

Hola, tesis doctoral en informática !!!

Índice general

1. Sistema de Recomendación Web	13
1.1. Personalización de la Web	17
1.2. Minería de Uso Web	20
1.2.1. DATOS	21
1.2.2. Selección	22
1.2.3. Preproceso	24
1.2.4. Transformación	25
1.2.5. Minería de Datos (DM)	35
1.2.6. Evaluación e Integración	38
1.2.7. CONOCIMIENTO	40
1.3. Minería de Datos	41
1.3.1. Mapas de Navegación Web	41
1.3.2. Reglas de Asociación	47
1.4. Publicaciones	65
Actas de HCII'05	65
Actas de Interacción'05	66
Actas de SICO'05	66
2. Minería de Reglas de Asociación	69
2.1. Conceptos básicos	71

2.1.1.	Tipo de Datos	72
2.1.2.	Primeros Algoritmos	73
2.1.3.	Formato de \mathcal{D}	77
2.1.4.	Fases de <i>ARM</i>	78
2.2.	<i>Minería de Itemsets Frecuentes</i>	79
2.2.1.	Algoritmos y estructuras	79
2.2.2.	Evaluación de diferentes implementaciones	92
2.3.	Generación de <i>reglas de asociación</i>	96
2.3.1.	genrules()	97
2.3.2.	Apriori2	102
2.4.	El <i>Ítem Raro</i>	108
2.4.1.	Estudio de ítems raros	108
2.4.2.	<i>Reglas de Oportunidad</i>	111
2.5.	Publicaciones	112
	Actas de HCII'07	112
	ESWA, vol. 35(3) 2008	113
	Actas de Interacción'10	114
3.	<i>ARM para Clasificación</i>	115
3.1.	Transacciones Estructuradas	117
3.1.1.	Tipos de <i>transacciones</i>	119
3.1.2.	Descripción de mushroom.dat	122
3.1.3.	Modelización de <i>Registros</i>	122
3.1.4.	Análisis eficiente de <i>Registros</i>	122
3.1.5.	Lectura de <i>datasets</i> comprimidos	126
3.2.	<i>Minería de Reglas de Clasificación Asociativa</i>	128
3.2.1.	Adaptación de la notación	130
3.3.	<i>Catálogo</i>	131
3.3.1.	Muestras	137
3.4.	<i>Catálogo Completo</i>	138
3.4.1.	Colecciones de <i>Catálogos Completos</i>	138
3.5.	Datos missing	139
3.6.	Publicaciones	141
3.6.1.	[...]	142
3.6.2.	[...]	143

4. Conclusiones y Trabajo Futuro	145
4.1. Flujos de Datos	148
A. Notación	151
A.1. Sistemas de Recomendación Web	151
A.2. Minería de Reglas de Asociación	151
A.3. Catálogos	151
B. Código	153
B.1. Sistemas de Recomendación Web	153
B.2. Minería de Reglas de Asociación	153
B.3. Catálogos	153
C. Datos utilizados	159
C.1. Sistemas de Recomendación Web	159
C.2. Minería de Reglas de Asociación	159
C.3. Catálogos	159
Índice de figuras	161
Índice de tablas	163
Índice de definiciones	165
Índice de listados	168
Índice alfabético	169
D. Sobre la bibliografía	173
Bibliografía	189

Motivación

La obtención, almacenamiento y distribución de grandes cantidades de datos está al alcance de todo el mundo gracias a la *Informática*, la *Telecomunicación* y el resto de ciencias que han impulsado el crecimiento de ambas. Entidades de todo tipo (personas físicas o jurídicas, empresas públicas y privadas, organizaciones de diversos ámbitos...) pueden disponer con facilidad de equipos informáticos y telemáticos capaces de recoger, almacenar y distribuir información de forma masiva. Esto las convierte en entidades propietarias de colecciones con enormes cantidades de datos en crudo. Si nos quedamos con estas aportaciones estaremos hablando de un problema de *infoxicación*, la «incapacidad de llevar a cabo un análisis eficiente de gran cantidad de información».

ABIERTO

Cada día surgen nuevos dispositivos capaces de medir, almacenar y distribuir nuevos datos añadiéndolos o combinándolos con los que ya tenemos, lo que redundará en el exceso de información cruda de que disponemos. Esto acrecentaría el problema planteado si no hubiera sido tratado desde todas las ciencias existentes (Arte, Medicina, Historia, Informática, Investigación Operativa...). Desde hace más de 4 décadas todos los investigadores se han encontrado con grandes y crecientes colecciones de datos en crudo y con la necesidad de extraer de ellos, de forma eficiente, parte del conocimiento que contienen. De aquí surgen múltiples estudios sobre cómo transformar tanta información dinámica en conocimiento.

La globalización de la *World Wide Web* (WWW) no ha hecho más que acrecentar el problema. Cada día más con la aparición y popularización de sistemas

de comunicación inalámbrica que permiten comunicar fácilmente dispositivos de proceso, almacenamiento y visualización de datos entre sí, sea cual sea su ubicación. La *WWW* es el medio de generación, almacenamiento y distribución masiva de datos más representativo. Ya en 1996 Etzioni se preguntaba si la enorme cantidad de información que estaba generando la *WWW* en su época era un lodazal o una mina de oro. Se estaban adaptando las bases de la *Minería de Datos* (DM) a la *Minería Web* (WM). Para comprender el comportamiento de los usuarios de la *WWW* se podían buscar *patrones* entre los datos que genera su propio uso y facilitar con ellos la obtención de los objetivos de cada usuario al navegar por la web, aparecía la *Minería de Uso Web* (WUM). Hoy en día sabemos que la *WWW* es una mina de oro, pero hay que saber convertir la información que contienen los millones de datos que genera a diario en conocimiento capaz de ser utilizado por los medios de que dispone cada persona en muchas de sus tareas cotidianas.

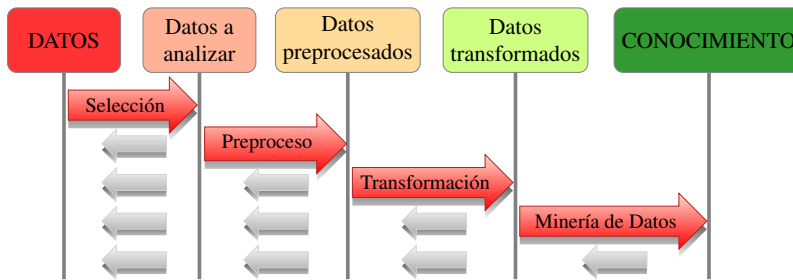


Figura 1: Proceso *KDD*

En la misma época estaba emergiendo con fuerza el *Knowledge Discovery in Databases* (KDD), «el proceso no trivial de identificar *patrones* de datos válidos, novedosos, potencialmente útiles y comprensibles» según Fayyad, Piatetsky-Shapiro y Smyth (1996). El proceso es puramente científico: la materia prima es una colección de datos a la que se le aplican cinco tareas secuenciales y recursivas (con retroalimentación) y cuyo producto final es el conocimiento. Fayyad, Piatetsky-Shapiro y Smyth inciden en la diferencia entre *Minería de Datos* y *Knowledge Discovery in Databases*, la primera es una tarea de la segunda (ver figura 1). Sin embargo algunos autores las confunden debido a que la recursividad del proceso de *KDD* hace que la *DM* tenga que ser ajustada y reutilizada con lo que podría considerarse un sub-proceso completo de adquisición de conocimientos.

Aunque he mencionado que la *WUM* es una derivación de la *DM* realmente

es un proceso específico y completo de *KDD* ya que se ha de estudiar qué datos se utilizarán de entre los disponibles, se han de preprocesar adecuadamente, transformar de modo que puedan ser analizados mediante *DM* y finalmente evaluar los resultados, todo ello de forma recursiva siempre que en el proceso aparezcan resultados intermedios que lo aconsejen.

Uno de los grandes paradigmas de la ciencia es «divide y vencerás», y es perfectamente aplicable a cualquier proceso de *KDD*. Cada investigador se centra en su especialidad para mejorar un pequeño aspecto de la ciencia. En mi caso he querido aportar mis conocimientos sobre algoritmia (matemáticas e informática), análisis de datos (estadística y matemáticas) y programación (informática) por lo que tras estudiar el estado del arte sobre el proceso completo de la *Minería de Uso Web* decidí dedicar mi investigación a la *Minería de Datos*.

Nuestros primeros trabajos se centraron en la *Personalización de la Web* (Anderson, 2002), concretamente en el análisis de los *Sistemas de Recomendación Web (WRS)* a través de la *Minería de Uso Web*, introduciendo nuevos métodos para la fase de *Minería de Datos* que permitieran aliviar los problemas ya sugeridos por otros investigadores. La publicación en congresos de nuestros trabajos nos ofreció la oportunidad de compartir estas propuestas con colegas de especialidades afines o complementarias e ir creciendo en conocimientos generales sobre el tema. Tras analizar las primeras tareas del *KDD* aplicadas al estudio del uso de la web llegamos a conclusiones similares al de resto de investigadores respecto a las primeras tareas del proceso: los datos iniciales se han seleccionar y preprocesar adecuadamente (fijando ciertos parámetros de forma empírica) y transformar en *sesiones de navegación*.

Nuestra primer propuesta fue el uso de *grafos* para representar las *sesiones de navegación* y analizar el uso de la web por parte de los internautas. Los *grafos* permiten estudiar las *secuencias* de páginas producidas por los usuarios en sus *sesiones de navegación*, de las que podemos extraer tanto la co-ocurrencia de páginas en una misma sesión como el orden en que fueron visitadas. Aunque este camino parecía adecuado su tratamiento sobre colecciones de datos muy grandes desbordaba las capacidades de las máquinas usadas para nuestra investigación.

En nuestra segunda propuesta decidimos cambiar el tratamiento dado a las *sesión de navegación*, considerándolas como colecciones de *transacciones*, grupos de páginas recorridos en la misma sesión sin importar el orden en que fueron visitadas, reduciendo las dimensiones del problema para aliviar la carga de procesamiento y almacenamiento de los datos a tratar. El trabajo de Agrawal y Srikant (1994a) nos abrió nuevos caminos con la introducción del algoritmo *Apriori*, un

elegante algoritmo capaz de encontrar gran cantidad de *Reglas de Asociación* entre los datos que estábamos manejando. Lo incorporamos a nuestro trabajo porque su sencillez daba mucho juego y podíamos adaptarlo a nuestro objetivo: encontrar *patrones* de navegación de los usuarios de un sitio web para su personalización (Borges y Levene, 1999). Los resultados obtenidos en esta fase se exponen en el capítulo 1.

Las *Reglas de Asociación* son una descripción de las co-ocurrencias existentes entre los diferentes ítems de una *transacción* (las diferentes páginas visitadas en un sitio web durante una *sesión de navegación*, en nuestro caso). Conceptualmente están más cercanas a la Estadística Descriptiva que a la *DM* pero en la práctica permiten abordar el análisis de inmensas colecciones de *transacciones* por lo que han sido abordadas desde el punto de vista informático debido principalmente al llamado *dilema del ítem raro*, que postula que los ítems menos frecuentes en el análisis pueden hacer inviable el estudio completo de la colección de datos debido a problemas de desbordamiento y falta de recursos. Este dilema nos preocupaba pues la catalogación de un ítem como «raro» sólo dependía del número de veces que aparecía en las *transacciones* (su frecuencia absoluta o *soporte* según el lenguaje de las *regla de asociación*) y no de otros factores que podrían ser más significativos.

En el capítulo 2 planteamos dos modificaciones al algoritmo *Apriori* original. Los datos nos proporcionan información sobre «*qué hace*» el usuario, las *reglas de asociación* nos pueden dar información sobre «*cómo lo hacen*». Esto nos sugirió la primera de nuestras aportaciones abordando una búsqueda de relaciones entre las *reglas de asociación* encontradas entre los datos analizados con la intención de agrupar a los usuarios de un sitio web en función de «*qué reglas cumplen*» y no únicamente en función de «*qué páginas visitan*». De aquí surgió la única publicación que tengo hasta la fecha en una revista científica de impacto dando mucha más visibilidad a mi aportación que la obtenida en los diferentes congresos en que hemos expuesto nuestras ideas (véase 2.5). Sin embargo esta solución incrementaba el *dilema del ítem raro*.

La segunda aportación presentada en este capítulo proponía una solución sencilla para tratar este dilema en situaciones en que los ítems menos frecuentes pueden ser de importancia, concretamente en su aplicación a los *WRS*. Se trataba de usar las *Reglas de Oportunidad* (junto a las *Reglas de Asociación*) de modo que un *Sistema de Recomendación Web* pudiera no sólo sugerir al usuario las páginas más relacionadas con aquellas que ya ha visitado si no que también pudiera sugerirle páginas menos frecuentes posiblemente relacionados con sus objetivos.

En el capítulo 4 se vuelven a tomar estos temas pues pueden ser mejorados utilizando técnicas estadísticas, de *clasificación* y de investigación operativa que no se han podido abordar en esta tesis.

El *dilema del ítem raro* seguía preocupándonos. Aunque trabajemos con mucha información simultáneamente los actuales sistemas informáticos han de ser capaces de ofrecer información válida sobre todos los ítems de una colección de *transacciones*. Hay diversos estudios sobre las limitaciones de la tecnología y algoritmia actual para afrontar un estudio completo de una colección grande de *transacciones* [libro que no consigo y limita el análisis de mushroom y chess], hay trabajos que ayudan a determinar qué ítems raros deben ser analizados, el dilema se aborda desde muchas perspectivas aplicando distintas técnicas del tipo «divide y vencerás». Pero si no nos esforzamos en incluir en un único proceso la información que contienen los ítems raros el conocimiento adquirido tendrá menos calidad y, sobre todo, no se podrán estudiar a fondo problemas tan actuales como las enfermedades raras, nuevos métodos de fraude. . .

La mayor aportación de esta tesis se expone en el capítulo 3, donde abordamos el estudio de ciertas colecciones de *transacciones* extensamente utilizadas por la comunidad científica, los *catálogos* y las muestras que se obtienen a partir de su estructura. En el [libro que...] ponen límite a la información que se puede analizar en colecciones de este tipo, como mushroom.dat. Expondremos que no debemos renunciar a ningún tipo de información ofrecida por ese fichero, podemos aprovechar la rigidez de su estructura para reducir drásticamente el número de elementos a analizar. Al aplicar la reducción propuesta podremos obtener todas las *Reglas de Asociación* contenidas en la colección de *transacciones* utilizando los algoritmos ya conocidos sobre esta materia.

Para terminar esta investigación proponemos el uso de estructuras informáticas adaptadas a la flexibilidad y agilidad necesaria en *Minería de Datos* proponiendo el uso de *Catálogos Completos* . . .

Sistema de Recomendación Web

Observar el comportamiento de los individuos de cierta población, anotarlo y analizarlo buscando *patrones* de comportamiento que permitan conocer mejor a dicha población es uno de los métodos más utilizados por la ciencia desde sus inicios. El objetivo de esta metodología es siempre el mismo: transformar *datos* en *conocimiento*, *experiencia* en *ciencia*. Las Matemáticas y la Estadística han aportado sus cimientos, contando con las necesidades y experiencias de las demás ciencias para modelizar correctamente los datos observados de modo que puedan ser analizados científicamente. La Informática aportó el *Knowledge Discovery in Databases (KDD)* como paradigma de Descubrimiento de Conocimiento en *Bases de Datos (DB)*, que derivaron en la *Minería de Datos (DM)* ante el crecimiento exponencial de contenido en las *DB*. Con la ayuda de la Telemática, junto a todas las ciencias que han impulsado su brutal crecimiento desde hace unas décadas, las ciencias de la tecnología proporcionan herramientas eficientes para la captación masiva de datos, su almacenamiento y su distribución. Uno de los objetivos de la ciencia actual, apoyada por la tecnología, es extraer en tiempo real el conocimiento que contienen esas grandes cantidades de datos.

Fayyad, Piatetsky-Shapiro y Smyth (1996) definieron el proceso de *Knowledge Discovery in Databases* como «*El proceso no trivial de identificar patrones de datos válidos, novedosos, potencialmente útiles y comprensibles*». La figura 1.1, extraída de su artículo, muestra una idea del proceso completo de *KDD*, formado por cinco fases recurrentes que permiten convertir los *datos disponibles* en *conocimiento*. Una *selección* de los datos disponibles convenientemente *preprocesados* y *transformados* pueden ser sometidos a técnicas de *Minería de Datos*

que descubran los *patrones* que contienen para ser convertidos en conocimiento. Friedman (1997) considera el proceso de *KDD* como un análisis exploratorio de datos para grandes *Bases de Datos*. Hay muchos autores que describen a su modo las diferentes fases del proceso de *KDD* (Rokach y Maimon, 2014), siguiendo todos ellos la propuesta de Fayyad.

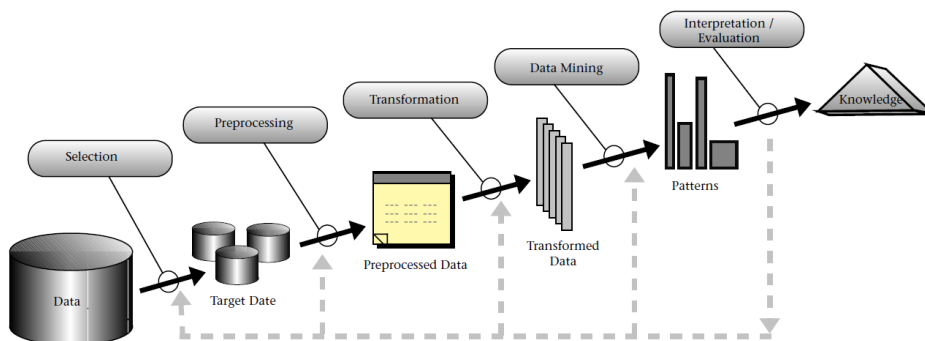


Figura 1.1: Fases del proceso de *KDD* [Fayyad et al. (1996)]

Un *Sistema de Recomendación (RS)* es un claro ejemplo del proceso de *KDD*. Se basa en la capacidad de almacenamiento y procesamiento que ofrece la tecnología, con estrategias diseñadas por expertos en su área (selección, preprocesamiento, transformación y análisis), y el objetivo de hacer recomendaciones de calidad de forma semi-automática en base a los *patrones* obtenidos.

Tradicionalmente ha sido una persona quien realizaba todo el proceso de captación de datos, análisis y conversión en conocimiento. Un típico ejemplo es el del médico que ha estudiado una carrera para conocer la experiencia de otros científicos en el mismo campo, al ejercer ha recabado datos sobre la salud de sus usuarios, los pacientes, realiza un análisis exponiendo los datos obtenidos a la experiencia (propia o derivada de sus estudios o comunicaciones con otros expertos) y cuando encuentra un *patrón* de comportamiento que se adapte al paciente al que está tratando emite un diagnóstico junto a una recomendación.

De un *Sistema de Recomendación* se espera precisamente esto, que convierta la información en conocimiento en base a comparar los datos de un individuo con *patrones* de comportamiento ya contrastados en la población a la que pertenece el individuo. Cuando esta población es muy dinámica (las enfermedades a que se enfrentan los médicos no son siempre conocidas o pueden evolucionar de modo que ya no sea válido el conocimiento previamente adquirido) el proceso se ha

de repetir cíclicamente aportando nuevos datos o conocimiento o renunciando a aquellos que no aporten valor al conocimiento final. Son muchas las disciplinas que se han sumado a la búsqueda de *Sistemas de Recomendación* que se auto-gestionen, como la aportación de Kouris, Makris y Tsakalidis (2005) utilizando métodos y técnicas de *Recuperación de la Información*.

Con la tecnología actual se espera que esta conversión sea de gran calidad y en tiempo real ya que se conocen muchos datos y se pueden analizar desde múltiples perspectivas mediante procesos informáticos muy rápidos. Esta expectativa se manifiesta en la gran cantidad de trabajos científicos que hacen su aporte para la obtención de *Sistemas de Recomendación* en casi todas las áreas de investigación. Volviendo a nuestro médico, podríamos decir que es un RS y cuanta más experiencia tenga (propia o ajena, pero de calidad) más rápidos y certeros serán sus diagnósticos y, por tanto, sus recomendaciones. Si nuestro médico utiliza la tecnología actual llegará un momento en uno de sus análisis en que querrá conocer nuevos datos de su paciente, otro investigador puede averiguar con qué dispositivo se pueden obtener los nuevos datos, otro investigador estudiará la influencia de este nuevo dato sobre el diagnóstico y si los resultados son «correctos» se puede implementar un sistema que realice de forma automática este proceso en tiempo real, con el coste de utilizar mayor cantidad de datos.

Ya hace más de dos décadas Frawley, Piatetsky-Shapiro y Matheus (1992) estimaban que la cantidad de información almacenada de forma digital se duplica cada 20 meses. Con los avances tecnológicos sobre digitalización, almacenamiento y transmisión de datos y su enorme abaratamiento estas estimaciones se quedan cortas hoy en día por lo que hay que seguir investigando el modo de analizar de forma eficiente cada vez más grandes cantidades de datos.

Los *Sistemas de Recomendación Web (WRS)* son un caso particular de los *Sistemas de Recomendación*, están más adaptados a la fase de recolección de datos que cualquier otro RS porque se usan únicamente con dispositivos informáticos unidos telemáticamente. Cada vez que un usuario utiliza un servicio web está enviando datos a un servidor, datos que el servidor puede almacenar en sus discos duros o en su memoria RAM para ser analizados inmediatamente o con posterioridad (generalmente se harán ambos análisis). Si el WRS ya «conoce» lo que suele hacer un determinado usuario de su sitio web puede mejorar su *experiencia de usuario (UX)* facilitándole enlaces de forma dinámica para que el usuario cumpla su objetivo con el menor esfuerzo posible. Si el sistema no conoce al usuario puede comparar su comportamiento con el del resto de usuarios del sitio web, si encuentra un *patrón* de comportamiento que se adapte al usuario

desconocido también podrá mejorar su *UX*. Sin embargo el crecimiento de información en la web es cada día mayor, llegando a cotas que hacen muy complejo el planteamiento expuesto.

Etzioni (1996) planteó ideas sobre la pobre utilización de la gran cantidad de datos que contienen y recogen los servidores web. Estaba emergiendo la *Minería Web* bajo tres perspectivas: *Minería de Estructura Web*, *Minería de Contenido Web* y *Minería de Uso Web*. Conociendo el contenido de la web, cómo está estructurado y cómo se usa se podría llegar a niveles altísimos de personalización de la web. Aplicando el conocimiento adquirido a través de estas tres perspectivas al uso de la web se podrían desarrollar *Sistemas de Recomendación Web* que operasen en tiempo real sobre todos los usuarios de la web, recogiendo nuevos datos de uso y analizándolos para obtener mejores resultados, conocimiento de mayor calidad y mejores recomendaciones.

Y. Cho, J. Kim y S. Kim (2002) proponen su propio *Sistema de Recomendación Web* para un e-comercio basándose en *Minería de Uso Web*, inducción de árboles de decisión, *Minería de Reglas de Asociación* y una taxonomía del producto comercializado. Cada vez tenemos más técnicas a disposición de la *KDD* y esto generará un flujo constante de investigaciones sobre esta materia. Al analizar los datos de uso de la *WWW* se pueden hacer análisis más precisos sobre los intereses o preferencias de los usuarios.

Schafer, Konstan y Riedl (2001) presentan una taxonomía sobre *Sistemas de Recomendación* centrándose en portales de e-comercio. Las técnicas a usar para personalizar las recomendaciones puede depender de muchos factores, entre los que podríamos citar:

- Las recomendaciones pueden estar dirigidas a usuarios específicos o a todos los usuarios del servicio.
- El objetivo de la recomendación puede ser predecir cuánto le gustará el producto recomendado a cierto usuario o bien obtener una lista de productos de alto interés para el usuario (problema de las top- N recomendaciones)
- Se puede tener preparada la recomendación u obtenerla bajo demanda, en cuyo caso la eficiencia de los algoritmos utilizados adquiere gran importancia.

Al iniciar nuestra investigación queríamos implementar un *Sistema de Recomendación Web* basado en el conocimiento adquirido mediante la aplicación de técnicas de *Minería de Uso Web* a los datos recogidos en un *fichero de log* por un

servidor web. Había ya mucha investigación al respecto y comenzamos a seleccionar artículos de investigación sobre la *WUM* como una realización específica de un proceso de *KDD*.

En las siguientes secciones describiremos las fases mostradas en la figura 1.1 con el enfoque utilizado en este trabajo, la *Minería de Uso Web*, haciendo una visión global de los *WRS* y profundizando en los aspectos que más afectan al proceso de investigación mostrado en este informe.

1.1. Personalización de la Web

Un *portal web* que sea capaz de adaptar su contenido a las necesidades de sus usuarios es un reto para los investigadores desde la popularización de la web hasta la actualidad. Para personalizar un sitio web, el desarrollador del sitio debe conocer las necesidades de sus usuarios. Perkowitz y Etzioni (1997a, 1997b, 1998, 2000a) proponen el término «sitios adaptativos» para designar los sitios web que adaptan semi-automáticamente sus páginas aprendiendo de los *patrones* de acceso de los propios usuarios.

Según Nielsen (1998):

Web personalization is much over-rated and mainly used as a poor excuse for not designing a navigable website . The real way to get individualized interaction between a user and a website is to present the user with a variety of options and let the user choose what is of interest to that individual at that specific time. If the information space is designed well, then this choice is easy, and the user achieves optimal information through the use of natural intelligence rather than artificial intelligence. In other words, I am the one entity on the world to know exactly what I need right now. Thus, I can tailor the information I see and the information I skip so that it suits my needs perfectly.

Nielsen pone en evidencia que la correcta personalización de una herramienta, un *portal web* en nuestro caso, está estrechamente relacionada con las expectativas del usuario. Algo difícil de lograr por el hecho de que los usuarios no tienen los mismos objetivos, deseos o necesidades al visitar el mismo *portal web*. El diseño de la herramienta no está en sus manos pero las herramientas que estamos analizando en esta investigación son fácilmente moldeables en su diseño y contenido por los administradores y desarrolladores del *portal web*.

Perkowitz y Etzioni (1998, 1999b, 1999a, 2000b) proponen personalizar un sitio web mediante su algoritmo PageGather, que crea dinámicamente índices con enlaces a otras páginas por las que podría estar interesado el usuario. Se trata de un caso de estudio de sus «sitios adaptativos», enfocado a la recomendación en tiempo real de las páginas que otros usuarios del sitio web visitan junto a la que está visitando nuestro usuario.

La WWW se ha convertido en un lugar en que cualquiera puede realizar gran variedad de tareas o utilizar multitud de servicios. Cuando se diseña un *portal web* se ha de tener en mente que las personas usan herramientas para conseguir sus objetivos con la máxima eficacia y eficiencia (Constantine y Lockwood, 1999), y un *portal web* puede ser considerado como una herramienta con la que los usuarios realizan sus tareas. Para personalizar un *portal web* necesitamos recoger información sobre los usuarios que lo visitan e intentar ofrecer una página web ajustada a las necesidades de nuestros visitantes (Nielsen, 1998). El principal objetivo de la personalización ha de ser la satisfacción del usuario por lo que el proceso de personalización no debe olvidar el fin de la usabilidad, la personalización de la web no es una excusa para diseños pobres. Esto es posible si los usuarios encuentran un *portal web* con un buen diseño de sus páginas, sus contenidos y, en general, con un buen diseño del sitio donde los principios de usabilidad y accesibilidad se hayan tenido en cuenta durante el proceso de diseño del *portal web* (Nielsen, 2000).

Si fuéramos capaces de ofrecer a nuestros usuarios información personalizada podríamos satisfacerles pues obtendrían lo que realmente buscan en el menor tiempo posible, o al menos en un tiempo asumible (Duyne, Landay y J. I. Hong, 2002).

Si un usuario no tiene fácil acceso a sus objetivos en un sitio web es muy probable que busque satisfacer sus necesidades en otro sitio web. Pero satisfacer las necesidades de diferentes usuarios no es posible sin tener información exhaustiva sobre sus objetivos y métodos de navegación. El desarrollador del sitio web puede esforzarse en conseguir un diseño que a él o a un grupo de usuarios de prueba les resulte cómodo y fácil de usar para la consecución de sus objetivos, pero la globalización de la web permite cada vez más el acceso al sitio web de usuarios de diferentes culturas, nivel de estudios o intereses y le permiten hacerlo desde diferentes dispositivos. La personalización del sitio web debería conseguir que un usuario particular sienta que el sitio se adapta a sus necesidades, con lo que es más fácil que cuando quiera utilizar un servicio de las características del sitio web en cuestión elija éste en lugar de otro que no se adapte a su modo de trabajar.

Todos los usuarios generan información al navegar a través de un sitio web. Cada uno de ellos tiene un objetivo al acceder al sitio web y lo intenta llevar a cabo, con mayor o menor eficiencia, a través del uso de los hiperenlaces facilitados por el desarrollador del sitio web. Es el usuario quien decide qué enlaces utilizar para lograr su objetivo, pero está sujeto al diseño del sitio web: aunque quisiera acceder desde la página *A* directamente a la página *C* no podrá hacerlo si el desarrollador no ha incluido un enlace a *C* en la página *A*. Según K.-J. Kim y S.-B. Cho (2007) la personalización de los servicios de búsqueda de la WWW es ya una realidad, presentan su propio sistema utilizando diferentes métodos de *DM* en un proceso de *KDD*.

Para personalizar un *portal web*, pues, necesitamos recoger toda la información que seamos capaces de convertir en conocimiento sobre su uso. Sin embargo los administradores del portal deben preservar la privacidad y confidencialidad de los datos recogidos (W3CP3P, 2002; Lam, Frankowski y Riedl, 2006; Rozenberg y Gudes, 2006). P3P 1.0, desarrollado por el *World Wide Web Consortium*, surgió como un estándar de la industria que proporciona una forma sencilla y automatizada para que los usuarios tengan más control sobre el uso de información personal en los sitios web que visitan. Cubren nueve aspectos de la privacidad en línea. Cinco de ellos giran en torno a los datos que pueden ser recogidos por el sitio web: *¿Quién está recogiendo estos datos?* *¿Exactamente qué información está siendo recopilada?* *¿Para qué fines?* *¿Qué información se comparte con los demás?* y *¿Quiénes son estos receptores de datos?*. Los cuatro aspectos restantes explican las políticas internas de privacidad del sitio: *¿Los usuarios pueden realizar cambios en cómo se utiliza su información?* *¿Cómo se resuelven los conflictos?* *¿Cuál es la política de retención de datos?* y, finalmente, *¿Dónde se puede encontrar las políticas detalladas en «forma legible por humanos»?*

Para garantizar el cumplimiento de estas recomendaciones decidimos hacer uso de nuestra metodología sobre los datos registrados por los servidores web en un archivo especial llamado *fichero de log* (W3CLog, 1996), donde se recoge sólo información anónima acerca de las acciones de los usuarios en un *portal web*.

Hasta aquí hemos visto que la personalización de la web es posible con la confluencia de muchos actores:

- Administradores que conozcan las condiciones de privacidad del sitio web y puedan controlar su aplicación.
- Desarrolladores y diseñadores que puedan presentar a cada usuario una

página web personalizada, en función de la información proporcionada por el usuario y de los *patrones* descubiertos por otros especialistas.

- Especialistas en *UX* que analicen la usabilidad del la página web personalizada obtenida.
- Especialistas en las áreas cubiertas en el *portal web* que evalúen la calidad de las recomendaciones.
- Especialistas en *Minería de Datos* capaces de analizar los datos de uso del sitio web y ofrecer *patrones* de comportamiento que puedan ser utilizados por los desarrolladores del sistema.

La metodología expuesta permitirá al administrador del sitio web analizar el comportamiento de los usuarios del sitio de forma permanente, lo que le permitirá rediseñar en todo o en parte el sitio web basándose en el *diseño centrado en el usuario*, «El diseño de la funcionalidad de un sitio debe estar dirigido por y para los usuarios.» (Baeza-Yates y Rivera Loaiza, 2003). En servicios con gran cantidad de usuarios se puede esperar que sean éstos quienes ayuden en su diseño, sin embargo la gran cantidad de sitios web dedicados a la misma temática hace que sea más fácil que el usuario cambie de servicio antes de emplear su tiempo en adaptar el que está utilizando, el administrador de un sitio web ha de ser capaz de adaptar los enlaces a las necesidades o preferencias de sus usuarios sin necesidad de su participación en el proceso (Fink, Kobsa y Nill, 1996; Joachims, Freitag y Mitchell, 1997; Ferré y col., 2001).

Aunque todos los actores tienen gran importancia en la obtención de un *WRS* de calidad, en este trabajo nos centraremos en el aporte de los especialistas en *Minería de Datos*. Analizaremos los tipos de datos con que hemos de trabajar y prepararemos estructuras que posibiliten su gestión de forma eficiente y realizable en un gran número de sistemas informáticos.

1.2. Minería de Uso Web

La personalización de la web ha generado muchas investigaciones debido a la capacidad de las páginas web para contener datos (noticias, artículos, palabras, imágenes...) y enlaces internos o a otras páginas web. Si se escriben en cada página los enlaces más adaptados al usuario que la está visitando es muy probable que su *experiencia de usuario* mejore. Esto genera cambios continuos

en la estructura del sitio web, alojando cada vez más páginas con contenido más variado. Al tratarse siempre de solicitudes de un cliente (usuario) a un servidor (sitio web) se guarda automáticamente mucha información sobre cada solicitud de página hecha por un usuario. Cuando un sitio web comienza a guardar estos datos es fácil observarlos mediante sencillos gráficos y tablas y mejorar la estructura y el contenido del sitio web. Sin embargo el número de datos recogido puede ir creciendo día a día, hora a hora, en función de la popularidad del sitio web. Cuando ya comienzan a ser muchos los datos guardados sobre el uso del sitio web comienzan a ser intratables mediante esos sencillos gráficos y tablas, perdemos gran parte de la información que contienen los datos por no estar usando las herramientas adecuadas para analizarlos. A grandes rasgos son las características que hicieron aparecer la *Minería Web (MW)* (Cooley, Mobasher y Srivastava, 1997; Pitkow, 1997; Kosala y Blockeel, 2000; Scime, 2004) y sus tres especialidades: *Minería de Estructura Web (WSM)*, *Minería de Contenido Web (WCM)* y *Minería de Uso Web (WUM)*. Todas ellas enfocan el proceso de *KDD* con el propósito de extraer el basto conocimiento que encierran los datos gestionados en la WWW. Todas ellas son necesarias para lograr la personalización de la web, cada una aporta conocimiento útil para las restantes.

