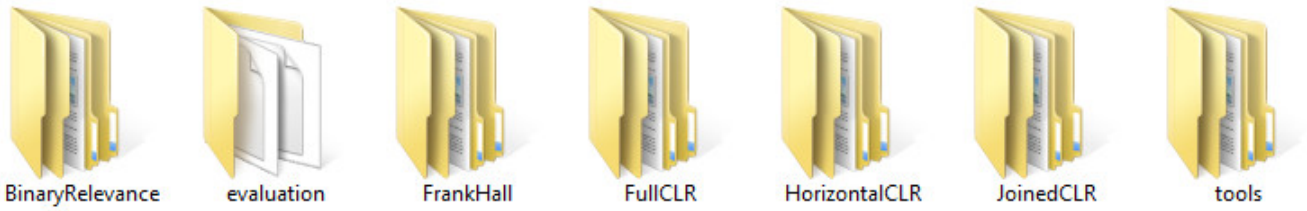

Figura 2.7: Vista desde Windows del contenido del paquete con sus 7 directorios.

A continuación se detalla el contenido de los directorios anteriormente referidos. Cada uno de los directorios correspondientes a los métodos contienen un grupo de clases necesarias para el desarrollo propio del método. Dado que las variaciones específicas entre ellos se encuentran a nivel del código cada uno contiene clases con igual nombre pero, lógicamente, con sus respectivas diferencias en el código.

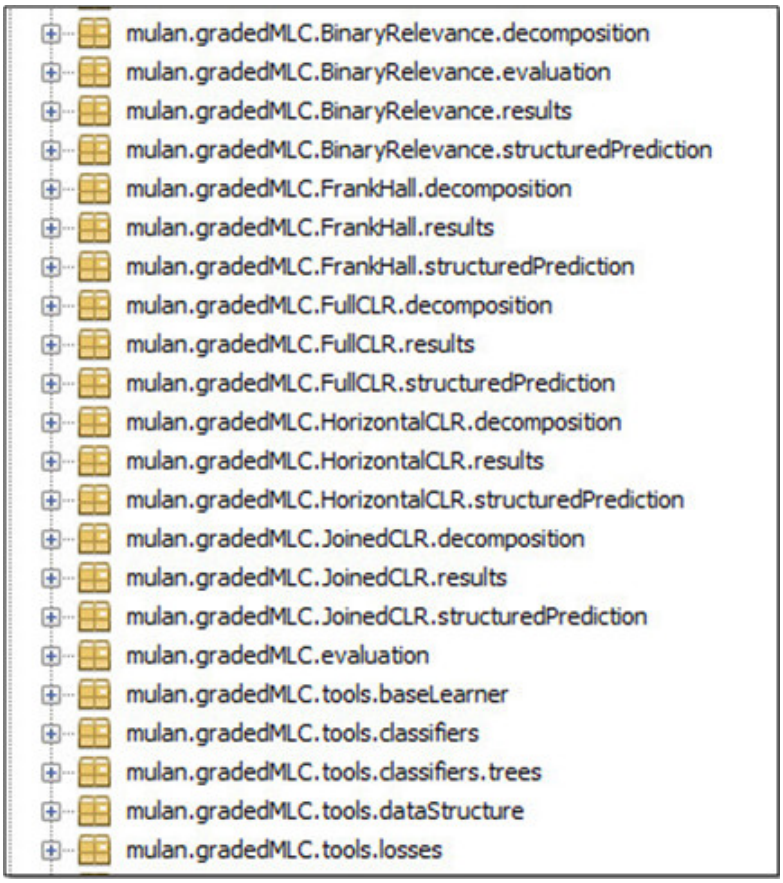

Figura 2.8: Estructura del paquete gradedMLC.

Directorios con elementos no comunes

Básicamente los elementos que varían están concentrados en la forma en que se descomponen y analizan los conjuntos de datos, la forma en que se toma conclusiones de ellos y la forma en que

se procesan y muestran los resultados incluyendo las evaluaciones de las métricas. De esta manera, cada uno de los métodos contiene tres subdirectorios con nombres comunes.

- ***decomposition***

El primero de los tres subdirectorios se denomina *decomposition* y contiene un conjunto de clases (figura 2.9) encargadas de realizar la descomposición del conjunto de instancias que se obtuvo del conjunto de datos. La clase *decomposer*.java es la que maneja todo el proceso y se auxilia, en el caso de la Clasificación Multietiqueta Gradual, de la clase GradedMultiLabelDecomposition.java donde se aplica la descomposición por pares y el Ranking de Etiquetas Calibradas en el caso de los métodos CLR. Esta última clase difiere incluso entre los métodos CLR donde, por ejemplo, mientras que el FullCLR no considera la intensidad de las preferencias, (o sea, en el momento de realizar la comparación por pares entre dos etiquetas λ_u y λ_v se realiza de forma binaria de manera que se obtiene 1 para la más preferida y 0 para la menos sin prestar atención a en cuanto más preferida es la escogida y en cuanto menos la discriminada) el JoinedCLR si presta atención a este aspecto. En las diferencias entre los métodos encontradas en las clases de este directorio radican gran parte de los parámetros a medir en el momento de escoger un determinado algoritmo para un determinado conjunto de datos.

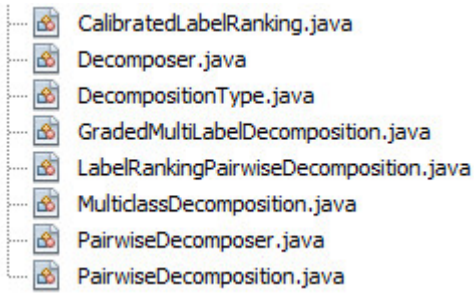

Figura 2.9: Conjunto de clases contenidas en el directorio decomposition

- ***structuredPrediction***

El segundo subdirectorio se denomina structuredPrediction y contiene un conjunto de clases (figura 2.10)que coinciden con diferentes métodos utilizados para convertir las preferencias binarias obtenidas por el método de comparación por pares en un ranking de etiquetas y clasificar las instancias de prueba.

- *Voting:* Interpreta cada preferencia binaria como un voto completo sin pesos(0 ó 1) para la clase preferida. Las etiquetas son rankeadas entonces de acuerdo al número de votos recibidos luego de la evaluación de todos los clasificadores base.

- *WeightedAndUnweightedVoting*

- *WeightedVoting*

En el caso de los métodos de Relevancia Binaria y Frank&Hall solo aceptan el denominado *voting* o votación que se encuentra en la clase homónima. Los métodos CLR tienen disponibles las tres opciones.

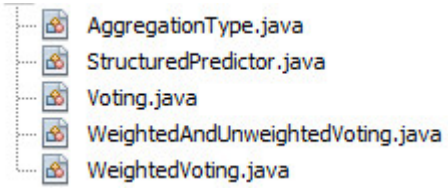

Figura 2.10: Conjunto de clases contenidas en el directorio structuredPrediction

- ***results***

El tercer subdirectorio denominado results está compuesto por un grupo de clases que permiten el procesamiento y manejo de cada uno de los resultados obtenidos y mostrarlos a los usuarios en el panel de salida del NetBeans. Las clases AFoldResult.java y ML_GradedFoldResult.java están orientadas a evaluar los resultados obtenidos con cada una de las métricas compatibles con el método en cuestión. La clase Result.java es una clase esencial dentro de este subdirectorio y ofrece un envoltorio donde se almacenan cada una de las características básicas de la corrida del algoritmo y sus resultados. Finalmente, la clase Configuration.java es, como su nombre lo indica donde se almacena toda la configuración inicial que necesita el algoritmo para comenzar a correr. En esta clase se definen elementos como: la letra que se utilizará de prefijo para nombrar las etiquetas calibradas en la corrida del algoritmo, el tipo de descomposición que se utilizará, el tipo de agregación (*voting, weightedvoting*,...), la ubicación del fichero con el conjunto de datos, el número de repeticiones, la ubicación del clasificador, entre algunas otras opciones.

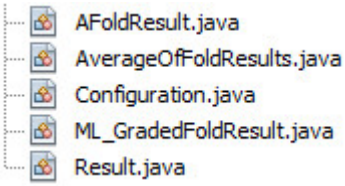

Figura 2.11:

Directorios comunes

- ***evaluation***

El primer directorio común es evaluation donde se encuentran las clases:

1. *EvaluationBinaryRelevance*.java
2. *EvaluationFrankHall*.java

3. *EvaluationHorizontalCLR*.java
4. *EvaluationFullCLR*.java
5. *EvaluationJoinedCLR*.java

correspondientes a cada uno de los métodos explicados al inicio del capítulo. Desde estas clases se comienzan a correr cada uno de los métodos correspondientes respectivamente. Contienen un mínimo de elementos para configurar como la estrategia de agregación a utilizar y las ubicaciones del directorio donde se encuentra\n el\los archivos con el conjunto de datos y del directorio donde creará un archivo texto con los resultados de las corridas, siendo estos son los tres parámetros que demandan estas clases para correr. Las clases también ofrecen la posibilidad de en caso de que existan archivo pertenecientes a variaciones de un mismo conjunto de datos, sean corridos de manera íntegra.

- ***tools***

Este directorio, como indica su nombre, son herramientas necesarias para los algoritmos o elementos comunes dentro de los propios algoritmos. *tools* se encuentra estructurado como se muestra en la figura 2.12.

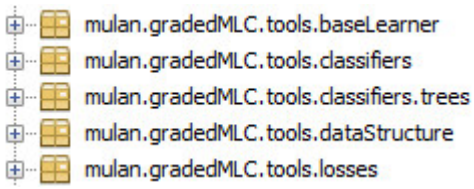

Figura 2.12: Estructura interna del directorio tools

baseLearner

El primer subdirectorio se denomina *baseLearner* y es el encargado de construir y manejar los clasificadores utilizando las instancias obtenidas de descomponer el conjunto de datos con el directorio *decomposition* previamente explicado. En este caso existen tres clases que facilitan la funcionalidad de este directorio, ellas son, primeramente *DecomposedDatasetLearner*.java que es la más directa con las clases de evaluación pues es llamada por estas para preparar las condiciones y configuración para los clasificadores. También se encuentra *BaseLearnerEnvironment*.java donde se prefija el medio a través del cuál se crearán los clasificadores, en el presente caso esto conduce a la tercera clase: *WekaBaseLearner*.java pues es precisamente la encargada de la construcción de los clasificadores base. Esta última clase hereda de *DecomposedDatasetLearner*.java y se auxilia del clasificador de árbol J48.

classifiers

Este subdirectorio contiene un conjunto reducido de clases del paquete de clasificadores del Weka 3.4.7 que permiten junto a las clases del subdirectorio anterior la construcción de los clasificadores base utilizados por los algoritmos.

dataStructure

Como su nombre lo indica este subdirectorio contiene las clases que permiten el almacenamiento y estructuración de las instancias de los conjuntos de datos y son por su importancia y necesidad las más utilizadas en los algoritmos planteados. Dentro de las más importantes se encuentran:

- ExtInstance.java: Es la clase encargada de almacenar los datos de cada una de las instancias de los conjuntos de datos con sus elementos por separado (instancias y clases) siendo una clase básica en los algoritmos.

- ExtInstances.java: Esta clase es donde se almacenan los conjuntos de instancias, específicamente en el atributo **ArrayList<ExtInstance> ext_Instances** que no es más que un arreglo de ExtInstance.

Dada la importancia básica otorgada a estas clases y la incompatibilidad con las instancias de los Weka más recientes que la versión 3.4.7, en este subdirectorio se incluyeron algunas clases reducidas del antiguo Weka que manejaban las instancias como *Instance*.java e *Instances*.java imprescindibles para el normal funcionamiento de los algoritmos.

Además se implementó una clase denominada *DatasetStatistics.java,* que similarmente a la implementada en MULAN muestra algunos datos de interés sobre los conjuntos de datos ya sean características básicas o datos estadísticos que ayudan a un posterior entendimiento de las corridas realizadas de los algoritmos sobre el conjunto de datos en cuestión.

losses

Este subdirectorio contiene un conjunto de clases correspondientes a las métricas planteadas para la evaluación de los resultados de clasificación de los métodos planteados y algunas clases auxiliares. En total se reunieron 12 métricas, algunas basadas en métricas clásicas y otras no. De ellas 4 son aplicables al método Frank&Hall, 11 al *BinaryRelevance* y las 12 a los métodos CLR como se muestra en la tabla 2.1.

Métricas	BinaryRelevance	Frank&Hall	HorizontalCLR	FullCLR	JoinedCLR
VerticalHammingDistance	X	X	X	X	X
OneErrorLoss	X	X	X	X	X
ZeroOneLoss	X	X	X	X	X
CIndex	X	X	X	X	X
PrecisionLoss	X		X	X	X
F1Loss	X		X	X	X
RecallLoss	X		X	X	X
FalloutLoss	X		X	X	X
AccuracyLoss	X		X	X	X
TpRate	X		X	X	X
FpRate	X		X	X	X
OptimalCalibrationHamming			X	X	X

Cuadro 2.1: Métricas compatibles con cada uno de los cinco métodos.

Debido a que para BinaryRelevance la interface *RankLoss*.java, y las métricas CIndex.java y *OneErrorLoss*.java que implementan a la primera contienen variaciones a las variantes utilizadas por el resto de los métodos se decidió cambiar el nombre de las mismas para introducirlan en el presente subdirectorio, siendo finalmente RankLossBR.java, CIndexBR.java y OneErrorLossBR.java.

Las métricas planteadas son utilizadas por las clases del subdirectorio ***result*** que como se explicó anteriormente son las encargadas de aplicarlas a los algoritmos.

2.2.3.1. Clases externas utilizadas por los algoritmos

Para la integración con MULAN como un paquete externo se crearon un conjunto de relaciones entre clases de los métodos y clases externas pertenecientes a la biblioteca Weka utilizada por el marco de trabajo antes mencionado, en su mayoría interfaces. Dentro de ellas se encuentran:

Clase	Tipo
weka.core.SerializedObject	Normal
weka.core.Utils	Normal
weka.core.OptionHandler	Interface
weka.core.Option	Normal
weka.core.Drawable	Interface
weka.core.Range	Normal
weka.core.Summarizable	Interface
weka.classifiers.Sourcable	Interface
weka.core.Matchable	Interface
weka.core.WeightedInstancesHandler	Interface
weka.core.ProtectedProperties	Normal
weka.core.SerializedObject	Normal
weka.core.UnassignedClassException	Excepción
weka.core.Copyable	Interface
weka.core.UnassignedDatasetException	Excepción
weka.core.matrix.*	Paquete
weka.core.AdditionalMeasureProducer	Interface

Figura 2.13:

Capítulo 3

Evaluación de la plataforma extendida

En este capítulo se detalla la estructura seguida por los ficheros de tipo .earff para facilitar la creación de nuevos ficheros. Se describen cada uno de los conjuntos de datos utilizados. Se mencionan las doce métricas existentes para los métodos explicando las más generales. Se muestran los resultados de la validación realizada a la integración de los cinco métodos a través de la comparación de los resultados antes y después del proceso y además se incluyen las salidas brindadas por los diferentes métodos utilizados. Se describe la estructura interna del paquete y finalmente se explica a grandes rasgos como insertar un nuevo clasificador y las ventajas de este nuevo paquete para MULAN..

3.1. Conjuntos de Datos Multietiqueta

En el presente proyecto se utilizaron cuatro conjuntos de datos estructurados de una manera tal que permitieran la experimentación con Clasificación Multietiqueta Gradual. A continuación se explica la estructura seguida en los archivos, una breve reseña del tópico de cada uno y un resumen con sus principales características.

3.1.1. Estructura de los archivos

Para la realización de este proyecto se utilizaron archivos con extensión .earff que no es más que una variación de la extensión .arff (utilizada tradicionalmente en el software MULAN para la Clasificación Multietiqueta simple) para permitir la presencia de información sobre los grados de las etiquetas. Para las especificaciones de los atributos de las etiquetas:

@ATTRIBUTE class [ML_Graded|(*labelA*),(*labelB*)(,...) × 0<1<2<(*...maxGrade*)|2] donde (*label*) denota el nombre de la etiqueta y (*maxGrade*) es el grado más alto posible y todo lo que no esta entre () es fijo en todo los conjuntos de datos. En la sección de @data, se utilizó el siguiente formato:

value1, value2 ... lastvalue, {labelA0,labelB0<labelC1<labelD2...}

suponiendo que *labelA* y labelB *tienen* grado 0, *labelC* tiene grado 1, *labelD* tiene grado 2, etc. En general, la información de la etiqueta es adicionada al final de la información de la instancia siguiendo el formato siguiente:

{index1 value1, index 2 value2, ...}, {*labelA0,labelB0*<l*abelC1*<*labelD2*...}

3.1.2. Conjuntos de datos

Se dará un breve resumen de los conjuntos de datos que se utilizaron para la experimentación (3.1):

BeLaE: Este conjunto de datos consiste en 1930 instancias que representan a un estudiante graduado cada una. Cada instancia tiene 50 atributos. Dos atributos, edad y sexo, caracterizan al estudiante, el resto de los 48 atributos representan cada uno la respuesta dada por el estudiante a una pregunta realizada al mismo, sobre preferencias para definir su futuro trabajo. Cada una de estas respuestas tiene un grado de '1'(sin ninguna importancia) a '5'(muy importante).

En vista a la ausencia de una caracterización más comprensiva e informativa de los estudiantes, Cheng, decidió utilizar un subconjunto de las respuestas como atributos adicionales en la caracterización de los estudiantes. Siguiendo esta idea se generaron 50 conjuntos de datos escogiendo aleatoriamente un subconjunto de n preguntas como etiquetas objetivo. El resto de los $50-n$ atributos fueron usados como características de las instancias. Se utilizaron $n = 5$ y $n = 10$. Dando lugar a los conjuntos de datos: BeLaE_m5 y BeLaE_m10.

movies: Se tomó un conjunto de datos de una guía de programas de TV denominada *TVSpielfilm.de* que clasificaba películas asignándoles grados a las categorías 'fun', 'action', 'sex', 'suspense' y 'sophistication' en lugar de dar una calificación general. Cada categoría tiene grados desde '0' a '3'. En total, contiene datos de 1967 películas. Se agregaron el título en inglés, el año, el nombre de director, los nombres de actores, los nombres de los personajes, los nombres de los escritores, el tiempo de duración, el país de origen, y el idioma.

medical: El conjunto de datos 'medical' consiste en 1953 informes de radiología de texto libres. Fueron reunidos en el 2007 por el Centro del Idioma Natural Médico del CMC donde se les pidió a tres compañías especializadas dieran su clasificación a través de sus códigos de clasificación enfermedad/diagnóstico. Se generó un conjunto de datos de GMLC considerando el nivel de acuerdo como la calidad de asignación.