En nuestra investigación nos centraremos en la *Minería de Uso Web*, que como todo proceso de *KDD* puede llevarse a cabo mediante la ejecución consecutiva y recursiva de diferentes tareas sobre los datos disponibles y los resultados intermedios obtenidos en el proceso (figura 1.1). En cada una de las siguientes secciones especificaremos qué tipo de datos están disponibles y qué tareas se pueden aplicar en este proceso para convertirlos en conocimiento útil que facilite la personalización de la web (Srivastava y col., 2000; Mobasher, Cooley y Srivastava, 2000; De Bra, Aroyo y Chepegin, 2004).

1.2.1. DATOS

La informática ayuda a virtualizar la realidad, las distintas ciencias determinan qué características tienen los individuos de cualquier población pero no podemos obtener todas esas características de la realidad ya que muchas de ellas aún no han sido descubiertas o no tenemos procedimientos para medirlas correctamente. Con los medios y métodos disponibles se ha de decidir qué parte de la realidad se guardará en formato digital mediante *Bases de Datos (DB)*, representada por Data en la figura 1.1.

La mayoría de entidades (empresas públicas y privadas, servicios...) dispo-

nen de *Bases de Datos* con cada vez más información sobre la propia entidad, sus usuarios o productos. Esta información crece constantemente en diferentes dimensiones: nuevos usuarios, nuevas características mensurables sobre cada usuario, nuevas interacciones con cada usuario, distribución geográfica de nuevas sucursales de la entidad, desaparición de sucursales. . .

Estas grandes *Bases de Datos* no sirven de nada si no son explotadas convenientemente. Pero las magnitudes que van adquiriendo a lo largo del tiempo, su dinamismo y constante crecimiento en diferentes dimensiones hacen inviable un estudio completo y comprensible de toda la información que contienen. Es necesario convertir los datos disponibles (Data) en conocimiento (Knowledge) para lo que es necesario planificar un proceso de *KDD*, o incluso varios procesos según el volumen de información del que se disponga y los objetivos perseguidos con su análisis.

En la *Minería de Uso Web* se busca conocer cómo se usa la web. Si nos centramos en un sitio web, los datos disponibles pueden ser tanto de la estructura como del contenido o del uso del sitio web o incluso de enlaces a otros sitios web. Los datos sobre estructura y contenido tienen mucha dependencia de los administradores, diseñadores y redactores de contenido del sitio web, sin embargo los datos de uso pueden guardar cierta independencia respecto a los actores mencionados. La información generada por el uso del sitio web puede ser almacenada por el servidor en que se aloja el sitio web mediante *ficheros de log*, de los que nosotros usamos los de formato *Extended Log File Format* cuyas especificaciones se encuentran en W3CLog (1996). Se trata de ficheros de texto plano con una línea por cada elemento solicitado al servidor por sus usuarios, línea en que aparece información sobre la IP que ha solicitado el elemento, instante de la solicitud, URI del elemento solicitado, protocolo, estado y otros datos que no van a ser utilizados en este trabajo.

1.2.2. Selección

Los datos recogidos por la entidad que quiere hacer el proceso completo de *KDD* pueden ser de diferente índole, incluyendo datos personales que no deberían ser accesibles por todos los investigadores involucrados en el proceso de *KDD*. La primera fase del proceso es la selección de atributos que serán realmente analizados entre los disponibles por la entidad propietaria de los datos. Los investigadores tendrán a su disposición lo que en la figura 1.1 se denomina Target Data, que realmente es una consulta realizada a la *Base de Datos* com-

pleta, Data. Aquí aparece por primera vez el caracter recursivo de este proceso: si los investigadores determinan que necesitan datos que ni siquiera están recogidos en la *Base de Datos* global podrían solicitar a la entidad la obtención de dichos datos, afectando al punto de origen del proceso.

Esta selección es fundamental para el proceso global de *KDD*, si los resultados de la evaluación final no son satisfactorios pueden provocar que se opte por una selección diferente de los registros y campos a incorporar en el Target Data. Cualquier decisión errónea en esta fase puede derivar en análisis que no conduzcan al descubrimiento real de conocimiento útil y previamente desconocido sobre la población en estudio.



Figura 1.2: Selección de datos

Si la *Base de Datos* en estudio pertenece a una empresa (pública o privada) son los responsables de la misma quienes han de decidir, aconsejados por los investigadores que llevarán a cabo el proceso de *KDD* y por las leyes vigentes en el país en que se realiza el estudio, qué datos y registros pasan a formar parte del Target Data, recurriendo a la codificación de los datos que sean necesarios para el análisis pero no deban aportar información sensible.

La selección en sí, unida a la experiencia de los investigadores encargados del proceso de *KDD*, determinará si se han de recabar en Data atributos y datos necesarios para el objetivo perseguido. Todas las fases del proceso son recurrentes y pueden generar parte del conocimiento buscado en el sentido de obligar a retroceder en el proceso para mejorar la calidad del conocimiento final adquirido.

En el caso de la *WUM* existen datos recogidos en el servidor al atender a demandas internas de sus usuarios, con información confidencial sobre los usuarios o sobre la estructura o contenido del sitio web. Este tipo de información no debería ser accedida por quienes analizan el uso del sitio web por lo que debería ser filtrada durante esta primera fase. Si fuera necesario se podrían codificar las propias IPs de los usuarios del sitio u otros datos que pueden revelar información personal para cumplir con las normativas sobre privacidad de datos del país en que se está llevando a cabo la investigación (Lam, Frankowski y Riedl, 2006; Rozenberg y Gudes, 2006).

1.2.3. Preproceso

El preproceso es una de las fases que más tiempo consume en cualquier proceso de *KDD*. No todos los valores registrados en el Target Data contienen información relevante para el estudio a realizar por lo que se deben eliminar todos aquellos que sólo aportan ruido o dificultan innecesariamente la ejecución de las siguientes fases del proceso.

Estas decisiones deben ser tomadas por expertos en la materia final del estudio, su experiencia puede agilizar enormemente el tiempo de obtención de resultados y la calidad de los mismos. La evaluación final puede ayudar también a tomar decisiones en esta fase si se detecta nula influencia de un dato incorporado en esta fase.

En el preproceso se decide qué hacer con los datos atípicos pues son datos que dificultan el estudio completo, decisión que aparecerá de nuevo en la fase de *Minería de Datos*. Pero su influencia puede ser esencial en algunas ocasiones. Hay investigadores que eliminan directamente estos datos atípicos, otros plantean un estudio exclusivo de estos datos. Sea cual sea la decisión tomada debería volverse a este punto una vez finalizado el proceso de *KDD* para comprobar si puede explicarse de modo científico la existencia de datos atípicos (en ocasiones debidos simplemente a malas mediciones en el proceso de captación de datos).

También hay que tomar decisiones sobre los datos missing, incluso provocando un estudio previo para hacer una *predicción* sobre estos datos y poder utilizar estas estimaciones en el resto del proceso. Si un atributo posee muchos datos missing y se decide eliminar el atributo del estudio podríamos estar renunciando a información de extrema calidad, si se utilizan estimaciones de los mismos pero no son estimaciones correctas el resultado final puede aparecer totalmente sesgado por esta circunstancia.



Figura 1.3: Preproceso

Al ser una de las fases que más recursos consume se llevará a cabo en horas en que el servidor tiene menor carga de trabajo y los resultados que ofrece serán usados posteriormente mediante algoritmos de extracción de información. Es fundamental crear los filtros adecuados al análisis posterior que se quiera realizar a estos

datos, la extracción en sí de datos no es necesariamente compleja (puede ser una

simple consulta a una *Base de Datos*) pero si no extraemos los datos adecuados la información obtenida puede carecer de relevancia.

Para estudiar el uso de un sitio web se deben conocer qué recursos ha solicitado realmente cada usuario. Considerando que una página web puede estar formada por gran cantidad de elementos: texto, imágenes, vídeos, anuncios externos o internos, enlaces... es fácil pensar que cada vez que un usuario accede a un sitio web va a generar un gran número de líneas en su *fichero de log*, una por cada elemento incluido en la página solicitada y alojado en el servidor. La primera decisión a la hora de extraer información de los *ficheros de log* consiste en eliminar todas las líneas que no se deben a la navegación a través del sitio web por parte del usuario si no a la propia estructura del sitio web: se han de detectar las páginas que realmente ha solicitado el usuario y descartar los elementos que son cargados en el navegador del usuario por estar enlazados en la página solicitada (Cooley, Mobasher y Srivastava, 1999).

Además de eliminar toda esta información de los datos seleccionados en la fase previa hay muchas páginas solicitadas por los usuarios que realmente no existen en el servidor, marcadas con estado 404, que deben ser eliminadas después de analizar si se deben a una solicitud incorrecta escrita por el usuario o a enlaces propios del sitio web que han cambiado su ubicación o han desaparecido. Existen otros problemas en el uso de *ficheros de log*, como la existencia de páginas cargadas en la caché del servidor o del cliente y cuyas solicitudes no aparecerán (Pitkow, 1997). Algunos autores proponen resolverlo con el uso de «huellas» (Villena y col., 2002). Otro problema es la existencia de agentes en la red (robots) que navegan automáticamente por todo un sitio web y cuyo comportamiento no debería ser considerado en el estudio ya que no reflejan el comportamiento de los usuarios a los que queremos adaptar el servicio.

Como en todos los procesos de *KDD*, el preproceso reduce notablemente el número final de datos a analizar en *WUM*. Sin embargo se debe vigilar que no se excluyan datos en esta fase que pudieran ser determinantes para la obtención de conocimiento de calidad.

1.2.4. Transformación

Una vez obtenidos los llamados Preprocessed Data en la figura 1.1 cambiaremos su aspecto para comprenderlos mejor nosotros y poder entonces adaptarlos para que puedan ser gestionados por las máquinas. Hasta ahora hemos hecho la *selección* de datos que formarán parte del estudio entre todos los disponibles,

hemos *preprocesado* estos datos de un modo muy genérico, básicamente eliminamos aquellos que pensamos (o sabemos) que no aportarán conocimiento al estudio que llevamos a cabo. Ahora toca darles más aspecto de información que de datos, usaremos una o varias estructuras que permitan modelizar los datos preprocesados de modo que tengan más sentido para el análisis a realizar.

En general no utilizaremos todos los datos preprocesados, están ahí para ser utilizados en diferentes estudios, o añadidos o eliminados en un estudio en particular pero no para tratarlos todos de golpe, seguimos hablando de colecciones muy grandes de datos. Los datos pueden ser sometidos a una reducción de dimensiones mediante una selección y extracción de características o bien el uso de muestreo. Un muestreo bien aplicado puede arrojar mucha información sobre el contenido del conjunto completo de datos preprocesados, lo que ayudaría a decidir qué datos serán los que más conocimiento aporten. La experiencia de los investigadores ayudará a decidir hechos como que «*el índice de masa corporal es más representativo en este estudio que la altura y peso del individuo por separado*», aunque ese índice se calcula a partir de los otros dos datos si en lugar de guardar en memoria dos valores guardamos sólo uno estaremos ganando recursos en la máquina para hacer otros procesos o guardar otros datos relevantes para el estudio.

Las cadenas de caracteres son un tipo de dato difícil de gestionar a nivel informático. Si tenemos datos en ese formato lo más conveniente es hacer una conversión usando códigos hash, p. ej., de modo que se identifiquen unívocamente con un número entero. Esto supondrá un gran ahorro de memoria y mayor rapidez en funciones de búsqueda y comparación, funciones comunes en la *DM*. Incluso estos códigos hash podrían ser codificados para trabajar con números enteros consecutivos. Si adecuamos el tipo de dato y sus valores al algoritmo que los va a manejar ganaremos en eficiencia.

Los datos numéricos pueden ser transformados mediante categorización o transformaciones funcionales de modo que puedan ser representados con menos recursos sin perder la información que contienen. El «*índice de masa corporal*» es un número en coma flotante, se suelen utilizar 32 o 64 bits para representar este tipo de números en memoria por lo que cuando tengamos muchísimos valores que guardar estaremos ocupando gran parte de la memoria disponible y quedará menos memoria para la ejecución de los algoritmos de *DM*. Para tareas como *regresión* o *clustering* es un formato muy adecuado, sin embargo en muchas otras tareas el valor numérico en sí no es utilizado pero se sospecha que la variable sí que es importante para el estudio, como le ocurre a la variable «edad» en múlti-

ples estudios. La edad, a pesar de ser un número entero, se convierte en *conceptos* como «Bebé», «Menor de 12 años», «Preadolescente»... para trabajar con menos valores diferentes, quizá necesitemos sólo 2 o 3 bits para almacenar este tipo de datos y tendríamos más memoria disponible para el algoritmo de *DM* a aplicar.

Esta fase es fundamental para lograr resultados más precisos, en ella intervienen los conocimientos sobre matemáticas y estadística de los investigadores, conocimiento pleno sobre las técnicas que se aplicarán a los datos transformados y buenos conocimientos sobre informática y programación si no se tienen las herramientas adecuadas para llevar a cabo el análisis. Un error cometido en muchos trabajos de *clustering* consiste en aplicar medidas de distancia numérica entre códigos numéricos para hacer averiguaciones sobre su «similitud» sin considerar que la distancia entre dos códigos no puede ser interpretada igual que la diferencia entre los dos números usados para su codificación. Errores informáticos y de programación hay muchos, se trata de manejar colecciones crecientes (casi infinitas) de datos con recursos finitos y se obtienen continuos problemas de desbordamiento de memoria.

Todo proceso de *KDD* se retroalimenta de cada conocimiento (parcial) adquirido en cualquiera de sus fases, obligando a retroceder para llegar finalmente a un conocimiento lo más completo posible a partir de los datos, recursos y tiempo disponible para el análisis. Una vez termine el proceso de *KDD* podremos replantearnos el proceso desde el principio o hacer pruebas con transformaciones diferentes que recojan lo mejor del conocimiento que ya hemos adquirido y sean capaces de observar cualquier cambio en los datos que estamos analizando. Siempre podemos descubrir *patrones* que encajan mejor con los individuos que estamos estudiando.

Volviendo a centrarnos en el proceso de *WUM*, los datos que vamos a procesar son básicamente las páginas que se han visitado y los enlaces utilizados para ello. Los definiré formalmente para ir exponiendo la notación usada en este informe.

Definición 1 (*Páginas del sitio web*). Sea P el conjunto de las M páginas del sitio web.

$$P = \{P_i, i = 1 \dots M\} \quad (1.1)$$

Definición 2 (*Enlaces internos del sitio web*). Sea E el conjunto de todos los enlaces internos del sitio web.

$$E = \{E_{ij}, i \neq j, i, j \in \{1 \dots M\}\} \quad (1.2)$$

Ambos conjuntos son parte de la estructura física del sitio web. La forma más natural de representarlos nos la proporciona la *Teoría de Grafos*. Un *grafo* es una estructura con muchas propiedades que permiten hacer sobre ellos gran cantidad de cálculos y, sobre todo, su representación gráfica es muy fácil de interpretar y manipular con aplicaciones informáticas. Básicamente son colecciones de *nodos* unidos (o no) entre sí mediante *ejes*. Si representamos ambos conjuntos utilizando las páginas como nodos los ejes serán los enlaces de E , como muestra la figura 1.4.

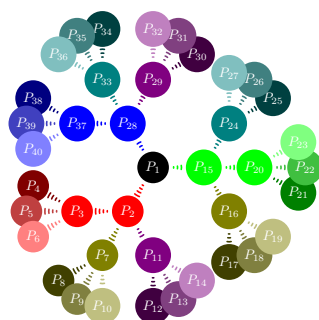


Figura 1.4: Estructura del sitio web

En muchos sitios web existe una página denominada «mapa del sitio». Se trata generalmente de un listado de las páginas de P organizado según los criterios del diseñador del mapa y que no considera los enlaces de E creados dinámicamente. Puede reorganizarse dinámicamente si cada página contiene metadatos con palabras clave asociadas a su contenido, pero conforme aumenta el número de páginas de P se va haciendo más difícil su correcta gestión y acaba por ser un mapa

pobrementemente actualizado.

Si el sitio web es de pequeñas dimensiones es muy fácil representar su estructura mediante un *grafo*, sin embargo si es un sitio web con muchas páginas visitables y muy dinámico en su contenido acaban siendo tan grandes los *grafos* que dejan de aportar información. La *Minería de Estructura Web* puede ayudar mucho en este aspecto al desarrollador del sitio web, mostrándole *grafos* similares al de la figura 1.4 simplemente leyendo una página del sitio web y recorriendo todos sus enlaces internos, permitiéndole centrarse en *grafos* más pequeños generados de forma dinámica.

En la fase de preproceso hemos obtenido un *fichero de log* reducido en que sólo aparecen las solicitudes de páginas realizadas por los usuarios del sitio web, lo que en la literatura se conoce como *clickstream* (Bucklin y Sismeiro, 2001), junto con el resto de variables seleccionadas para el análisis (IP del solicitante, fecha, hora, estado...). Con los *clickstreams* sólo sabemos *qué* visitan los usuarios en nuestro sitio web pero no *cómo* lo hacen. Hemos de utilizar un modelo de datos que represente el comportamiento de un usuario de un sitio web, sien-

do la *sesión de navegación* un buen punto de partida. Una *sesión de navegación* es una *secuencia* de interacciones entre el usuario y un sitio web que podría ser generalizada del siguiente modo:

1. El usuario tiene un objetivo (o varios) que le llevan a entrar en nuestro sitio web.
2. Si no encuentra directamente lo que busca se fija en los enlaces existentes en la página en que está, siempre que su diseño doten de usabilidad a dichos enlaces, o incluso utiliza Ctrl+F para buscar en la página pistas sobre su objetivo.
3. Si no encuentra nada puede volver a la página anterior (con Alt+← o Retroceso, con lo que no solicita nada al servidor pues suele estar guardada en la caché de cliente) o bien puede dar por terminada su *sesión de navegación*.
4. Si encuentra un enlace a otra página de nuestro sitio web que le parece el camino más apropiado para cumplir con su objetivo lo utilizará, solicitando una nueva página en su sesión. Vuelve a encontrarse en la situación planteada en el punto 2.
5. Si no encuentra nada su *sesión de navegación* ha terminado.

En el *fichero de log* reducido no se refleja por qué ha abandonado la sesión el usuario, y si se ha usado la caché del cliente es posible que se encuentren situaciones no previstas en el sitio web si la siguiente página registrada por el usuario en el *fichero de log* no puede provenir de la página actual por no existir el enlace. Sólo si está registrado y ha iniciado sesión identificada en el sitio web podemos recabar información sobre su navegación real, concretamente sobre la finalización de la sesión. Puede abandonar nuestro sitio web satisfecho por haber encontrado fácilmente su objetivo o bien incómodo por haber perdido el tiempo buscándolo en nuestro sitio web. Como lo que queremos es descubrir cómo usan nuestro sitio web vamos a plantear como primera hipótesis que los usuarios abandonan el sitio web satisfechos por haber encontrado en la última página de su *sesión de navegación* el objetivo que tenía desde el principio. Se supone que los usuarios que vuelven a nuestro sitio web es porque les ha resultado útil y es su comportamiento el que estamos analizando, cada vez con más datos. Los usuarios

que no vuelven acaban por dejar de encajar en los *patrones* de navegación que hayamos descubierto y sus datos acabarán por ser ignorados por los algoritmos de *DM*.

Para estudiar el comportamiento de nuestros usuarios hay que construir las *sesiones de navegación* de cada uno de ellos, agrupando adecuadamente todos los clickstreams que haya producido, considerando que cada usuario está identificado por la IP utilizada. La asociación usuario-IP no es totalmente correcta pues debido a las características de las redes internas de comunicación es posible que diferentes usuarios utilicen la misma IP, además de existir las IPs dinámicas, un mismo usuario podría usar diferentes IPs en sus visitas al sitio web, más hoy en día que puede acceder a la WWW desde múltiples lugares y dispositivos. Si no hay un registro efectivo del usuario no podemos identificarlo unívocamente, sin embargo la solución tomada es la máxima información que podremos extraer de *ficheros de log* anónimos.

Un problema que presenta la obtención de *sesiones de navegación* es que no suele haber información precisa sobre las mismas en el *fichero de log* ya que el protocolo html carece de estado. Los usuarios de un sitio web son en su mayoría anónimos por lo que la finalización de una *sesión de navegación* se determina de forma heurística. Se han propuesto diversas alternativas: fijando un umbral a la duración máxima de una sesión (2, 4 u 8 horas, p.ej.), o al intervalo de tiempo máximo entre dos clickstream consecutivos (10 o 30 minutos, p.ej.), o una combinación de ambas, o incluso si ya se tienen clasificadas las páginas del sitio web en diferentes temáticas un cambio de temática podría interpretarse como un final de sesión. Existen muchos trabajos que abordan este tema (He y Göker, 2000; Huang, Peng, An, Schuurmans y Cercone, 2003; Huang, Peng, An y Schuurmans, 2004a, 2004b).

Definición 3 (*Sesión de navegación*). Una sesión de navegación es un vector ordenado de las páginas visitadas por un usuario en un intervalo de tiempo determinado por $maxClick$ y $maxSession$ con información sobre el tiempo transcurrido entre la visita de cada par de páginas del vector

$$s_i = (userID, p_1, u_1, p_2, u_2 \dots p_{n_i-1}, u_{n_i-1}, p_{n_i}) \quad (1.3)$$

Con este procedimiento obtendremos N vectores como el expuesto en la ecuación 1.3, donde i enumera las diferentes sesiones, $userID$ identifica al usuario que ha hecho la solicitud, p_j representa la página solicitada y u_j el tiempo que permanece en ella o, mejor dicho, el tiempo transcurrido entre la solicitud de la

página p_j y la de p_{j+1} (realmente no sabemos qué hizo el usuario durante ese tiempo). n_i es el número de páginas visitadas en la sesión i -ésima, con lo que el número de registros del *fichero de log* reducido coincide con $\sum_{i=1}^N n_i$. Nótese que desconocemos u_{n_i} pues tras la visita de la página p_{n_i} finaliza la sesión, si quisiéramos usar el tiempo de permanencia en todas las páginas de una sesión deberíamos estimar u_{n_i} .

Definición 4 (Conjunto de *sesiones de navegación*). Sea S el conjunto de todas las sesiones de navegación obtenidas tras la fase de transformación

$$S = \{s_i, i = 1 \dots N\} \quad (1.4)$$

Listado 1.1: Algoritmo de obtención de *sesiones de navegación*

```

sesiones =  $\emptyset$ ;
sesiones_abiertas =  $\emptyset$ ;

forall registroi, i...N do
  select IPi, pi y ti from registroi;
  select sesion from sesiones_abiertas where (IPi ∈ sesion);
  if (∃ sesion) then
    if (ti - sesion.tfin < maxClick) and
      (ti - sesion.t0 < maxSession) then
      add (ti - sesion.tfin, pi) to sesion;
      sesion.tfin = ti;
    else
      CierraSesion(sesion)
      CreaSesion(IPi, pi, ti);
    end
  else
    CreaSesion(IPi, pi, ti);
  end
end

forall sesion ∈ sesiones_abiertas do
  CierraSesion(sesion)
end

```

El algoritmo para obtener las *sesiones de navegación* a partir del *fichero de log* reducido (véase el listado 1.1) no es costoso computacionalmente hablando. En nuestros primeros experimentos utilizamos una duración máxima de sesión de 4 horas y un intervalo máximo de 30 minutos entre dos clickstreams para considerar que una sesión ha finalizado. En el pseudocódigo del algoritmo se

utilizan los valores *maxSession* y *maxClick* para referirnos a estos umbrales respectivamente.

La función *CreaSesion()* recibe como parámetro los tres valores recogidos en el *registro* en que nos encontramos, crea la nueva sesión y la incorpora al conjunto de sesiones abiertas. Como aún no sabemos cuánto tiempo permanece en la página p_i usaremos dos variables auxiliares que marcarán el inicio y fin de la sesión mientras la estamos construyendo, en este caso t_i .

Listado 1.2: Función *CreaSesion()*

```
t0 = tfin = ti;
insert (IPi, t0, tfin, pi) into sesiones_abiertas;
```

La función *CierraSesion()* recibe como parámetro la sesión que ha de cerrarse, a la que se eliminan los datos auxiliares t_0 (inicio de sesión) y t_{fin} (fin de sesión). Estos datos podrían guardarse si van a ser utilizados en las siguientes fases del proceso de *WUM*, para seleccionar sesiones en función de su horario, día de la semana, temporada... En nuestro caso no los utilizamos por lo que podemos prescindir de ellos. Finalmente, la función mueve la sesión del conjunto de sesiones abiertas al conjunto de sesiones, resultado final del algoritmo.

Listado 1.3: Función *CierraSesion()*

```
delete t0 and tfin from sesion;
insert sesion into sesiones;
delete sesion from sesiones_abiertas;
```

Mediante esta transformación ya tenemos los datos modelados en función de lo que estamos buscando: conocer el comportamiento de los usuarios de nuestro sitio web para aprender a sugerir a un usuario las páginas que podría estar buscando en función de las páginas que está visitando en tiempo real. Esta fase consiste, pues, en transformar datos crudos en estructuras que modelen mejor a los individuos que queremos estudiar, en este caso las *sesiones de navegación* de los usuarios de un sitio web.



Figura 1.5: Transformación

En la siguiente fase se transformarán de nuevo de los datos, esta vez para adaptarlos a los algoritmos de *Minería de Datos* a los que serán sometidos. Ya tenemos los datos transformados y guardados mediante una estructura que es fácil de describir usando

gran variedad de métodos estadísticos de análisis de datos en casi cualquier

dispositivo informático.

La Estadística y la Informática llevan muchos años trabajando conjuntamente, con las aportaciones de otras ciencias, en extraer información más o menos compleja de cualquier colección de datos estructurada. Podríamos realizar complejos análisis estadísticos a las *sesiones de navegación* que tenemos pero estamos en un proceso de *Minería de Uso Web* y usaremos estos datos con técnicas de *Minería de Datos*. Podemos realizar sencillos análisis estadísticos descriptivos sobre los datos que tenemos. Sencillos porque a pesar de haber seleccionado, preprocesado y transformado los datos consiguiendo una enorme reducción respecto a la cantidad de datos iniciales aún tenemos muchos datos. Si observamos qué datos tenemos almacenados en las *sesiones de navegación* es fácil obtener conocimiento básico sobre las páginas visitadas por los usuarios del sitio web.

Nosotros estamos trabajando desde la perspectiva de la *WUM* por lo que nos centraremos en la información que tenemos sobre el uso del sitio web, las *sesiones de navegación*. Es inmediato deducir que podemos extraer dos nuevos conjuntos de datos, para conocer qué páginas de P y qué enlaces de E se utilizan en realidad en nuestro sitio web. De hecho el conjunto de páginas visitadas lo podríamos obtener antes de haber creado las *sesiones de navegación*.

Definición 5 (*Páginas visitadas del sitio web*). Sea $P' \subseteq P$ el conjunto de todas las páginas visitadas en el sitio web y $flog_{red}$ el fichero de log reducido.

$$P' = \{P_i \in P \mid P_i \in flog_{red}\} = \{P_i \in P \mid P_i \in S\} \quad (1.5)$$

El primer análisis que se puede llevar a cabo es el estudio de $\neg P'$, las páginas de P que nunca visitan nuestros usuarios. Si son irrelevantes o su contenido ya está en otra página de P' sería bueno eliminarlas, junto a los enlaces que apuntan a ellas, para reducir las dimensiones del sitio web sin perder su verdadera utilidad, ganando en usabilidad. Si son importantes habrá que plantear si debe mejorarse E o simplemente mejorar el texto de los mensajes de los enlaces que apuntan a estas páginas para que sean comprendidos por los usuarios del sitio web. Deben anotarse para poder comprobar en futuros análisis del sitio web si ha tenido efecto el cambio realizado.

La frecuencia de uso de cada página, N_i , es un indicador de su popularidad, pero aún no sabemos si una página es popular por ser el *objetivo* de muchos de nuestros usuarios o simplemente lo es como página de *transición* entre otras

páginas. Si el conocimiento adquirido al final del proceso de *WUM* se integra adecuadamente se podrá observar un cambio en estas frecuencias, en tal caso es posible que páginas que sólo se usan como *transición* desaparecen de muchas las nuevas *sesiones de navegación*.

Definición 6 (Tiempo de permanencia en las páginas visitadas). Sea \vec{u}_i el conjunto de tiempos de permanencia en la página P_i .

$$\vec{u}_i = (u_1, \dots, u_{N_i}) \quad (1.6)$$

En \vec{u}_i guardamos el tiempo de permanencia de algunas de las N_i visitas que ha recibido la página P_i . Si P_i es la última página de una *sesión de navegación* no sabemos cuánto tiempo ha permanecido el usuario en esa página durante esa sesión por lo que no es un dato que conozcamos. Se pueden hacer rápidas estimaciones a partir de \vec{u}_i . Su media, moda o intervalos de confianza podrían darnos información sobre el uso de las páginas del sitio web y orientarnos hacia la siguiente fase, la aplicación de una técnica de *DM* para extraer conocimientos más profundos sobre los datos en estudio.

El conjunto de enlaces de E realmente utilizados por nuestros usuarios forma parte de la información que contienen las *sesiones de navegación*.

Definición 7 (Enlaces internos utilizados en el sitio web). Sea $E' \subset E$ el conjunto de todos los enlaces internos utilizados por los usuarios del sitio web. Se define E'_{jk} cuando existe alguna sesión que contiene la secuencia P_j, n_j, P_k .

$$E' = \{E_{jk} \in E, \mid \exists s_i \in S \mid (P_j, n_j, P_k) \in s_i, j \neq k, j, k \in \{1 \dots M\}\} \quad (1.7)$$

La comparación entre E y E' nos permitirá preguntarnos por qué no se usan los enlaces de $\neg E'$. Para reducir las dimensiones de $\neg E'$ basta con aplicar lo que ya podemos inferir de P' , que si $P_i \notin P' \Leftarrow E_{ij} \wedge E_{ji} \notin E'$. El uso de enlaces está estrechamente relacionado con el uso de las páginas del sitio web, la mayoría de información obtenida al analizar P' coincidirá con la obtenida al analizar E' , sin embargo en E' podemos observar la relación encontrada por los usuarios entre las páginas de nuestro sitio web sin recurrir a técnicas de *DM*.

El número de visitas recibidas por la página P_k , N_k , será mayor o igual al número de enlaces de E' que conduzcan a la página P_k . Sólo coincidirán cuando la página P_k no sea nunca la primera página visitada en una sesión.

La transformación hecha a los datos nos hace comprender mejor lo que estamos estudiando pero aún son muchos datos y no basta con análisis descriptivos para extraer el conocimiento que contienen. En la siguiente fase se abre un gran abanico de posibilidades, ya sabemos qué datos tenemos, cuántos son, podemos preguntarnos qué conocimiento contienen desde diferentes perspectivas y aplicando nuevas transformaciones a nuestro modelo para conseguir datos más adaptados a las herramientas que usaremos para su análisis.

1.2.5. Minería de Datos (DM)

Llegados a esta fase hemos de decidir qué tarea de *Minería de Datos* se adapta mejor a los datos que tenemos y al objetivo final del estudio: *clustering* (Ng y J. Han, 1994; Perkowitz y Etzioni, 1999a, 2000b; Goethals, 2003; Saglam y col., 2006; Tsay, T.-J. Hsu y J.-R. Yu, 2009), *clasificación* (B. Liu y W. Hsu, 1996; B. Liu, W. Hsu y Ma, 1998; Rokach y Maimon, 2014) o *regresión* (Kohavi y Quinlan, 1999). Los objetivos de la *DM* son básicamente *predicción* y descripción, y ambos están estrechamente relacionados. La *predicción* se engloba en las técnicas de *DM* supervisadas mientras que la descripción incluye técnicas no supervisadas de *DM* como *clasificación* o *visualización*. El nexo entre ambos objetivos es evidente en el análisis de grandes colecciones de datos: si no se realiza un buen proceso descriptivo de los mismos será difícil obtener un buen modelo de *predicción*.

En la *Minería de Datos* se debe plantear bien el problema a resolver. El proceso de *DM* es sólo una fase del proceso de *KDD*, una pequeña parte del engranaje como muestra la figura 1.1, pero si no se ejecuta correctamente no puede obtenerse conocimiento válido y previamente desconocido. Una vez seleccionados, preprocesados y transformados los datos y elegida la tarea de *DM* que queremos aplicar hemos de decidir con qué algoritmo abordaremos esta fase y qué parámetros se ajustan mejor a los datos existentes y los resultados que esperamos obtener.

Actualmente es una disciplina en la que trabajan investigadores de todas las ramas del saber. Colecciones masivas de datos (en aumento), dispositivos capaces de almacenarlas, distribuirlas y procesarlas y nuevas metodologías para su conversión en conocimiento están al alcance de todos.

La Estadística y la Informática son las ciencias que más aportan a la *Minería de Datos*. No hay estudio de *DM* que no intente justificar la bondad de sus resultados con la Estadística, aunque las justificaciones utilizadas no están siempre

basadas en un profundo conocimiento de la Estadística. No sería posible la *DM* sin el concurso de la Informática, aunque muchas aportaciones no son realmente útiles a la comunidad científica hasta que se desarrollan con eficiencia.

En esta fase se trata de aprovechar la potencia de cálculo de los actuales dispositivos informáticos para analizar relaciones, similitudes y diferencias entre los datos transformados, creando *patrones* que pueden ser visualizados de distintos modos para conseguir con ello que el investigador o el propio sistema detecte *patrones* específicos o extraños. Perkowitz y Etzioni (1998, 1999a, 1999b, 2000b) utilizan estos *patrones* para dividir el sitio web en clusters, conjuntos de páginas que son visitadas de forma conjunta y que son analizadas por el administrador del sitio web para evaluar los resultados y mejorar la *experiencia de usuario* si las agrupaciones obtenidas son correctas. Los análisis de *clustering* pretenden dividir la población en grupos cuyos individuos sean homogéneos entre sí, de modo que los diferentes clusters sean lo más heterogéneos posible entre sí, basándose en diferentes medidas de similitud. Schafer, Konstan y Riedl (2001) dividen la gran población de usuarios de un e-comercio en diferentes clusters, de modo que cuando se quiere hacer una recomendación a un usuario se consultará únicamente en el cluster en que mejor se identifique al usuario con lo que el número de datos a procesar será mucho menor y los resultados pueden ser mucho más acertados si se ha hecho correctamente la asignación.

La Estadística es la base de la *Minería de Datos* por lo que muchas de sus técnicas se aprovechan, como el muestreo aplicado a los datos originales en H. C. Ozmutlu, Spink y S. Ozmutlu (2002) tras el que aseguran obtener un subconjunto manejable de datos con la misma información que el conjunto original.

La *Minería de Reglas de Asociación (ARM)* es una técnica de *DM* propuesta por Agrawal, Imielinski y A. Swami (1993b). Gran parte de la presente investigación gira en torno a la *ARM*. En ciertos trabajos se ha tratado como una tarea análoga a la de *clasificación* (Kohavi y Quinlan, 1999), derivando en la llamada *Clasificación Asociativa* (B. Liu, W. Hsu y Ma, 1998). La *clasificación* se enfoca hacia la *predicción* y la *ARM* descubre asociaciones entre los valores de los atributos en estudio, lo que puede aprovecharse en algoritmos como CMAR (Thabtah, Cowling y Hamoud, 2006) o AR+OPTBP (Kahramanli y Allahverdi, 2009).

La ejecución del algoritmo seleccionado no tiene por qué ser única, generalmente se llevarán a cabo diversos ajustes que eviten la obtención de resultados ya conocidos y proporcionen conocimiento previamente desconocido sobre la población en estudio. Cuanto más dinámica sea la población de origen de los datos en proceso más elaborada ha de ser esta fase pues se puede llegar a concluir que

el conocimiento previo a este estudio ya no es válido o que es necesario obtener nuevos datos que permitan llegar a nuevas conclusiones del análisis.

Volviendo al proceso de *Minería de Uso Web* en estudio, una vez tenemos el conjunto de *sesiones de navegación* podemos hacer rápidas averiguaciones basándonos en la estadística descriptiva e inferencial para ayudarnos a entender mejor los datos que tenemos. Las *sesiones de navegación* son los datos de los que vamos a extraer el conocimiento. Para tener un rápido acceso a ellas lo conveniente es guardarlas en una *Base de Datos* y codificarlas de modo que en lugar de anotar las páginas web con largas cadenas de caracteres se guarde un valor numérico que las identifique unívocamente y que sea fácil de manejar por un dispositivo informático, el uso de códigos hash es muy apropiado para este menester.

Las *sesiones de navegación* representan *secuencias*, conjuntos de ítems ordenados que son fácilmente modelables como cadenas de Markov y estudiadas mediante *n*-gramas o analizadas mediante algoritmos adaptados a las *secuencias* como AprioriAll, AprioriSome o DynamicSome (Agrawal y Srikant, 1995) y GSP (Srikant y Agrawal, 1996).

El análisis de la *DB* de *sesiones de navegación* puede estar dirigido a diferentes objetivos, interesándonos en esta fase de la investigación la *predicción* de las páginas web que desean visitar nuestros usuarios. En esta línea se han propuesto muy diversos planteamientos sobre cómo sintetizar la información que contiene la (generalmente enorme) *Base de Datos* de sesiones. Entre otras soluciones se propone el uso de *Hypertext Probabilistic Grammar* (Borges y Levene, 1999), el uso de *Programación Genética Basado en Gramática* (Hervás-Martínez, Romero Morales y Ventura Soto, 2004a, 2004b), el sistema de *predicción* WhatNext, basado en *n*-gramas (Su y col., 2000) o el modelo multi-step dynamic *n*-gram (Sun y col., 2002).

En nuestros primeros pasos en esta investigación, desarrollados en la sección 1.3.1, nos inclinamos por el uso de *grafos* dirigidos para obtener *patrones* de navegación de los usuarios en la web. Los *grafos* son modelos matemáticos con los que se pueden expresar las conexiones que realizan los usuarios entre las páginas de un sitio web al navegar por él.

El uso de *grafos* aporta mucha información al proceso de *WUM*, sin embargo al seguir creciendo nuestras colecciones de datos se va complicando el uso de este tipo de estructuras. Tras una nueva revisión del estado del arte sobre la materia comprobamos que en muchos estudios se ignoraba este orden, viendo una *sesión de navegación* como un conjunto (no ordenado) de páginas visitadas en una

misma sesión. Al no considerar este orden pasamos de tener *secuencias* a tener *transacciones*, lo que dio lugar a la investigación mostrada en la sección 1.3.2.

Al convertir una *sesión de navegación* en una *transacción* se pierde toda la información sobre los enlaces utilizados y sobre el tiempo de permanencia en cada página por lo que hemos de analizar qué información contiene la estructura de datos que hemos seleccionado. Una *transacción* es simplemente un conjunto de datos, en nuestro caso un conjunto de páginas visitadas, ni siquiera se guarda el número de veces que se ha visitado la misma página en una *sesión de navegación*. Ahora cada uno de los registros que tenemos contiene simplemente un conjunto de páginas web que un usuario decidió visitar durante una sesión en nuestro sitio web. Si muchos usuarios agrupan las mismas páginas podremos pensar que hay algo que las relaciona por lo que si un usuario estuviera visitando una de esas páginas podríamos recomendarle visitar el resto de páginas del mismo grupo.

Esto nos obligaba a abordar el uso de técnicas de *Minería de Reglas de Asociación (ARM)*, que permiten extraer información sobre qué páginas están relacionadas entre sí a través del análisis de las *transacciones* obtenidas. Variamos nuestra investigación hacia el uso de *ARM* por la sencillez del algoritmo *Apriori* y la fácil comprensión de las reglas obtenidas. Pronto descubrimos que a pesar de la reducción de datos obtenida con este cambio eran todavía muchas las reglas obtenidas y sería difícil tratarlas simultáneamente sin antes aplicar otro tipo de preproceso y transformación de los datos a minar.

1.2.6. Evaluación e Integración

La evaluación e interpretación de los *patrones* obtenidos, teniendo en cuenta los objetivos planteados en el proceso completo, ha de poner el foco en la comprensión y utilidad del modelo obtenido. Se debe documentar correctamente en esta fase el conocimiento adquirido para poder utilizarlo en posteriores procesos. El primer problema que se presenta en esta fase es el exceso de conocimiento adquirido. Hemos cambiado una gran cantidad de datos iniciales por una gran cantidad de conocimiento ¿podremos analizar todo este conocimiento en profundidad? Un sistema informático puede ayudar al analista del proceso en este punto, como apuntan B. Liu y W. Hsu (1996), utilizando diferentes criterios para decidir qué conocimiento es más *útil* en el caso concreto que estamos evaluando.

La integración de este conocimiento en otro sistema que sea capaz de obtener provecho del mismo es el objetivo final de cualquier proceso de *KDD*. El proceso se ha realizado a partir de una «foto fija» de la población en estudio, los datos

obtenidos y seleccionados en las primeras fases. Ahora han de integrarse en un sistema que sea capaz de traducir el conocimiento adquirido en beneficios para los usuarios y los propietarios del sistema (mayor usabilidad, personalización, recomendaciones más acertadas, ventas. . .). Esta transición no es necesariamente sencilla pero se escapa al proceso expuesto en este trabajo.

La última fase del proceso es la que da validez al mismo: integrar el conocimiento al sistema de modo que pueda ser usado y evaluar los resultados obtenidos. En este punto se ha de comprobar qué conocimientos se tenían al principio del proceso y qué conocimientos nuevos se han adquirido tras el análisis de los datos.

Al afrontar un proceso de *Minería de Uso Web* se hace bajo la creencia de que un usuario no puede decidir sobre el diseño del sitio, pero muchos usuarios sí podrían decidir. Si un gran número de usuarios visitaran conjuntamente las páginas *A*, *F* y *H*, el webmaster podría definir los enlaces entre ellas, lo que mejoraría su rango de alcance. Una mejora en la facilidad de uso del sitio web que no requiere la intervención del webmaster es la *predicción*: presentar a cualquier usuario que visite la página *A*, enlaces a las páginas *F* y *H*, aunque realmente no se encuentren en el diseño inicial de la página *A*. Para lograrlo se debe estudiar previamente el uso del sitio, detectando si un porcentaje considerable de usuarios que solicitan el recurso *A* también solicitan los recursos *F* y *H* en la misma sesión (Ng y J. Han, 1994). Existen varios trabajos planteando diversas alternativas a esta solución (Su y col., 2000; Sun y col., 2002; X. Chen y X. Zhang, 2003), todos ellos con un esquema común: considerar la información que aportan todos los usuarios, analizarla más o menos exhaustivamente para obtener *patrones* de comportamiento de los usuarios en general y resumirla de modo que con pocos recursos podamos ayudar al máximo al usuario anónimo, mediante la *predicción* del próximo recurso que solicitará.

Esta fase no la pudimos llevar a cabo durante el transcurso de esta investigación por no tener acceso al código de un sitio web real en explotación. Si no podemos modificar dinámicamente las páginas sugeridas por el sitio web a sus usuarios en función de las páginas que está visitando no podremos contrastar los datos iniciales con los nuevos datos. No podemos observar si una página ha dejado de ser visitada por servir únicamente para transitar entre dos páginas entre las que no existía originalmente un enlace directo. No podemos observar si un enlace ha dejado de usarse por ser inicialmente un enlace que los usuarios esperaban les llevara a su objetivo debido al texto engañoso del enlace. No podemos observar si los usuarios dejan de abandonar nuestro sitio web por encontrarlo más adaptado

a su modo de trabajar.

Esta fase es fundamental para un *Sistema de Recomendación Web* por lo que este informe sería incompleto si no hubiéramos cambiado el rumbo de nuestra investigación, dedicando nuestros esfuerzos a la tarea de *Minería de Datos* del proceso inicial abordado.

1.2.7. CONOCIMIENTO

El *conocimiento* es el resultado del proceso de *KDD*. No es algo único que una vez adquirido pueda ser anotado en un libro para no necesitar ser «calculado» de nuevo. Los datos originales cambiarán, aumentarán o dejarán de tener relevancia por la desaparición de un artículo, su análisis puede dar como resultado nuevo *conocimiento*, complementario, independiente o contrario al adquirido en procesos previos. Su correcta gestión es la que logra una adaptación real de los servicios de una empresa a los usuarios del servicio.

El proceso de *KDD* ha de realizarse continuamente, estudiar el acierto de los parámetros usados en sus distintas fases y hacer variaciones de los mismos para mejorar la calidad del conocimiento adquirido, comprobar el dinamismo de la población en estudio y el efecto que producen los cambios realizados al servicio. No siempre se puede utilizar todo el conocimiento adquirido, hay que decidir qué cambios son realmente aplicables en la empresa que ha realizado el proceso de descubrimiento de conocimiento.

En un *Sistema de Recomendación Web* se busca conocer lo que quiere el usuario antes de que éste lo solicite para poder preparar enlaces a las páginas que se le van a sugerir en función de las que el usuario vaya visitando. Un sitio web con muchas páginas debería tener una estructura de navegación bien diseñada para que los usuarios pudieran obtener con facilidad las páginas de su interés. Este diseño suele ser realizado por expertos en contenidos o en diseño web, y a menudo no tienen en cuenta los aspectos de usabilidad y/o adaptabilidad de un sitio web (Peñarrubia, Fernández-Caballero y González, 2004). El usuario real del sitio nunca interviene en este diseño, si deseara las páginas *A*, *F* y *H*, pero el experto decide que existe una relación más directa entre las páginas *A*, *B* y *C*, éstas serán las páginas con enlaces de navegación y no aquellas que realmente el usuario quería visitar. Esto ocasionará que el usuario emplee mucho tiempo en encontrar lo que busca o en el peor de los casos que no lo encuentre, a no ser que se aplique un buen proceso de *Minería de Uso Web* que permita la adaptación progresiva de las recomendaciones del sitio web, lo que podría derivar

en un pleno conocimiento de cómo usan el sitio web sus usuarios y una perfecta adaptación del sitio web a sus usuarios.

1.3. Minería de Datos

La abundancia de grandes colecciones de datos y la necesidad de extraer de ellos conocimiento desconocido ha lanzado a muchos investigadores de diversas disciplinas al desarrollo de algoritmos y metodologías con el objetivo común de descubrir *patrones* ocultos entre los datos (J. Han, Kamber y Pei, 2011). Una vez vistas las fases del proceso de *Minería de Uso Web* nos centraremos en las posibilidades que nos aporta la *Minería de Datos* en función de los datos obtenidos tras la fase de transformación, en nuestro caso las *sesiones de navegación*. Aún deberemos transformar estas sesiones en otro tipo de datos que sean realmente manejables por los algoritmos de *DM* que decidamos aplicar. De momento tenemos un conjunto de N sesiones, cada una de ellas con la siguiente forma:

$$s_i = (userID, p_1, u_1, p_2, u_2 \dots p_{n_i-1}, u_{n_i-1}, p_{n_i}), i = 1 \dots N$$

Estas *sesiones de navegación* pueden ser representadas de diferentes modos, destacando dos modelos matemáticos sobre el resto: las *secuencias* y las *transacciones*. En las *secuencias* es importante el orden en que se visitaron las páginas en una misma *sesión de navegación*, en las *transacciones* sólo nos interesamos por el hecho de que dos o más páginas han sido visitadas en la misma sesión. En una *secuencia* se puede averiguar cuántas veces aparece una página en cada sesión, incluso si el usuario simplemente ha refrescado su navegador (F5) y disponemos de mecanismos para detectarlo, en una *transacción* no se suele guardar esta información por lo que sólo sabremos si una página ha sido visitada o no durante una sesión.

Decidimos comenzar por el estudio de *secuencias* para poder analizar qué importancia tiene el orden de navegación en el análisis de las *sesiones de navegación* de nuestro sitio web (Agrawal y Srikant, 1995; Srikant y Agrawal, 1996; Borges y Levene, 1999). Es de esperar que cuanto más información analicemos más conocimiento adquiriremos por lo que sugerimos el uso de *Mapa de Navegación Web* para representar las *sesiones de navegación* y analizarlas mediante técnicas de Teoría de Grafos.

1.3.1. Mapas de Navegación Web

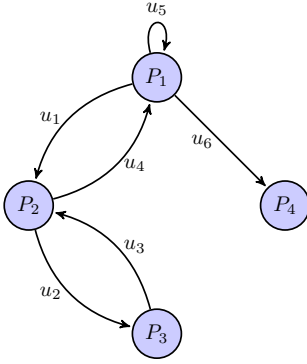


Figura 1.6: Sesión de navegación

Para analizar mejor una *sesión de navegación* vamos a dibujarla. Primero dibujamos un círculo representando p_1 , luego dibujamos p_2 (si es distinta a p_1) y una flecha que va de p_1 a p_2 y en ella anotamos u_1 . Seguimos dibujando las páginas que no estén ya en el dibujo y la flecha que la une con la anterior página dibujada, con su u_i correspondiente. Cuando completamos el dibujo observamos que tenemos un *grafo* compuesto y dirigido que modeliza una *sesión de navegación* y sobre el que se podrán aplicar muchos conocimientos derivados de la Teoría de Grafos.

Es un enfoque matemático para observar una *secuencia*. Para crear cada *grafo* hay que leer una *sesión de navegación* e ir agregando nodos y ejes a un *grafo* vacío, considerando si ya existe el nodo a insertar. Los nodos son las páginas visitadas p_j , el tiempo u_j que ha permanecido el usuario en dicha página es el peso del enlace a la siguiente página visitada en la sesión, p_{j+1} . La última página de la sesión no genera ningún eje. Cada sesión $s_i, i = 1 \dots N$, se convierte en el *grafo* G_i .

$$\begin{aligned}
 G_i = & (P', E') \mid \\
 & P' = \{P'_j, j = 1 \dots n_i\} \wedge \\
 & E' = \{(E'_{jk}, u_j), 1 \leq j \leq n_{i-1}, 1 \leq k \leq n_i\}
 \end{aligned} \tag{1.8}$$

La representación gráfica de una *sesión de navegación* puede ayudar al analista a comprender mejor el uso que se hace del sitio web. Cada nodo hoja puede catalogarse como un *objetivo* del usuario o un intento fallido tras el que decide volver por donde vino. Si permanece mucho tiempo en ese nodo es más probable que sea su objetivo, en cuyo caso deberíamos facilitarle acceso directo desde la página en que inició la búsqueda de su objetivo. Todos los nodos en que permanece mucho tiempo un usuario podrían catalogarse como objetivos en la sesión. Aquellos en que permanece poco tiempo se pueden catalogar como recursos intermedios que podrían no ser visitados si existieran enlaces directos a sus objetivos. Pero todo esto lo han de mostrar los propios datos, y en la fase de evaluación debería comprobarse la calidad de las estimaciones hechas.

Si catalogamos cada página P_j^i como *objetivo* o *de transición* en función de su vector \vec{u}_j , definido mediante la ecuación 1.6, ya estamos haciendo estimaciones, intentando que sea un dato concreto, el tiempo de permanencia en una página, quien nos permita hacer una primera *clasificación* de las páginas del sitio web. Gráficamente se aprecia en seguida el efecto de esta *clasificación*, como muestra la figura 1.7. Hemos de recordar que estamos trabajando con *DM* por que serán muchos los *grafos* que obtengamos, y además los quereamos tratar en tiempo real por lo que este tipo de representaciones sólo se hará para que el analista pueda ver la información que tiene desde distintas perspectivas.

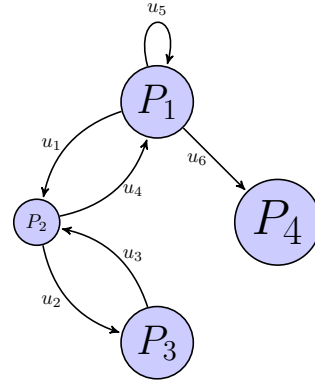


Figura 1.7: Sesión de navegación ponderada

No es casualidad que la estructura elegida sea un *grafo* compuesto, es el efecto de utilizar papel y lápiz para dibujar una *sesión de navegación*. Sin embargo un *grafo* compuesto utiliza muchos recursos de memoria para guardar los múltiples ejes que pueden unir dos nodos. Podemos guardar casi la misma información en un *grafo* simple y dirigido cuyos nodos contienen la página visitada y el vector de tiempos de permanencia y sus ejes están ponderados con el número de veces que se ha usado ese enlace en esa sesión, como muestra la figura 1.8. En este caso no podremos relacionar los tiempos de permanencia con la página de la que proviene, al trabajar con *grafos* compuestos sí teníamos esa información pero no sabíamos como usarla por lo que vamos a prescindir de ella trabajando con *grafos* simples.

Incluso la información sobre \vec{u}_j se podría sustituir por un único valor representativo del vector de tiempos de permanencia en la página p_j (su media aritmética, moda...). Depende del uso que le vayamos a dar puede ser conveniente sustituir todos esos datos por uno que los describa adecuadamente y liberar memoria para los costosos procesos de *Minería de Datos* a que vamos a someter los datos. Como en todo proceso de *KDD* el valor utilizado para representar \vec{u}_j puede producir variaciones sensibles en los resultados obtenidos por lo que se deberían hacer pruebas con diferentes valores representativos de esta variable hasta encontrar uno que proporcione conocimiento de mayor calidad.

Con todas estas transformaciones hemos definido el *Mapa de Navegación*

Web (MNW), mostrado en la figura 1.8, del que queremos extraer toda la información posible sobre el uso del sitio web en estudio.

Definición 8 (*Mapa de Navegación Web*).

$$MNW_i = (P', E') \mid \begin{aligned} P' &= \left\{ \left(P'_j, \vec{u}_j \right), j = 1 \dots n_i \right\} \wedge \\ E' &= \left\{ \left(E'_{jk}, n_{jk} \right), 1 \leq j \leq n_{i-1}, 1 \leq k \leq n_i \right\} \end{aligned} \quad (1.9)$$

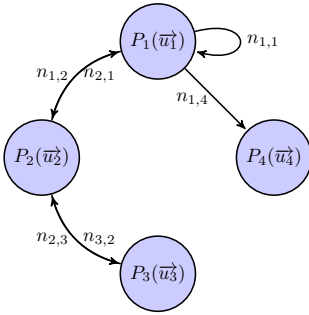


Figura 1.8: MNW

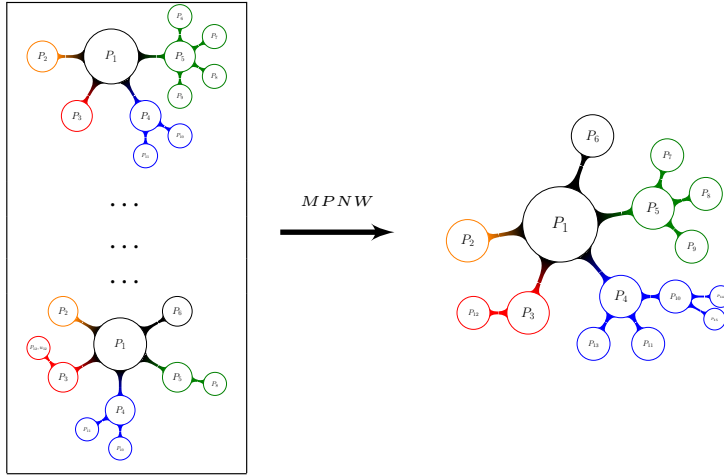
Si tenemos el mapa del sitio web podemos comprobar qué enlaces existentes en E no se utilizan y preguntarnos por qué. En muchas ocasiones es sólo por cuestión de su ubicación, tamaño o etiqueta (por no ser muy explícita o no estar bien redactada). Hay que recordar que si estamos trabajando en un sitio web que hace recomendaciones de forma dinámica los enlaces de E cambiarán constantemente y estas comprobaciones podrían carecer de sentido si se analizan datos antiguos.

La unión de todos los *Mapas de Navegación Web* de un mismo usuario proporciona su *Mapa Personal de Navegación Web*, mostrado en la figura 1.9.

Definición 9 (*Mapa Personal de Navegación Web*).

$$MPNW_{userID} = \bigcup_{i=1}^N MNW_i \mid sesion_i.user = userID \quad (1.10)$$

En el *Mapa Personal de Navegación Web* está recogida toda la información sobre las *sesiones de navegación* realizadas por el usuario por lo que cada vez que el usuario solicita una página del sitio web se busca dicha página en su $MPNW_{userID}$ y, si está, se recomendarán algunas de las páginas que estén conectadas con la solicitada mediante un eje o un camino, de dos ejes como máximo para no ralentizar esta fase, ordenando las recomendaciones en función del tiempo de permanencia que tiene cada una de las páginas sugeridas en su $MPNW_{userID}$. Podemos ir más allá y extender la búsqueda de páginas a recomendar a otro nivel utilizando caminos más largos y comprobando si las nuevas sugerencias son mejor acogidas por los usuarios del sitio web.

Figura 1.9: Obtención del *MPNW*

Si tenemos m usuarios, $userID = 1 \dots m$, podemos modelizar el uso general del sitio web mediante la unión de todos sus *MPNW*, obteniendo un *Mapa de Uso del Sitio Web (MUSW)*.

Definición 10 (*Mapa de Uso del Sitio Web*).

$$MUSW = \bigcup_{userID=1}^m MPNW_{userID} \quad (1.11)$$

El *MUSW* se puede utilizar para hacer recomendaciones a usuarios que no hayan estado antes en nuestro sitio web, ya que no tendremos información sobre su $MPNW_{userID}$. Se usará del mismo modo que usamos los mapas personales, buscando en el *grafo* la página solicitada por el usuario y añadiéndole enlaces con recomendaciones basadas en los ejes que salen de dicha página en el *grafo*, ponderadas por el tiempo de permanencia de las páginas destino de cada eje.

Si un usuario ya ha estado en nuestro sitio web con anterioridad y tenemos su $MPNW_{userID}$ podemos ofrecerle mejor información si tenemos en cuenta tanto su $MPNW$ como el *MUSW*. Para hacerlo en tiempo real nos hemos de limitar a consultar únicamente qué enlaces tiene la página solicitada por el usuario en ambos mapas y extraer inmediatamente el mejor conocimiento de estos datos.

Estamos intentando trabajar con muchos datos simultáneamente. Al catalogar las páginas se consigue una gran reducción de datos, si no se pierde información

habremos hecho un gran avance. Aún así seguimos trabajando con muchos datos, sería deseable poder dividir el gran conjunto de *grafos* que queremos analizar en conjuntos más pequeños y gestionables. Existen para ello muchas técnicas de *clustering* (Ng y J. Han, 1994), la idea siempre es la misma: agrupar a los individuos que más se parezcan entre sí y más se diferencien del individuo representativo de los otros grupos. La cuestión es ahora determinar qué función de «similitud» se debe aplicar a nuestros mapas de navegación y cuál será su coste computacional.

Nuestros *grafos* están compuestos por las páginas del sitio web, los enlaces utilizados para navegar a través de ellas y el tiempo estimado de permanencia en cada página (o su catalogación como *objetivo* o *de transición* o *intento fallido*). ¿Cómo determinamos que dos *sesiones de navegación* son «parecidas»? En primer lugar deberíamos observar si contienen los mismos nodos, si se han visitado las mismas páginas o al menos un conjunto de ellas coinciden, a continuación podríamos observar la similitud en cuanto a tiempo de permanencia en las páginas visitadas en ambas sesiones y, por último, la coincidencia en el uso de enlaces haría más próximos ambos *grafos*. Comparar todos estos datos en una colección grande de *grafos* no es tarea trivial, no podemos esperar resultados en tiempo real pues hay que hacer muchas operaciones complejas para determinar si dos *grafos* son similares:

1. calcular su intersección, comprobar qué páginas del primer *grafo* están en el segundo,
2. comprobar si la intersección obtenida es representativa para ambos *grafos*,
3. comparar los tiempos de permanencia en las páginas seleccionadas,
4. comprobar si los enlaces utilizados en ambas sesiones presentan similitudes.

Si trasladamos esto a la comparación de más de dos *grafos* se complica el asunto. La intersección de muchos *grafos* puede no contener ningún nodo con lo que habríamos perdido toda la información disponible. Al obtener la intersección de dos *grafos* deberíamos considerar también los nodos (páginas visitadas) que no pertenecen a la intersección pero sí a uno de los *grafos*, lo que se podría conseguir a nivel de usuario haciendo la intersección de un *grafo* con el *Mapa Personal de Navegación Web* del usuario o con el *Mapa de Uso del Sitio Web* y considerando

los nodos adyacentes a estas intersecciones en el mapa utilizado como posibles páginas a recomendar al usuario.

A pesar de disponer de muchas herramientas de Teoría de Grafos para llevar a cabo el estudio actual, al poner a prueba nuestra propuesta no se pudieron guardar todos los resultados obtenidos para poder manipularlos en tiempo real. Uno de los retos de la *Minería de Datos* es extraer de enormes cantidades de datos la máxima información siendo conscientes de que trabajamos siempre con recursos limitados, lo que nos hizo replantear el análisis propuesto reduciendo de nuevo los datos a gestionar. Aunque los enlaces utilizados por los usuarios son quienes nos dan más información sobre la navegación a través de nuestro sitio web decidimos prescindir de ellos para centrarnos en analizar las páginas visitadas en una *sesión de navegación*, sin conservar el orden en que se visitó cada página ni el tiempo de permanencia estimado para cada una de ellas. Aún así el número de datos a gestionar era muy grande por lo que tuvimos que buscar metodologías aplicables a esta nueva situación.

1.3.2. Reglas de Asociación

Las *Reglas de Asociación* fueron introducidas por Agrawal, Imielinski y A. Swami (1993b) y la *Minería de Reglas de Asociación* popularizada con su algoritmo *Apriori* (Agrawal y Srikant, 1994a, 1994b), cuyo pseudocódigo se muestra en los listados 1.4, 1.5, 1.6 y 1.7. Las definen a través del ejemplo de la cesta de la compra en un supermercado.

1. El supermercado tiene M productos distintos.
2. Cada «cesta de la compra» se registra mediante un dispositivo informático en forma de *ticket*, que puede contener información del cliente (tarjetas de fidelización) y del empleado, fecha y hora de la compra y un listado de los productos adquiridos por el cliente junto con detalles como cantidad, precio, descuentos. . .
3. De cada ticket consideramos únicamente el conjunto de productos adquiridos, sin importarnos su orden en el *ticket* ni si se ha adquirido una o más unidades. Convertimos el *ticket* en una *transacción*.
4. Si comparamos cientos o miles de *transacciones* podremos describir la relación de co-existencia de dos o más productos en una misma compra. Estas relaciones nos permitirán afirmar que, en la muestra observada, «el

10 % de clientes compran pan y leche en la misma compra». O incluso que «el 60 % de los clientes que compran pan también compran leche en la misma compra».

5. Este conocimiento debe convertirse en estrategias de marketing para facilitar a sus clientes la construcción de su próxima «cesta de la compra». Debe extrapolarse a la población de clientes el conocimiento adquirido al analizar la muestra. Y debe analizarse la muestra con algoritmos rápidos pues cada día se pueden generar cientos o miles de *transacciones* con información aún sin analizar.

Según su propia experiencia, tras generar gran número de *reglas de asociación* a partir de gran cantidad de *transacciones* de un supermercado descubrieron que «los viernes, muchos de los clientes compran pañales y cervezas», lo que les llevó a sugerir al propietario del comercio que los viernes promocionara ambos productos de manera conjunta. La base científica de las *Reglas de Asociación* es la Estadística Descriptiva, sin embargo hacer un planteamiento similar con métodos estadísticos no nos llevaría a descubrir algo tan sorprendente ya que en Estadística hemos de formular la hipótesis antes de observar los datos, y llegar a formular esa hipótesis concretamente resulta difícil de imaginar.

Formalicemos su propuesta para poder entenderla mejor. El punto de partida es un conjunto de M ítems al que llamaremos \mathcal{I} . Tenemos muchas *transacciones*, subconjuntos de \mathcal{I} obtenidos en determinadas circunstancias, y nos preguntamos qué ítems están presentes de forma conjunta en las *transacciones*.

Definición 11 (Población de ítems). *Sea \mathcal{I} un conjunto con M ítems distintos. Llamaremos itemset a cualquier subconjunto de \mathcal{I} , o bien k -itemset si queremos indicar en su nombre su tamaño.*

$$\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_M\} \quad (1.12)$$

En la cesta de la compra, \mathcal{I} es el conjunto de todos los productos existentes en el comercio. En una tarea de *clasificación* se suelen codificar todos los pares atributo=valor mediante números enteros consecutivos, con lo que \mathcal{I} contiene únicamente un conjunto de números que representan todos los valores que pueden tomarse en los diferentes atributos en estudio. En *Minería de Uso Web* es el conjunto de páginas del sitio web (ver definición 1, pág. 27).

Definición 12 (*Transacción*). Una transacción \mathcal{T} es un itemset, un subconjunto de \mathcal{I} , obtenido en determinadas circunstancias.

$$\mathcal{T} \subseteq \mathcal{I} = \{I_j \mid I_j \in \mathcal{I}, j = 1 \dots k\} \quad (1.13)$$

En la cesta de la compra una *transacción* es parte de la información recogida en un ticket, es el conjunto de productos que ha comprado el cliente en una compra. O podríamos crear las *transacciones* con el conjunto de productos que han comprado los clientes identificados durante el último mes, con lo que tendríamos menos transacciones y podríamos buscar *patrones* de compra entre nuestros clientes, no entre cada una de sus compras individuales. En un problema de *clasificación* cada registro en el que se han anotado los valores que toma un individuo para cada uno de los atributos en estudio es una *transacción*. En *WUM* una *transacción* es el conjunto de páginas que ha visitado un usuario en una *sesión de navegación*.

Definición 13 (Almacén \mathcal{D}). Un almacén \mathcal{D} es un conjunto de transacciones.

$$\mathcal{D} = \{\mathcal{T}\} \quad (1.14)$$

He definido el almacén de *transacciones*, \mathcal{D} , para diferenciarlo de la *DB* que contiene todos los datos que podemos tener recogidos. La *Minería de Reglas de Asociación*, como técnica de *Minería de Datos* sólo es una fase de un proceso de *KDD* por lo que se aplicará después de seleccionar, pre-procesar y transformar los datos originales. Cualquier cambio en los parámetros usados en estas fases provocará la obtención de un almacén \mathcal{D} diferente al que habría que analizar por completo.

En la cesta de la compra podríamos crear \mathcal{D} con todos los tickets de un día, de una semana, de un día concreto de la semana durante los últimos 3 meses... En *WUM* podríamos crear \mathcal{D} a partir de las *sesiones de navegación* usando criterios similares o de geolocalización de la IP... No tiene sentido en muchos casos intentar aplicar técnicas de *Minería de Datos* a colecciones extremas de datos, a todo el historial de tickets o al conjunto de todas las *sesiones de navegación* de nuestro sitio web. Si el almacén \mathcal{D} es tan grande tendremos que renunciar a mucha de la información que posee, recordemos que estamos investigando cómo

extraer el máximo de información de un gran conjunto de datos utilizando equipos informáticos al alcance de todos los investigadores. Las aportaciones de esta tesis se pueden extrapolar al uso de supercomputadoras si se dispone de alguna de ellas para verificar resultados y se tiene un buen conocimiento de su uso, pero no es asunto a tratar en este informe.

Al poder crear \mathcal{D} de un modo rápido y con diferentes criterios se abre el campo de posibilidades de uso de las *reglas de asociación*. Si podemos generar un almacén \mathcal{D} con información específica para un análisis y extraer de él todas las *reglas de asociación* representativas en un tiempo adecuado para el estudio que estamos haciendo podremos adaptar nuestros análisis utilizando la información proporcionada por los propios datos, que pueden volver a ser generados con nuevos criterios y analizados para obtener conocimiento de mayor calidad o para confirmar o descartar el conocimiento previo al análisis.

El objetivo del análisis es encontrar las *Reglas de Asociación* que contiene el almacén \mathcal{D} , reglas con la forma $X \rightarrow Y$ que nos informan sobre el número de veces que aparece el *itemset* Y en las transacciones que contienen el *itemset* X .

Definición 14 (Regla de Asociación). *Una Regla de Asociación es una expresión del tipo $a_i \rightarrow c_i$, donde a_i y c_i son dos itemsets mutuamente excluyentes ($a_i \cap c_i = \emptyset$).*

$$R_i = \{a_i \rightarrow c_i\} \quad (1.15)$$

a_i recibe el nombre de antecedente y c_i el de consecuente de la regla de asociación.

Las *reglas de asociación* se obtienen al examinar a fondo un almacén de *transacciones* y lo único que nos dirán es qué relaciones de co-ocurrencia existen entre los ítems en estudio. La aportación de la *Minería de Reglas de Asociación* es la posibilidad real de hacer el análisis de tremendas colecciones de conjuntos de datos. Algo tan aparentemente simple se ha de llevar a la práctica con cierta prudencia pues los recursos disponibles en la mayoría de dispositivos informáticos son limitados. A pesar de contar con muchas herramientas derivadas de la *Teoría de Conjuntos*, al tratar de implementarlas en un dispositivo informático y manipular grandes cantidades de datos aparecerán enseguida problemas de desbordamiento de memoria, por lo que se entiende que la *Minería de Reglas de Asociación* sea una disciplina más investigada en el ámbito de la Informática que en el de la Estadística.