Conj. Datos	Inst.	Atrib.	Grad.	Etiq.	Avg. Grad.	i=1	i=2	i=3	i=4	i=5
BeLA-E n=5	1930	45	5	5	2.50	7.95	13.04	23.89	31.43	23.69
BeLA-E n=10	1930	40	5	10	2.50	7.95	13.04	23.89	31.43	23.69
Movies	1967	27002	4	5	0.72	50.26	31.13	15.18	3.43	-
Medical	103	1602	4	10	0.03	90.23	3.30	6.10	0.37	-

Cuadro 3.1: Características de los conjuntos de datos

3.2. MÉTRICAS PARA LA DESCRIPCIÓN DE LOS CONJUNTOS DE DATOS

A la hora de plantear un desarrollo experimental sobre un modelo que genere clasificadores multietiqueta, así como de discutir los resultados obtenidos por éste, es importante determinar cómo de multietiqueta es el conjunto de datos con el que se valida. Para ello se han propuesto varias métricas.

La métrica más sencilla es el denominado *distinct* [49], el cual es el número de combinaciones de distintas etiquetas presentes en un conjunto de datos. Además del número de combinaciones, se han propuesto otras dos medidas, la cardinalidad y la densidad [49]. Si consideramos $|Y_i|$ el número de etiquetas de la instancia i, la cardinalidad de un conjunto de datos D viene dada por el promedio de etiquetas asociadas a cada instancia.

$$cardinalidad(D) = \frac{1}{m}\sum_{i=0}^{m}|Y_i| \tag{3.1}$$

Sin embargo intuitivamente parece razonable considerar también el número total de etiquetas posible. Por ejemplo podemos tener dos conjuntos de cardinalidades similares, pero puede ocurrir que el número de posibles etiquetas máximo que pueda tener asignado una instancia varíe enormemente. Para evitar la distorsión que supone comparar cardinalidades entre conjuntos con diversos número de etiquetas se define la densidad como la cardinalidad entre el número total de etiquetas $|L|$.

$$densidad\,(D) = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i|}{|L|} \tag{3.2}$$

3.2.1. MÉTRICAS PARA DATOS GRADED MULTIETIQUETA

Para los problemas de Clasificación Multietiqueta Gradual se generalizaron algunas métricas comunes a la Clasificación Multietiqueta. Todas las métricas son computadas individualmente en las instancias y promediadas antes en el conjunto de prueba. A continuación se explican algunas de las métricas utilizadas:

3.2.1.1. *Hamming Loss*

En la Clasificación Multietiqueta Gradual *Hamming Loss* puede ser generalizado para medir la perdida real en las subtareas de las reducciones tanto horizontal como vertical. Cheng mostró que en ambas funciones es igualmente aplicable esta métrica. Por simplicación *Hamming loss* es más usado en la versión vertical:

$$HammLoss(\hat{y}_x, y_x) = \frac{\sum_{i=1}^{n} AE(\hat{y}_x^i, y_x^i)}{(m-1)*n} \tag{3.3}$$

con $AE : MxM \rightarrow N, AE(\mu_i, \mu_j) = |i-j|$. Así, *Hamming Loss* en GMLC denota la desviación media de los grados predichos para la etiqueta con los que realmente posee. De cualquier manera el uso de distancia entre grados para la evaluación de predicciones ordinales es cuestionable desde un punto de vista teórico, sin embargo es un útil indicador de la efectividad de clasificación en esta técnica.

3.2.1.2. *Vertical 0-1 Loss*

Esta métrica mide el porcentaje de etiquetas con grados incorrectamente asignados. Contrariamente a la *Hamming Loss,* no considera el tamaño de la diferencia de grados.

$$Vert01(\hat{y}_x, y_x) = \frac{1}{n} \sum_{i=1}^{n} \Pi(\hat{y}_x^i \neq y_x^i) \tag{3.4}$$

3.2.1.3. *C-Index*

La métrica *C-Index* es una generalización de *Rank Loss*. Llevada al caso gradual, esta métrica mide el error del *ranking* por pares entre un par de etiquetas de dos conjuntos diferentes: $P_x^{'i}, P_x^{'j}, i < j$. Esencialmente, cuenta el número de pares de etiquetas incorrectamente ordenados con diferente grado en el *ranking*.

$$C - Index(h_x, P_x^{'1}, ..., P_x^{'m}) = \frac{\sum_{\mu_i < \mu_j} \sum_{(\lambda, \lambda') \in P_x^{'i} \times P_x^{'j}} S(h_x(\lambda), h_x(\lambda'))}{\sum_{\mu_i < \mu_j} |P_x^{'i} \times P_x^{'j}|} \tag{3.5}$$

3.2.1.4. *One Error Rank Loss*

Esta métrica es una generalización de *One Error Loss* utilizada para *rankings* en la clasificación multietiqueta. Cheng la utiliza para medir si la etiqueta más alta en el *ranking* tiene el grado más alto posible μ_m. El inconveniente de esta versión es que si una instancia fuera del conjunto de prueba no tiene una etiqueta con una relevancia del grado más alto posible, *One Error* no puede ser cero, aun cuando la clasificación de la instancia está completamente correcta. Para resolver este problema se propuso una versión cambiada denominada O*ne Error Rank Loss* que compara el grado real de la etiqueta más altamente rankeada con el grado más alto de todas las etiquetas de una instancia.

$$OneErr(\hat{y}_x^i, y_x^i) = \frac{1}{m-1} AE(\max_{1 \leq i \leq n} y_x^i, \max_{1 \leq j \leq n} y_x^j) \tag{3.6}$$

3.2.1.5. *Optimistic Hamming Loss*

Bajo algunas circunstancias, CLR tiende a estimar por debajo o por encima de la posición correcta de la etiqueta virtual. Para solucionar esto se propuso evaluar el funcionamiento del *ranking* trucando la posición de la etiqueta virtual: se posicionó el punto de corte luego de realizado el método para que la distribución de los grados fuera la correcta. De cierta manera esto permite computar métricas de biparticionado aun cuando el algoritmo subyacente solo puede predecir *rankings*.

Se generalizó este método a GMLC y al caso de *ranking* multipartito. Así pues, se define el particionado trucado $P_x^{''1}, P_x^{''2}, ...$sobre un *ranking* como $|P_x^{''i}| = |P_x^{'i}|$ y $s_x(\lambda_u) \leq s_x(\lambda_v)$ si $\lambda_u \in P_x^{'i}, \lambda_v \in P_x^{'j}, \mu_i \prec \mu_j$. Dada la correspondiente predicción(en la forma de $\hat{y}^{''}$), entonces se obtiene el *Optimistic Hamming Loss* de la siguiente manera:

$$OptHammLoss = HammLoss(y_x^{''}, y_x) \tag{3.7}$$

3.3. Validación

A pesar de su gran importancia y su extensa aplicabilidad en la actualidad cada vez más repleta de datos sin procesamiento inteligente, la Clasificación Multietiqueta Gradual es un tema extremadamente poco analizado contando con muy escasa bibliografía (poco más de tres artículos) y muy pocas soluciones o alternativas. Debido a esto, su estudio se encuentra limitado y en el caso de estos métodos solo pueden ser comparados entre sí pues no existen otros que ofrezcan valores comparativos mejores. Para la verificación se compararon los valores arrojados por las métricas de los métodos individuales y los valores de la versión unificada de todos los métodos integrados entre sí, ya dentro de MULAN. Ambos obtuvieron idénticos resultados en cada uno de los métodos y para cada uno de los conjuntos de datos utilizados. (En el caso de los algoritmos que utilizan CLR la métrica **OptimalCalibrationHamming** se sustituyó por sus siglas **OCH** por cuestiones de espacio)

Métricas\Conj.Datos	**BR**		**FH**	
	Indep.	*Integ.*	*Indep.*	*Integ.*
VerticalHammingDistance	0.6454	0.6454	0.1800	0.1800
OneErrorLoss	0.2770	0.2770	0.1862	0.1862
ZeroOneLoss	0.9493	0.9493	0.5561	0.5561
CIndex	0.4989	0.4989	0.3049	0.3049
PrecisionLoss	0.0506	0.0506	-	-
F1Loss	0.0506	0.0506	-	-
RecallLoss	0.0506	0.0506	-	-
FalloutLoss	0.9493	0.9493	-	-
AccuracyLoss	0.9620	0.9620	-	-
TpRate	0.0506	0.0506	-	-
FpRate	0.0193	0.0193	-	-
OCH	-	-	-	-

Figura 3.1: Resultados de las corridas de los dos primeros métodos métodos sobre **BeLaE_m10.**

Métricas\Conj.Datos	**FCLR**		**HCLR**		**JCLR**	
	Indep.	*Integ.*	*Indep.*	*Integ.*	Indep.	Integ.
VerticalHammingDistance	0.3351	0.3351	0.1724	0.1724	0.2726	0.2726
OneErrorLoss	0.1051	0.1051	0.0928	0.0928	0.1608	0.1608
ZeroOneLoss	0.7697	0.7697	0.5912	0.5912	0.6619	0.6619
CIndex	0.2225	0.2225	0.2087	0.2087	0.2399	0.2399
PrecisionLoss	0.2302	0.2302	0.4087	0.4087	0.3379	0.3379
F1Loss	0.2302	0.2302	0.4087	0.4087	0.3379	0.3379
RecallLoss	0.2302	0.2302	0.4087	0.4087	0.3379	0.3379
FalloutLoss	0.7697	0.7697	0.5912	0.5912	0.6619	0.6619
AccuracyLoss	0.9692	0.9692	0.9763	0.9763	0.9338	0.9338
TpRate	0.2302	0.2302	0.4087	0.4087	0.3379	0.3379
FpRate	0.0157	0.0157	0.0120	0.0120	0.0348	0.0348
OCH	0.2046	0.2046	0.0473	0.0473	0.0366	0.0366

Cuadro 3.2: Resultados de las corridas de los métodos CLR sobre **BeLaE_m10.**

3.4. Resultados mostrados por el marco de trabajo

Clases Evaluation

Cada una de las clases principales del marco de trabajo devuelve un conjunto de informaciones relacionadas con la tarea específica que realizan. Existen seis clases principales en el directorio *evaluation,* cinco de ellas se encargan de estructurar y correr cada uno de los cinco algoritmos presentados. La última de las seis utiliza la clase *DatasetStatistic*.java en el subdirectorio dataStructure para mostrar una determinada cantidad de datos de interés acerca del conjunto de datos sobre el que correrán los algoritmos.