Definición 15 (Minería de Reglas de Asociación). *La Minería de Reglas de Asociación es el proceso de búsqueda de Reglas de Asociación en grandes almacenes de transacciones utilizando los recursos informáticos disponibles y en un tiempo apropiado.*

El análisis propuesto es útil para muchas disciplinas en la época de la tecnología y el *Big Data*. Las *transacciones* sirven para modelar gran cantidad de estudios y las *reglas de asociación* puede convertirse en conocimiento muy útil en distintas áreas de investigación si se adquiere a tiempo de poder ser utilizado.

- La cesta de la compra se puede generalizar a cualquier tipo de comercio, recibiendo mucha atención en el ámbito del *e-comercio*, donde muchas recomendaciones se hacen en base al conocimiento adquirido por el uso de *reglas de asociación*.
- En muchos estudios de *clasificación* se codifican los datos observados a cada individuo en un registro con pares atributo=valor y se realiza el proceso de *clasificación* observando la co-ocurrencia de los datos de los distintos registros. Las *reglas de asociación* proporcionan justo esta información por lo que han generado el estudio de las *reglas de clasificación* (B. Liu, W. Hsu y Ma, 1998; Thabtah, Cowling y Hamoud, 2006; Kahramanli y Allahverdi, 2009).
- Cuando un médico solicita diferentes análisis está adquiriendo datos del paciente, un conjunto de pares atributo=valor que puede ser observado como una *transacción* si los valores son categóricos o se pueden categorizar. Si pudiera comparar esta *transacción* con las *transacciones* de otros pacientes a los que se está estudiando o ya han sido diagnosticados quizá tendría información de mayor calidad para poder emitir un diagnóstico o solicitar un nuevo análisis.
- En un sondeo socio-político con respuestas categóricas se podría tratar cada encuesta individual como una *transacción*. La interpretación del sondeo se podría beneficiar del descubrimiento de alguna *regla de asociación* difícil de encontrar usando métodos puramente estadísticos.
- En *Minería de Uso Web* se pueden convertir las *sesiones de navegación* en *transacciones* y estudiar qué páginas relacionan entre sí los usuarios. Si se pueden obtener las «mejores» *reglas de asociación* en tiempo real se podrá usar esta información para mejorar un *Sistema de Recomendación Web*.

Para medir la calidad de las *Reglas de Asociación* se utilizan dos valores, el *soporte* y la *confianza* de la regla. Antes de definirlos hemos de exponer el significado de *soporte* de un *itemset*, que siempre tendrá relación con el tamaño del almacén de *transacciones*, $|\mathcal{D}|$.

Definición 16 (*Soporte de un itemset*). *El soporte de un itemset es su frecuencia en \mathcal{D} , el número de veces que aparece el itemset en el almacén de transacciones.*

$$\text{soporte}(X) = |\mathcal{T}_X| = n_X, \text{ siendo } \mathcal{T}_X = \{\mathcal{T} \in \mathcal{D} / X \in \mathcal{T}\} \quad (1.16)$$

Se utilizan las dos versiones de la frecuencia para hablar del soporte. En la anterior ecuación aparece el soporte absoluto y a continuación se muestra el soporte relativo, puesto en relación al número de transacciones con que estamos trabajando.

$$\text{soporte}_{\text{relativo}}(X) = \frac{n_X}{|\mathcal{D}|} \quad (1.17)$$

En muchos estudios se supone que el *soporte* de un *itemset* es representativo de la frecuencia de ese *itemset* en la población de la que procede la muestra analizada. Si esta suposición fuera correcta sería más probable encontrar este *itemset* en esta población que cualquier otro con menor *soporte* por lo que se utiliza para estimar probabilidades sobre la población de origen de la muestra \mathcal{D} .

La limitación de recursos en los dispositivos en que se ejecutan los algoritmos de *Minería de Reglas de Asociación* obliga a hacer alguna definición más. El principal problema es el uso de memoria RAM cuando tenemos muchos ítems sobre los que guardar información. De \mathcal{I} se pueden obtener $2^M - M - 1$ *itemsets* distintos con más de un ítem, y de cada k -*itemset* se pueden obtener hasta $2^k - 2$ *reglas de asociación* por lo que si M es grande estamos hablando de registrar millones de datos y frecuencias, tantos que pueden llegar a desbordar la capacidad de almacenamiento del dispositivo en que se ejecute el algoritmo de *ARM*.

La primera solución propuesta consiste en ignorar los ítems que tengan menor *soporte* para aprovechar la memoria sólo para guardar los ítems más frecuentes, lo que se supone que aportará información para un conjunto más amplio de individuos de la población. Es una suposición que no debe satisfacer las inquietudes del analista ya que cuando trabajamos con colecciones muy grandes de *transacciones* puede darse el caso que decidamos ignorar los ítems cuyo *soporte* sea inferior al 0.5 %, lo que muchos investigadores califican como «poco representativo» de la población en estudio. Si ese aparentemente pequeño 0.5 % supone

que ignoremos la información que proporciona un ítem que aparece en miles de transacciones no deberíamos dar por terminado el estudio.

Definición 17 (*Soporte mínimo*). *El soporte mínimo, minSup , es un valor fijado antes de llevar a cabo la búsqueda de itemsets frecuentes. Si el almacén \mathcal{D} es muy grande y el equipo en que se ejecute el algoritmo de Minería de Reglas de Asociación tiene recursos insuficientes para el análisis completo de \mathcal{D} se debe fijar un soporte mínimo para que no se produzcan desbordamientos de memoria RAM, lo que se logrará ignorando los itemsets cuyo soporte sea inferior a minSup .*

Definición 18 (*Itemset frecuente*). *Un itemset frecuente es un itemset con soporte en \mathcal{D} igual o superior a minSup .*

Definición 19 (*Conjunto de itemsets frecuentes, \mathcal{L}*). *Sea \mathcal{L} es el conjunto de todos los itemsets frecuentes de \mathcal{D} . Sean \mathcal{L}_k los conjuntos de k -itemsets frecuentes de \mathcal{D} .*

$$\mathcal{L} = \cup_k \mathcal{L}_k \quad (1.18)$$

Técnicamente el valor más grande que puede tomar k coincide con la longitud de la *transacción* más larga de \mathcal{D} . Sin embargo al trabajar con *soporte* mínimo es muy probable que k no alcance dicho valor.

Definición 20 (*Soporte de una Regla de Asociación*). *El soporte de la Regla de Asociación $X \rightarrow Y$ es la frecuencia del itemset que la forma, $X \cup Y$, en \mathcal{D} .*

$$\text{soporte}(X \rightarrow Y) = \text{soporte}(X \cup Y) = n_{XY} \quad (1.19)$$

$$\text{soporte}_{\text{relativo}}(X \rightarrow Y) = \frac{\text{soporte}(X \cup Y)}{|\mathcal{D}|} = \frac{n_{XY}}{|\mathcal{D}|} \quad (1.20)$$

El *soporte* de una *regla de asociación* nos indica con qué frecuencia se encuentra esta regla entre las transacciones de la muestra \mathcal{D} .

Definición 21 (*Confianza de una Regla de Asociación*). *La confianza de una regla es la frecuencia con que aparece el consecuente, Y , en las transacciones en que está el antecedente, X .*

$$\text{confianza}(X \rightarrow Y) = \frac{\text{soporte}(X \cup Y)}{\text{soporte}(X)} = \frac{n_{XY}}{n_X} \quad (1.21)$$

Si la *regla de asociación* $X \rightarrow Y$ tiene *soporte* s y *confianza* c su lectura es sencilla de comprender:

«En el $s\%$ de las *transacciones* de \mathcal{D} está el *itemset* $(X \cup Y)$. En el $c\%$ de las *transacciones* en que está presente el *itemset* X también está presente el *itemset* Y »

Se utilizan mucho en tareas de *predicción* interpretando la *confianza* como una probabilidad aunque para hacerlo correctamente debería estudiarse más a fondo la población de origen de los datos.

Definición 22 (*Confianza mínima*). La *confianza mínima*, minConf , es un valor fijado antes de llevar a cabo la generación de reglas de asociación para evitar problemas de desbordamiento de memoria RAM, lo que se logrará ignorando las reglas de asociación cuya confianza sea inferior a minConf .

En todo proceso de *Minería de Reglas de Asociación* hay dos fases bien diferenciadas, la *Minería de Itemsets Frecuentes (FIM)* y la obtención de las *reglas de asociación* a partir de los *itemsets* frecuentes encontrados. La parte más costosa del proceso es la primera pues se ha de recorrer por completo el almacén \mathcal{D} para comprobar qué ítems se relacionan entre sí y cuáles lo hacen de manera frecuente (superando el *soporte* mínimo fijado en el estudio). En esta fase se guarda la información obtenida conforme se va leyendo \mathcal{D} , con el riesgo de tener desbordamiento de memoria RAM si no se hacen costosas operaciones de comprobación cada vez que se va a necesitar algo más de RAM. En la segunda fase se utiliza el conjunto de *itemsets* frecuentes encontrado en la primera y se comprueban todas las reglas que se pueden derivar de ellos, ofreciendo como resultado las reglas que superen la *confianza* mínima fijada para el estudio.

Existen muchos algoritmos de *Minería de Reglas de Asociación* entre los que destacan *AIS* (Agrawal, Imielinski y A. Swami, 1993b), *SETM* (Houtsma y A. N. Swami, 1993), *Apriori*, *Apriori-TID* y *AprioriHybrid* (Agrawal y Srikant, 1994a, 1994b), *Partition* (Savasere, Omiecinski y Navathe, 1995), *DHP* (Park, M. Chen y P. Yu, 1995, 1997), *Count Distribution*, *Data Distribution* y *Candidate Distribution* (Agrawal y Shafer, 1996), *Eclat*, *MaxEclat*, *Clique* y *MaxClique* (Zaki, Parthasarathy y col., 1997), *DIC* (Brin y col., 1997), *Basic*, *Cumulate* y *EstMerge* (Srikant y Agrawal, 1997), *MaxMiner* (Bayardo, 1998), *RangeApriori* (Groth y Robertson, 2001), *PincerSearch* (D.-I. Lin y Kedem, 2002) o *FP-Growth* (J. Han, Pei y Yin, 2000; J. Han, Pei, Yin y Mao, 2004).

Entre todos ellos hemos seleccionado *Apriori* por tratarse de un algoritmo sencillo, fácil de implementar y de modificar o ajustar a nuevas aportaciones a la *Minería de Reglas de Asociación*. De hecho este algoritmo es la base de muchos de los algoritmos propuestos hasta la fecha. Como todo algoritmo de *ARM*, *Apriori* consta de dos fases:

1. *Minería de Itemsets Frecuentes*. Se fija $minSup$, el *soporte* mínimo, y se recorre \mathcal{D} las veces que sea necesario hasta obtener todos los *itemsets* frecuentes que contiene, aquellos cuyo *soporte* sea igual o superior a $minSup$. Los listados 1.4, 1.5 y 1.6 muestran esta fase en el algoritmo *Apriori*.
2. *Obtención de Reglas de Asociación*. Se fija $minConf$, la *confianza* mínima, y se recorre el conjunto de *itemsets* frecuentes \mathcal{L} obteniendo las reglas del tipo $itemset_1 \Rightarrow itemset_2$ con *confianza* igual o superior a $minConf$. El listado 1.7 muestran esta fase en el algoritmo *Apriori*.

Definición 23 (Conjunto de candidatos a k -*itemsets* frecuentes, \mathcal{C}_k). \mathcal{C}_k es el conjunto de todos los k -*itemsets* que podrían ser frecuentes en \mathcal{D} . Se genera a partir de \mathcal{L}_{k-1} .

Listado 1.4: Algoritmo *Apriori*

```

 $\mathcal{L}_1 = \{large\ 1 - itemsets\};$ 
for ( $k = 2; \mathcal{L}_{k-1} \neq \emptyset; k++$ ) do begin
   $\mathcal{C}_k = \text{apriori-gen}(\mathcal{L}_{k-1});$ 
  forall transactions  $t \in \mathcal{D}$  do begin
     $\mathcal{C}_t = \text{subset}(\mathcal{C}_k, t);$ 
    forall candidates  $c \in \mathcal{C}_t$  do
       $c.count++$ ;
  end
   $\mathcal{L}_k = \{c \in \mathcal{C}_k / c.count \geq minSup\};$ 
end

Answer =  $\mathcal{L} = \bigcup_k \mathcal{L}_k$ 

```

La primera fase se inicia fijando el valor del *soporte* mínimo, $minSup$, leyendo \mathcal{D} y anotando la frecuencia de todos los ítems que contiene, obteniendo \mathcal{C}_1 , el conjunto de todos los *candidatos* a 1-*itemset*. A continuación descartamos los ítems de \mathcal{C}_1 cuya frecuencia (*soporte*) no satisfaga el *soporte* mínimo establecido ($soporte < minSup$) y obtenemos los 1-*itemsets* frecuentes, los ítems que

están en \mathcal{D} con *soporte* mínimo, \mathcal{L}_1 . A partir de \mathcal{L}_1 se generan los candidatos a 2-*itemsets* frecuentes, \mathcal{C}_2 , y se comprueba en \mathcal{D} cuáles tienen *soporte* mínimo, obteniendo \mathcal{C}_2 . El proceso se repetirá con \mathcal{C}_k y \mathcal{L}_k mientras se sigan encontrando conjuntos de k -*itemsets* frecuentes en \mathcal{D} . En el listado 1.4 se entiende que ya se ha hecho la primera lectura de \mathcal{D} por lo que conocemos la frecuencia de todos los ítems de \mathcal{D} y hemos obtenido ya \mathcal{L}_1 descartando los ítems poco frecuentes.

La generación de candidatos, \mathcal{C}_k , se realiza mediante la función *a priori – gen*, que recibe como argumento \mathcal{L}_{k-1} , el conjunto de $(k-1)$ -*itemsets* frecuentes. Se realiza en dos fases, la unión y la poda.

Listado 1.5: Función *a priori – gen*: unión

```
insert into  $\mathcal{C}_k$ 
select  $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$ 
from  $\mathcal{L}_{k-1}p, \mathcal{L}_{k-1}q$ 
where  $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2},$ 
       $p.item_{k-1} < q.item_{k-1};$ 
```

En la unión (ver listado 1.5) se obtienen todos los k -*itemsets* fruto de la unión de dos *itemsets* de \mathcal{L}_{k-1} con raíz común (cuyos primeros $k-2$ ítems coinciden). A nivel de implementación hay que situarse en cada hoja de \mathcal{L}_{k-1} y añadirle \mathcal{C}_k , el vector de todos sus «hermanos menores». Es en esta función donde más recursos de memoria RAM son necesarios debido a que cada hoja del árbol \mathcal{L} generará un vector de pre-candidatos (excepto la última hoja de cada rama, por no tener «hermanos menores»). Es por tanto en esta función donde se puede presentar el problema de desbordamiento de memoria cuando se ejecuta el algoritmo con ciertas colecciones de datos.

Listado 1.6: Función *a priori – gen*: poda

```
forall itemsets do
  forall  $(k-1)$ -subsets  $s$  of  $c$  do
    if  $(s \notin \mathcal{L}_{k-1})$  then
      delete  $c$  from  $\mathcal{C}_k$ 
```

En la poda (ver listado 1.6) se eliminan de $\mathcal{L}_{k-1} \cdot \mathcal{C}_k$ los k -*itemsets* que contengan algún $(k-1)$ -*itemsets* no frecuente (no presente en \mathcal{L}_{k-1}) pues, a priori, no será un k -*itemset* frecuente. Esta función es la que da nombre al algoritmo. La poda supone recorrer por completo $\mathcal{L}_{k-1} \cdot \mathcal{C}_k$ y para cada uno de sus elementos realizar la búsqueda de todos sus $(k-1)$ -*itemsets* en \mathcal{L}_{k-1} . Consume mucho tiempo porque las búsquedas en \mathcal{L}_{k-1} son muy laboriosas, hay que buscar el primer elemento en \mathcal{L}_1 , localizar su rama \mathcal{L}_2 y buscar en ella el segundo elemento y

seguir hasta que no se encuentre uno de los elementos o lleguemos al nivel \mathcal{L}_{k-1} y lo encontremos en cuyo caso no habría que hacer nada. Si no se ha encontrado se libera la memoria reservada para el pre-candidato por lo que no puede producir desbordamiento de memoria.

Una vez generados todos los candidatos a k -itemset frecuente se procede a contarlos en \mathcal{D} (listado 1.4). Inicialmente se ha asignado *soporte* nulo a todos los candidatos en \mathcal{C}_k . Se lee cada *transacción* de \mathcal{D} y se busca en $\mathcal{L}_{k-1}-\mathcal{C}_k$ cada k -itemset de la *transacción*, incrementando su *soporte* si se encuentra entre los candidatos. La lectura completa de \mathcal{D} consume mucho tiempo si está guardado en disco duro, sin embargo no siempre es posible guardarlo en memoria RAM para agilizar esta fase del algoritmo. Cuando se termina de leer \mathcal{D} se comprueban qué candidatos tienen *soporte* mínimo y se eliminan los que no lo tienen, liberando memoria en esta fase.

Una vez conocemos todos los *itemsets* con *soporte* mínimo que contiene \mathcal{D} , \mathcal{L} , procederemos a obtener las *Reglas de Asociación* calculando la confianza de todas las reglas que contienen y guardando las que superen la *confianza* mínima, *minConf*.

El listado 1.7 muestra el código que proponen Agrawal y Srikant para generar las *reglas de asociación*. Aunque es un código muy sencillo supone un uso intensivo de procesador y de memoria para guardar los resultados que se van obteniendo. Si no se tienen que utilizar simultáneamente todas las *reglas de asociación* obtenidas es preferible guardarlas en disco conforme se van obteniendo para asegurarnos de no provocar un desbordamiento de memoria. Realmente no es un problema costoso computacionalmente hablando pero una mala implementación de la función o el uso de una estructura inadecuada para almacenar \mathcal{L} podría provocar tiempos de ejecución muy elevados.

La obtención de millones de *Reglas de Asociación* no supone el final del proceso. Hay que extraer conocimiento de ellas y para ello hay que usar técnicas de visualización de grandes colecciones de datos, técnicas de comparación que permitan observar los cambios producidos sobre el conocimiento que ya teníamos. . .

Listado 1.7: Función *genrules*()

```
forall large itemsets  $l_k, k \geq 2$  do
  call genrules( $l_k, l_k$ );

// The genrules generates all valid rules  $\tilde{a} \Rightarrow (l_k - \tilde{a}), \forall \tilde{a} \subset a_m$ 
procedure genrules( $l_k$ : large  $k$ -itemset,
                   $a_m$ : large  $m$ -itemset)
```

```

1)  $A = \{(m-1)\text{-itemsets } a_{m-1} \mid a_{m-1} \subset a_m\};$ 
2) forall  $a_{m-1} \in A$  do begin
3)    $conf = \text{support}(l_k) / \text{support}(a_{m-1});$ 
4)   if ( $conf \geq \text{minconf}$ ) then begin
5)     output the rule  $a_{m-1} \Rightarrow (l_k - a_{m-1})$ ,
        with confidence =  $conf$  and support =  $\text{support}(l_k)$ ;
6)   if ( $m-1 > 1$ ) then
7)     call  $\text{genrules}(l_k, a_{m-1})$ ; // to generate rules
                                     // with subsets of  $a_{m-1}$ 
                                     // as the antecedents
8)   end
9) end

```

Técnicas de ARM aplicadas a un Sistema de Recomendación Web

En la sección 1.2.4 describimos cómo obtener *sesiones de navegación* de *archivos de log*. Si convertimos cada sesión en una *transacción* podemos obtener fácilmente un almacén \mathcal{D} en que cada *transacción* representa un conjunto de páginas visitado por un usuario en un espacio de tiempo determinado. Si las páginas que visitó el usuario tienen relación entre sí es fácil que encontremos otras *transacciones* que contengan subconjuntos de páginas similares a las de esta sesión. Volvemos a la misma idea que en la sección 1.3.1 pero ahora manejaremos menos datos, nos preocuparemos sólo por estudiar a fondo las relaciones entre las páginas representadas por los nodos de los *grafos*.

La *Minería de Reglas de Asociación* tiene un planteamiento muy sencillo que concuerda en cierto modo con nuestras necesidades. Queremos saber cómo se usa un sitio web, tenemos *sesiones de navegación* que podemos observar como un simple *conjunto de páginas visitado por un usuario en una sesión, transacciones*. Si transformamos todas las *sesiones de navegación* en *transacciones* (ver listado 1.8) obtendremos un almacén de *transacciones*, al que llamaremos \mathcal{D} . Los subconjuntos de datos (*itemsets*) más frecuentes de las *transacciones* de \mathcal{D} nos informarán de que «es frecuente encontrar en una misma sesión *este subconjunto de páginas*». Las *reglas de asociación* que obtengamos de \mathcal{D} nos harán deducir que «si un usuario visita *este conjunto de páginas* entonces es probable que también visite *este otro conjunto de páginas*», lo que es un conocimiento muy útil para un *Sistema de Recomendación Web*, sobre todo si podemos obtenerlo de un modo rápido en situaciones en que \mathcal{D} sea muy dinámico.

Si queremos ir más allá podemos pensar que cada vez que un usuario entra en nuestro sitio web está interesado en un objetivo concreto y visita un «conjunto

Listado 1.8: Algoritmo de obtención de \mathcal{D}

```

 $\mathcal{D} = \emptyset$ ;

forall  $sesion_i, i = 1 \dots N$  do
    transaccion =  $\emptyset$ ;
    forall  $p_j \in sesion_i, j = 1 \dots n_i$  do
        if  $p_j \notin transaccion$ 
            insert  $p_j$  into  $transaccion$ ;
        end
    insert  $transaccion$  into  $\mathcal{D}$ 
end

```

concreto de páginas web», P . Con esta hipótesis, si encontramos que muchos usuarios también visitan P en sus *sesiones de navegación* podemos pensar que son páginas que cubren un mismo objetivo. En un *Sistema de Recomendación Web* sería útil esta información cuando un usuario entrara en una de las páginas de P , podríamos recomendarle que visite el resto de páginas de P .

El uso más inmediato de una *regla de asociación* en un *Sistema de Recomendación Web* es la predicción. Cuando un usuario solicita la página P_i , si tenemos información fácil de procesar podremos recomendarle que visite otras páginas, aquellas en que aparezca P_i como *antecedente* y tengan mejor *soporte* o *confianza*. Si aplicamos el algoritmo de *ARM* a todo el almacén \mathcal{D} tardaremos demasiado tiempo en obtener todas las *reglas de asociación* que contiene con P_i como *antecedente* por lo que debemos seleccionar un grupo de transacciones representativo para hacer el proceso de *Minería de Datos* en tiempo real.

Al analizar las sesiones obtenidas en el *fichero de log* de un servidor propio comprobamos que habían usuarios de los que teníamos diversas sesiones por lo que comenzamos a pensar en ese caso, por ir de menor a mayor complejidad. Decidimos probar tiempos de ejecución del algoritmo en el conjunto de transacciones de cada usuario que había realizado al menos una visita al sitio web obteniendo tiempos de ejecución muy por debajo del segundo, lo que permitiría su uso en tiempo real en un *Sistema de Recomendación Web*.

Si un usuario está visitando el grupo de páginas X y disponemos de reglas cuyo *antecedente* sea X podemos utilizar los *consecuentes* de dichas reglas como páginas recomendadas. Si son muchas las reglas cuyo *antecedente* sea X habrá que decidir qué criterio se utiliza para determinar qué *consecuentes* se recomiendan. Todas las reglas se obtienen de las transacciones de \mathcal{D} que contienen a X

por lo que el *soporte* y la *confianza* de la regla nos proporcionan la misma información y podemos utilizar indistintamente cualquiera de las dos medidas para decidir qué reglas se utilizan en el WRS.

Existen otras medidas como el *lift*, que hace intervenir el *soporte* del *consecuente*.

Definición 24 (*lift*). *El lift de una Regla de Asociación compara la frecuencia de la regla con la frecuencia del consecuente. Si llamamos n_{XY} al número de veces que aparece el itemset $(X \cup Y)$ en \mathcal{D} , su lift sería*

$$lift(X \rightarrow Y) = \frac{soporte(X \rightarrow Y)}{n_Y} = \frac{n_{XY}}{n_X \cdot n_Y} \quad (1.22)$$

Su interpretación se hace en términos de probabilidad. Si hubiera independencia estadística entre los sucesos A («la transacción contiene el *itemset* X ») y B («la transacción contiene el *itemset* Y ») podríamos comprobarlo a partir de las probabilidades $P(A)$, $P(B)$ y $P(A \cap B)$ debido a que dos sucesos independientes cumplen que $P(A \cap B) = P(A) \cdot P(B)$. Utilizando las frecuencias observadas en \mathcal{D} podríamos pensar que dos sucesos son independientes si el producto de sus frecuencias coincide con la frecuencia conjunta de ambos. Utilizando n_X , n_Y y n_{XY} para expresar estas frecuencias si detectamos que $n_X \cdot n_Y = n_{XY}$ podemos pensar que la presencia de Y en las *transacciones* de \mathcal{D} es independiente de que también esté presente X . Si ocurriera esto tendríamos que $lift(X \rightarrow Y) = \frac{n_{XY}}{n_X \cdot n_Y} = 1$ por lo que el *lift* debe compararse con la unidad para que nos proporcione alguna información.

- si $lift(X \rightarrow Y) > 1$ se deduce que el conjunto de páginas Y es más popular entre los que han visitado ya X que por sí misma por lo que es más probable que un usuario que ya ha visitado X quiera visitar Y . El hecho de que el usuario haya visitado X «favorece» la probabilidad de que quiera visitar Y en la misma *sesión de navegación*.
- Si $lift(X \rightarrow Y) < 1$, la presencia de X disminuye la probabilidad de que aparezca Y en la misma *sesión de navegación*. Quizá no sea buena idea recomendar Y a un usuario que ya ha visitado X .
- Si $lift(X \rightarrow Y) = 1$, la presencia de X no influye en la probabilidad de que aparezca Y en la misma *sesión de navegación*, son dos sucesos

independientes. Quizá no sea buena idea recomendar Y a un usuario que ya ha visitado X a no ser que no tengamos otra recomendación mejor.

An y Cercone, 2001 proponen el uso de otras medidas de calidad para determinar qué *Reglas de Asociación* se deben descartar. Recordemos que se generan gran cantidad de reglas y no se pueden utilizar todas ellas de un modo sensato, en un *Sistema de Recomendación Web* deberíamos recomendar un conjunto pequeño de páginas, no todas las que pueden tener relación con las visitadas por el usuario. Hay muchos trabajos en esta línea, intentando aportar rigurosidad estadística a cálculos tan simples como el *lift* (no es muy riguroso hacer una estimación de independencia estadística basándonos únicamente en un cociente de frecuencias muestrales).

Sea cual sea el índice que utilicemos para decidir qué *reglas de asociación* se utilizarán en la recomendación primero hemos de obtenerlas. Ya sabemos todo lo que necesitamos para ello, y hemos decidido qué algoritmo utilizaremos por lo que hay que decidir qué lenguaje de programación usaremos y qué estructuras se adaptarán mejor al tipo de datos que queremos almacenar y a su eficiencia en las labores de búsqueda, inserción y eliminación de elementos. La lectura de los trabajos de investigación sobre las dificultades de la implementación de algoritmos de *Minería de Reglas de Asociación* (Goethals, 2003; Borgelt, 2004; Bodon, 2004, 2005) nos hizo pensar en detalles muy interesantes sobre la implementación de *Apriori* y nos hizo tener una buena base antes de afrontar nuestra propia implementación del algoritmo. Al probar nuestra primera versión de *Apriori* nos tropezamos pronto con el *dilema del ítem raro*, obteniendo continuos desbordamientos de memoria RAM. Se resolvía puntualmente incrementando el *soporte* mínimo pero nos parecía que tenía que haber mejores soluciones.

En la fase de *Minería de Itemsets Frecuentes* se hace la mayor reserva de memoria RAM, concretamente en la fase de unión de la función *apriori-gen()*. La mayoría de memoria reservada se libera tras la fase de poda de esta función, pero si ya hay desbordamiento no se llegará a producir la poda. Decidimos unir ambas fases en una sola, como muestra el listado 1.9, para no reservar más memoria RAM de la que necesitamos para guardar C_k y ahorrar tiempo de computación pues las operaciones de reserva y liberación dinámica de memoria pueden ser muy costosas (sobre todo cuando se utiliza mucha memoria RAM y se ha de recurrir a la memoria virtual gestionada en el disco duro).

En la implementación de \mathcal{L} se ha de utilizar una estructura que utilice pocos recursos de memoria para guardar sus datos y en la que sea fácil realizar búsquedas.

Listado 1.9: Función *apriori – gen*: unión y poda

```

insert into  $C_k$ 
select  $c = \{p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}\}$ 
from  $L_{k-1}p, L_{k-1}q$ 
where ( $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2},$ 
       $p.item_{k-1} < q.item_{k-1}$ )
and
  forall ( $k-1$ )-subsets  $s$  of  $c$ 
     $s \in \mathcal{L}_{k-1}$ ;

```

das, recordemos que se trabajará con muchos datos dentro de esta estructura y cualquier proceso de búsqueda, inserción o eliminación puede ser realmente muy costoso en tiempo si no acertamos con la estructura utilizada. Al ir adaptando el algoritmo a nuestras necesidades y trabajar con conjuntos no excesivamente grandes de datos en esta fase de la investigación pudimos proponer una poda relajada que reducía el número de accesos a \mathcal{L} y el número de elementos a eliminar de C_k en esta fase, lo que supondrá un aumento de rendimiento a cambio de no liberar temporalmente algo de memoria RAM.

Listado 1.10: Función *apriori – gen*: poda relajada

```

forall itemsets do
  forall 2-subsets  $s$  of  $c$  do
    if ( $s \notin \mathcal{L}_2$ ) then
      delete  $c$  from  $C_k$ 

```

Al aplicar la poda relajada la función con la unión y la poda queda como muestra el listado 1.11. Por un lado no esperamos desbordamiento de memoria porque no se reservará espacio para todos los posibles candidatos ya que muchos de ellos serán descartados antes de realizar la reserva, por otro lado se ahorran miles de búsquedas en \mathcal{L}_{k-1} por lo que se gana en eficiencia al realizar esta modificación al algoritmo original.

En un primer estudio teórico se previó que el número de candidatos obtenidos con el algoritmo modificado sería muy superior al obtenido utilizando el algoritmo original. Sin embargo, al experimentar con datos de *ficheros de log* reales se observó que las sesiones producidas por un mismo usuario de nuestro sitio web presentaban *patrones* muy marcados y eran muy homogéneas en cuanto a

Listado 1.11: Función *apriori* – *gen*: unión y poda relajada

```

insert into  $C_k$ 
select  $c = \{p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}\}$ 
from  $L_{k-1}p, L_{k-1}q$ 
where ( $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2},$ 
       $p.item_{k-1} < q.item_{k-1}$ )
and
      ( $p.item_{k-1}, q.item_{k-1}$ )  $\in L_2$ ;

```

las páginas que contenían. Esto permitió realizar un estudio completo de todas las páginas, es decir, sin tener que considerar *soporte* mínimo, y permitió generar predicciones para cualquier página visitada anteriormente por el usuario. La *confianza* o el *lift* servirían para dar prioridad a las sugerencias.

Para probar la eficiencia del algoritmo modificado se preprocesaron las solicitudes hechas a un servidor web durante los 22 últimos días del mes de julio del 2004. En la lectura del *fichero de log* se observaron 2 977 380 solicitudes a recursos, 94 442 de las cuales no pudieron ser servidas (errores en la petición y/o peticiones a páginas no existentes), 2 437 378 peticiones fueron realizadas a recursos auxiliares a las páginas solicitadas por los usuarios (imágenes, ficheros de estilo, archivos multimedia, etc.) En el análisis se emplearon 445 560 solicitudes, conteniendo 2 158 recursos diferentes.

Para este estudio se definió una *sesión de navegación* como la *secuencia* de solicitudes realizada desde la misma dirección IP, de modo que entre una petición y la siguiente no transcurrieran más de 15 minutos, y entre la primera petición y la última de una sesión no transcurrieran más de 8 horas. Así se obtuvieron un total de 30 559 sesiones realizadas desde 11 391 direcciones IP, obteniendo así información sobre los accesos de 11 391 usuarios distintos.

En la tabla 1.1 se muestran los tiempos de ejecución del algoritmo modificado sobre las 30 559 *transacciones* producidas desde las 11 391 direcciones IP, identificando cada dirección IP con un usuario distinto y haciendo un análisis exclusivo para las *transacciones* de cada usuario.

Se obtuvieron tiempos inferiores a los 30 milisegundos en el 99.9 % de las ejecuciones del algoritmo modificado (menos de 15 milisegundos en el 79.7 % de los casos analizados). El análisis de las sesiones realizadas por un robot que realizó una visita al sitio web duró algo más de 1.5 segundos, en este caso aplicamos un *soporte* mínimo (absoluto) de 5 y 10 (sólo consideraríamos los *itemsets*

Cuadro 1.1: Resumen de tiempos de ejecución del algoritmo modificado

Tiempo (milisegundos)	Frecuencia	(%)
$t < 15$	9 079	79.7
$15 \leq t < 30$	2 305	20.2
$30 \leq t < 45$	4	0.0
$145 \leq t < 60$	2	0.0
$60 \leq t$	1	0.0

Cuadro 1.2: Resumen de tiempos de ejecución del algoritmo modificado

$ D $	$ I $	Transacción más larga	Soporte mínimo (absoluto)	Tiempo (milisegundos)
3	11	11	1	1
11	13	10	1	1
20	51	26	1	1
30	139	57	1	49
51	127	21	1	16
86	602	222	1	1 540
			5	467
			5	187
275	73	52	1	30
364	114	22	1	16

que estuvieran al menos en 5 o 10 transacciones) para comprobar la drástica reducción de tiempo empleado a cambio de no obtener toda la información de D (como muestra la tabla 1.2). Por último hubieron seis casos en que las direcciones IP se correspondían con ordenadores públicos por lo que las *sesiones de navegación* que proporcionaron seguramente fueron realizadas por diferentes usuarios, mostrando comportamientos muy heterogéneas y cuyos análisis se realizaron en tiempos de entre 30 y 60 milisegundos.

Los resultados obtenidos nos dan pié a pensar que este tipo de análisis es válido para determinar qué sugerencias se pueden hacer a cualquier usuario que ya haya visitado nuestro sitio web ya que se están utilizando los datos generados por

el propio usuario en anteriores visitas al sitio y se obtienen las recomendaciones mientras se están enviando los recursos solicitados por la actual visita con lo que se podrán enviar también las sugerencias que determine nuestro algoritmo como mejores para el usuario actual en la visita actual.