Las primeras mencionadas se estructuran de la misma manera teniendo como diferencias el funcionamiento de sus componentes (figura 3.2). Primeramente, el main se encarga de recibir la ubicación donde se encuentra el fichero .earff y la dirección donde se escribirá la salida, para esto se auxilia de la clase BufferedReader del paquete de Entrada/Salida de Java con la lee el contenido del fichero .earff almacenándolo en un Reader que luego pasa como parámetro al constructor de la clase interna ExtInstances del paquete en cuestión Algorítmo 3.1. Con el constructor de esta clase se crea un objeto donde se almacenan cada uno de los campos del archivo por separado, incluyendo Atributos, Clases, Instancias, Grados, etc.

Algoritmo 3.1 Segmento de código correspondiente a la lectura del fichero .earff y la creación del objeto data de tipo ExtInstances donde se estructuran y almacenan las instancias.

```
for(int i = 0; i < 1; i++)
Configuration.positionOfEARFFFile = inputFilePath;
System.out.println(".Earff: -(i+1));
try
Reader r = null;
r = new BufferedReader(new FileReader(Configuration.positionOfEARFFFile));
ExtInstances data = null;
data = new ExtInstances(r);
(...)
```

A continuación se crea una instancia de la clase *Result* que primeramente contiene el conjunto de instancias y todas sus características almacenadas en el objeto de la clase ExtInstances que se le pasó como parámetro a su constructor, además en este objeto de tipo Result se adicionan como atributos todas las características y parámetros que se utilizarán para correr los algoritmos. Estos campos son llenados con la ayuda de la clase *Configuration.java* de la que se extraen los parámetros iniciales de la corrida como se muestra en el Algoritmo 3.2.

Algoritmo 3.2 Segmento correspondiente a la creación del objeto *Result* y la asignación de los campos extraidos de *Configuration*

```
Result results = new Result(data);
results.setSymmetric(Configuration.isSymmetricClassifier);
results.setDecompositionType(Configuration.decompositionType);
results.setAggregationType(Configuration.aggregationType);
results.setBaseLearnerEnvironment(Configuration.baseLearnerEnvironment);
results.setClassifier(Configuration.positionOfClassifier);
results.setRandom(Configuration.seedForEvaluation);
results.setFolds(Configuration.numberOfFolds);
EvaluationBinaryRelevance eval = new EvaluationBinaryRelevance(results);
eval.crossValidateModel();
```

Una vez realizado lo anterior se procede a crear una instancia de la propia clase cuyo constructor solo necesita una variable de tipo *Result* donde se almacenarán, como bien supone su nombre,

los resultados de la corrida y a través del método *.toString()* se organizan los resultados para su devolución al usuario donde se muestran los valores obtenidos por las métricas y los valores predichos para las instancias de prueba. Finalmente se llama al método *crossValidateModel()* donde se aplica la Validación Cruzada al modelo propuesto.

- Validación Cruzada

Luego de procesar y almacenar las instancias y configurar el modelo, se procede a aplicar la Validación Cruzada con el método *crossValidateModel()* implementado dentro de la misma clase de evaluación. Este método, primeramente se encarga de dividir las instancias que recibió en un conjunto de entrenamiento y uno de prueba. Para esto se auxilia de los métodos **trainCV** y **testCV** pertenecientes a la clase *ExtInstances* que ubican de forma ordenada y diferente cada una de las instancias en los dos subconjuntos prestando especial atención al número de **Folds** establecido por el usuario (por defecto: 10). Luego de obtener el conjunto de entrenamiento y de prueba una instancia de la clase *Decomposer* perteneciente al directorio *Decomposition* es encargada de descomponer el conjunto de prueba realizando la comparación por pares con la técnica CLR. Otra clase de este mismo directorio, DecomposedDatasetLearner, con ayuda de Weka, se encarga de construir el modelo y de aprender los clasificadores con los que se obtendrán las predicciones de las instancias de prueba. Finalmente, tras aplicar los clasificadores sobre el conjunto de prueba, con las clases del directorio *structuredPrediction*, especialmente de *StructuredPredictor.java* se procede a ordenar los resultados y guardarlos en la variable mencionada inicialmente bajo el nombre de *results.* Una vez terminado el proceso se procede a devolver los valores resultantes extrayendolos de la variable *results* y mostrándolos en pantalla a través del panel de resultados de NetBeans.

Algoritmo 3.3

```
for (int fold = 0; fold < results.getFolds();fold++)
long time = System.currentTimeMillis();
// decompose train set
Decomposer decomposition = Decomposer.forName(results.getDecompositionType()); decomposition.setSymmetric(results.isSymmetric());
ExtInstances trainingSet = data.trainCV(results.getFolds(), fold);
decomposition.buildDecomposedDatasets(trainingSet);
// build the base classifiers for decomposed datasets
DecomposedDatasetLearner models = DecomposedDatasetLearner.forName(results.getEnvironment());
models.setDecomposedDatasets(decomposition.getDecomposedDatasets());
models.buildClassifiers(results.getClassifierName());
models.setClassToNumber(decomposition.getClassToNumber());
// decompose test set
Instances test = data.testCV(results.getFolds(), fold, decomposition.getClassToNumber().keySet());
// Structured Prediction with base classifiers
StructuredPredictor pred = StructuredPredictor.forName(results); pred.classifyInstances(test, models, trainingSet); (...)
```

En la figura 3.2 se observa gráficamente de una manera muy general el funcionamiento de las clases de evaluación.

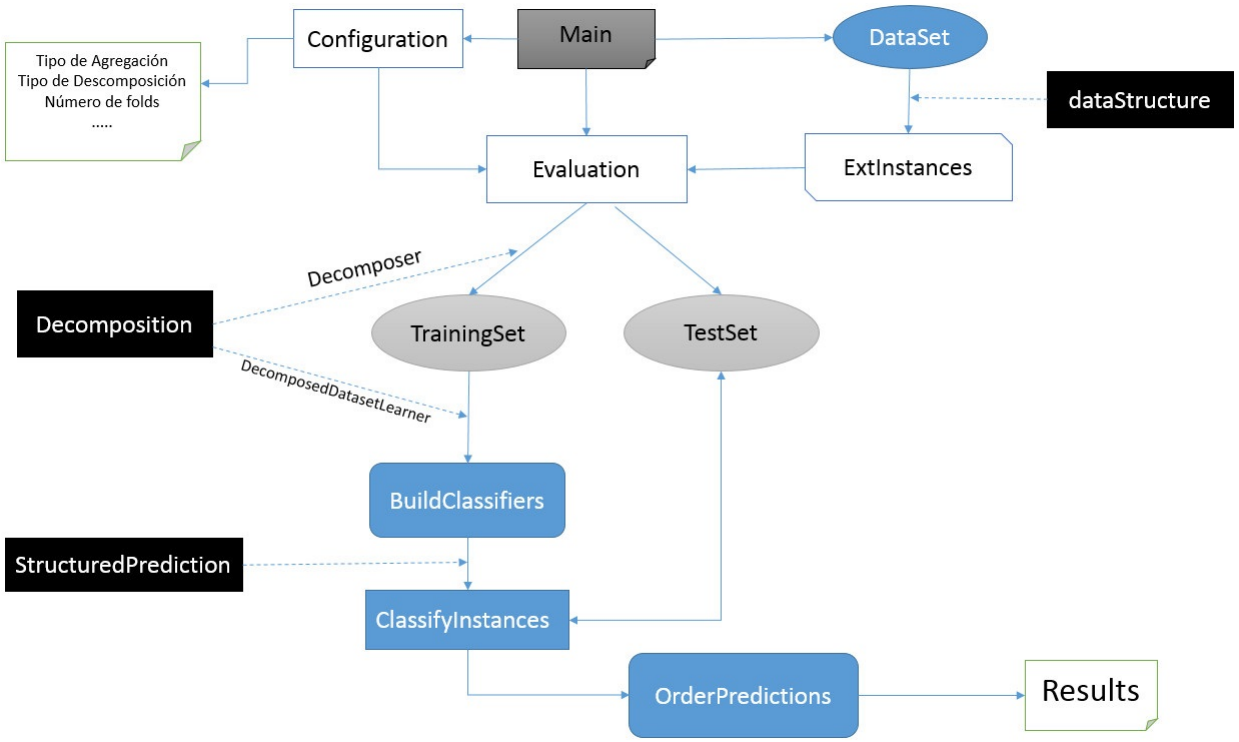

Figura 3.2: Funcionamiento básico de las clases Evaluation de cada uno de los métodos

A continuación se muestra la salida obtenida tras correr el algoritmo BinaryRelevance sobre el conjunto de datos BeLaE_m10 con 10 *folds*. Sin embargo, por cuestiones de tamaño solo se mostraran los *folds* 1,2 y 10 dejando solamente un pequeño conjunto de la etiquetas y grados predichos. Las salidas de todos los algoritmos se encuentran estructuradas de la misma manera teniendo solo como diferencias los valores obtenidos.

Evaluation Date : Sun Jun 07 22:32:58 EDT 2015
DecompositionType : PairwiseDecomposer
AggregationType : Voting
BaseLearnerEnvironment : WekaBaseLearner Classifier : mulan.gradedMLC.tools.classifiers.trees.J48 Classification
Type : ML_Graded Symetric : true
Number of Folds : 10
Number of Instances : 1930

=== Fold 1 ==========
Fold Computation Time: 22564 ms
=== Losses: ==========
AccuracyLoss = 0.9625906735751295
TpRate = 0.06476683937823834
FpRate = 0.019086391033097175
RecallLoss = 0.06476683937823834
PrecisionLoss = 0.06476683937823834
F1Loss = 0.06476683937823834
FalloutLoss = 0.9352331606217616
ZeroOneLoss = 0.9352331606217616
VerticalHammingDistanceLo = 0.6290155440414508
CIndexBR = 0.49740932642487046
OneErrorLossBR = 0.27849740932642486
relevant Labels:
[VERSTCHE1, NOTHELFE2, PRESTIGE2, FAEHIGEN3, UNTERNEH3, MIVERBES3, UMENSEIN3, LEISTUNG3, AZEITSEL4, MENLEIT4]
[PRESTIGE0, MENLEIT0, NOTHELFE1, LEISTUNG1, VERSTCHE2, FAEHIGEN3, MIVERBES3, UMENSEIN3, UNTERNEH4, AZEITSEL4]
[NOTHELFE0, PRESTIGE0, UNTERNEH3, VERSTCHE3, AZEITSEL3, MENLEIT3, LEISTUNG3, FAEHIGEN4, MIVERBES4, UMENSEIN4]
[AZEITSEL1, NOTHELFE2, MIVERBES2, VERSTCHE2, PRESTIGE2, MENLEIT2, LEISTUNG2, FAEHIGEN3, UNTERNEH3, UMENSEIN3]

[UMENSEIN1, MENLEIT1, NOTHELFE2, UNTERNEH2, PRESTIGE2, FAEHIGEN3, MIVERBES3, VERSTCHE3, AZEITSEL3, LEISTUNG3]

[PRESTIGE0, UNTERNEH1, NOTHELFE2, AZEITSEL2, MIVERBES3, UMENSEIN3, VERSTCHE3, MENLEIT3, FAEHIGEN4, LEISTUNG4]

[VERSTCHE1, UNTERNEH2, PRESTIGE2, NOTHELFE3, FAEHIGEN3, MIVERBES3, AZEITSEL3, MENLEIT3, LEISTUNG3, UMENSEIN4]

[AZEITSEL1, NOTHELFE3, UNTERNEH3, UMENSEIN3, VERSTCHE3, PRESTIGE3, FAEHIGEN4, MIVERBES4, MENLEIT4, LEISTUNG4]

[PRESTIGE1, NOTHELFE2, UNTERNEH2, UMENSEIN2, AZEITSEL2, MENLEIT2, FAEHIGEN3, MIVERBES3, VERSTCHE3, LEISTUNG3]

(.....)

=== Micro Averaged Confusion Matrix ===

a_____ b actual class

92765 1805 | a = negative

1805 125 | b = positive

=== Fold 2 ==========

Fold Computation Time: 21923 ms

=== Losses: ==========

AccuracyLoss = 0.9620932642487047

TpRate = 0.05233160621761658

FpRate = 0.019340171301681294

RecallLoss = 0.05233160621761658

PrecisionLoss = 0.05233160621761658

F1Loss = 0.05233160621761658

FalloutLoss = 0.9476683937823834

ZeroOneLoss = 0.9476683937823834

VerticalHammingDistanceLo = 0.6433937823834197

CIndexBR = 0.5 OneErrorLossBR = 0.2927461139896373

relevant Labels:

[NOTHELFE0, UNTERNEH1, VERSTCHE3, PRESTIGE3, FAEHIGEN4, MIVERBES4, UMENSEIN4, AZEITSEL4, MENLEIT4, LEISTUNG4]

[NOTHELFE2, PRESTIGE2, MENLEIT2, LEISTUNG2, FAEHIGEN3, UNTERNEH3, MIVERBES3, UMENSEIN3, VERSTCHE3, AZEITSEL4]

[MIVERBES1, PRESTIGE1, NOTHELFE2, VERSTCHE2, AZEITSEL2, MENLEIT2, LEISTUNG2, FAEHIGEN3, UNTERNEH3, UMENSEIN3]

[UMENSEIN2, NOTHELFE3, UNTERNEH3, VERSTCHE3, MENLEIT3, LEISTUNG3, FAEHIGEN4, MIVERBES4, PRESTIGE4, AZEITSEL4]

[PRESTIGE1, MENLEIT1, MIVERBES3, UMENSEIN3, VERSTCHE3, AZEITSEL3, LEISTUNG3, NOTHELFE4, FAEHIGEN4, UNTERNEH4]

[UNTERNEH2, PRESTIGE2, NOTHELFE3, FAEHIGEN3, MIVERBES3, AZEITSEL3, MENLEIT3, LEISTUNG3, UMENSEIN4, VERSTCHE4]

[PRESTIGE1, MENLEIT1, NOTHELFE2, UMENSEIN2, AZEITSEL2, FAEHIGEN3, UNTERNEH3, VERSTCHE3, LEISTUNG3, MIVERBES4]

[PRESTIGE2, MENLEIT2, NOTHELFE4, FAEHIGEN4, UNTERNEH4, MIVERBES4, UMENSEIN4, VERSTCHE4, AZEITSEL4, LEISTUNG4]

[MIVERBES0, PRESTIGE1, FAEHIGEN2, LEISTUNG2, NOTHELFE3, VERSTCHE3, AZEITSEL3, MENLEIT3, UNTERNEH4, UMENSEIN4]

[AZEITSEL2, MENLEIT2, FAEHIGEN3, UNTERNEH3, MIVERBES3, UMENSEIN3, VERSTCHE3, PRESTIGE3, NOTHELFE4, LEISTUNG4]

[AZEITSEL1, NOTHELFE2, UNTERNEH2, UMENSEIN2, VERSTCHE2, PRESTIGE2, MIVERBES3, MENLEIT3, LEISTUNG3, FAEHIGEN4]

[LEISTUNG2, NOTHELFE3, FAEHIGEN3, UNTERNEH3, MIVERBES3, UMENSEIN3, VERSTCHE3, PRESTIGE3, AZEITSEL3, MENLEIT3]

[PRESTIGE2, MENLEIT2, NOTHELFE3, FAEHIGEN3, UNTERNEH3, MIVERBES3, UMENSEIN3, LEISTUNG3, VERSTCHE4, AZEITSEL4]

[PRESTIGE0, AZEITSEL0, UNTERNEH1, UMENSEIN1, MENLEIT1, NOTHELFE2, MIVERBES2, VERSTCHE2, FAEHIGEN3, LEISTUNG3]

(...)

=== Micro Averaged Confusion Matrix ===

a _____b actual class

92741 1829 | a = negative

1829 101 | b = positive

=== Fold 3 ==========

(...)

=== Fold 10 ==========

Fold Computation Time: 21493 ms

=== Losses: ==========

AccuracyLoss = 0.961699481865285

TpRate = 0.04248704663212435

FpRate = 0.019541080680977055

RecallLoss = 0.04248704663212435

PrecisionLoss = 0.04248704663212435

F1Loss = 0.04248704663212435

FalloutLoss = 0.9575129533678757

ZeroOneLoss = 0.9575129533678757

VerticalHammingDistanceLo = 0.6507772020725389

CIndexBR = 0.49740932642487046

OneErrorLossBR = 0.2694300518134715

relevant Labels:

[UMENSEIN2, PRESTIGE2, LEISTUNG2, NOTHELFE3, FAEHIGEN3, UNTERNEH3, MIVERBES3, VERSTCHE4, AZEITSEL4, MENLEIT4]

[NOTHELFE1, UNTERNEH2, MIVERBES2, UMENSEIN2, VERSTCHE2, MENLEIT2, FAEHIGEN3, PRESTIGE3, AZEITSEL3, LEISTUNG3]

[UNTERNEH1, VERSTCHE1, PRESTIGE2, AZEITSEL2, MENLEIT2, NOTHELFE3, FAEHIGEN3, MIVERBES3, UMENSEIN3, LEISTUNG3]

[NOTHELFE1, AZEITSEL1, PRESTIGE2, FAEHIGEN3, UNTERNEH3, UMENSEIN3, VERSTCHE3, MENLEIT3, MIVERBES4, LEISTUNG4]

[PRESTIGE1, NOTHELFE2, MIVERBES2, AZEITSEL2, MENLEIT2, FAEHIGEN3, UNTERNEH3, VERSTCHE3, LEISTUNG3, UMENSEIN4]

[PRESTIGE0, MIVERBES1, AZEITSEL1, MENLEIT2, NOTHELFE3, UNTERNEH3, UMENSEIN3, VERSTCHE3, FAEHIGEN4, LEISTUNG4]

[PRESTIGE2, NOTHELFE3, FAEHIGEN3, UNTERNEH3, MIVERBES3, UMENSEIN3, MENLEIT3, LEISTUNG3, VERSTCHE4, AZEITSEL4]