1.4. Publicaciones

La investigación científica ha de divulgarse, es el mejor modo que tenemos para recibir retroalimentación de la comunidad científica. En este primer periodo nuestra investigación se difundió a través de tres trabajos, aceptados para su exposición en dos Congresos Internacionales y un Simposio. Con el primero de ellos presentamos en Las Vegas (HCII'05¹) el trabajo expuesto en la sección 1.3.1. La investigación mostrada en la sección 1.3.2 dio lugar a dos ponencias en Granada (Interacción'05² y SICO'05³).

Personalization through Inferring User Navigation Maps from Web Log Files, 2005

Botella Beviá, F., Lazcorreta Puigmartí, E., Fernández-Caballero, A. y González López, P. Personalization through Inferring User Navigation Maps from Web Log Files. *Proceedings of the 11th International Conference on Human-Computer Interaction*, Las Vegas, 2005.



Resumen

One of the most challenging areas in human computer interaction is personalization. Nonetheless, usable and well design sites can not be replaced by personalization, which must be used in just measure. In order to avoid dealing with personal data from users and thus not to worry about privacy, we decide to work with anonymous data. Moreover, these data can be found in all web server logs. In this paper, we propose a methodology for simple personalization. Amazon.com does with book recommendations based only

¹http://www.hci.international/index.php?module=conference&CF_op=view&CF_id=4



²<http://aipo.es/congresos/interaccion/interacción-2005>

³http://cedi2005.ugr.es/2005/simposio_s16_sico.shtml

⁴http://cio.umh.es/files/2011/12/CIO_2005_03.pdf

on anonymous data about user's navigation in the website. Our target is to facilitate the navigation of users by means of suggested links towards the next page that users likely want to go. And this can be accomplished in real time using an intelligent agent system or at user requirements design by running a specific process in the system that offers instantaneously the suggested links.

Mejora de la usabilidad y la adaptabilidad mediante técnicas de Minería de Uso Web, 2005

Botella Beviá, F., Lazcorreta Puigmartí, E., Fernández-Caballero, A. y González López, P. Mejora de la usabilidad y la adaptabilidad mediante técnicas de Minería de Uso Web. *Actas del VI Congreso Interacción Persona-Ordenador*, 299–306, Granada, 2005.  ⁵




Resumen

Una de las áreas más activas en la Minería de Uso Web es la *predicción*. La personalización de un sitio web es uno de los objetivos fundamentales de la *predicción*. Si se estudia el comportamiento de los usuarios de un sitio web, es posible conocer qué páginas suelen visitar y cuáles son las siguientes páginas que visitan. Sería de gran ayuda para los usuarios si pudieran tener los dos o tres enlaces a las páginas que más suelen visitar en un sitio web. En este trabajo proponemos un método, basado en el algoritmo *Apriori*, que permite inferir las páginas web que un usuario visita a partir de los datos registrados en los *ficheros de log* del servidor web de visitas previas. Estos enlaces serán sugeridos al usuario en un área concreta de la página web lo que permitirá mejorar la usabilidad y adaptabilidad del sitio web.

Técnica de minería de datos para la adaptación de sitios Web, 2005

Botella Beviá, F., Lazcorreta Puigmartí, E., Fernández-Caballero, A. y González López, P. Técnica de minería de datos para la adaptación

⁵http://www.researchgate.net/profile/Antonio_Fernandez-Caballero/publication/228963180_Mejora_de_la_usabilidad_y_la_adaptabilidad_mediante_tcnicas_de_minera_de_uso_Web/links/0912f509c0f3f3f63d000000?origin=publication_detail

de sitios Web. *Actas del Simposio de Inteligencia Computacional (SICO)*, 455–462, Granada, 2005. ISBN 84-9732-444-7.   ⁶

Resumen

En este trabajo presentamos el uso de técnicas de *Minería de Datos* y la búsqueda de técnicas de *predicción* sobre los *ficheros de log* de un servidor web para mejorar la facilidad de uso del sitio. Se intenta adaptar el sitio web a las necesidades reales de los usuarios, de modo que el usuario reciba información detallada que le permitirá «recordar» qué páginas visitó en anteriores ocasiones donde también solicitó la página actual, así como información sobre el uso general del sitio web.

⁶http://www.dsi.uclm.es/personal/AntonioFdez/nais/nais/investigacion/publicaciones/congresos_2005/SICO2005.pdf

Minería de Reglas de Asociación

La línea de investigación se va definiendo poco a poco. En sus inicios teníamos la intención de crear un *Sistema de Recomendación Web (WRS)* capaz de adaptarse al usuario en sitios web con gran cantidad de páginas y de usuarios generando gran cantidad de datos sobre su uso. Lo planteamos como un proceso de *Minería de Uso Web (WUM)*, un proceso específico de *Knowledge Discovery in Databases (KDD)* en el que los datos y el conocimiento final giran en torno al uso de la *World Wide Web (WWW)*. Como en todo proceso de *KDD*, la *WUM* se divide en cinco tareas secuenciales y recurrentes: *Selección, Preproceso, Transformación, Minería de Datos* e «*Integración y Evaluación*». Las tres primeras tareas se han expuesto a fondo en la sección 1.2, la *Minería de Datos* ha ocupado un papel especial en la sección 1.3 pero no se ha dejado resuelta y no ha sido posible llevar a cabo la parte de integración en un sitio web real que genere gran cantidad de datos de uso, por lo que no ha sido posible evaluar el conocimiento adquirido.

La *Minería de Datos* comienza a tomar protagonismo en nuestra investigación. Las *Reglas de Asociación* son relaciones muy interesantes y los algoritmos que las obtienen están recibiendo cada vez más atención de la comunidad científica, como muestran las revisiones y comparaciones elaboradas por Hipp, Güntzer y Nakhaeizadeh (2000), Zhao y Bhowmick (2003) y Goethals (2003). Aparecen decenas de algoritmos y estructuras de datos para gestionar los problemas que van surgiendo conforme se va avanzando en esta nueva disciplina donde la mayor dificultad está en administrar correctamente los recursos disponibles para

estudiar las cada vez más grandes cantidades de datos a analizar. Nosotros mismos proponemos en la sección 1.3.2 algunos cambios sobre uno de los algoritmos más destacados en la *Minería de Reglas de Asociación, Apriori*. Decidimos profundizar en este elegante algoritmo, complicarlo lo justo para que se pueda adaptar a nuestras necesidades y que pueda incorporar nuestras aportaciones. Decidimos poner a prueba nuestras propuestas para lo que había que desarrollar el código y comenzamos a observar más a fondo el código que otros investigadores habían puesto a nuestra disposición a través de su publicación en las actas de FIMI'03 (Zaki y Goethals, 2003).

Al analizar a fondo el algoritmo *Apriori* descubrimos que la generación de *reglas de asociación* se plantea de un modo en que se repiten gran cantidad de cálculos. Propusimos un algoritmo alternativo para el sugerido originalmente por Agrawal y Srikant (1994b) en el que se ahorran muchos costosos procesos de búsqueda lo que redundaba en ganancia de tiempo sin pérdida de ningún tipo de información.

Conforme más profundizamos en el estudio de *Reglas de Asociación* más vemos la necesidad de reducir las dimensiones de los datos a tratar. Podemos abordar el estudio completo de «pequeñas colecciones» de datos utilizando *Apriori* para obtener resultados en tiempo real. Pero cuando nos enfrentamos a grandes colecciones de datos siempre hemos de renunciar al análisis de muchos de ellos. En este capítulo intentaremos utilizar *Apriori* para dividir los datos iniciales en grupos más homogéneos en función de las reglas que cumpla cada registro. Si el *clustering* obtenido es correcto se reducirán enormemente las necesidades de recursos pues en cada grupo habrá un número menor de ítems a procesar, podríamos obtener más y mejor información de cada uno de los grupos a partir de los mismos datos.

Otro de los puntos de interés de esta investigación está motivado por la existencia del *dilema del ítem raro*. En el capítulo anterior lo resolvimos parcialmente estudiando únicamente las transacciones (sesiones) que procedían de un mismo usuario con lo que se reducía notablemente el número de datos a procesar y se podía llevar a cabo un análisis individual en tiempo real para alimentar un *WRS*. Sin embargo este planteamiento no puede aplicarse a grandes colecciones de datos por los problemas de desbordamiento de memoria ya citados. Este capítulo finaliza con una aportación con la que pretendemos aliviar este problema utilizando toda la información disponible en \mathcal{D} , posibilitando el análisis de los *ítems raros* mejor relacionados con los ítems frecuentes y su uso en un *RS* mediante la definición de *Reglas de Oportunidad*.

2.1. Conceptos básicos

Agrawal, Imielinski y A. Swami (1993b) son pioneros en la *Minería de Reglas de Asociación*. La definición formal que dan a esta disciplina es muy elemental¹:

Let $\mathcal{I} = \{I_1, I_2, \dots, I_N\}$ be a set of binary attributes, called items. Let \mathcal{D} be a database of transactions. Each transaction t is represented as a binary vector, with $t[k] = 1$ if t bought the item I_k , and $t[k] = 0$ otherwise. There is one tuple in the database for each transaction. Let X be a set of some items in \mathcal{I} . We say that a transaction t satisfies X if for all items I_k in X , $t[k] = 1$.

By an association rule, we mean an implication of the form

$$X \rightarrow I_j$$

where X is a set of some items in \mathcal{I} , and I_j is a single item in \mathcal{I} that is not present in X . The rule $X \rightarrow I_j$ is satisfied in the set of transactions \mathcal{D} with the confidence factor $0 \leq c \leq 1$ iff at least $c\%$ of transactions in \mathcal{D} that satisfy X also satisfy I_j .

Given the set of transactions \mathcal{D} , we are interested in generating all rules that satisfy certain additional constraints of two different forms: Syntactic Constrains and Support Constrains.

Definen \mathcal{I} como un conjunto de N atributos binarios y \mathcal{D} como una *Base de Datos de transacciones*, vectores binarios de N elementos cuyo valor k -ésimo indica si el ítem I_k está o no presente en la *transacción*. El *consecuente* de una *Regla de Asociación* es un ítem de \mathcal{I} no presente en el *antecedente* de la regla. Las investigaciones posteriores sobre este tema amplían esta definición para estudiar reglas en que el *consecuente* sea un *k-itemset*, como se indica en la definición 14. Por último definen la *Minería de Reglas de Asociación* como la disciplina que busca en un gran almacén \mathcal{D} las *Reglas de Asociación* que cumplen ciertas restricciones, bien de tipo sintáctico (como la presencia de cierto ítem en el antecedente o el consecuente) o de *soporte* (obteniendo únicamente las reglas más frecuentes).

¹He adaptado la nomenclatura a la utilizada en este informe.

Las restricciones no son imposiciones si no necesidades. Si \mathcal{I} está formado por N ítems, para tratar una transacción tendremos que utilizar N valores (ceros o unos), y se podrán obtener hasta 2^N *itemsets* distintos², pudiendo actuar como *antecedente* de una *regla de asociación* $2^N - 2$ al descontar \emptyset e \mathcal{I} . Considerando que cada 1-*itemset* puede ser el *antecedente* de hasta $2^{N-1} - 1$ *reglas de asociación* distintas, cada 2-*itemset* puede ser el *antecedente* de $2^{N-2} - 1$ reglas distintas... es fácil descubrir que el número de *Reglas de Asociación* que se podría obtener es enorme: $\sum_{i=1}^{N-1} \binom{N}{i} (2^{N-i} - 1)$. Como la mayoría de datos se ha de manipular en memoria RAM para poder acceder a ellos rápidamente, sólo si trabajamos con pocos ítems y pocas *transacciones* podremos evitar las restricciones.

2.1.1. Tipo de Datos

Al definir las transacciones como vectores binarios se puede interpretar que es mejor guardar los datos a nivel de bit ya que sólo son necesarios dos valores para representar cada dato, lo que convertiría a \mathcal{D} en una matriz de ceros y unos cuyas $|\mathcal{D}|$ filas representan cada una de las *transacciones* y sus N columnas cada uno de los ítems de \mathcal{I} . Es fácil pensar en trabajar con matrices de estas características para aprovechar la potencia de los ordenadores al trabajar a nivel de bit, sin embargo no se plantea nadie este formato hasta el trabajo de Dong y M. Han (2007), que proponen comprimir \mathcal{D} en una BitTable y usar el algoritmo *BitTableFI*. Indican que esta estructura posibilita una rápida generación de candidatos y de su recuento por utilizar funciones de unión e intersección de bits, obteniendo mejor rendimiento que los algoritmos con que se compara. En la sección 4 volveremos a este planteamiento como propuesta de trabajo futuro.

Trabajar con uniones e intersecciones de bits puede ser muy eficiente para un procesador si se consigue programar de forma óptima, sin embargo al desarrollar un programa con un lenguaje de alto nivel no es fácil trabajar a nivel de bit (en C++ se guarda un valor bool usando un Byte, ocho bits) por lo que la mayoría de implementaciones hechas sobre algoritmos de *Minería de Reglas de Asociación* no utilizarán esta definición de \mathcal{I} . Generalmente se representa el ítem I_k utilizando el número k con lo que las *transacciones* no se guardan como N -tuplas de ceros y unos si no como conjuntos de números enteros. El formato más utilizado

²Si $N = 100$ tendremos $2^{100} \approx 1,3 \cdot 10^{30}$ posibles *itemsets* en las *transacciones* si no contamos repeticiones, lo que da una idea del número de relaciones que se pueden encontrar entre los *itemsets* de \mathcal{D} cuando éste es muy grande y el número de ítems distintos también lo es.

para guardar \mathcal{D} de este modo es escribiendo en un fichero de texto plano cada *transacción* en una línea del fichero, lo que se conoce como *representación horizontal de \mathcal{D}* . Algunos algoritmos están especialmente diseñados para aprovechar la *representación vertical de \mathcal{D}* , en que en cada línea se escribe el par TID *item* de modo que cada *transacción* ocupe tantas líneas consecutivas como ítems contenga. A lo largo de este informe se tratará \mathcal{D} como un conjunto de números enteros y no una matriz de bits.

2.1.2. Primeros Algoritmos

En este primer artículo proponen el algoritmo *AIS* (ver algoritmo 2.1) que se basa en múltiples lecturas de la *Base de Datos* de transacciones \mathcal{D} . Comienzan con un conjunto vacío de *itemsets* frecuentes, \mathcal{L} , y un conjunto de *itemsets* al que llaman frontera, \mathcal{F} , y que contiene inicialmente el conjunto vacío. En cada lectura de \mathcal{D} , $k = 1 \dots$, se realizan los siguientes pasos:

1. Se vacía el conjunto de candidatos, \mathcal{C}_k .
2. Se busca cada $k - 1$ -*itemset* de la frontera en cada *transacción*.
3. Si se encuentra se extiende con todos los ítems de la *transacción* y se añade la extensión al conjunto \mathcal{C}_{k+1} , creándolo con *soporte* 1 si no existe o incrementando su *soporte* si ya existe.
4. Al terminar la k -ésima lectura de \mathcal{D} se vacía la frontera, se guardan en \mathcal{L}_k los candidatos que tienen *soporte* mínimo y se añaden a la frontera los que se estima que se podrán extender en la siguiente iteración.
5. Si la frontera no está vacía se vuelve al paso 1.

Empiezan a encontrarse los problemas de recursos intrínsecos a esta disciplina, llegando a proponer un algoritmo que guarda en disco la información que está en memoria y no es necesaria. Este algoritmo se llamará cada vez que se necesite memoria para almacenar nuevos datos, lo que restará algo de eficiencia al proceso completo.

Proponen también una serie de heurísticas para estimar el *soporte* de un k -*itemset* antes de generar su candidato, de modo que si se estima que tendrá un *soporte* bajo no se genere el candidato y el sistema no caiga por falta de memoria para almacenar los candidatos. La generación de candidatos se hace bajo la

suposición de independencia de los ítems presentes en cada transacción. Si los *itemsets* X e Y son disjuntos y tienen *soporte* x e y respectivamente, estiman que el *itemset* $X \cup Y$ tendrá *soporte* $z = x \cdot y$, por lo que esperan que será frecuente en \mathcal{D} si $z \geq \text{minSup}$.

Listado 2.1: Algoritmo AIS, 1993

```

procedure LargeItemsets
begin
  let Large set  $\mathcal{L} = \emptyset$ ;
  let Frontier set  $\mathcal{F} = \{\emptyset\}$ ;

  while  $\mathcal{F} \neq \emptyset$  do begin
    -- make a pass over the database
    let Candidate set  $\mathcal{C} = \emptyset$ ;
    forall database tuples  $t$  do
      forall itemsets  $f$  in  $\mathcal{F}$  do
        if  $t$  contains  $f$  then begin
          let  $\mathcal{C}_f$  = candidate itemsets that are extensions of  $f$ 
            and contained in  $t$ ;
          forall itemsets  $c_f$  in  $\mathcal{C}_f$  do
            if  $c_f \in \mathcal{C}$  then
               $c_f.\text{count} = c_f.\text{count} + 1$ ;
            else begin
               $c_f.\text{count} = 0$ ;
               $\mathcal{C} = \mathcal{C} + c_f$ ;
            end
          end
        end
      end
    end

    -- consolidate
    let  $\mathcal{F} = \emptyset$ ;
    forall itemsets  $c$  in  $\mathcal{C}$  do begin
      if  $\text{count}(c)/\text{dbsize} > \text{minsupport}$  then
         $\mathcal{L} = \mathcal{L} + c$ ;
        if  $c$  should be used as a frontier in the next pass then
           $\mathcal{F} = \mathcal{F} + c$ ;
        end
      end
    end
  end
end
end

```

Houtsma y A. N. Swami (1993) sugieren trabajar directamente sobre las *Bases de Datos de transacciones* aprovechando las funciones propias de los *Gestor de Bases de Datos* en lugar de desarrollar aplicaciones específicas para trabajar con almacenes \mathcal{D} guardados como texto plano. Proponen el algoritmo *SETM* (ver

algoritmo 2.2), que genera los candidatos al leerlos en las transacciones, igual que *AIS*, y guarda una copia de cada *itemset* junto al TID de la transacción que lo contiene. Al trabajar con *DB* en disco no necesita muchos recursos de memoria para guardar tanta información pero no puede competir en eficiencia con programas preparados específicamente para extraer la misma información de un fichero de texto plano. Sin embargo la asociación de cada *itemset* con el TID de la transacción en que está contenido puede agilizar un estudio más profundo de ciertos *itemsets* por lo que no descartamos el uso de esta idea cuando podamos realizar en paralelo estas sentencias que no benefician la eficiencia del propósito actual, encontrar los *itemsets* frecuentes presentes en \mathcal{D} , pero cuyos resultados pueden ser guardados en ficheros cuyo análisis posterior puede proporcionar gran cantidad de conocimiento sobre la población en estudio.

Listado 2.2: Algoritmo SETM, 1993

```

 $k := 1$ ;
sort  $R_1$  on item;
 $C_1 :=$  generate counts on  $R_1$ ;
repeat
   $k := k + 1$ ;
  sort  $R_{k-1}$  on  $trans\_id, item_1, \dots, item_{k-1}$ ;
   $R'_k :=$  merge-scan  $R_{k-1}, R_1$ ;
  sort  $R'_k$  on  $item_1, \dots, item_{k-1}$ ;
   $C_k :=$  generate counts on  $R'_k$ ;
   $R_k :=$  filter  $R'_k$  to retain supported patterns;
until  $R_k = \{\}$ 

```

Como se está buscando la co-existencia de ítems en *transacciones* se puede aprovechar el orden lexicográfico del código utilizado para representar a cada ítem. El *soporte* del *itemset* XY coincide con el de YX por lo que si al guardar los *itemsets* lo hacemos con sus ítems ordenados lexicográficamente sólo guardaremos el *itemset* XY . Este orden se va a aprovechar para hacer más eficientes las operaciones a realizar por el algoritmo.

Las funciones del algoritmo son consultas a la *Base de Datos*. Para obtener R'_k se ejecuta la consulta del listado 2.3, que extiende cada k -*itemset* frecuente encontrado en el paso anterior con los ítems frecuentes de \mathcal{D} que sean lexicográficamente mayores que el mayor de los ítems del k -*itemset*. Concretamente en la extensión de un k -*itemset* basta con añadir uno a uno los ítems de la *transacción* que contienen al k -*itemset* y son lexicográficamente mayores al mayor de los ítems del k -*itemset*.

Listado 2.3: Algoritmo SETM, función merge-scan

```

INSERT INTO  $R'_k$ 
SELECT  $p.trans\_id, p.item_1, \dots, p.item_{k-1}, q.item$ 
FROM  $R_{k-1} p, SALES q$ 
WHERE  $q.trans\_id = p.trans\_id$  AND
 $q.item > p.item_{k-1}$ 

```

C_k se obtiene contando los *itemsets* del conjunto R'_k y guardando sólo los que tienen *soporte* mínimo, como muestra el listado 2.4.

Listado 2.4: Algoritmo SETM, selección de candidatos

```

INSERT INTO  $C_K$ 
SELECT  $p.item_1, \dots, p.item_k, COUNT(*)$ 
FROM  $R'_k p$ 
GROUP BY  $p.item_1, \dots, p.item_k$ 
HAVING COUNT (*)  $\geq min\_support$ 

```

El conjunto auxiliar R_k lo obtienen seleccionando las tuplas de R'_k que pueden ser extendidas, como muestra el listado 2.5.

Listado 2.5: Algoritmo SETM, *itemsets* extendibles

```

INSERT INTO  $R_k$ 
SELECT  $p.trans\_id, p.item_1, \dots, p.item_k$ 
FROM  $R'_k p, C_k q$ 
WHERE  $p - item_1 = q.item_1$  AND
...
 $p - item_k = q.item_k$ 
ORDER BY  $p.trans\_id, p.item_1, \dots, p.item_k$ 

```

Estos primeros algoritmos de *Minería de Reglas de Asociación* utilizan la misma estrategia para evitar desbordamiento de memoria al generar los candidatos a *itemset* frecuente. Como el número de candidatos es teóricamente muy grande cuando trabajamos con grandes almacenes \mathcal{D} y muchos ítems en \mathcal{I} , y como es de esperar que el número de *itemsets* en \mathcal{D} sea sensiblemente menor que el teórico, en estos algoritmos se propone reservar memoria únicamente para contar los *itemsets* que realmente están en \mathcal{D} , es decir, se reserva memoria cada vez que se encuentra un *itemset* en \mathcal{D} para el que aún no hemos hecho dicha reserva.

Este planteamiento consigue su objetivo a costa de eficiencia, ya cada una de las millones de operaciones de reserva de memoria que se llevan a cabo consume un tiempo y su acumulación resulta en un tiempo de ejecución muy superior al que conseguiríamos si hiciéramos una buena estimación de los *itemsets* contenidos en \mathcal{D} y realizáramos la reserva de memoria en una única instrucción. En los algoritmos que se muestran en la sección 2.2 se adoptará esta última estrategia.

En ambos artículos se trata por encima la generación de *reglas de asociación*. En su definición se especifica que el consecuente es un único ítem por lo que una vez conocemos todos los *k-itemsets* frecuentes basta con recorrerlos uno a uno y separarlos en dos *itemsets*, el antecedente con $k - 1$ ítems y el consecuente con el ítem restante, con lo que se obtendrán todas las *Reglas de Asociación* de \mathcal{D} que tienen *soporte* mínimo.

2.1.3. Formato de \mathcal{D}

En la sección 2.1.1 hemos cuestionado el modo de guardar los datos de \mathcal{D} para el análisis, adelantando que la codificación de los ítems de \mathcal{I} mediante números enteros es el formato más utilizado por los desarrolladores de algoritmos de *ARM*. Agrawal, Imielinski y A. Swami (1993b) no especifican nada sobre este asunto en su artículo, mientras que Houtsma y A. N. Swami (1993) proponen el uso de funciones ya implementadas en los *Gestores de Bases de Datos (DBMS)*. Como la *Minería de Reglas de Asociación* es una fase del proceso de *KDD* lo más inmediato es usar *Bases de Datos* para guardar \mathcal{D} , sin embargo la eficiencia de las funciones de un *DBMS* es notablemente inferior a la obtenida usando lenguajes de programación compilados si son muchos los datos a procesar.

Agrawal, Imielinski y A. Swami (1993a) proponen dotar a los *DBMS* de nuevas operaciones bien implementadas para poder hacer *Minería de Datos* sobre *clasificación*, *asociación* y *secuencias* directamente sobre la *DB* utilizando combinaciones de dichas operaciones para obtener *patrones* válidos para los tres modelos abordados.

Hay más investigadores que buscan el uso de *DBMS* para generar *Reglas de Asociación*, como Houtsma y A. N. Swami (1995) que proponen el estudio de *ARM* mediante lenguajes nativos de *Bases de Datos*, SQL concretamente.

Otro trabajo que busca el uso de *DBMS* para extraer reglas de asociación es el de Holsheimer y col. (1995). En su favor está la flexibilidad que permiten los *DBMS* para tratar los datos desde múltiples perspectivas, en su contra el tiempo necesario para obtener resultados frente a los algoritmos específicos de

ARM desarrollados mediante lenguajes de programación compilado. Una de las propuestas de este artículo es el uso de la jerarquía de los ítems para reducir el tamaño del repositorio a analizar, concretamente hablan de la cesta de la compra y de descubrir reglas que involucren genéricamente al ítem «cerveza» en lugar de usar como ítem cada una de las marcas de cerveza que están en \mathcal{I} , idea que será tomada por otros investigadores para reducir las dimensiones del problema y poder llevar a cabo estudios más amplios.

Aunque los algoritmos presentados de forma teórica a lo largo de esta investigación no especifican el formato físico de los datos a procesar, las comparaciones realizadas sobre los distintos algoritmos no sería correcta si no se experimenta en condiciones similares por lo que en la mayoría de los casos encontraremos implementaciones realizadas con lenguajes de programación compilados. De hecho, los almacenes \mathcal{D} disponibles en <http://fimi.ua.ac.be/data/> y <https://archive.ics.uci.edu/ml/datasets.html> tienen formato de texto plano, la mayoría de ellos en formato horizontal.

2.1.4. Fases de *ARM*

Como se ha visto en la sección anterior y exponen Agrawal, Imielinski y A. Swami en su primer artículo, la *ARM* se compone de dos fases bien diferenciadas:

1. En primer lugar hay que encontrar los conjuntos de ítems que están presentes en un porcentaje mínimo de *transacciones* de \mathcal{D} .
2. A continuación se descubren las *Reglas de Asociación* que se deducen de esos conjuntos.

La primera fase es la que mayores aportaciones científicas ha recibido en la *ARM* pues al trabajar con grandes colecciones de *transacciones* puede consumir más recursos de los disponibles o tardar más tiempo del adecuado para dar por finalizado el análisis. Cabe destacar que la tecnología utilizada para el desarrollo de los algoritmos expuestos en esta sección ha evolucionado notablemente desde los inicios de esta metodología, las comparaciones realizadas a través de los artículos presentados pueden haber variado debido a que actualmente tenemos a nuestra disposición recursos de memoria y de cómputo muy superiores a los utilizados en las dos décadas precedentes. En la sección 2.2 se estudian las propuestas más interesantes sobre esta fase, *Minería de Itemsets Frecuentes*, y se muestra una comparativa sobre la implementación del algoritmo más utilizado en la *Minería de Reglas de Asociación*.

La segunda fase es elemental y no consume tiempo ni recursos excesivos en comparación con la primera por lo que no ha habido grandes avances desde sus inicios excepto en el estudio de las métricas que mejores reglas de asociación proporcionan, dado que el número de reglas generadas puede ser demasiado grande para ser analizado correctamente. En la sección 2.3 se aborda esta fase, la generación de *reglas de asociación* a partir del conjunto de *itemsets* frecuentes, y se presenta una propuesta para ejecutarla de un modo más eficiente, propuesta que forma parte de nuestro artículo más citado.

2.2. *Minería de Itemsets Frecuentes*

Uno de los cuellos de botella de la *Minería de Reglas de Asociación* es la búsqueda de los *itemsets* frecuentes presentes en \mathcal{D} . Es una fase en que los consumos de recursos se han de optimizar, sobre todo las necesidades de memoria RAM y el uso de procedimientos con alta carga de proceso. Es tal su importancia que ha creado su propia disciplina, la *Minería de Itemsets Frecuentes (FIM)*.

En las siguientes secciones expondremos una selección de los muchos trabajos planteados sobre esta disciplina, organizados cronológicamente y destacando las principales aportaciones de cada uno de ellos. En la sección 2.2.1 se mostrarán algunos algoritmos y estructuras de datos especializadas en esta tarea, su lectura puede aportar muchas ideas para seguir investigando. En la sección 2.2.2 nos centraremos en la implementación real de los trabajos propuestos y mostramos los resultados que obtuvimos al evaluar diferentes implementaciones del algoritmo *Apriori*.

2.2.1. Algoritmos y estructuras

La atención mostrada por la comunidad científica a la búsqueda de algoritmos y estructuras de datos específicos para la *Minería de Itemsets Frecuentes* se revela en la gran cantidad de trabajos presentados. Debido al gran volumen de información que se ha de manipular se han propuesto todo tipo de representaciones de los almacenes \mathcal{D} destinados a reducir su volumen sin perder la información que contienen mediante el uso de estructuras especializadas o de taxonomías, técnicas basadas en el muestreo de los almacenes \mathcal{D} para la extracción de la máxima cantidad de conocimiento oculto, técnicas basadas en paralelismo para lograr mayor velocidad en la ejecución. . .

Agrawal y Srikant (1994a) proponen el algoritmo más estudiado en *ARM*, *Apriori* (ver algoritmo 1.4), y un par de alternativas que se adaptan mejor a algunas colecciones de *transacciones*, *AprioriTID* y *AprioriHybrid*.

La clave de *Apriori* está en la generación de candidatos, que es la fase del algoritmo que más recursos consume y que más incide en la eficiencia del resto de operaciones del algoritmo. Su propuesta se basa en la siguiente propiedad, que es la que da nombre al algoritmo: *si un candidato a k -itemset contiene un $k - 1$ -itemset que no es frecuente, a priori sabemos que no puede ser frecuente*. Gracias a esta observación en su algoritmo sólo se generan los candidatos a k -itemset frecuente cuyos subconjuntos de $k - 1$ ítems sean todos frecuentes. Esto supone un uso mayor de procesador en esta fase (tiempo de ejecución) en beneficio de un menor uso de memoria principal ya que se reduce notablemente el número de *itemsets* que teóricamente se pueden obtener del conjunto \mathcal{I} de ítems en estudio.

La principal diferencia entre *Apriori* y los dos primeros algoritmos expuestos radica en la generación de candidatos. En *Apriori* no se requiere una lectura de \mathcal{D} para obtener los candidatos pues se obtienen a partir de los *itemsets* frecuentes encontrados en una lectura previa.

La separación de las funciones de generación y poda hacen más legible el algoritmo (ver funciones 1.5 y 1.6), pero su implementación es poco eficiente pues requiere una reserva de memoria excesiva en la primera fase. En Interacción'05 (pg. 66) propusimos una solución eficiente a este problema uniendo ambas funciones en una sola (véanse los listados 1.9, 1.10 y 1.11).

En el mismo artículo proponen el algoritmo *AprioriTID* (ver algoritmo 2.6), que se basa en una representación vertical de \mathcal{D} . Este cambio de representación de \mathcal{D} surge por la poca eficiencia de la representación horizontal usada en *Apriori* para algunos bancos de transacciones específicos. Esta observación sigue presente en muchas investigaciones y pone de manifiesto que no existe un algoritmo «mejor» que todos para cualquier banco de transacciones. En un intento de lograr este objetivo proponen en el mismo artículo el algoritmo *Apriori-Hybrid*, que combina características de *Apriori* y *AprioriTID* aprovechando que el primero es más eficiente para valores pequeños de k y conforme va aumentando mejora la eficiencia del segundo.

También plantean algunas mejoras en la búsqueda de *reglas de asociación*, aunque se trata de la fase menos problemática de la *ARM* y serán expuestas en la sección 2.3.

De forma independiente a la presentación de *Apriori* Mannila, Toivonen y Verkamo (1994) observan que la generación de candidatos puede ralentizar o incluso

Listado 2.6: Algoritmo AprioriTID, 1994

```

 $\mathcal{L}_1 = \{ \text{large 1-itemsets} \};$ 
 $\overline{\mathcal{C}}_1 = \text{database} \setminus \mathcal{D};$ 
for ( $k = 2; L_{k-1} \neq \emptyset; k++$ ) do begin
   $\mathcal{C}_k = \text{apriori-gen}(\mathcal{L}_{k-1});$  //New candidates
   $\overline{\mathcal{C}}_k = \emptyset;$ 
  forall entries  $t \in \overline{\mathcal{C}}_{k-1}$  do begin
    // determines candidate itemsets in  $\mathcal{C}_k$  contained
    // in the transaction with identifier  $t.TID$ 
     $\mathcal{C}_t = \{c \in \mathcal{C}_k | (c - c[k]) \in t.\text{set-of-itemsets} \wedge (c - c[k-1]) \in t.\text{set-of-itemsets}\};$ 
    forall candidates  $c \in \mathcal{C}_t$  do begin
       $c : \text{count}++;$ 
    end
    if  $\mathcal{C}_t \neq \emptyset$  then
       $\overline{\mathcal{C}}_k += \langle t.TID, \mathcal{C}_t \rangle;$ 
    end
   $\mathcal{L}_k = \{c \in \mathcal{C}_k | c : \text{count} \geq \text{minsup}\};$ 
end
Answer =  $\bigcup_k \mathcal{L}_k;$ 

```

abortar la ejecución del algoritmo de *ARM* y proponen un análisis que permite eliminar los candidatos innecesarios para el estudio. Comparan su algoritmo, *OCD*, con *AIS* (ver listado 2.1), indicando a partir de sus experimentos que mejora notablemente su rendimiento. Basan su reducción de candidatos a procesar en un análisis combinatorio de los k -itemsets frecuentes obtenido en la k -ésima lectura de \mathcal{D} , una forma compleja de obtener los mismos candidatos que proporciona *Apriori*. También proponen el uso de muestreo para obtener las reglas de asociación mediante el análisis de parte de \mathcal{D} , sin embargo lo justifican pobremente y sólo dicen que se obtienen buenos resultados en algunas pruebas que han hecho. Mencionan continuamente el trabajo de Agrawal y Srikant (1994a) insistiendo en que obtienen resultados similares de forma totalmente independiente, partiendo del conjunto \mathcal{C}'_k , un superconjunto del generado por *Apriori* y coincidente con el obtenido en la fase de unión. Dos años más tarde presentan conjuntamente Agrawal, Mannila y col. (1996), una revisión conjunta de sus trabajos.

Park, M. Chen y P. Yu (1995) hacen una propuesta muy interesante para reducir el número de candidatos: leer los $(k+1)$ -itemsets de cada transacción mientras se hace el recuento de \mathcal{C}_k y convertirlos en un entero mediante una función hash, si al finalizar la lectura actual de \mathcal{D} algún código hash no tiene soporte mínimo no es necesario crear candidatos para los $(k+1)$ -itemsets que producen

Listado 2.7: Algoritmo DHP (fase 1), 1995

```

s = a minimum support;
set all the buckets of  $H_2$  to zero; /* hash table */
forall transaction  $t \in \mathcal{D}$  do begin
    insert and count 1-items occurrence in a hash tree;
    forall 2-subsets  $x$  of  $t$ 
         $H_2[h_2(x)] ++$ ;
end
 $\mathcal{L}_1 = \{c \mid c.count > s, c \text{ is a leaf node of the hash tree}\};$ 

```

ese código hash.

Listado 2.8: Algoritmo DHP (fase 2), 1995

```

k = 2;
 $\mathcal{D}_k = \mathcal{D}$  /* database for large k-itemsets */
while ( $|\{x \mid H_k[x] \geq s\}| \geq LARGE$ ) {
    /* make a hash table */
    gen_candidate( $\mathcal{L}_{k-1}, H_k, C_k$ );
    set all the buckets of  $H_{k+1}$  to zero;
     $\mathcal{D}_{k+1} = \emptyset$ ;
    forall transactions  $t \in \mathcal{D}_k$  do begin
        count_support( $t, C_k, k, \hat{t}$ ); /*  $\hat{t} \subseteq t$  */
        if  $|\hat{t}| > k$  do begin
            make_hasht( $\hat{t}, H_k, k, H_{k+1}, \hat{t}$ );
            if  $|\hat{t}| > k$  then  $\mathcal{D}_{k+1} = \mathcal{D}_{k+1} \cup \{\hat{t}\}$ ;
        end
    end
    end
     $L_k = \{c \in C_k \mid c.count \geq s\}$ ;
    k ++;
}

```

Proponen el algoritmo *DHP* (ver listados 2.7, 2.8 y 2.9) que reduce notablemente el número de candidatos generados por *Apriori*. La propuesta es muy acertada pero en ciertas circunstancias puede provocar una ralentización del algoritmo ya que obliga en cada lectura de \mathcal{D} a la codificación de un gran número de $(k+1)$ -itemsets sin retener más información que su número hash (para evitar un uso excesivo de memoria utilizan una función hash no unívoca, i.e., dos $(k+1)$ -itemsets distintos pueden generar el mismo número hash, por lo que si encuentran que un número hash determinado tiene *soporte* mínimo no pueden garantizar que los itemsets que generan dicho número tenga también *soporte* mínimo).

Si el número de ítems distintos de \mathcal{D} , $|\mathcal{I}|$, es muy grande y las transacciones

Listado 2.9: Algoritmo DHP (fase 3), 1995

```

gen_candidate( $\mathcal{L}_{k-1}, H_k, \mathcal{C}_k$ );
while ( $|\mathcal{C}_k| > 0$ ) {
   $\mathcal{D}_{k+1} = \emptyset$ ;
  forall transactions  $t \in \mathcal{D}_k$ 
    count_support( $t, \mathcal{C}_k, k, \hat{t}$ ); /*  $\hat{t} \subseteq t$  */
    if ( $|\hat{t}| > k$ ) then  $\mathcal{D}_{k+1} = \mathcal{D}_{k+1} \cup \{\hat{t}\}$ ;
  end
   $\mathcal{L}_k = \{c \in \mathcal{C}_k \mid c.count \geq s\}$ ;
  if ( $|\mathcal{D}_{k+1}| = 0$ ) then break;
   $\mathcal{C}_{k+1} = \text{apriori\_gen}(\mathcal{L}_k)$  /* refer to Agrawal and Srikant (1994) */
   $k++$ ;
}

```

no tienen una longitud excesiva, este algoritmo permite reducir drásticamente el *soporte* mínimo fijado por el analista, umbral que ha de ser grande en este tipo de *datasets* debido a la magnitud que puede alcanzar el número de candidatos a 2-itemsets. Los autores indican que en posteriores niveles este número se reduce por las propias características del repositorio por lo que podría prescindirse de la costosa generación de códigos hash.

J. Han y Fu (1995) extienden el estudio de *reglas de asociación* añadiendo información a la *Base de Datos* de transacciones. Proponen añadir al estudio una taxonomía de los ítems del servicio de modo que puedan obtener información a múltiples niveles, reduciendo mediante agrupación el número de ítems a procesar y pudiendo obtener mejor información en el análisis. La primera lectura de \mathcal{D} trata a todos los ítems según su pertenencia al máximo nivel de la taxonomía usada, de modo que la mayoría de ítems (que son pocos en comparación con el total al estar agrupados) tengan *soporte* mínimo y puedan seguir siendo analizados. En la siguiente lectura de \mathcal{D} sólo se consideran las transacciones que contienen ítems frecuentes al primer nivel de la taxonomía, tratándolos ahora como ítems de un nivel inferior. Se prosigue con sucesivas lecturas de \mathcal{D} y niveles inferiores en la taxonomía hasta detenerse cuando no se encuentran nuevos *k-itemsets*. Proponen los algoritmos ML_T2L1, ML_T1LA, ML_TML1 y ML_T2LA.

Conforme van aumentando las propuestas sobre *ARM* se incrementan notablemente los repositorios de transacciones existentes, con lo que los algoritmos van perdiendo su eficiencia y los analistas se ven obligados a incrementar el *soporte* mínimo para poder llevar a cabo el estudio. Para resolver el problema de la magnitud, Savasere, Omiecinski y Navathe (1995) proponen el algoritmo *Parti-*

tion (ver algoritmo 2.10), que comienza dividiendo \mathcal{D} en n particiones y realiza el análisis en varias fases, utilizando un *soporte* mínimo inferior al fijado por el analista pues es de esperar que el *soporte* de cada *itemset* en una partición será menor a su *soporte* en \mathcal{D} .

Listado 2.10: Algoritmo Partition, 1995

```

P = partition_database( $\mathcal{D}$ )
n = number of transactions
for i = 1 to n do begin // Phase I
    read_in_partition( $p_i \in P$ )
     $\mathcal{L}^i$  = gen_large_itemsets( $p_i$ )
end
for (i = 2;  $\mathcal{L}_i^j \neq \emptyset, j = 1, 2, \dots, n; i++$ ) do
     $\mathcal{C}_i^G = \bigcup_{i=1,2,\dots,n} \mathcal{L}_i^j$  // Merge Phase
for i = 1 to n begin // Phase II
    read_in_partition( $p_i \in P$ )
    forall candidates  $c \in \mathcal{C}^G$  gen_count( $c, p_i$ )
end
 $\mathcal{L}^G = \{c \in \mathcal{C}^G | c.count \geq minSup\}$ 

Answer =  $\mathcal{L}^G$ ;

```

La fase I del algoritmo realiza n iteraciones sobre \mathcal{D} , una por cada partición creada. Durante la iteración i sólo se considera la partición p_i . La función `gen_large_itemsets` toma una partición p_i como input y genera los *k-itemsets* locales de todos los tamaños, $\mathcal{L}_1^i, \mathcal{L}_2^i, \dots, \mathcal{L}_k^i$ como output. En la fase de mezcla son combinados todos los *itemsets* frecuentes de la misma longitud de las n particiones. El conjunto de candidatos globales de longitud j se obtiene con $\mathcal{C}_j^G = \bigcup_{i=1}^n \mathcal{L}_j^i$. En la fase II el algoritmo cuenta el *soporte* de todos los candidatos globales en las n particiones y obtiene todos los *itemsets* con *soporte* mínimo. El algoritmo lee \mathcal{D} una vez en la fase I y otra vez en la fase II, lo que supone un gran ahorro en el proceso de lectura de datos del disco.

Mueller (1995) hace un buen análisis de los algoritmos existentes sobre *ARM* y propone los algoritmos *SEAR* (basado en una nueva estructura de datos, *SPTID*), *SPEAR*, *SPINC* y una versión paralela de los dos primeros.

Toivonen (1996) propone un método de muestreo con el que encontrar todas las *reglas de asociación* de una *Base de Datos* con una única lectura de la misma, si no se busca excesiva precisión, y con una segunda lectura si buscamos mayor precisión. La idea consiste en guardar en memoria una muestra representativa de \mathcal{D} y obtener los *itemsets* frecuentes según un *soporte* mínimo inferior al planteado

en el estudio. Al trabajar sólo con parte de \mathcal{D} y poder alojarlo en memoria se gana mucho en eficiencia, pudiendo obtenerse de forma muy rápida los principales *patrones* presentes en \mathcal{D} y, si se considera necesario, se puede hacer una segunda lectura de \mathcal{D} para analizar las transacciones no presentes en la muestra inicial. El uso de muestras de \mathcal{D} para obtener información preliminar de las *reglas de asociación* que contiene se ha mostrado eficiente en muchos artículos pero en ninguno de ellos se analiza la representatividad de la muestra ya que proceder a su análisis conllevaría un gasto de tiempo que iría en contra de uno de los principales propósitos de los algoritmos de *ARM*, su inmediata respuesta.

Agrawal y Shafer (1996) hacen su incursión en la ejecución de algoritmos de *ARM* en paralelo, con el fin de repartir entre varios procesadores la enorme carga de cómputo y memoria requeridos para su proceso cuando trabajamos con grandes *Bases de Datos de transacciones*. Proponen tres algoritmos - *Count Distribution*, *Data Distribution* y *Candidate Distribution* - con los que ponen a prueba las dificultades de comunicación entre procesadores, uso de memoria compartida y sincronización de memorias locales. Se trata de un interesante artículo de referencia si se quiere paralelizar la obtención de *reglas de asociación*. En los tres algoritmos parten de una misma base: dividen \mathcal{D} en particiones disjuntas que son repartidas entre los N equipos que realizarán el proceso, usando procesadores, memoria principal y en disco totalmente independientes y compartiendo únicamente los resultados necesarios.

El algoritmo *Count Distribution* se limita a realizar en cada paso la generación y recuento de candidatos de la partición que le corresponde, combinando los resultados al final del mismo. En esta aproximación no se aprovecha la posibilidad de utilizar memoria compartida, ya que cada equipo utiliza su propia memoria para evaluar el árbol completo de *itemsets* frecuentes. Para aprovechar el incremento de memoria principal al añadir nuevos equipos proponen el algoritmo *Data Distribution* cuya primera fase es idéntica a la del algoritmo *Count Distribution* pero a continuación cada procesador hace el recuento de candidatos mutuamente excluyentes con el resto de procesadores. Al compartir todos el mismo repositorio \mathcal{D} se debe utilizar un equipo que permita una rápida comunicación entre procesadores. Ambos algoritmos requieren de una sincronización de todos los procesadores al finalizar cada paso para intercambiar recuentos o *itemsets* frecuentes, lo que causa paradas de procesadores si el flujo de trabajo no está correctamente balanceado.

El algoritmo *Candidate Distribution* pretende resolver este problema al no requerir sincronización hasta el final del proceso. En el paso l , determinado heu-

rísticamente, el algoritmo divide \mathcal{L}_{l-1} entre los procesadores de modo que el procesador P^i pueda generar un único C_m^i ($m \geq l$) independiente del resto de procesadores ($C_m^i \cap C_m^j = \emptyset, i \neq j$). Simultáneamente, las transacciones se replican selectivamente de modo que cada procesador pueda hacer el recuento de C_m^i de forma independiente al resto de procesadores. Hasta el paso l se utilizará el algoritmo *Count Distribution* o *Data Distribution*, a partir de $k = l$ se procede con el algoritmo *Candidate Distribution*.

Brin y col. (1997) proponen un nuevo algoritmo para el minado de reglas de asociación, *DIC*, que reduce el número de lecturas de la *Base de Datos* y genera un número menor de candidatos a *itemset* que los algoritmos basados en muestras. Se basan en ir creando un árbol similar a \mathcal{L} leyendo \mathcal{D} en bloques de M transacciones, de los que va deduciendo qué *itemsets* pueden ser frecuentes y cuáles no. También hablan de *reglas de implicación*, que no sólo miden la co-ocurrencia de ítems en transacciones, considerando lo que ellos llaman «verdadera implicación» entre *antecedentes* y *consecuentes* normalizados. Proponen ideas sobre el reordenamiento de los ítems presentes en \mathcal{D} para lograr mayor eficiencia. Introducen el concepto de *convicción* como alternativa a la *confianza*, obteniéndola a partir de $P(A)P(\neg B)/P(A, \neg B)$.

Srikant y Agrawal (1997) presentan el algoritmo *Basic* (ver algoritmo 2.11) de extracción de *reglas de asociación* de una colección de transacciones cuyos ítems están jerarquizados en una taxonomía. Este enfoque, a diferencia de futuros trabajos que también usan taxonomías, no reduce el número de ítems a procesar en la fase de *FIM* pues considera como ítems tanto a los ítems de \mathcal{D} como a sus ancestros en la jerarquización, completando todas las transacciones de \mathcal{D} con los ancestros de cada uno de sus ítems (sin duplicados). Realmente se aumenta el conjunto de ítems a considerar, \mathcal{I} , los ítems poco frecuentes desaparecerán del estudio pero quedarán en las transacciones sus ancestros de modo que siempre podremos extraer información sobre ellos aunque sea a otro nivel de la taxonomía utilizada.

Una vez utilizado el algoritmo *Basic* piensan en algunas optimizaciones del mismo, que darán lugar al algoritmo *Cumulate* (ver algoritmo 2.12).

1. **Filtrado de ancestros añadidos a las transacciones.** No añaden a las transacciones todos los ancestros de sus ítems ya que alguno de ellos, o incluso el propio ítem, no forman parte de los candidatos a estudiar.
2. **Pre-cómputo de ancestros.** En lugar de recorrer la taxonomía para conocer los ancestros de un ítem se crea una tabla de ancestros para cada ítem.

Listado 2.11: Algoritmo Basic, 1997

```

 $\mathcal{L}_1 := \{\text{frequent 1-itemsets}\};$ 
 $k := 2;$  //  $k$  represents the pass number
while ( $\mathcal{L}_{k-1} \neq \emptyset$ ) do begin
     $\mathcal{C}_k :=$  New candidates of size  $k$  generated from  $\mathcal{L}_{k-1};$ 
    forall transactions  $t \in \mathcal{D}$  do begin
        Add all ancestors of each item in  $t$  to  $t$ , removing any duplicate;
        Increment the count of all candidates in  $\mathcal{C}_k$  that are contained in  $t$ ;
    end
     $\mathcal{L}_k :=$  All candidates in  $\mathcal{C}_k$  with minimum support;
     $k := k + 1;$ 
end
Answer :=  $\bigcup_k \mathcal{L}_k;$ 

```

3. **Eliminación de itemsets que contienen un ítem y su ancestro.** Justificado en los siguientes lemas:

Lema 1. *El soporte de un itemset X que contiene tanto el ítem x como su ancestro \hat{x} coincide con el soporte del itemset $X - \hat{x}$.*

Lema 2. *Si \mathcal{L}_k , el conjunto de k -itemsets frecuentes, no incluye ningún itemset que contenga tanto un ítem como su ancestro, el conjunto de candidatos \mathcal{C}_{k+1} generado no contendrá ningún itemset que contenga tanto un ítem como su ancestro.*

También presentan el algoritmo *EstMerge* (ver algoritmo 2.13), que en cada lectura de \mathcal{D} hace un muestreo previo basado en el subconjunto \mathcal{D}_S para obtener una mejor estimación de los candidatos a k -itemset, \mathcal{C}_k . Esta lectura extra de ciertas transacciones de \mathcal{D} inicialmente consumen algo más de recursos y de tiempo, sin embargo cuando se lee \mathcal{D} para obtener \mathcal{L}_k no se buscará ninguno de los candidatos que no han superado el *soporte* mínimo corregido para la muestra (con la muestra se utiliza un *soporte* mínimo reducido por el factor 0.9), con lo que el tiempo empleado con la muestra se recupera ampliamente al ahorrar muchísimas operaciones de búsqueda y conteo de candidatos.

Srikant, Vu y Agrawal (1997) proponen tres nuevos algoritmos, *Reorder*, *MultipleJoins* y *Direct*, para introducir restricciones sobre los ítems a procesar dentro de los algoritmos existentes de *ARM*. Justifican su trabajo en que a menudo el analista necesita hacer un filtrado de *reglas de asociación* en función de la presencia o ausencia de cierto ítem (o de su ancestro o descendiente en una

Listado 2.12: Algoritmo Cumulate, 1997

```

Compute  $\mathcal{T}^*$ , the set of ancestors of each item, from  $\mathcal{T}$  // Optimization 2
 $\mathcal{L}_1 := \{\text{frequent 1-itemsets}\}$ ;
 $k := 2$  //  $k$  represents the pass number
while ( $\mathcal{L}_{k-1} \neq \emptyset$ ) do begin
     $\mathcal{C}_k :=$  New candidates of size  $k$  generated from  $\mathcal{L}_{k-1}$ ;
    if ( $k = 2$ ) then // Optimization 3
        Delete any candidate in  $\mathcal{C}_2$  that consists of an item and its ancestor;
        Delete any ancestors in  $\mathcal{T}^*$  that are not present in any of the
            candidates
        in  $\mathcal{C}_k$  // Optimization 1
    forall transactions  $t \in \mathcal{D}$  do begin
        forall item  $x \in t$  do
            Add all ancestors of  $x$  in  $\mathcal{T}^*$  to  $t$ ;
            Remove any duplicates from  $t$ ;
            Increment the count of all candidates in  $\mathcal{C}_k$  that are contained in  $t$ ;
        end
         $\mathcal{L}_k :=$  All candidates in  $\mathcal{C}_k$  with minimum support;
         $k := k + 1$ ;
    end
Answer :=  $\bigcup_k \mathcal{L}_k$ ;

```

taxonomía) y que es mucho más eficiente aplicar ese filtro antes de proceder con el algoritmo de *ARM* que después de haberlo hecho.

Zaki, Parthasarathy y col. (1997) proponen cuatro nuevos algoritmos, *Eclat*, *MaxEclat*, *Clique* y *MaxClique*. Todos ellos se basan en el clustering de *itemsets* y en el esquema *divide y vencerás* y en todos se propone leer \mathcal{D} sólo una vez, con el ahorro de tiempo que ello supone. *Eclat* utiliza la representación vertical de \mathcal{D} y genera un número de candidatos mayor que *Apriori* debido a que no guarda el soporte de todos los *itemsets* leídos. Su principal ventaja sobre *Apriori* está en que sólo realiza una lectura de \mathcal{D} , lo que supone una ejecución más rápida en general en una época en que el acceso a disco era aún demasiado costosa.

Bayardo (1998) afronta el problema de la presencia de k -*itemsets* largos en \mathcal{D} que dificultan su análisis con los métodos tradicionales, citando básicamente *Apriori*. Propone el algoritmo *Max-Miner* para extraer únicamente los *itemsets* frecuentes *maximales*, aquellos que no tienen un superconjunto que sea también frecuente. Dado que todo *itemset* frecuente es un subconjunto de un *itemset* frecuente maximal, la salida de su algoritmo representa implícitamente y de modo conciso a todos los *itemsets* frecuentes presentes en \mathcal{D} . El ahorro que supone el almacenamiento de sólo los *itemsets* frecuentes maximales frente a *Apriori* proporciona a su algoritmo mayor rapidez de ejecución en algunas *Bases de Datos*,

Listado 2.13: Algoritmo EstMerge, 1997

```

 $\mathcal{L}_1 := \{\text{frequent 1-itemsets}\};$ 
Generate  $\mathcal{D}_S$ , a sample of the database, in the first pass;
 $k := 2$ ; //  $k$  represents the pass number
 $\mathcal{C}_1'' := \emptyset$ ; //  $\mathcal{C}_k''$  represents candidates of size  $k$  to
// be counted with candidates of size  $k+1$ 
while ( $\mathcal{L}_{k-1} \neq \emptyset$  or  $\mathcal{C}_{k-1}'' \neq \emptyset$ ) do begin
     $\mathcal{C}_k :=$  New candidates of size  $k$  generated from  $\mathcal{L}_{k-1} \cup \mathcal{C}_{k-1}''$ ;
    Estimate the support of the candidates in  $\mathcal{C}_k$  by making a pass over  $\mathcal{D}_S$ ;
     $\mathcal{C}_k' :=$  Candidates in  $\mathcal{C}_k$  that are expected to have minimum support and
    candidates all of whose parents are expected to have minimum
    support;
    Find the support of the candidates in  $\mathcal{C}_k' \cup \mathcal{C}_{k-1}''$  by making a pass over  $\mathcal{D}$ ;
    Delete all candidate in  $\mathcal{C}_k$  whose ancestors in  $\mathcal{C}_k'$  do not have minimum
    support;
     $\mathcal{C}_k'' :=$  Remaining candidates in  $\mathcal{C}_k$  that are not in  $\mathcal{C}_k'$ ;
     $\mathcal{L}_k :=$  All candidates in  $\mathcal{C}_k'$  with minimum support;
    Add all candidates in  $\mathcal{C}_{k-1}''$  with minimum support to  $\mathcal{L}_{k-1}$ ;
     $k := k + 1$ ;
end
Answer :=  $\bigcup_k \mathcal{L}_k$ ;

```

sin embargo este ahorro puede perderse cuando queremos obtener información más detallada sobre algunos *itemsets* frecuentes que no sean maximales.

Holt y Chung (1999) aplican los algoritmos *Apriori* y *DHP* al minado de *Bases de Datos* de texto y proponen los algoritmos *Multipass-Apriori*, *M-Apriori* y *Multipass DHP (M-DHP)* específicos para tratar *Bases de Datos* de texto usando técnicas de *ARM*.

J. Han, Pei y Yin (2000) introducen una nueva estructura, *FP-Tree*, para guardar los *itemsets* presentes en \mathcal{D} , y con ella el algoritmo *FP-Growth*, con el que evitan la generación de candidatos, uno de los cuellos de botella del algoritmo *Apriori*. En su artículo proponen otro algoritmo para construir la estructura *FP-Tree* en la que recoger toda la información relevante de \mathcal{D} . *FP-Tree* realmente es una representación compacta de \mathcal{D} para poder manipularlo en la memoria principal del ordenador. Es por ello que si el número de ítems distintos que contiene es muy grande las prestaciones de *FP-Growth* se reducen notablemente teniendo que trabajar con un *soporte* alto para poder llevar a cabo su ejecución.

Hipp, Güntzer y Nakhaeizadeh (2000) presentan una comparativa entre los cuatro algoritmos más representativos de *ARM*: *Apriori*, *FP-Growth*, *Partition* y *Eclat*, concluyendo que no existen grandes diferencias en cuestión de tiempo

de ejecución y que la propia distribución de \mathcal{D} puede influir en el hecho de que un algoritmo funcione mejor en una colección de datos concreta respecto a los demás algoritmos.

Özel y Güvenir (2001) proponen el algoritmo *PHP*, basado en *DHP* pero utilizando hashing perfecto con lo que no es necesario hacer doble recuento de los candidatos a $(k + 1)$ -*itemsets* frecuentes. No especifican la función hash que usan ni aseguran que su método pueda ser utilizado con cualquier colección \mathcal{D} , lo que no es descartable pues el número de candidatos a $(k + 1)$ -*itemsets* puede ser tan grande que no sea viable su almacenamiento en un vector de códigos hash.

D.-I. Lin y Kedem (2002) hacen una propuesta interesante desde el punto de vista de la *Minería de Itemsets Frecuentes*, completar la búsqueda bottom-up de *itemsets* frecuentes mediante una búsqueda top-down simultánea que puede reducir drásticamente el número de candidatos generado por el algoritmo y el número de lecturas de \mathcal{D} . El único inconveniente de este algoritmo es que no se guarda el *soprote* real de todos los *itemsets* frecuentes cuyos candidatos no han sido generados, con lo que sería necesaria una nueva lectura de \mathcal{D} para obtenerlo. En su artículo indican que los algoritmos de *ARM* utilizados son eficientes cuando los *itemsets* frecuentes de \mathcal{D} son cortos, decreciendo su eficiencia conforme aumenta su longitud. Presentan un nuevo algoritmo que combina las búsquedas bottom-up y top-down, *Pincer-Search*, para mantener y actualizar una nueva estructura de datos, *MFC S* (*Maximum Frequent Candidate Set*) con la que obtener el conjunto de *itemsets* frecuentes maximales, *MFS*, sin necesidad de usar un conjunto grande de candidatos, lo que indican que ahorrará mucho tiempo de CPU al proceso por ahorrarse búsquedas y conteo de candidatos que no llegarán a ser *itemsets* frecuentes.

En su artículo presentan un esquema general muy claro sobre el proceso de *Minería de Itemsets Frecuentes*:

Throughout the execution, the set of all *itemsets* is partitioned, perhaps implicitly, into three sets:

1. *frequent*: This is the set of those *itemsets* that have so far been discovered as frequent.
2. *unfrequent*: This is the set of those *itemsets* that so far have been discovered as infrequent.
3. *unclassified*: This is the set of all the other *itemsets*.

Initially, the frequent and the infrequent sets are empty. Throughout the execution, they grow at the expense of the unclassified set. The execution terminates when the unclassified set becomes empty, and then, of course, all the maximal frequent *itemsets* are discovered.

Holt y Chung (2002) presentan el algoritmo *IHP* mostrando con sus experimentos que se comportan mejor que *Apriori* y *DHP* en colecciones con transacciones largas. En la primera lectura de \mathcal{D} , *IHP* realiza el recuento de cada ítem de \mathcal{I} además de una TID Hash Table (THT) para cada ítem utilizando una función hash, lo que supone un aumento de requerimiento de memoria principal pero limitado por el valor máximo obtenido en la función hash y no por el número total de transacciones como ocurría en Partition. Su principal aportación consiste en la eliminación de candidatos mediante una comprobación previa de las THT de cada ítem presente en el k -candidato a comprobar, esta reducción supone un recuento más rápido de los candidatos dado su menor número. También incorporan técnicas de reducción de ítems en transacciones y de *transacciones* que no pueden generar más k -*itemsets* frecuentes, técnicas ya revisadas en la literatura expuesta en esta sección.

Cheung y Zaiane (2003) proponen una nueva estructura, CATS-Tree, que extiende la idea de la estructura FP-Tree para mejorar la compresión de \mathcal{D} y permitir el minado de *itemsets* frecuentes sin generación de candidatos. A destacar la escalabilidad de esta estructura que permite modificar el *soporte* mínimo sin tener que reconstruirla, lo que hace posible también la inserción o eliminación de transacciones sin repetir todo el proceso de *ARM*.

Hay diversas aportaciones para optimizar el uso de memoria RAM y el acceso a los datos almacenados. D.-I. Lin y Kedem (2002) proponen almacenar los *itemsets* en una estructura capaz de gestionarlos de un modo más eficiente, permitiendo hacer su recuento en dos sentidos, comenzando con 1-*itemsets* e incrementando su tamaño (del mismo modo que hace *Apriori*) y almacenando simultáneamente los k -*itemsets* frecuentes que va encontrando en \mathcal{D} para detectar los frecuentes en sentido inverso, reduciendo su tamaño. T.-P. Hong, Kuo y S.-L. Wang (2004) proponen un algoritmo que busca *reglas de asociación* difusas en transacciones con datos cuantitativos, reduciendo tiempos de ejecución en esta fase. En su tesis, G. Liu (2005) presenta una detallada descripción de su investigación sobre *Minería de Itemsets Frecuentes*. G. Liu, Lu y J. X. Yu (2007) proponen el uso de CFP-tree, una estructura compacta para el almacenamiento de *itemsets* y optimizada para realizar búsquedas. L. Liu y col. (2007) utilizan pa-

ralelismo para optimizar el proceso. Tsay, T.-J. Hsu y J.-R. Yu (2009) presentan otra estructura, FIU-tree, con la que sólo se ha de acceder dos veces a \mathcal{D} para encontrar todos los *itemsets* frecuentes que contiene. Muchas de estas publicaciones son posteriores a este periodo de nuestra investigación por lo que no pudieron ser incorporadas en su momento.

2.2.2. Evaluación de diferentes implementaciones

La *Minería de Reglas de Asociación* va adquiriendo más atención en el mundo científico gracias a la tecnología, que cada vez nos permite adquirir mayor cantidad de datos y nos proporciona herramientas para su análisis. Todas las disciplinas científicas se benefician de esta situación, y es en la Informática donde se centran las mayores expectativas sobre este tema. En 2003 y 2004 se realizaron dos Workshops para discutir sobre la *implementación* de los algoritmos de *FIM*, ambos con el sugerente título *Workshop on Frequent Itemset Mining Implementations* (FIMI'03³ y FIMI'04⁴).

La implementación de cualquier aportación teórica es esencial para poder probar su valía, una mala implementación puede provocar que no se siga con un desarrollo teórico que en la práctica da «malos» resultados. Estamos hablando de tratar con enormes cantidades de datos y queremos hacerlo en un tiempo prudencial y obtener el máximo de información, conocimiento de calidad que podría pasar desapercibido entre tal cantidad de datos. Esta tarea es cosa de informáticos, expertos en programación que sean capaces de desarrollar código eficiente. A raíz de la revisión de FIMI encontramos que podíamos probar algunas de las implementaciones presentadas en los Workshops, incluso revisar su código para modificarlo o aprender de él.

Elegimos *Apriori* como base para nuestra implementación. Se trata de un algoritmo flexible que sirve de base para experimentar con las propuestas vistas al principio de esta sección y con nuestras aportaciones. En este periodo nos sirvió mucho la investigación de Bodon (2003, 2004, 2005, 2006; Rácz, Bodon y Schmidt-Thieme, 2005). Bodon es uno de los desarrolladores de A C++ Frequent Itemset Mining Template Library⁵ cuya intención es dotar a los investigadores de herramientas específicas para la tarea de *FIM*. Es una librería escrita en C++ que trabaja con la *STL* para garantizar su funcionalidad y modularidad.

³<http://fimi.ua.ac.be/fimi03/>

⁴<http://fimi.ua.ac.be/fimi04/>

⁵http://www.cs.bme.hu/~bodon/en/fim_env/index.html

Bodon mantiene una página web con su propia implementación de *Apriori*⁶, de donde podemos descargarnos los ejecutables para Linux y Windows, el código fuente y documentación detallada sobre el código y algunas consideraciones teóricas. Hemos utilizado dos desarrollos más del algoritmo, el primero realizado por Bart Goethals⁷ y el segundo por Christian Borgelt^{8,9}. En la sección «Implementaciones» del FIMI¹⁰ hay mucho más trabajo para revisar, pero nosotros nos queríamos centrar en el algoritmo *Apriori*.

Borgelt desarrolla su código en C mientras que Bodon y Goethals utilizan C y C++. C es un lenguaje muy eficiente aunque el código generado en C puede llegar a ser muy difícil de mantener y modificar. C++ aún no llega a la excelencia de C sin embargo puede utilizar toda su eficiencia y permite trabajar con clases que facilitan enormemente el desarrollo, mantenimiento y modificación del código. En las conclusiones de esta sección influye mucho esta circunstancia, el código de Borgelt ganará en efectividad frente a las otras dos implementaciones pero su mantenimiento o expansión para añadirle nuestras aportaciones requerirá de muchas más horas de trabajo jugando con el riesgo de hacer un mal uso de una *estructura* que no es capaz de autogestionarse (las clases sí que permiten tener un control total sobre ellas).

Para llevar a cabo los experimentos usamos dos almacenes \mathcal{D} publicados en FIMI, BMS-POS.dat y BMS-View1.dat, y datos propios de un servidor al que teníamos acceso en ese momento. Los datos propios fueron seleccionados y preprocesados a partir del *fichero de log* del servidor, en la etapa de transformación convertimos las páginas web (largas cadenas de caracteres) en códigos numéricos, creando ficheros con los que poder reconocer las páginas a partir del código utilizado en la fase de *Minería de Datos*. Con BMS-POS.dat y BMS-View1.dat no pudimos hacer lo mismo porque no teníamos información sobre las páginas web correspondientes a los números utilizados para codificarlas.

Las ejecuciones se hicieron en similares condiciones para poder hacer comparaciones válidas. Todas las aplicaciones usadas en esta tesis se han ejecutado utilizando los recursos que tenemos la mayoría de investigadores, uno de los objetivos es que se puedan aplicar todas las propuestas realizadas sin necesidad de un supercomputador. Este objetivo se ha mantenido hasta el final de esta in-

⁶<http://www.cs.bme.hu/~bodon/en/apriori/>

⁷<http://adrem.ua.ac.be/~goethals/software/>

⁸<http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html#assoc>

⁹En el CD están las tres implementaciones, en la carpeta bin.

¹⁰<http://fimi.ua.ac.be/src/>

vestigación en que se sugiere la ejecución de algoritmos de *Minería de Datos* utilizando *smartphones* u otros dispositivos «ligeros» con los que cualquier investigador puede llegar a obtener resultados válidos y en tiempos prudentes en cualquier momento y lugar.

Para decidir qué *soporte* mínimo se utilizaría en cada prueba hicimos antes algunos ensayos comenzando con *soporte* mínimo nulo (con lo que se buscan todos los *itemsets* presentes en \mathcal{D}) obteniendo en los tres casos problemas de desbordamiento de memoria y pérdida de la información obtenida tras la salida inesperada de la ejecución de los programas. Fuimos incrementando el *soporte* mínimo hasta alcanzar la primera ejecución completa y a partir de ahí decidimos el rango de valores que usaríamos como *soporte* mínimo para cada uno de los almacenes \mathcal{D} .

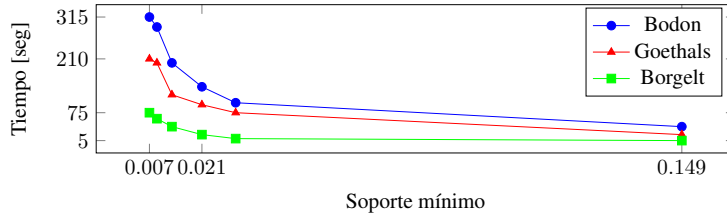


Figura 2.1: Tiempos de ejecución con la colección BMS-POS.dat

La colección de datos BMS-POS.dat (11.4MB) tiene 515 597 transacciones con 1 657 ítems distintos y un total de 3 360 020 ítems. Los tiempos de ejecución para valores del *soporte* mínimo entre 0.007 % y 0.149 % se muestran en la figura 2.1.