[NOTHELFE2, PRESTIGE2, FAEHIGEN3, MIVERBES3, AZEITSEL3, MENLEIT3, LEISTUNG3, UNTERNEH4, UMENSEIN4, VERSTCHE4]

[PRESTIGE0, NOTHELFE1, UNTERNEH2, MIVERBES2, VERSTCHE2, AZEITSEL2, UMENSEIN3, MENLEIT3, LEISTUNG3, FAEHIGEN4]

[NOTHELFE1, PRESTIGE1, AZEITSEL2, FAEHIGEN3, UNTERNEH3, MIVERBES3, UMENSEIN3, VERSTCHE3, LEISTUNG3, MENLEIT4]

[PRESTIGE1, MIVERBES2, LEISTUNG2, NOTHELFE3, UNTERNEH3, MENLEIT3, FAEHIGEN4, UMENSEIN4, VERSTCHE4, AZEITSEL4]

[PRESTIGE1, LEISTUNG2, NOTHELFE3, UNTERNEH3, MIVERBES3, VERSTCHE3, AZEITSEL3, MENLEIT3, FAEHIGEN4, UMENSEIN4]

[MIVERBES1, PRESTIGE1, LEISTUNG1, NOTHELFE2, UNTERNEH2, UMENSEIN2, AZEITSEL2, MENLEIT2, FAEHIGEN4, VERSTCHE4]

[UMENSEIN2, PRESTIGE2, AZEITSEL2, MENLEIT2, NOTHELFE3, FAEHIGEN3, UNTERNEH3, MIVERBES3, VERSTCHE3, LEISTUNG3]

[MENLEIT1, LEISTUNG2, NOTHELFE3, FAEHIGEN3, UMENSEIN3, VERSTCHE3, PRESTIGE3, AZEITSEL3, UNTERNEH4, MIVERBES4]

[PRESTIGE1, MIVERBES2, VERSTCHE2, MENLEIT2, NOTHELFE3, AZEITSEL3, LEISTUNG3, FAEHIGEN4, UNTERNEH4, UMENSEIN4]

[PRESTIGE0, LEISTUNG0, NOTHELFE2, MENLEIT2, MIVERBES3, UMENSEIN3, FAEHIGEN4, UNTERNEH4, VERSTCHE4, AZEITSEL4]

(...)

=== Micro Averaged Confusion Matrix ===
a b actual class
92722 1848 | a = negative
1848 82 | b = positive

=== Total Macro Average =======
PrecisionLoss = 0.050673575129533674 RecallLoss = 0.050673575129533674
AccuracyLoss = 0.9620269430051813 F1Loss = 0.050673575129533674
FpRate = 0.019374008670825844
VerticalHammingDistanceLo = 0.6454015544041452
FalloutLoss = 0.9493264248704663
ZeroOneLoss = 0.9493264248704663
CIndexBR = 0.4989637305699482
TpRate = 0.050673575129533674
OneErrorLossBR = 0.27707253886010363
Total Computation Time: 215934.0 ms

Como se observa a simple vista al comienzo se ofrece información sobre la configuración de algunos parámetros a tomar en cuenta en la corrida del algoritmo y otros como el tiempo y la fecha de corrida con objetivos organizativos para el momento de manipular los ficheros con los resultados obtenidos. Luego comienzan a mostrarse los *Folds* desde el 1 hasta el número preestablecido en la clase *Configuration*.java. Dentro de cada *fold,* primero se muestra el tiempo de duración y luego el valor obtenido en cada una de las métricas (11 en el ejemplo por ser *Binary Relevance*). Finalmente son mostradas el ranking de etiquetas de las instancias de prueba y los respectivos grados asignados a cada una de las etiquetas por los clasificadores entrenados. Tras la información específica de los *folds* se agregan los valores generales de las métricas obtenidos de la unión de todos los resultados de los *folds*.

Clase EstimationOfStatistics

Ésta clase orientada a ofrecer información básica y estadística del conjunto de datos pasado por parámetros también arroja resultados. Estos pueden ayudar a un mejor entendimiento de la información obtenida en los algoritmos. A continuación se muestra la salida obtenida tras correr la clase usando el conjunto de datos BeLaE_m10:

Statistics of Current Dataset
Relation Name: BeLa-E
Dataset Type: ML_Graded
Number of Classes: 10

Number of Grades: 5

Classes and Grades: [LEISTUNG, MENLEIT, AZEITSEL, PRESTIGE, VERSTCHE, UMENSEIN, MIVERBES, UNTERNEH, FAEHIGEN, NOTHELFE, 0<1<2<3<4]

Number of Instances: 1930

Number of Attributes: 40

Number of Numeric Attributes: 40

For

Grade: 0: Times the grade appear in the instances: 731.0 Parcial Sum for grade: 0.0

Grade: 1: Times the grade appear in the instances: 2014.0 Parcial Sum for grade: 2014.0

Grade: 2: Times the grade appear in the instances: 4806.0 Parcial Sum for grade: 9612.0

Grade: 3: Times the grade appear in the instances: 7307.0 Parcial Sum for grade: 21921.0

Grade: 4: Times the grade appear in the instances: 4442.0 Parcial Sum for grade: 17768.0

Number of times the label: LEISTUNG was prefered with the higher grade: 445.0

Number of times the label: MENLEIT was prefered with the higher grade: 244.0

Number of times the label: AZEITSEL was prefered with the higher grade: 484.0

Number of times the label: PRESTIGE was prefered with the higher grade: 64.0

Number of times the label: VERSTCHE was prefered with the higher grade: 632.0

Number of times the label: UMENSEIN was prefered with the higher grade: 578.0

Number of times the label: MIVERBES was prefered with the higher grade: 468.0

Number of times the label: UNTERNEH was prefered with the higher grade: 390.0

Number of times the label: FAEHIGEN was prefered with the higher grade: 865.0

Number of times an instance is related to a label with the higher grade: 4442.0

3.5. ¿Cómo incorporar un nuevo clasificador a la plataforma extendida?

A continuación se ofrece una idea general de como introducir un nuevo clasificador a partir de la estructura que siguen los métodos existentes. Para la incorporación de un nuevo clasificador al paquete integrado es necesario conocer y dominar primero la estructura y funcionamiento de las clases de evaluación explicadas en la sección anterior. El nuevo algoritmo debe utilizar las clases básicas de almacenamiento y preprocesamiento de instancias propuestas en el subdirectorio *dataStructure*, así como seguir la estructura de los ficheros .earff diseñados específicamente para ser usados por esta herramienta y con especial atención en ranking de las etiquetas y asignación de grados a cada una de ellas.

Primeramente se debe crear un nuevo directorio bajo la dirección **mulan.gradedMLC.[*nombre del algoritmo*]** donde "nombre del algoritmo" es precisamente sustituido por el nombre del método a introducir. En este directorio se almacenarán los tres subdirectorios no comunes entre los diferentes métodoss, dígase *decomposition, result* y *structuredPrediction*.

- *Decomposition*

Los cinco algoritmos propuestos utilizan Comparación por pares y CLR para reducir el problema gradual a un problema más sencillo, por lo que las clases descomponedoras presentes en el paquete solo obedecen a estas técnicas. Es necesario ubicar en el directorio **mulan.gradedMLC.[*nombre del algoritmo*].decomposition** la clase correspondiente al método de transformación que se le aplicará al conjunto de datos.

- Result

Las clases en el directorio *result* están implementadas para almacenar los parámetros iniciales requeridos para la corrida del modelo, así como obtener y mostrar los resultados arrojados por los algoritmos. Dado que las ya implementadas contienen elementos que utilizan etiquetas calibradas y se basan en dar respuestas con ranking, en este directorio del nuevo método se deben implementar las clases que manipulan los resultados y cubren las funcionalidades anteriormente expuestas pero siguiendo la estructuración de salida con ranking que ofrecen los métodos existentes.

- structuredPrediction

En este directorio se encuentran las clases correspondientes a varias estrategias que permiten realizar la clasificación de las instancias de prueba a través de diferentes maneras de analizar los votos de cada uno de los clasificadores bases. La clase central es *structuredPredictor.java* de la que heredan las otras implementadas hasta el momentos, también llamadas estrategias de agregación. Una vez más estas estrategias se rigen por los resultados obtenidos de los métodos que utilizan ranking, por lo que resulta necesario su reimplementación o adaptación para otro tipo de metodología.

Finalmente resulta necesaria la creación de una nueva clase de evaluación en el directorio *Evaluation* bajo el nombre *Evaluation*[nombre del nuevo clasificador] para seguir el orden de nombramiento del paquete gradedMLC. Esta nueva clase debe encontrarse bajo la ubicación *mulan.gradedMLC.evaluation.* El siguiente paso es estructurar dicha clase de la misma manera que se muestra en la sección anterior, llamando y usando las nuevas clases ubicadas en las locaciones explicadas anteriormente.

3.6. Utilidades y ventajas de la integración

Aunque los cinco métodos integrados se encuentran en su primera versión y son extremadamente jóvenes, en comparación con los propuestos en otras temáticas diferentes, muestran muy buenos

resultados y utilizan elementos que aumentan su rendimiento como el método de transformación de problemas: Comparación por pares y Ranking de Etiquetas Calibradas. Hasta el momento el marco de trabajo dedicado a problemas multietiquetas, MULAN, no proporcionaba ninguna utilidad que permitiera trabajar con datos graduales siendo una limitante dada la alta presencia de este tipo de datos en los problemas de clasificación. La integración de estos métodos convirtió a MULAN en una herramienta más potente capaz de obtener resultados confiables para todas las problemáticas analizadas.