La figura 2.2 muestra los tiempos de ejecución obtenidos al analizar el almacén \mathcal{D} BMS-View1.dat (2.1MB), que contiene 149 639 líneas con el par {transacción, ítem} (representación vertical de \mathcal{D}), con 497 ítems distintos. Lo convertimos en un fichero en que cada línea contuviera sólo una transacción (representación horizontal de \mathcal{D}), BMS-View2.dat¹¹ (1MB), pues las implementaciones que estamos utilizando leen este formato de ficheros, obteniendo un total de 59 602 transacciones.

Los datos propios (5.1MB) recogían 585 149 transacciones con 2 213 ítems distintos y un total de 1 732 617 ítems. Los tiempo de ejecución de las implementaciones puestas a prueba con este almacén \mathcal{D} se muestran en la figura 2.3,

¹¹Disponible en el CD adjunto, en la carpeta datos

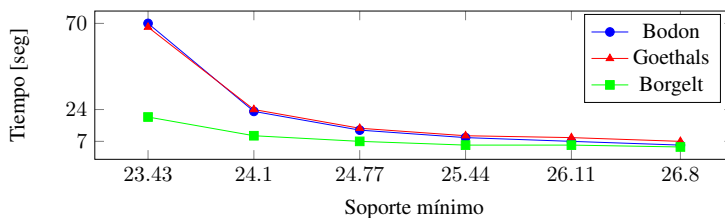


Figura 2.2: Tiempos de ejecución con la colección BMS-View1.dat

con *soporte* mínimo entre 0.51 % y 17.09 %. En este caso se observan tiempos de ejecución muy grandes (alcanzando los 11 minutos al usar la implementación de Goethals con *soporte* mínimo del 0.51 %), lo que pone en evidencia, al compararlo con los tiempos de ejecución de BMS-POS.dat, que además del número de transacciones y de datos a procesar (este último almacén \mathcal{D} contiene el doble de datos que el nuestro) en los procesos de *Minería de Reglas de Asociación* son importantes las relaciones presentes en sus ítems, es decir, puede afectar más al análisis la propia distribución de los datos que su cantidad.

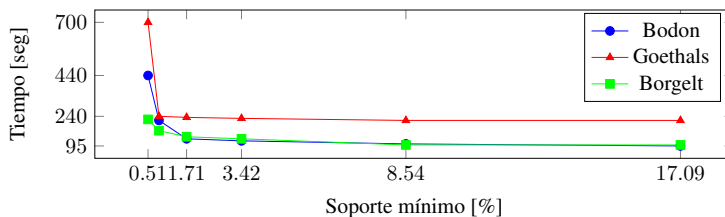


Figura 2.3: Tiempos de ejecución con la colección de datos propios

La implementación de Borgelt, realizada con C, era la más eficiente de las tres en todas las ejecuciones excepto para valores intermedios del *soporte* mínimo (entre 1.71 % y 17.09 %) en el análisis de nuestros datos en que los tiempos son sensiblemente menores al usar la implementación de Bodon.

Los resultados fueron tan dispares que nos sumergimos en el código usado para analizar el porqué de estas diferencias y poder aprovechar alguna de ellas para introducir nuestras aportaciones o bien crear una nueva implementación con lo mejor que pudiéramos extraer de cada una de ellas. Descubrimos que todos tenían alguna parte mejor implementada que las otras, pero intentar modificarla no parecía fácil a pesar de que Bodon estaba replanteando su trabajo para trabajar

con estructuras más modulares, las clases y librerías que aporta el lenguaje C++, ganando en legibilidad a cambio de perder la velocidad de C.

Al repasar el código e intentar modificar algunos de sus aspectos descubrimos ciertos procesos que podían optimizarse fácilmente, lo que nos llevó a una nueva publicación, por primera vez en una revista de impacto, descrita en la siguiente sección.

2.3. Generación de *reglas de asociación*

Una vez resuelta la fase de *Minería de Itemsets Frecuentes* se utiliza el conjunto \mathcal{L} de *itemsets* frecuentes para obtener las *Reglas de Asociación* que contiene el almacén de transacciones \mathcal{D} . Ya tenemos todos los *itemsets* cuyo *soporte* sea superior al mínimo prefijado, minSup , pero aún no tenemos las *Reglas de Asociación*, para ello hemos de recorrer todo \mathcal{L} , de cada uno de los *itemsets* frecuentes que contiene obtener todas sus particiones en dos subconjuntos no vacíos y comparar la *confianza* de las dos reglas que genera con el umbral mínimo establecido para el estudio, minConf .

Apriori es un algoritmo hermoso, sencillo y moldeable, de ahí la atención que ha recibido por parte de la comunidad científica. Está descrito de un modo muy general por lo que se puede modificar con facilidad cualquiera de sus partes sin perder su esencia. La mayoría de los trabajos expuestos en la sección 2.2 y de los que mostraremos en la sección 2.4 se centran en mejorar la primera fase de este algoritmo, la *Minería de Itemsets Frecuentes*, debido a su gran necesidad de recursos y de procesos de lectura/escritura en disco. Si leemos con atención el párrafo anterior podremos ver que la fase de generación de *reglas de asociación* también tiene mucha carga de procesos:

1. Hemos de leer cada *itemset* frecuente de \mathcal{L} , lo que se conseguirá mediante un simple bucle que recorra todos sus nodos.
2. Cada *itemset* se divide en un par de subconjuntos no vacíos, X_1 y X_2 , que serán el antecedente y consecuente de dos *Reglas de Asociación* diferentes, $X_1 \rightarrow X_2$ y $X_2 \rightarrow X_1$.
3. Para obtener la *confianza* de cada regla hemos de buscar en \mathcal{L} el *soporte* de X_1 y de X_2 , calcular su cociente y determinar si tienen o no confianza mínima.

Los dos primeros pasos son elementales, sin embargo el paso 3 es más complejo de lo que parece a simple vista. Para encontrar el *soporte* de cada k -*itemset* hemos de hacer k búsquedas en \mathcal{L} . Primero hemos de localizar en \mathcal{L}_1 el nodo que representa a su primer ítem, una vez encontrado entramos en la rama que se deriva de él y localizar en \mathcal{L}_2 el nodo que representa a su segundo ítem, siguiendo el proceso hasta localizar su k -ésimo ítem en \mathcal{L}_k . Aunque usemos algoritmos de búsqueda eficiente en cada rama tenemos que hacer cientos de comparaciones para encontrar en \mathcal{L} cada uno de los *itemsets* obtenidos en el paso 2, lo que nos hizo pensar que es una fase en la que podría aplicar alguna optimización.

2.3.1. `genrules()`

En esta sección nos vamos a detener en la obtención de las *reglas de asociación*, concretamente en la función `genrules()` que encuentra las reglas que verifican nuestros *itemsets* frecuentes. Aparentemente es la parte más rápida del algoritmo *Apriori*. El algoritmo original propuesto por Agrawal y Srikant (1994a) se muestra en el listado 1.7. `genrules()` recibe como parámetros dos k -*itemsets*, el primero es siempre l_k y el segundo es un subconjunto de l_k , del que se extraerán los *antecedentes* de las *reglas derivadas* de l_k . En cada llamada a `genrules()` se obtiene la *confianza* de una regla del tipo $a_m \Rightarrow c_i$, donde $c_i = l_k - a_m$, lo que supone dos llamadas a la función *soporte*, función que ha de recorrer \mathcal{L} hasta el nivel determinado por el número de ítems del k -*itemset* que recibe como parámetro: $k + (k - m)$ búsquedas.

Agrawal y Srikant observaron la siguiente característica:

We showed earlier that if $a \Rightarrow (l - a)$ does not hold, neither does $\tilde{a} \Rightarrow (l - \tilde{a})$ for any $\tilde{a} \subset a$. By rewriting, it follows that for a rule $(l - c) \Rightarrow c$ to hold, all rules of the form $(l - \tilde{c}) \Rightarrow \tilde{c}$ must also hold, where \tilde{c} is a non-empty subset of c . For example, if the rule $AB \Rightarrow CD$ holds, then the rules $ABC \Rightarrow D$ and $ABD \Rightarrow C$ must also hold.

Tras lo cual proponen un segundo método más rápido cuando *minConf* es grande: si detectamos que $\text{conf}(ABC \Rightarrow D) < \text{minConf}$ ya sabemos que no es necesario comprobar las reglas $AB \Rightarrow CD$, $AC \Rightarrow BD$, $BC \Rightarrow AD$, $A \Rightarrow BCD$, $B \Rightarrow ACD$, $C \Rightarrow ABD$.

En nuestro estudio, en que queremos detectar reglas sin altas *confianzas*, esto no es un adelanto si no una comprobación más que ralentizaría la ejecución del algoritmo.

Podemos mejorar notablemente el rendimiento del algoritmo analizando las características de las reglas que genera. El siguiente ejemplo ilustra nuestras observaciones al respecto.

Ejemplo

Sea el k -itemset $l_k = \{1, 2, 3, 4, 5, 6\}$.

1. La ejecución de $genrules(l_k, l_k)$ realizará una llamada recursiva a $genrules(l_k, \{1, 2, 3, 4, 5\})$ y ésta realiza una llamada a $genrules(l_k, \{1, 2, 3, 4\})$, entre otras.
2. Posteriormente, la llamada inicial a $genrules(l_k, l_k)$ realizará una llamada a $genrules(l_k, \{1, 2, 3, 4, 6\})$ y ésta llamará de nuevo a $genrules(l_k, \{1, 2, 3, 4\})$, entre otras.

En consecuencia, se llama a $genrules(l_k, \{1, 2, 3, 4\})$ dos veces, lo que generará duplicados de todas las llamadas hechas con los subconjuntos de $\{1, 2, 3, 4\}$ como segundo parámetro. Esto genera un gran número de búsquedas y cálculos repetidos, exponencial en función de k , que no aportan nada al estudio pues se trata de cálculos ya realizados y convenientemente almacenados.

Esta característica hace aún más ineficiente el algoritmo original si recordamos el modo en que se almacena el repositorio de *itemsets* frecuentes, \mathcal{L} , ya que para buscar el *soporte* de un k -itemset hemos de recorrer \mathcal{L} desde su raíz buscando cada uno de los ítems que forman l_k . Si hacer una búsqueda ya es costoso por tener que hacer k búsquedas individuales, repetirla cuando ya se ha hecho es un desperdicio de recursos.

El listado 2.14 presenta nuestra propuesta, una modificación de la función $genrules()$ en la que hemos respetado la numeración del pseudocódigo original (véase el listado 1.7) para que sea más fácil identificar los cambios que introdujimos.

El primer cambio consiste en añadir como parámetro de $genrules()$ el *soporte* de l_k . Con ello evitamos numerosas llamadas a la función $soporte()$ aprovechando el momento en que se ejecuta $genrules()$. No se hace ninguna llamada a $soporte(l_k)$ pues al procesar cada k -itemset de \mathcal{L} ya lo tenemos localizado y podemos consultar directamente su *soporte* en lugar de buscarlo, ahorrando las múltiples llamadas recursivas que hace el algoritmo original.

Listado 2.14: Función *genrules()* modificada

```

forall large itemsets  $l_k, k \geq 2$  do
     $supp\_l\_k = support(l_k)$ ;
    call genrules( $l_k, l_k, supp\_l\_k$ );

// The genrules generates all valid rules  $\tilde{a} \Rightarrow (l_k - \tilde{a})$ ,
// for all  $\tilde{a} \subset a_m$ 
procedure genrules( $l_k$ : large  $k$ -itemset,
                     $a_m$ : large  $m$ -itemset,
                     $supp\_l\_k$ : double)
x)  static a_m_processed; // set containing the sets of  $l_k$  processed
    // in previous calls to genrules
x)  if ( $m == k$ )
x)  a_m_processed.clear(); // Initialize a_m_processed for each new  $l_k$ 
1)   $A = \{(m-1)\text{-itemsets } a_{m-1} \mid a_{m-1} \subset a_m\}$ ;
2)  forall  $a_{m-1} \in A$  do begin
x)    if  $a_{m-1} \in a\_m\_processed$  then
x)    continue;
x)    a_m_processed.add( $a_{m-1}$ );
3)     $conf = supp\_l\_k / support(a_{m-1})$ ;
4)    if ( $conf \geq minconf$ ) then begin
7)        output the rule  $a_{m-1} \Rightarrow (l_k - a_{m-1})$ ,
            with confidence =  $conf$ 
            and support =  $support(l_k)$ ;
8)    if ( $m - 1 > 1$ ) then
9)        call genrules( $l_k, a_{m-1}, supp\_l\_k$ ); // to generate rules with
                                                // subsets of  $a_{m-1}$  as the
                                                // antecedents

10) end
11) end

```

El cambio más relevante es el uso de la variable estática *a_m_processed* y su inicialización cada vez que entramos en un nivel nuevo de \mathcal{L} . En el bucle sólo hay que comprobar si el *itemset* ya ha sido utilizado, si lo ha sido lo ignoramos y trabajamos con el siguiente *itemset*, si no lo ha sido se anota como utilizado y se sigue el flujo normal del algoritmo original. Se utilizan pocos recursos para guardar este vector en memoria ya que estamos centrados en un único k -*itemset* y *a_m_processed* contendrá únicamente sus subconjuntos.

Para probar la eficiencia de nuestra propuesta frente a la propuesta original aplicamos ambos algoritmos con diferentes *soportes* y *confianzas* mínimos sobre los almacenes de transacciones *foodmart* y *T40I10D100K* contando el número de reglas que se analizan con cada uno de los métodos y el número de reglas que contiene cada almacén \mathcal{D} en esas condiciones.

Las tablas 2.1 y 2.2 muestran los resultados obtenidos en la ejecución de

ambos algoritmos. En las columnas bajo el epígrafe «analizadas» se indica el número de reglas analizadas con cada método y entre paréntesis la relación entre este valor y las reglas finalmente encontradas. También se muestra el tiempo de ejecución en cada una de las situaciones tratadas para constatar que la reducción de cálculos influye en el tiempo necesario para llevarlas a cabo.

fichero	<i>minSup</i>	<i>minConf</i>	reglas		encontradas
			analizadas	original	
foodmart	0.005 %	90 %	905 498	757 085	137 917
(Cálculos)			(656.6 %)	(548.9 %)	
Tiempo			3.6 sg	3.1 sg	
		75 %	932 697	768 962	149 711
			(623.0 %)	(513.6 %)	
			3.8 sg	3.2 sg	
		50 %	1 304 152	865 132	290 441
			(449.0 %)	(297.9 %)	
			5.7 sg	4.5 sg	
		25 %	1 437 624	870 460	338 221
			(425.1 %)	(257.4 %)	
			6.3 sg	4.8 sg	
		5 %	1 442 456	870 464	340 375
			(423.8 %)	(255.7 %)	
			6.3 sg	4.8 sg	
		1 %	1 442 456	870 464	868 427
			(166.1 %)	(100.2 %)	
			10.8 sg	6.9 sg	
		0 %	1 442 456	870 464	870 464
			(165.7 %)	(100 %)	
			10.7 sg	7.0 sg	

Cuadro 2.1: Reglas analizadas y encontradas (foodmart)

La figura 2.4 ilustra los resultados obtenidos al analizar el almacén foodmart. Es fácil observar en estas gráficas que la reducción de cálculos realizados y de tiempo es mayor cuanto menor es la *confianza* mínima utilizada.

La primera observación a destacar de estos resultados es que la eficiencia de nuestro método depende de su implementación. Cuando no se impone un umbral de *confianza* (*minConf* = 0 %) estamos obligando al algoritmo a mostrar todas las reglas existentes. En estas condiciones nuestro método analiza únicamente las reglas que contiene el almacén \mathcal{D} mientras que el método original analiza un 65.7 % más de las reglas encontradas en el análisis de foodmart y hasta un 13 016.0 % más en el peor de los casos estudiados, al analizar T40I10D100K con *soporte* mínimo del 0.5 %.

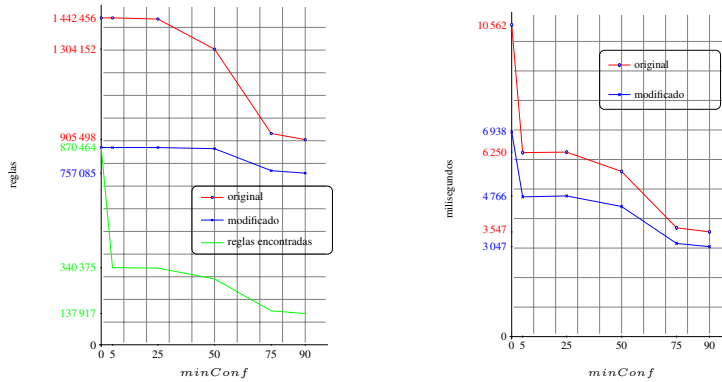


Figura 2.4: genrules() original vs. nuestra modificación (foodmart, $\text{minSup} = 0,005\%$)

fichero	minSup	minConf	reglas		encontradas
			analizadas		
			original	modificado	
T40I10D100K	5 %	50 %	30	30	0
(Cálculos)			()	()	
Tiempo			0 msg	0 msg	
	1 %	50 %	5 193 340 (1 789.1 %)	398 818 (137.4 %)	290 273
		0 %	30.3 sg 5 912 979 (1 474.2 %)	4.1 sg 401 096 (100 %)	401 096
	0.5 %	50 %	30.3 sg 606 619 662 (12 309.9 %)	4.1 sg 6 302 454 (127.9 %)	4 927 893
		0 %	4 772.7 sg 846 495 481 (13 116.0 %)	91.2 sg 6 453 924 (100 %)	6 453 924
			14 266.6 sg	106.1 sg	

Cuadro 2.2: Reglas analizadas y encontradas (T40I10D100K)

Al incrementar la *confianza* mínima se analizan más reglas de las que finalmente serán aceptadas ya que muchas de ellas no superarán el umbral fijado. A pesar de que se acercan los valores obtenidos usando ambos algoritmos siempre se produce un gran ahorro al utilizar nuestra propuesta.

Todo esto también se refleja en el tiempo de ejecución, notablemente inferior en todas las pruebas realizadas para el algoritmo modificado, hasta el extremo de tardar menos de 2 minutos en una ejecución en que el algoritmo original empleó

más de 237 minutos (T40I10D100K con $minSup = 0,5\%$ y $minConf = 0\%$). Si aplicamos nuestra propuesta a las transacciones realizadas por un único usuario serán aún menores los tiempos, consiguiendo resultados en tiempo real para alimentar un *Sistema de Recomendación Web*.

2.3.2. Apriori2

Otra de nuestras propuestas consistió en crear un fichero similar a \mathcal{D} en el que sustituimos las *transacciones* originales por el conjunto de *reglas de asociación* que satisface cada una de ellas. La idea original era aplicar *Apriori* a este nuevo conjunto, \mathcal{R} , para analizar a través de las reglas que cumple cada transacción «cómo» se comportan los usuarios que las generan en lugar de estudiar «qué» ítems relacionan los usuarios en una transacción.

Reescribiendo \mathcal{D} para obtener \mathcal{R}

Para estudiar el comportamiento de un usuario es preferible analizar qué reglas «verifica» en vez de centrar el análisis en los ítems que forman dichas regla. Supongamos que un usuario visita los lunes un *portal web* solicitando las páginas A , B y C y un conjunto de páginas P_1 . Los viernes este usuario visita las páginas A , G y H y un conjunto de páginas P_2 . Supongamos también que las páginas de P_1 y P_2 no están relacionadas frecuentemente con la página A y que nuestro sistema de recomendación no considera variables temporales por lo que no puede determinar si es lunes o viernes. Cuando el usuario entra en el *portal web* y solicita la página A es muy probable que le recomendemos visitar la página B o G si estamos usando los métodos tradicionales de recomendación. Si el usuario visita a continuación G el sistema recomendará la página H , a continuación alguna de las páginas de P_2 y por último la página B con menor probabilidad. De cualquier modo el hecho de que visite la página A no es tan significativo como la posibilidad de usar las reglas de comportamiento ya guardadas en nuestro sistema (las reglas que contienen la relación entre los k -itemsets AGH y P_2). Nuestro objetivo es aprovechar la capacidad que tenemos de analizar las reglas que se verifican en nuestro *portal web* de forma independiente al número de páginas que contienen, proporcionar un método para medir la relación existente entre las reglas de comportamiento de nuestro *portal web* y usando únicamente las transacciones proporcionadas por los usuarios.

Si estudiamos las reglas verificadas por cada transacción del repositorio original \mathcal{D} , podemos definir el conjunto de reglas \mathcal{R} que contiene una línea por cada

transacción de \mathcal{D} conteniendo las reglas que verifica la transacción original. El objetivo de esta conversión, $\mathcal{D} \hookrightarrow \mathcal{R}$, es analizar en un segundo estudio las reglas que contiene \mathcal{R} utilizando el algoritmo *Apriori*.

Inicialmente esta tarea es simplemente exploratoria y se necesitan muchas horas para convertir el repositorio original \mathcal{D} en grandes repositorios \mathcal{R} con diferentes umbrales de *soporte* y *confianza*. Los repositorios de reglas tienden a ser mucho más grandes que los de transacciones. Si una transacción verifica una regla que contiene un determinado *k-itemset* l_k también verificará todas las reglas de los *itemsets* contenidos en l_k . Por ejemplo, si la transacción $ABCD$ genera una regla también generará las cincuenta reglas que se deducen de sus subconjuntos:

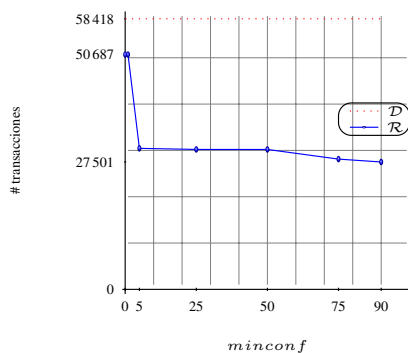
- 14 reglas de 4 ítems: $ABC \rightarrow D, ABD \rightarrow C, \dots D \rightarrow ABC$
- 24 reglas de 3 ítems: $AB \rightarrow C, AB \rightarrow D, \dots D \rightarrow BC$
- 12 reglas de 2 ítems: $A \rightarrow B, A \rightarrow C \dots D \rightarrow C$

Si usamos un umbral de confianza bajo, una transacción de \mathcal{D} con dos ítems generará una línea con 2 reglas en \mathcal{R} , una transacción con 3 ítems generará 12 reglas, una transacción con 4 ítems generará 50 reglas. \mathcal{R} crecerá exponencialmente en función de la longitud de las transacciones de \mathcal{D} .

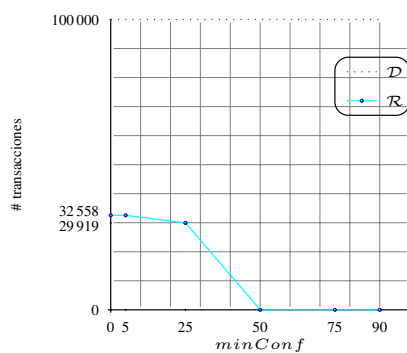
El desmesurado tamaño de los repositorios \mathcal{R} obtenidos nos condujo a hacer un análisis exploratorio de las diferencias existentes entre \mathcal{R} y \mathcal{D} con el fin de reducir los datos que forman \mathcal{R} sin perder la información que contiene.

Análisis descriptivo de las diferencias existentes entre \mathcal{D} y \mathcal{R}

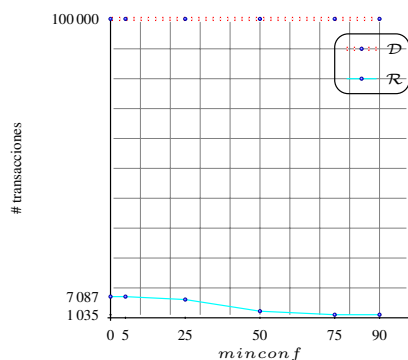
En la sección TAL se expuso una modificación del algoritmo de generación de reglas con la que se puede trabajar con umbrales de confianza muy pequeños. La razón por la que buscamos esta reducción, que en algunos casos consistía en la eliminación del umbral, se entiende mejor al revisar los resultados obtenidos, que se muestran en la figura 2.5 y el cuadro 2.3. Conforme se reduce el valor de *confianza* mínima se pierde información de las transacciones en estudio. Por ejemplo, al usar un *soporte* mínimo del 5 % en el repositorio T40I10D100K las reglas obtenidas se derivan únicamente de un 32.6 % de las transacciones del repositorio, de lo que se deduce que más del 67 % de las transacciones originales son completamente ignoradas por el análisis.



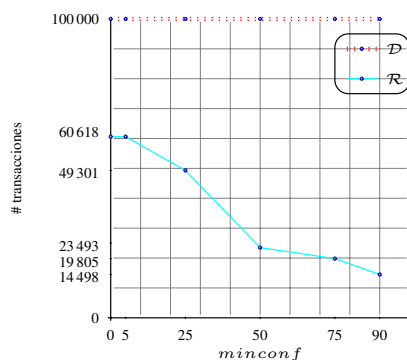
(a) foodmart 0.005 %



(b) T40I10D100K 5 %



(c) T10I4D100K 1 %



(d) T10I4D100K 0.5 %

Figura 2.5: \mathcal{D} vs \mathcal{R} Cuadro 2.3: \mathcal{D} vs \mathcal{R}

fichero	$minSup$	$minConf$	transacciones		
			\mathcal{D}	\mathcal{R}	\mathcal{D} útil
foodmart	0.005 %	90 %	58 418	27 501	47.1 %
		75 %		28 125	48.1 %
		50 %		30 201	51.7 %
		25 %		30 212	51.7 %
		5 %		30 440	52.1 %
		1 %		50 687	86.8 %
		0 %		50 687	86.8 %
T40I10D100K	5 %	50 %	100 000	0	0 %
		25 %		29 919	29.9 %
		5 %		32 558	32.6 %
		0 %		32 558	32.6 %

Continúa en la página siguiente...

Cuadro 2.3 – ... continuación

fichero	$minSup$	$minConf$	transacciones		\mathcal{D} útil
			\mathcal{D}	\mathcal{R}	
	1 %	50 %		92 227	92.2 %
		25 %		99 956	100.0 %
		5 %		99 956	100.0 %
		1 %		99 956	100.0 %
		0 %		99 956	100.0 %
	0.5 %	50 %		99 835	99.8 %
T10I4D100K	5 %	50 %	100 000	0	0 %
		25 %		15 475	15.5 %
		5 %		32 558	32.6 %
	1 %	50 %		2 211	2.2 %
		25 %		6 109	6.1 %
		5 %		7 087	7.1 %
		1 %		7 087	7.1 %
		0 %		7 087	7.1 %
	0.5 %	50 %		23 493	23.5 %
		25 %		49 301	49.3 %
		5 %		60 618	60.6 %
		1 %		60 618	60.6 %
		0 %		60 618	60.6 %

Existen muchas aproximaciones en el estado del arte que buscan la reducción de las dimensiones del problema a tratar utilizando técnicas estadísticas basadas en la frecuencia de los ítems, sin embargo no se tienen en cuenta ciertos detalles, como el hecho de que los resultados son sólo significativos para un reducido grupo de individuos de la población en estudio.

En este trabajo usamos valores para el *soporte* y *confianza* mínimos que puedan ser utilizados en ordenadores personales convencionales sin generar problemas de falta de recursos, de modo que podamos atender al mayor número de individuos de la población en estudio. El tamaño de los repositorios \mathcal{R} obtenidos con estos parámetros es tan grande que no podemos analizarlos directamente por lo que proponemos un nuevo algoritmo para dividir \mathcal{R} en diferentes repositorios que puedan ser analizados convenientemente.

División automática del repositorio de reglas

Existen muchas investigaciones proponiendo métodos para dividir los repositorios originales de transacciones, pero todos ellos se basan en un alto control y *clasificación* del conjunto de ítems disponibles en el sistema. Nuestra propuesta no usa esta información adicional porque se obtiene de una gestión del sistema y no de del propio uso del mismo. Definimos un conjunto de familias de reglas para

dividir el repositorio \mathcal{R} y agrupar aquellas reglas que los usuarios relacionan entre sí cuando interactúan con el sistema. A continuación se describe el algoritmo que proponemos.

1. Seleccionar la regla con mayor *soporte*, en caso de empate seleccionar la regla con mayor confianza y en caso de empate seleccionar la regla que contenga más ítems.

$$R_1 = \max_i \{ \text{soporte}(R_i) \} \wedge$$

$$(\text{confianza}(R_1) = \max_j \{ \text{confianza}(R_j) \mid \text{soporte}(R_j) = \text{soporte}(R_1) \})$$

Esta regla será el elemento principal de la primera familia de reglas, \mathcal{F}_1 .

2. Dividir \mathcal{R} en subconjuntos de reglas. El primer repositorio, \mathcal{R}_1 , contendrá todas las líneas de \mathcal{R} que tengan la regla R_1 y el segundo repositorio, \mathcal{R}_∞ , contendrá el resto de líneas de \mathcal{R} .
3. Ejecutar el algoritmo *Apriori* sobre \mathcal{R}_1 y aplicar de nuevo el paso 1 para seleccionar R_2 , la regla de \mathcal{R}_1 que aparece con mayor frecuencia junto a R_1 .
4. Comprobar el *soporte* de R_2 en \mathcal{R}_∞ : si el *soporte* de R_2 en \mathcal{R}_1 es mayor que su *soporte* en \mathcal{R}_∞ se añade R_2 a la familia \mathcal{F}_1 ; en otro caso se elimina R_2 de \mathcal{R}_1 .
5. Volver al paso (3) mientras queden reglas no clasificadas en \mathcal{R}_1 .

Cuando se ha completado la definición de la primera familia de reglas se eliminan todas las reglas pertenecientes a \mathcal{F}_1 del repositorio original de reglas, \mathcal{R} , y las transacciones que queden vacías. A continuación se utiliza el mismo procedimiento para generar \mathcal{F}_2 , la segunda familia de reglas. El algoritmo finaliza cuando \mathcal{R}_∞ no contiene reglas asociadas, es decir, cuando todas las transacciones de \mathcal{R}_∞ contienen únicamente una regla.

Propuesta de sistema de recomendación

Hemos definido un *Sistema de Recomendación* en el que aplicar la modificación en dos pasos del algoritmo *Apriori*. El objetivo principal es obtener recomendaciones personalizadas para los usuarios de nuestro portal de educación, generando recomendaciones en tiempo real en forma de enlaces. El proceso se define a continuación:

1. Un usuario entra en el sistema y selecciona el ítem A .
2. El sistema sólo puede usar la información almacenada sobre los ítems directamente relacionados con A , es decir, las reglas originales cuyo antecedente es A – esta es la recomendación clásica basada en *Reglas de Asociación*. La primera recomendación se basa únicamente en frecuencias (el sistema recomendará los consecuentes de las reglas con mayor *confianza* que tengan al ítem A como antecedente).
3. El usuario selecciona un segundo ítem, B .
4. A partir de ahora podemos usar la nueva información que hemos recogido sobre el comportamiento de la población de usuarios, las reglas que verifican en su visita a nuestro *portal web*. Obtenemos las reglas derivadas del k -itemset l_k y buscamos la familia a la que pertenecen, \mathcal{F}_i .
 - Si \mathcal{F}_i contiene reglas derivadas de las reglas verificadas por el usuario en esta visita, entonces hay una *confianza* del 100 % entre las reglas descubiertas y las reglas verificadas por el usuario. En este caso el sistema recomendará los ítems de las reglas descubiertas con mayor *soprote*. A diferencia de los métodos clásicos, esta propuesta descubre reglas que no tienen los ítems visitados por el usuario como antecedentes; esto es importante ya que el análisis realizado ignora por completo el orden de selección de los ítems.
 - Si \mathcal{F}_i no contiene reglas derivadas se usarán las reglas con mayor *confianza* (y mayor *soprote* en caso de empate) en relación con las reglas verificadas por el usuario. Adicionalmente, para resolver el problema de encontrar recomendaciones basadas en el antecedente, es posible encontrar reglas con cualquier ítem seleccionado por el usuario. Esto no puede llevarse a cabo con el método clásico.
5. El usuario selecciona un tercer ítem, C .
6. Repetimos el tercer paso buscando una o más familias que contengan las reglas derivadas del k -itemset $l_k = \{A, B, C\}$.

Esta aproximación conduce a una nueva experiencia con nuestro sistema educacional. Los usuarios se pueden sentir más confortables en su interacción con el sistema ya que pueden llevar a cabo sus tareas más rápidamente cuando utilizan los enlaces recomendados por el sistema.

2.4. El Ítem Raro

ABIERTO

Desde la irrupción de la informática en pequeñas y grandes empresas se guarda en *Bases de Datos* todo tipo de información, entre otra las *transacciones* que modelan la cesta de la compra o las *sesiones de navegación* de un usuario a través de un *portal web*. El volumen de datos almacenados crece a diario por lo que su análisis inicial se realiza con técnicas como la *Minería de Reglas de Asociación*, capaces de extraer información sobre la coexistencia de ítems en grandes colecciones de datos.

El uso de *soporte* mínimo mediante *Minería de Reglas de Asociación* permite abordar el problema sobre grandes *Bases de Datos* con la tecnología actual pero impide obtener información sobre una gran cantidad de los ítems en estudio, ítems que tiene menor *soporte* del fijado como mínimo para ser estudiados por lo que se denominan *ítems raros*.

El *dilema del ítem raro* ha sido estudiado por muchos investigadores para obtener reglas de asociación que involucren a ítems poco frecuentes, sin embargo todas las propuestas realizadas sobrecargan las tareas del algoritmo usado y de sus analistas y no aportan reglas de uso frecuente. En la sección 2.4.1 expondremos estas propuestas.

En la sección 2.4.2 se introducen las *Reglas de Oportunidad*, que permiten a los algoritmos de búsqueda de reglas de asociación encontrar simultáneamente información de interés sobre los ítems que no superen el *soporte* mínimo con un consumo mínimo de recursos, sin intervención de los analistas y aportando conocimiento útil. Con esta propuesta conseguimos aliviar parte del problema generado por el *dilema del ítem raro*.

2.4.1. Estudio de ítems raros

ABIERTO

Los *ítems raros* son ítems que por alguna razón no aparecen con frecuencia en las *transacciones*. En una cesta de la compra los *ítems raros* suelen ser muy caros o duraderos, y también pueden ser novedades que tardarán en ser comprados de forma frecuente. En un *portal web* los *ítems raros* son páginas muy especializadas, de baja calidad o páginas de novedades.

Los ítems exclusivos (por su alto precio, larga durabilidad o especialización) serán siempre *ítems raros* aunque se incorporen en un *Sistema de Recomendación* ya que son infrecuentes por naturaleza. Sin embargo, los ítems nuevos deberían ser incorporados a los *Sistemas de Recomendación* hasta que alcancen la cate-

ría de ítems frecuentes, momento en el cual ya pueden ser tratados mediante la búsqueda clásica de *regla de asociación*.