Con el paquete incorporado al MULAN, este puede procesar una nueva extension de archivo, para almacenar los conjuntos de datos estructurados de manera tal que se integre en un solo fichero la informacion que antes se guardaba en dos. Esta nueva estructura mostró la utilización de preferencias en el momento de almacenar los grados y la claridad brindada por el uso de ranking para las preferencias en los empates. Se incorporaron cuatro conjuntos de datos estructurados de la manera mencionada que son ejemplos claves para la conformación de nuevos conjuntos. También brinda un grupo de doce métricas rediseñadas especialmente para tratar el caso específico de la clasificación multietiqueta gradual.

Los métodos utilizados contienen gran flexibilidad permitiendo la variación de determinados parámetros ubicados en diferentes clases. Por ejemplo, dependiendo del algoritmo se puede escoger entre tres diferentes tipos de agregación para la clasificación de instancias de prueba con los resultados arrojados por los distintos clasificadores. En la clase Configuration.java encontrada en el subdirectorio results de cada uno de los métodos pueden ser cambiados varios valores: el número de folds a utilizar, el prefijo otorgado a las etiquetas calibradas, el número de la semilla para la evaluación, la ubicación del conjunto de datos y la ubicación donde se escribirán los resultados. En caso de que un mismo conjunto de datos se encuentre dividido en varios ficheros el algoritmo es capaz de leer cada uno de los ficheros de manera automática y devolver un resultado general. Además, se implementó una nueva clase cuya funcionalidad es brindar al usuario un conjunto de informaciones básicas y estadísticas sobre el conjunto de datos introducido como parámetro permitiendo al usuario saber si es conveniente el uso del conjunto de datos o si debe cambiar algún parámetro para la corrida o simplemente utilizar algún algoritmo específico.

CONCLUSIONES

1. En el presente proyecto se estableció un marco referencial sobre Clasificación Multietiqueta Gradual recolectándose los artículos y documentos que centraban su atención en el tema hasta la fecha. Debido a la escasa bibliografía con que cuenta esta rama de la clasificación solo se encontraron cuatro artículos, de los cuales, dos ofrecían soluciones.

2. Se utilizaron cinco métodos propuestos en uno de estos artículos por la Universidad Técnica de Darmstadt en Alemania que contienen algoritmos basados en métodos de transformación para reducir el problema gradual a uno más simple, específicamente Comparación por pares y Ranking de etiquetas calibradas, que demostraron ser mejores que los desarrollados hasta la fecha.

3. Al comparar los valores de las métricas para los métodos independientes e integrados a MULAN sobre cada uno de los cuatro conjuntos de datos se obtuvieron resultados iguales.

4. La creación del nuevo paquete supuso una mejora a MULAN, pues amplió su radio de solución de problemas permitiendole la cacidad de manejar también los vinculados a Clasificación Multietiqueta Gradual.

RECOMENDACIONES

Como recomendaciones:

- Crear módulo para generar conjunto de datos graduales de manera automática.
- Enriquecer el paquete con nuevos clasificadores y métricas.

REFERENCIAS BIBLIOGRÁFICAS

[1] E. Gibaja and S. Ventura, "Multi-label learning: a review of the state of the art and ongoing research," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 6, pp. 411–444, 2014.

[2] C. Brinker, E. L. Mencía, and J. Furnkranz, "Graded multilabel classification by pairwise comparisons," in *Data Mining (ICDM), 2014 IEEE International Conference on*, pp. 731–736, IEEE, 2014.

[3] S. V. Eva Gibaja, "A tutorial on multi-label learning," *ACM Comput*, pp. 1–39, 2010.

[4] E. L. M. J. Furnkranz, E. Hullermeier and K. Brinker, "Multi-label classification via calibrated label ranking," *Machine Learning*, vol. 73, pp. 133–153, 2008.

[5] J. Ramon Quevedo, O. Luaces, and A. Bahamonde, "Multi label classifiers with a probabilistic thresholding strategy," *Pattern Recognition*, vol. 45, pp. 876–883, Feb. 2012.

[6] A. S. F.Aiolli, F. Sebastiani, "Preference learning for category -ranking based interactive text categorization," pp. 2034–2039, 2007.

[7] T. C.D.Nguyen, T.A.Dung, "Text classification for dag-structured categories," pp. 1–18, 2005.

[8] P. Q. T. Goncalves, "A preliminary approach to the multilabel classification problem of portuguese juridical documents," pp. 435–444, 2003.

[9] J. F. Loza, "Efficient pairwise multilabel classification for large-scale problems in the legal domain," pp. 30–36, 2008.

[10] A. B. J. Yearwood, M. Mammadov, "Profiling phishing emails based on hyperlink information," pp. 120–127, 2010.

[11] J. d. C.-A. R. M.-B. J. M. Carmona-Cejudo, M. Baena-Garcia, "Feature extraction for multi-label learning in the domain of email classification," pp. 30–36, 2011.

[12] P. R. T. Sobol-Shikler, "Classification of complex information: Inference of co-occurring affective states from their expressions in speech," vol. 32, pp. 1284–1297, July 2010.

[13] A. A. F. R. T. Alves, M. R. Delgado, "Knowledge discovery with artificial immune systems for hierarchical multi-label classification of protein functions," vol. 4, pp. 1–29, July 2010.

[14] M. R. T. J. Jung, "Gene function prediction using protein domain probability and hierarchical gene ontology information," pp. 1–4, 2008.

[15] Y. R. J. T.-H.-J. Z. G.-J. Qi, X.-S. Hua, "Twodimensional multilabel active learning with an efficient online adaptation model for image classification," pp. 1880–1897, 2009.

[16] X. W. X. S. H. J. Wang, Y. Zhao, "A transductive multi-label learning approach for video concept detection," pp. 17–26, 2011.

[17] C. Y. N. Z. J. Fan, Y. Shen, "Structured max-margin learning for inter-related classifier training and multilabel image annotation," *IEEE Transactions on Image Processing*, vol. 20, pp. 837–854, March 2011.

[18] C. D. H. Wang, H. Huang, "Multi-label feature transform for image classifications," *Lecture Notes in Computer Science*, vol. 6314, pp. 793–806, 2010.

[19] G.-B. H. W. Zong, "Face recognition based on extreme learning machine," *Neurocomputing*, May 2011.

[20] Y. P. X. H. H. Wang, C. Lin, "Application of improved random forest variables importance measure to traditional chinese chronic gastritis diagnosis," *Medicine and Education*, pp. 84–89, December 2008.

[21] J. R. H. Su, M. Heinonen, "Structured output prediction of anti-cancer drug activity," *Lecture Notes in Computer Science*, vol. 6282, no. 38-49, 2010.

[22] J. H. W.-J. N. Barrett, "Applying natural language processing toolkits to electronic health records - an experience report," *Studies in health technology and informatics*, vol. 143, pp. 441–446, 2009.

[23] J. B. Y. S. K. B.-A. K. M. S. D. Wu, L. Lu, "Stratified learning of local anatomical context for lung nodules in ct images," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2791–2798, 2010.

[24] P. R. F. Pachet, "Improving multilabel analysis of music titles: A large-scale validation of the correction approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, pp. 335–343, February 2009.

[25] Z. R. W. Jiang, A. Cohen, "Polyphonic music information retrieval based on multi-label cascade classification system," *Studies in Computational Intelligence*, vol. 251, pp. 117–137, 2009.

[26] H.-M. W. S.-D. L. H.-Y. Lo, J.-C. Wang, "Cost-sensitive multi-label learning for audio tag annotation and retrieval," *IEEE Transactions on*, vol. 13, pp. 518–529, June 2011.

[27] M. P. E. E. Myint, "An approach for mulit-label music mood classification," *3rd International Conference on Signal Processing Systems,*, p. 290, July 2010.

[28] E. S.-J. K. C. Ganapathy, J.-H. Kang, "Classification techniques for assessing student collaboration in shared wiki spaces," *Lecture Notes in Computer Science*, vol. 6738, pp. 456–458, 2011.

[29] F. W. S. N. Ahsan, "Impact analysis of scrs using single and multi-label machine learning classification," *ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2010.

[30] A. P. D. Thorleuchter, D. V. den Poel, "A compared r&dbased and patent-based cross impact analysis for identifying relationships between technologies," *Technological Forecasting and Social Change*, vol. 77, pp. 1037–1050, September 2010.

[31] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: a lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, pp. 2038–2048, July 2007.

[32] J. FÃŒrnkranz, E. HÃŒllermeier, E. Loza MencÃa, and K. Brinker, "Multilabel classification via calibrated label ranking," *Machine Learning*, vol. 73, no. 2, pp. 133–153, 2008.

[33] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[34] E. Spyromitros, G. Tsoumakas, and I. Vlahavas, "An empirical study of lazy multilabel classification algorithms," in *Artificial Intelligence: Theories, Models and Applications* (J. Darzentas, G. Vouros, S. Vosinakis, and A. Arnellos, eds.), vol. 5138 of *Lecture Notes in Computer Science*, pp. 401–406, Springer Berlin / Heidelberg, 2008.

[35] J. L. A. Jiménez, "Modelos de aprendizaje basados en programación genética para clasificación multi-etiqueta," p. 207, 2013.

[36] J. XU, "An extended one-versus-rest support vector machine for multi-label classification," *Neurocomputing*, vol. 74(17), pp. 3114–3124, 2011.

[37] M. MYINT, E. E.; PWINT, "An approach for mulit-label music mood classification," *3rd International Conference on Signal Processing Systems*, 2010.

[38] A. GOLD, K.; PETROSINO, "Using information gain to build meaningful decision forests for multilabel classification," *IEEE 9th International Conference on Development and Learning (ICDL)*, pp. 58–63, 2010.

[39] M. A.-J. J. L. V. S. GIBAJA, E.; VICTORIANO, "A tdidt technique for multi-label classification," *10th International Conference on. Intelligent Systems Design and Applications*, pp. 519–524, 2010.

[40] M.-L. ZHANG, "Ml-rbf: Rbf neural networks for multi-label learnin," *Neural Processing Letters*, vol. 29, pp. 61–74, 2009.

[41] M.-L. Z. X.-H. ZHOU, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering,*, vol. 18, pp. 1338–1351, 2006.

[42] R. R. L. K. REFORMAT, "A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation," *Data & Knowledge Engineering*, vol. 64, pp. 171–197, 2008.

[43] C. B. L. . P. LARRAÑAGA, "Multi-dimensional classification with bayesian networks," *International Journal of Approximate Reasoning*, vol. 52, pp. 705–727, 2011.

[44] R. V. D. G. X. L. D. A. CARVALHO, "A new approach for multi-label classification based on default hierarchies and organizational learning," *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, vol. 2017-2022, 2008.

[45] W. Cheng, K. Dembczyski, and E. Hallermeier, *Graded Multilabel Classification: The Ordinal Case*.

[46] J. Furnkranz and E. Hullermeier, "Pairwise preference learning and ranking," *In Proc. ECML-03, Cavtat-Dubrovnik*, vol. 145-156, 2003.

[47] W. C. J. Furnkranz, E. Hullermeier and K. Brinker, "Label ranking by learning pairwise preferences," *Artificial Intelligence*, vol. 172, pp. 1897–1916, Nov.2008.

[48] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "MULAN: a java library for multi-label learning," *J. Mach. Learn. Res.*, vol. 12, pp. 2411–2414, July 2011.

[49] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," *Data mining and knowledge discovery handbook*, pp. 667–685, 2010.

Anexos

A continuación se muestran de manera más amplia los resultados obtenidos por cada uno de los algoritmos sobre los cuatro conjuntos de datos utilizando las doce métricas propuestas.

(En el caso de los algoritmos que utilizan CLR la métrica **OptimalCalibrationHamming** se sustituyó por sus siglas **OCH** por cuestiones de espacio)

Métricas\Conj.Datos	**BeLaE_m5**		**BeLaE_m10**		**Medical**		**Movies**	
	Indep.	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*
VerticalHammingDistance	0.6333	0.6333	0.6454	0.6454	0.6504	0.6504	0.4035	0.4035
OneErrorLoss	0.3794	0.3794	0.2770	0.2770	0.2550	0.2550	0.2433	0.2433
ZeroOneLoss	0.9286	0.9286	0.9493	0.9493	0.9419	0.9419	0.6506	0.6506
CIndex	0.4961	0.4961	0.4989	0.4989	0.5000	0.5000	0.4316	0.4316
PrecisionLoss	0.0713	0.0713	0.0506	0.0506	0.0579	0.0579	0.3493	0.3493
F1Loss	0.0713	0.0713	0.0506	0.0506	0.0579	0.0579	0.3493	0.3493
RecallLoss	0.0713	0.0713	0.0506	0.0506	0.0579	0.0579	0.3493	0.3493
FalloutLoss	0.9286	0.9286	0.9493	0.9493	0.9419	0.9419	0.6506	0.6506
AccuracyLoss	0.9257	0.9257	0.9620	0.9620	0.9623	0.9623	0.9349	0.9349
TpRate	0.0713	0.0713	0.0506	0.0506	0.0579	0.0579	0.3493	0.3493
FpRate	0.0386	0.0386	0.0193	0.0193	0.0192	0.0192	0.0342	0.0342

Figura 3.3: Resultados de las corridas con ***Binary Relevance*** sobre los 4 conjuntos de datos.

Métricas\Conj.Datos	**BeLaE_m5**		**BeLaE_m10**		**Medical**		**Movies**	
	Indep.	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*
VerticalHammingDistance	0.1918	0.1918	0.1800	0.1800	0.2449	0.2449	0.2711	0.2711
OneErrorLoss	0.1454	0.1454	0.1862	0.1862	0.2713	0.2713	0.2174	0.2174
ZeroOneLoss	0.5710	0.5710	0.5561	0.5561	0.6506	0.6506	0.5879	0.5879
CIndex	0.2551	0.2551	0.3049	0.3049	0.2985	0.2985	0.3912	0.3912

*Figura 3.4: Resultados de las corridas con **Frank&Hall** sobre los 4 conjuntos de datos.*

Métricas\Conj.Datos	**BeLaE_m5**		**BeLaE_m10**		**Medical**		**Movies**	
	Indep.	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*
VerticalHammingDistance	0.1819	0.1819	0.1724	0.1724	0.2124	0.2124	0.2582	0.2582
OneErrorLoss	0.0667	0.0667	0.0928	0.0928	0.0931	0.0931	0.1608	0.1608
ZeroOneLoss	0.6354	0.6354	0.5912	0.5912	0.6823	0.6823	0.6379	0.6379
CIndex	0.1643	0.1643	0.2087	0.2087	0.2189	0.2189	0.2399	0.2399
PrecisionLoss	0.3645	0.3645	0.4087	0.4087	0.3176	0.3176	0.3619	0.3619
F1Loss	0.3645	0.3645	0.4087	0.4087	0.3176	0.3176	0.3619	0.3619
RecallLoss	0.3645	0.3645	0.4087	0.4087	0.3176	0.3176	0.3619	0.3619
FalloutLoss	0.6354	0.6354	0.5912	0.5912	0.6823	0.6823	0.6379	0.6379
AccuracyLoss	0.3645	0.3645	0.9763	0.9763	0.9727	0.9727	0.9362	0.9362
TpRate	0.3645	0.3645	0.4087	0.4087	0.3176	0.3176	0.3619	0.3619
FpRate	0.0264	0.0264	0.0120	0.0120	0.0139	0.0139	0.0335	0.0335
OCH	0.0841	0.0841	0.0473	0.0473	0.0705	0.0705	0.1222	0.1222

Figura 3.5: Resultados de las corridas con ***HorizontalCLR*** sobre los 4 conjuntos de datos.

Métricas\Conj.Datos	**BeLaE_m5**		**BeLaE_m10**		**Medical**		**Movies**	
	Indep.	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*
VerticalHammingDistance	0.3229	0.3229	0.3351	0.3351	0.3327	0.3327	0.7380	0.7380
OneErrorLoss	0.0787	0.0787	0.1051	0.1051	0.0909	0.0909	0.1116	0.1116
ZeroOneLoss	0.6975	0.6975	0.7697	0.7697	0.7163	0.7163	0.9426	0.9426
CIndex	0.1867	0.1867	0.2225	0.2225	0.2294	0.2294	0.2540	0.2540
PrecisionLoss	0.3024	0.3024	0.2302	0.2302	0.2836	0.2836	0.0573	0.0573
F1Loss	0.3024	0.3024	0.2302	0.2302	0.2836	0.2836	0.0573	0.0573
RecallLoss	0.3024	0.3024	0.2302	0.2302	0.2836	0.2836	0.0573	0.0573
FalloutLoss	0.6975	0.6975	0.7697	0.7697	0.7163	0.7163	0.9426	0.9426
AccuracyLoss	0.9441	0.9441	0.9692	0.9692	0.9713	0.9713	0.9057	0.9057
TpRate	0.3024	0.3024	0.2302	0.2302	0.2836	0.2836	0.0573	0.0573
FpRate	0.0290	0.0290	0.0157	0.0157	0.0146	0.0146	0.0496	0.0496
OCH	0.2081	0.2081	0.2046	0.2046	0.1856	0.1856	0.5788	0.5788

Figura 3.6: Resultados de las corridas con ***FullCLR*** sobre los 4 conjuntos de datos.

Métricas\Conj.Datos	**BeLaE_m5**		**BeLaE_m10**		**Medical**		**Movies**	
	Indep.	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*	*Indep.*	*Integ.*
VerticalHammingDistance	0.1617	0.1617	0.1617	0.1617	0.2073	0.2073	0.2726	0.2726
OneErrorLoss	0.0667	0.0667	0.0928	0.0928	0.0931	0.0931	0.1608	0.1608
ZeroOneLoss	0.5677	0.5677	0.5611	0.5611	0.6656	0.6656	0.6619	0.6619
CIndex	0.1643	0.1643	0.2087	0.2087	0.2189	0.2189	0.2399	0.2399
PrecisionLoss	0.4322	0.4322	0.4388	0.4388	0.3343	0.3343	0.3379	0.3379
F1Loss	0.4322	0.4322	0.4388	0.4388	0.3343	0.3343	0.3379	0.3379
RecallLoss	0.4322	0.4322	0.4388	0.4388	0.3343	0.3343	0.3379	0.3379
FalloutLoss	0.5677	0.5677	0.5611	0.5611	0.6656	0.6656	0.6619	0.6619
AccuracyLoss	0.9545	0.9545	0.9775	0.9775	0.9733	0.9733	0.9338	0.9338
TpRate	0.4322	0.4322	0.4388	0.4388	0.3343	0.3343	0.3379	0.3379
FpRate	0.0236	0.0236	0.0114	0.0114	0.0135	0.0135	0.0348	0.0348
OCH	0.0638	0.0638	0.0366	0.0366	0.0654	0.0654	0.1366	0.1366

Figura 3.7: Resultados de las corridas con ***JoinedCLR*** sobre los 4 conjuntos de datos.