J. Han y Fu (1995) proponen dividir \mathcal{D} mediante una jerarquía y buscar las *reglas de asociación* tanto entre ítems como entre las categorías en las que sitúan a cada uno de ellos. Con esta estrategia se puede reducir el número de ítems a vincular, cuando se trabaja con *ítems raros* se puede ver su relación con otros ítems mediante la categoría a la que pertenecen por lo que todos los ítems pueden estar representados en alguna *regla de asociación* y se puede redirigir el análisis para estudiar mejor la relación de dichos *ítems raros* con el resto de ítems de la población. El procedimiento es correcto pero conlleva análisis específicos y la creación y mantenimiento de la jerarquía propuesta, lo que puede provocar una ralentización del análisis y falta de actualidad si aparecen nuevos ítems en la población y no se realiza en paralelo una correcta jerarquización de los mismos.

B. Liu, W. Hsu y Ma (1999) y Palshikar, Kale y Apte (2007) ponen el énfasis en la distribución de los ítems presentes en \mathcal{D} . La investigación precedente sobre *Minería de Reglas de Asociación* considera una distribución uniforme de los ítems de \mathcal{D} , lo que no es correcto en la mayoría de las ocasiones y produce una gran cantidad de reglas de asociación que no aportan información de calidad sobre las relaciones reales existentes entre los ítems de \mathcal{D} . Esa información de baja calidad presenta otro problema: no deja suficientes recursos para analizar otra información de mayor calidad pero menor presencia en la *Base de Datos*. Para resolver esto proponen el uso de *soportes* mínimos múltiples en lugar del clásico *soporte* mínimo único, de modo que los ítems con mayor *soporte* en \mathcal{D} necesiten más evidencias para mostrar relaciones entre sí que los ítems con poca presencia, a los que los algoritmos clásicos simplemente ignoran considerándolos *ítems raros*.

B. Liu, W. Hsu y Ma, 1999 proponen usar múltiples *soportes*, de modo que la relación entre dos ítems frecuentes sea considerada sólo si es una relación muy frecuente en \mathcal{D} . De este modo se alivia considerablemente la carga de memoria requerida por el algoritmo y permite abordar el estudio sobre un número mayor de ítems. La asignación de *soporte* a cada ítem puede hacerse por parte del analista o bien teniendo en cuenta el propio *soporte* de cada ítem en \mathcal{D} . En Kiran y Reddy, 2009 se propone una modificación interesante del algoritmo propuesto por B. Liu, W. Hsu y Ma, 1999, incorporando al estudio medidas de tendencia de los ítems de \mathcal{D} .

Con la primera propuesta de B. Liu, W. Hsu y Ma, 1999 se pueden incorporar los ítems nuevos asignándoles un *soporte* mínimo muy bajo mientras sean

nuevos, sin embargo el analista debe decidir en algún momento cuándo debe modificar su *soporte* mínimo y qué nuevo *soporte* asignarle.

En las restantes propuestas el *soporte* mínimo se obtiene en función del *soporte* real del ítem, con lo que muchos *ítems raros* se incorporan al sistema con facilidad, sin embargo siguen quedando muchos ítems sobre los que no obtenemos información pues si intentamos incorporarlos al estudio se detiene la ejecución del algoritmo por falta de recursos. Además las reglas que proporcionan siempre sugieren una relación entre un *ítem raro* como *antecedente* y uno que no lo es como *consecuente*, lo que en un *Sistema de Recomendación* no es práctico porque se usaría únicamente cuando el usuario solicite un *ítem raro*, lo que ocurre en muy pocas ocasiones dada la naturaleza del ítem solicitado.

candidatos a considerar que haría abortar la ejecución del algoritmo por falta de recursos.

Input: D, sm (*soporte* mínimo) y om (*oportunidad* mínima)

Output: RO (*Reglas de Oportunidad*) y RA (*Reglas de Asociación*) presentes en D

/* Obtener frecuencia de todos los ítems y 2-itemsets de \mathcal{D} */

foreach *transacción* Ti en D

foreach i1 en Ti

Incrementa(FP1[i1]);

foreach (i1; i2) en Ti

Incrementa(FP1[i1]->FP2[i2]);

/* Extraemos las RO */

foreach i1 en FP1

if (FP1[i1] >= sm) then

foreach i2 en FP1[i1]->FP2

if (FP1[i2] < sm y FP2[i2]/FP1[i2] >= om) then

Añadir RO(it1 it2);

Algoritmo 1: ORFind - Algoritmo de búsqueda de Reglas de Oportunidad

Al aplicar el algoritmo clásico tras la ejecución del algoritmo 1 a los repositorios T10I4D100K, T40I10D100K y kosarak hemos obtenido información de interés sobre todos los ítems de cada repositorio en un tiempo inferior al empleado con la implementación de las propuestas de [10] y [11].

Las reglas de asociación obtenidas entre ítems frecuentes tienen el suficiente *soporte* como para ser utilizados como patrones de comportamiento del colectivo estudiado. Las reglas de oportunidad obtenidas vinculan los ítems infrecuentes a los frecuentes, permitiendo a un sistema de recomendación saber cuál es el

Lo que queda de sección no está formateado

momento más oportuno para recomendar un ítem infrecuente de cuyo uso no tenemos aún información suficiente.

2.4.2. Reglas de Oportunidad

Las *Reglas de Oportunidad* son *Reglas de Asociación* interpretadas en sentido inverso. Pueden ser útiles cuando aparece el *dilema del ítem raro* en nuestra colección de datos. El principal motivo de la aparición de este problema son los recursos de memoria utilizados, si tenemos que guardar mucha información sobre muchos datos podemos llegar a desbordar la memoria del sistema y perder todo el trabajo hecho hasta el momento. Hay muchas propuestas para aliviar este problema, todas ellas buscando perder cierta información que por algún motivo se va a considerar irrelevante a cambio de ganar espacio en memoria para poder analizar otros datos de la colección con información más relevante.

ABIERTO

En un Sistema de Recomendación Web se utiliza información sobre taxonomías, contenido semántico y uso del portal para hacer sugerencias a sus usuarios. Sin embargo cuando un ítem es nuevo no puede obtenerse información de su uso hasta que no logra un cierto *soporte*. Las reglas de oportunidad permiten incorporar en el sistema la información de uso de los ítems nuevos de modo automático.

Las RO contienen los ítems frecuentes que son visitados conjuntamente con los ítems sin *soporte* mínimo. Esto nos permite sugerir un *ítem raro* a los usuarios que visitan alguna de las páginas que más favorecen la presencia de dicho ítem, p.ej. la página A.

Si la sugerencia es acertada es de esperar que el *ítem raro* vaya aumentando su *soporte* y se incorpore de modo natural al proceso de estudio de reglas de asociación, mostrando una asociación cada vez más fuerte a la página A.

Si la sugerencia no es acertada el ítem seguirá siendo raro y su asociación con la página A se irá diluyendo a favor de una asociación con otras páginas con las que realmente sea más afín. Esto se debe a que el número de veces que aparece el *ítem raro* en \mathcal{D} es pequeño y cualquier aparición nueva del ítem modificará sustancialmente los porcentajes en que se basan los algoritmos de búsqueda de *Reglas de Asociación* y *Reglas de Oportunidad*.

Referencias

[4] Tseng, M.-C., Lin, W.-Y. Efficient mining of generalized association rules with nonuniform minimum support. *Data & Knowledge Engineering* 62 (1), pp. 41-64. 2007

[6] Kouris, I.N., Makris, C.H., Tsakalidis, A.K. Using information retrieval


techniques for supporting data mining. *Data & Knowledge Engineering* 52 (3), pp. 353-383. 2005

2.5. Publicaciones

El trabajo expuesto en la sección ?? se presentó en Beijing (HCII'07¹²). La investigación mostrada en la sección 2.3 dio lugar a la publicación de un artículo en la revista *ESWA*¹³, lo que posibilitó mucho más la difusión de nuestro trabajo. Por último presentamos en Barcelona (Interacción'09¹⁴) el trabajo expuesto en la sección 2.4.

Selecting the Best Tailored Algorithm for Personalizing a Web Site, 2007

En este artículo presentamos los resultados obtenidos al comparar diferentes implementaciones del algoritmo *Apriori*. Para seguir en la línea que estamos investigando hemos de crear nuestra propia implementación si queremos introducir los resultados obtenidos hasta la fecha y los que puedan ir surgiendo en el curso de la investigación. El código de las implementaciones puestas a prueba nos resultará útil para trabajar en nuestra propia implementación.

Botella Beviá, F., Lazcorreta Puigmartí, E., Fernández-Caballero, A., González López, P., Gallud, J.A. y Bia Platas, A. Selecting the Best Tailored Algorithm for Personalizing a Web Site. *Proceedings of the 12th International Conference on Human-Computer Interaction (HCII)*, Beijing (China), 2007. 

Resumen

Automatic personalization for dynamic web systems has been carried out by several methods and techniques, like data mining or Web usage mining. The *Apriori* algorithm is one of the most used for web personalization. We have designed a web recommender system based on this algorithm to select the best user-tailored links. We needed an efficient algorithm to preserve updated our system in

¹² http://www.hci.international/index.php?module=conference&CF_op=view&CF_id=5




¹³ <http://www.journals.elsevier.com/expert-systems-with-applications/>

¹⁴ <http://interaccion2009.aipo.es>

real time. We tested several implementations of *Apriori* algorithm but none of them looked to run properly with our data. We tested these implementations with different access log files of public domain (BMSWebView.dat, BMS-POS) and with our log data. We decided to develop a new implementation that better adapts to our data.

Towards personalized recommendation by two-step modified *Apriori* data mining algorithm, 2008

El artículo

Lazcorreta Puigmartí, E., Botella Beviá, F. y Fernández-Caballero, A. Towards personalized recommendation by two-step modified *Apriori* data mining algorithm. *Expert Systems with Applications*, 35(3):1422–1429, 2008.  ¹⁵ ¹⁶

Resumen

In this paper a new method towards automatic personalized recommendation based on the behavior of a single user in accordance with all other users in web-based information systems is introduced. The proposal applies a modified version of the well-known *Apriori* data mining algorithm to the log files of a web site (primarily, an e-commerce or an e-learning site) to help the users to the selection of the best user-tailored links. The paper mainly analyzes the process of discovering association rules in this kind of big repositories and of transforming them into user-adapted recommendations by the two-step modified *Apriori* technique, which may be described as follows. A first pass of the modified *Apriori* algorithm verifies the existence of association rules in order to obtain a new repository of transactions that reflect the observed rules. A second pass of the proposed *Apriori* mechanism aims in discovering the rules that are really inter-associated. This way the behavior of a user is not determined by "what he does" but by "how he does". Furthermore, an efficient


¹⁵ http://cio.umh.es/files/2011/12/CIO_2007_26.pdf

¹⁶ https://www.researchgate.net/publication/222690209_Towards_personalized_recommendation_by_two-step_modified_Apriori_data_mining_algorithm?ev=pub_cit

implementation has been performed to obtain results in real-time. As soon as a user closes his session in the web system, all data are recalculated to take the recent interaction into account for the next recommendations. Early results have shown that it is possible to run this model in web sites of medium size.

Recomendaciones en sistemas web mediante el estudio de *ítems raros* en transacciones, 2010

En Interacción'10 presentamos las *Reglas de Oportunidad* a través del siguiente artículo:

Lazcorreta Puigmartí, E., Botella Beviá, F. y Fernández-Caballero, A. Recomendaciones en sistemas web mediante el estudio de ítems raros en transacciones. *Actas del XI Congreso Internacional de Interacción Persona-Ordenador*, 385–388, 2010. 

Resumen

Las recomendaciones en *portales web* se enriquecen cuando incorporan información sobre el uso real del portal. Una de las herramientas que proporcionan dicha información es el *ARM* (ARM) en las sesiones de navegación de los usuarios a través del portal. Si el número de páginas del portal es muy grande la *ARM* se tropieza con el *dilema del ítem raro*, que postula que no se puede encontrar información sobre las páginas poco visitadas sin obtener una explosión de *reglas de asociación*. Este dilema ha sido estudiado desde una perspectiva que sobrecarga las tareas del algoritmo usado y de sus analistas y no aporta reglas de uso frecuente. En este artículo se introduce un nuevo enfoque al dilema que permite a los algoritmos de *ARM* encontrar simultáneamente información de interés sobre los páginas del portal poco visitadas, con un consumo mínimo de recursos, sin intervención de los analistas y aportando conocimiento útil.

ARM para Clasificación

Uno de los problemas de la búsqueda de *reglas de asociación* mediante equipos informáticos es su necesidad de memoria RAM para guardar todos los *item-sets* frecuentes (\mathcal{L}) en un lugar de rápido acceso para ser más eficientes en la búsqueda de *reglas de asociación*. El *dilema del ítem raro* puede aparecer en cualquier momento, dependiendo de los datos que contenga el almacén \mathcal{D} que estemos analizando, y se siguen proponiendo ideas para aliviarlo como el uso de umbrales automáticos para el *soporte* propuesto por Sadhasivam y Angamut-hu (2011). Sería interesante poder averiguar, en base a información básica del fichero \mathcal{D} , qué requisitos de memoria RAM necesitamos, de cuáles disponemos y, con ello, deducir hasta qué *soporte* mínimo podemos trabajar con esa colección específica de datos en este equipo en particular. Hay estudios sobre ello [cita sobre mushroom.dat de gAcademy]. Aunque este proceso puede llevar algo de tiempo y restar eficiencia global al algoritmo de *FIM* permite ahorrar el tiempo perdido cuando el programa deja de funcionar por falta de RAM y no es capaz de ofrecernos ningún resultado.

ABIERTO

Esta reflexión tiene sentido si pensamos en grandes *datasets* pero nos llamó la atención en ficheros tan «pequeños» como chess.dat o mushroom.dat. mushroom.dat es una colección de datos ampliamente utilizada debido a su publicación en UCI - Machine Learning Repository¹, donde podemos encontrar una completa descripción de los datos en sí y de su aparición en el mundo de la investigación. Encontramos información sobre el uso de este almacén \mathcal{D} en artículos

¹<https://archive.ics.uci.edu/ml/datasets/Mushroom>

de *clasificación*.

La Minería de Reglas de *Clasificación* (CRM) toma pronto las bondades de la Minería de Reglas de Asociación (ARM) derivando en una nueva línea de investigación en auge denominada Minería de Reglas de Clasificación Asociativa (CARM) (B. Liu, W. Hsu y Ma, 1998; Bayardo, 1998; Y. Wang, Xin y Coenen, 2008) donde se aprovecha el potencial que tienen las *reglas de asociación* para descubrir clasificadores precisos.

Añadir más
citas

Estos ficheros han sido analizados en muchos artículos desde el punto de vista de la *Minería de Reglas de Asociación* clásica (Suzuki, 2004; Borgelt, 2004; Thabtah, Cowling y Hamoud, 2006; Y. Wang, Xin y Coenen, 2008; Li y N. Zhang, 2010; Malik y Raheja, 2013; Ritu y Arora, 2014; Sahoo, Das y Goswami, 2014). Se proponen nuevas medidas, como la *utilidad* de los *itemsets* (Wu y col., 2012) para aliviar el *dilema del ítem raro*. En todos los casos se ha de recurrir al *soporte* mínimo para poder llevar a cabo el análisis en un tiempo prudencial (algo más de 2 segundos en el caso de Ritu y Arora, un estudio muy reciente cuyos gráficos coinciden con los de Malik y Raheja). Intentamos aplicar técnicas de *Minería de Datos* a colecciones relativamente pequeñas (mushroom.dat sólo contiene $8\,124 \times 23$ datos) y no podemos profundizar o trabajar en tiempo real si no aplicamos recortes drásticos de información. Hoy en día, almacenes \mathcal{D} de este tamaño deberían poder analizarse en tiempos razonables sin prescindir de ninguno de sus datos.

La Minería de Reglas de Clasificación Asociativa (CARM) transforma los datasets que han de ser analizados desde la perspectiva de CRM para que puedan ser vistos como conjuntos de transacciones y hacer un análisis basado en ARM. Para ello convierten los valores utilizados en CRM, formado por pares atributo=valor, en atributos binarios que tendrán el significado «Si aparece el atributo binario X entonces el registro toma el valor Y en el atributo original \mathcal{A}_i ». Ha habido grandes avances en *Clasificación* gracias a esta pequeña transformación, sin embargo también se ha perdido información muy importante sobre el dataset en estudio, que no será aprovechada si no la incorporamos a la colección de transacciones que queremos analizar con ARM.

Mejorar

Antes de obtener información sobre mushroom.dat comenzamos a analizar los resultados obtenidos en nuestros propios experimentos sobre este almacén \mathcal{D} . Al observarlos encontramos una estructura concreta: todas las filas («*transacciones*») tienen el mismo número de datos y en la primera columna sólo aparece un 1 o un 2, en la segunda sólo un 3, 4 o 5... Se trata de una estructura muy rígida y con exceso de información. Al descubrir que el 1 aparecía en $X\,XXX\,transaccio-$

nes dedujimos que el 2 aparecería en las restantes $8124 - XXXX$. En el proceso de *Minería de Itemsets Frecuentes* estábamos desperdiciando esta información y nos entreteníamos en contar el número de veces que aparecía el ítem 2. También descubrimos que el ítem 3 aparecía en un total de $YYYY$ transacciones, en $ZZZZ$ ocasiones junto al ítem 1, luego en las restantes $YYYY - ZZZZ$ veces que aparece ha de estar junto al ítem 2. No necesitamos averiguar nada sobre el ítem 2, lo que averigüemos sobre el ítem 1 nos dará toda la información que tiene \mathcal{D} sobre el ítem 2. No necesitamos utilizar memoria RAM para guardar información sobre el ítem 2, tendremos memoria RAM libre para analizar los *ítems raros* más frecuentes de \mathcal{D} .

Tras llevar a cabo la investigación que se expone en este capítulo podemos afirmar que, aunque todas las citas usadas en los párrafos anteriores han experimentado con `mushroom.dat` para aliviar su *dilema del ítem raro*, `mushroom.dat` y muchas colecciones de datos de similares características realmente no contienen *Ítems Raros*.

Lichman, 2013 ponen a disposición de investigadores una buena colección de datos. Con ellos podemos probar nuestras propuestas y, lo que es más importante, hacer comparaciones con otros trabajos ya publicados, en situaciones similares y con los mismos datos. Aunque esto no siempre es así, muchos de los experimentos relatados en la amplia bibliografía de esta tesis han usado aparentemente los mismos datos pero al profundizar en ellos observamos pequeñas diferencias que pueden alterar los resultados de las comparaciones llevadas a cabo. En la sección 3.5 mostraremos un caso en el que los autores modifican los datos originales, lo que nos llevó a experimentar con dos datasets distintos diseñados para el mismo problema de *clasificación* y nos permitió llegar a conclusiones interesantes que de otro modo quizá no habríamos descubierto.

3.1. Transacciones Estructuradas

Desde sus inicios la *ARM* se ha protegido del *dilema del ítem raro* mediante la definición de un *soporte* mínimo que impide estudiar los ítems de menor soporte (*ítems raros*) para evitar problemas de desbordamiento de memoria y poder ofrecer resultados en poco tiempo. En la evolución del *ARM* hay múltiples aportaciones que alivian este problema pero siempre a costa de renunciar al estudio de algunas relaciones existentes entre los datos en estudio.

En los primeros trabajos sobre *ARM* se trataba de evitar la búsqueda de re-

laciones entre los *ítems raros* (Agrawal, Imielinski y A. Swami, 1993b; Agrawal y Srikant, 1994a; Park, M. Chen y P. Yu, 1997). En B. Liu, W. Hsu y Ma, 1999, proponen el uso de múltiples soportes mínimos para que un *itemset* sea considerado frecuente sólo si su *soporte* es grande en relación al *soporte* de los ítems que lo forman. De este modo se ahorra espacio en memoria que puede ser utilizado para analizar ítems con menor *soporte*. El uso de un *soporte* relativo para cada *itemset* basado en la *confianza* de las *reglas de asociación* que generan sus ítems, lo que permite guardar información de los *itemsets* con menor *soporte* que el *soporte* mínimo que superen su *soporte* relativo (Yun y col., 2003), garantizando que las reglas que generan sean de calidad. En Hu e Y.-L. Chen, 2006, se propone una mejora del algoritmo propuesto en B. Liu, W. Hsu y Ma, 1999, para dotarlo de escalabilidad. En Tseng y W.-Y. Lin, 2007, trabajan con las mismas ideas de *soporte* múltiple pero reducen el número de *itemsets* a guardar en función de su *lift*, que mide la mejora que produce la presencia de un ítem en el soporte del resto de ítems que contiene. En Kiran y Reddy, 2009, utilizan múltiples *soportes* basados en la distribución conjunta de los ítems y no sólo en su distribución individual como proponían en B. Liu, W. Hsu y Ma, 1999. Todas estas aportaciones reducen el número de *itemsets* almacenados en memoria, con lo que pueden almacenarse otros *itemsets* menos frecuentes pero con mejor información sobre la población en estudio.

Cuando trabajamos con un número grande de ítems diferentes y una gran cantidad de *transacciones* el *dilema del ítem raro* sólo es resoluble utilizando mucho tiempo y/o potentes computadores. Lo que sorprende es que este dilema aparezca en colecciones de datos con dimensiones reducidas, como ocurre con muchos repositorios utilizados en artículos sobre el problema de *Clasificación* que incorporan técnicas de *Minería de Reglas de Asociación*.

Es el caso de chess.dat y mushroom.dat. chess recoge información sobre la posición final de una serie de piezas en el juego del ajedrez, contiene tan solo 75 ítems distintos en un total de 3 196 transacciones. mushroom recoge el valor de una serie de atributos medidos en ciertas setas, utilizando 119 ítems distintos en 8 124 transacciones. Todos los investigadores que han trabajado con estas colecciones han tratado sus transacciones del mismo modo que la clásica *transacción* formada por la «cesta de la compra». En esta sección mostramos que se puede dar un tratamiento diferente sin perder las ventajas del ARM ni aplicar nuevos algoritmos.

3.1.1. Tipos de *transacciones*

La definición de *transacción* (véase pg. 49) es muy elemental: «conjunto de ítems observados en ciertas circunstancias». Esta simple definición permite modelar muchas colecciones de datos mediante *transacciones*, desde la clásica «cesta de la compra» hasta el estudio de una *sesión de navegación* web – como si se tratara de un conjunto de páginas web visitadas conjuntamente – pasando por la observación de las características de individuos que tienen enfermedades similares o la detección de fraude en el uso de tarjetas de crédito. En este capítulo estamos trabajando con la observación de ciertas características de los individuos de una población, ejemplarizando en el fichero `mushroom.dat`.

Si suponemos que no es casual el hecho de formar una *transacción* agrupando unos ítems y no otros, y observamos un conjunto grande de *transacciones*, no es difícil pensar que encontraremos en ellas conjuntos de ítems que se repiten, a los que llamamos *itemsets* frecuentes o patrones. Por ejemplo, en la cesta de la compra estos patrones nos pueden sugerir que «el cliente que compra el producto *A* suele comprar conjuntamente el producto *B*». En el análisis de navegación en un portal web podemos encontrar la regla que sugiere que «el usuario que visita la página *A* suele acudir en la misma visita a la página *B*» o incluso que «la página *A* debe tratar sobre el mismo tema que la página *B*», pues observamos que los usuarios las visitan conjuntamente con mucha frecuencia. En el problema de *clasificación* podríamos descubrir que «la mayoría de setas que tienen *estas características* son venenosas».

Es importante destacar que *ARM* suele trabajar sin conocimiento previo de la población en estudio, dejando que sean los datos quienes muestren esa información. Todos los algoritmos de *ARM* buscan extraer el máximo conocimiento de una gran cantidad de *transacciones* en el menor tiempo posible. Al aplicarlos se pueden hacer ajustes para cada estudio concreto, generalmente en base a estudios previos hechos sobre la misma población. Pero al plantearlos se debe procurar que sean aplicables a todas las poblaciones en estudio.

A lo largo de su evolución, *ARM* se ha ido desarrollando en base a conceptos teóricos y a su aplicación en ciertos *datasets*. Muchos desarrollos han resultado poco competitivos al ser aplicados en repositorios concretos, lo que genera una constante duda sobre si existe un algoritmo mejor que otro o si todo depende de los datos sobre los que son aplicados. Es el caso de `chess.dat` y `mushroom.dat`: pese a tener dimensiones reducidas no se suelen utilizar con *soporte* mínimo bajo, pues su análisis mediante *ARM* produce una explosión de reglas que o bien

no pueden ser guardadas en memoria o bien tardan tanto tiempo en obtenerse que se pierde su utilidad en servicios que deben ser ágiles, como los *Sistema de Recomendación*.

En ninguno de estos trabajos se ha tenido en cuenta que realmente estamos analizando dos tipos bien diferenciados de *transacciones*:

- la clásica «cesta de la compra» de la que surge el *ARM*, compuesta por un número indeterminado de ítems de \mathcal{I} y sin ninguna restricción, y
- las *transacciones* derivadas de la transformación de observaciones en el problema de *Clasificación*, compuestas por un número concreto de ítems de \mathcal{I} y con una estructura fija.

Para entenderlo mejor nos referiremos a la colección de datos mushroom.dat. En su diseño se consideraron 22 atributos diferentes (color, textura...) de un total de 8 124 setas diferentes, clasificadas cada una de ellas como «venenosa» o «comestible». Para dar a estos datos forma de *transacción* y poder estudiarlos mediante *ARM* se asigna un código único a cada uno de los pares *atributo*-valor. Así, los códigos 1 y 2 corresponden a la *clasificación* realizada, los códigos 3, 4, 5, 6, 7 y 8 indican el color. Cada seta es observada y se anota mediante una *transacción* su color, etc., de modo que todas las *transacciones* están compuestas por 23 ítems, su clase más el valor de cada uno de los 22 atributos medidos.

Una característica de este tipo de *transacción* que la diferencia de la cesta de la compra es la dependencia estructural entre los ítems de la población en estudio. Si una *transacción* tradicional contiene un ítem determinado no podemos asegurar que no contenga cualquier otro ítem de \mathcal{I} . Sin embargo si una *transacción* formada por códigos *atributo*-valor contiene el código de un *atributo*-valor concreto sabemos con seguridad que no contendrá otro código perteneciente al mismo atributo. En el caso de mushroom.dat, si una *transacción* contiene el código 1 (*venenosa*) no contendrá el código 2 (*comestible*).

Otra característica de este tipo de *transacciones* es la gran densidad de su matriz de correlación. En la práctica nos encontramos con que la dificultad de trabajar con *transacciones* de este tipo se debe a la gran cantidad de relaciones existentes en muchos de los *datasets* utilizados. Cada par *atributo*-valor está relacionado con casi todos los valores del resto de *atributos*, hecho que no ocurre en la clásica cesta de la compra. Es esto lo que provoca la falta de recursos y el exceso de tiempo empleado cuando intentamos analizar a fondo estas colecciones de datos con los algoritmos clásicos de *ARM*, si una *transacción* de este tipo no

tuviera una matriz de correlación densa y tuviera un número reducido de ítems su análisis no presentaría ningún problema.

Este tipo de *transacciones* sirve para modelizar muchos experimentos reales, como hacen Carmona y col., 2012, con las *sesiones de navegación* de los usuarios de un portal de comercio electrónico. Los datos que recogen están previamente diseñados en forma de *atributos* de modo que se observa el navegador usado por el usuario, el tipo de visitante, las palabras clave usadas... Todos los *atributos* se han dividido en un pequeño conjunto de valores representativos y para formar las *transacciones* han anotado el valor concreto de cada atributo en una visita al portal.

A partir de ahora utilizaremos el término *registro* para referirnos a las *transacciones estructuradas*.

Los *registros* pueden presentar problemas en su análisis cuando existe relación entre la mayoría de sus ítems. Sin embargo, antes de proceder a su análisis podemos asegurar que los distintos ítems que forman un *atributo* son, por definición, excluyentes. Esta característica no ha sido utilizada en ninguno de los estudios previos sobre *ARM* que hemos revisado, por lo que proponemos su uso para que no se pierda información contenida en los datos, información que no sabemos aún si es de calidad. Esto nos permitirá abordar problemas que con las propuestas existentes no se pueden resolver por falta de recursos de memoria o bien por tardar mucho más tiempo del aceptado por servicios que han de ser ágiles como los *Sistemas de Recomendación*.

La primera gran diferencia entre una *transacción* sin restricciones y un *registro* aparece en el conjunto \mathcal{I} , que ha sido diseñado para un experimento concreto. El *dataset* en que se recojan los datos obtenidos sobre los individuos de la población en estudio presentará características que no sólo se deben a los individuos estudiados, la elección de los *atributos* a medir y de los valores que puede tomar cada *atributo* son fundamentales y cualquier cambio en ellos producirá un *dataset* diferente en el que podrán existir más o menos relaciones de co-ocurrencia entre los ítems que contiene. Otra diferencia entre los *datasets* creados con *transacciones* y los creados con *registros* se debe a que en los primeros un individuo puede generar más de una *transacción* mientras que en los segundos cada individuo tiene asociada un único *registro*, lo que produce colecciones de datos menores de las que deberíamos poder extraer más información. Por último los *registros* tienen todos el mismo número de ítems y una restricción fundamental: en cada una de ellos aparece uno y sólo uno de los posibles valores de cada uno de los *atributos* en estudio. Utilizaremos esta restricción para reducir las dimensiones

del problema a resolver y poder utilizar los algoritmos existentes sobre *ARM*.

3.1.2. Descripción de *mushroom.dat*

ABIERTO

3.1.3. Modelización de *Registros*

ABIERTO

Definir *atributos*, *clases* ...

Definición 25 (*Registro*). En un problema de Clasificación con k atributos en estudio, un registro r es un k -itemset, un subconjunto ordenado de k elementos de \mathcal{I} en el que el i -ésimo ítem representa el valor tomado por un individuo en el atributo \mathcal{A}_i .

$$r \subseteq \mathcal{I} = \{I_j \mid I_j \in \mathcal{I}, j = 1 \dots k\} \quad (3.1)$$

Definición 26 (*Registro Clasificado*). En un problema de Clasificación con k atributos en estudio, un Registro Clasificado es un $(k+1)$ -itemset, un subconjunto ordenado de $k+1$ elementos de \mathcal{I} formado por un registro y la clase a la que pertenece.

3.1.4. Análisis eficiente de *Registros*

ABIERTO

Antes de leer un *dataset* de *registros* ya conocemos su estructura y sabemos que cada *registro* contiene exactamente k ítems. El tamaño de las *transacciones* en un problema de *ARM* no suele ser constante por lo que los algoritmos utilizados no pueden usar esta información cuando las *transacciones* son realmente *registros*. Se podrían modificar dichos algoritmos para aprovechar este hecho pero nosotros hemos preferido reducir la información a procesar y utilizar los algoritmos ya conocidos en esta materia.

Lema 3. Un registro no puede contener dos valores representantes del mismo atributo.

El hecho de que cada ítem de un *registro* represente a uno de los *atributos* en estudio implica que si en un *registro* aparece el ítem que representa a un valor concreto de un *atributo* no podrán aparecer el resto de ítems asociados a ese *atributo*. Es por ello que si eliminamos del *dataset* un valor concreto de uno de los *atributos* no perderemos información: si un *registro* no contiene información sobre un *atributo* es porque el valor que toma en ese *atributo* es precisamente aquel que hemos eliminado del *dataset*.

memoria el código que representa a cada ítem y un puntero que una los nodos del árbol.

Aplicando el lema 3 podemos reducir notablemente el tamaño de \mathcal{L} , como muestra el listado 3.2. Como en un *registro* sólo puede aparecer una *clase* y un valor de cada *atributo* no encontraremos nunca los *itemsets* (1,2), (3,4), (3,5), (4,5) y (6,7) por lo que sólo necesitamos 35 nodos para tener la misma información sobre reglas de asociación que contiene la colección de datos que estamos tratando.

Listado 3.2: \mathcal{L} reducido

1 - 3 - 6	2 - 3 - 6	3 - 6
- 7	- 7	- 7
- 4 - 6	- 4 - 6	4 - 6
- 7	- 7	- 7
- 5 - 6	- 5 - 6	5 - 6
- 7	- 7	- 7
- 6	- 6	6
- 7	- 7	7

Por último, si eliminamos un valor de cada *atributo*, aplicando el lema 4 vemos que toda la información que se puede guardar en el árbol \mathcal{L} reducido mostrado en el listado 3.2 se puede comprimir de modo que sólo sean necesarios 9 nodos, los mostrados en el listado 3.3, en que se han eliminado los ítems 2, 5 y 7. El árbol \mathcal{L} compacto se puede expandir con simples cálculos numéricos hasta obtener el árbol \mathcal{L} completo si fuera necesario obtener todos sus nodos, lo habitual es que sólo queramos obtener el valor almacenado en un nodo del árbol \mathcal{L} completo y no sea necesario recrear todo el árbol, aunque tenemos la capacidad de obtener toda la información que contiene.

Listado 3.3: \mathcal{L} compacto

1 - 3 - 6	3
4 - 6	4
6	6

Al ser de reducido tamaño es fácil transmitir *catálogos* comprimidos entre distintos dispositivos, lo que facilitaría su procesamiento por el dispositivo que ha recibido los datos sin necesidad de mayor interacción. Lo que hay que conseguir son estructuras y métodos adecuados para el procesamiento de estos ficheros usando pocos recursos.

En muchos casos el número de nodos del árbol reducido puede ser mayor que el obtenido con el algoritmo original. Sin embargo sólo estamos guardando el *soporte* en cada nodo por lo que no guardamos ni el ítem que tiene ese *soporte* (8, 16 o 32 bits) ni el puntero a su hermano menor (32 o 64 bits) por lo que el hecho de contener más nodos no garantiza que se necesite más memoria RAM para guardar su contenido. Sustituir búsquedas por cálculos sobre índices reducirá mucho el tiempo de ejecución del algoritmo.

Un *catálogo* comprimido contiene en muy poco espacio la misma información que un *catálogo*.

- N , el número de registros del *catálogo*.
- Los valores correspondientes a cada atributo junto a su frecuencia en el *catálogo* (\mathcal{C}_1 ampliado con información de A_i).
- Los valores menos frecuentes de cada atributo.

Con esta información es inmediato reconstruir el *catálogo* original, pero no necesitaremos hacerlo ya que vamos a trabajar con el *catálogo* comprimido y la información extra disponible. De hecho esta información nos permitirá crear el árbol \mathcal{L} completo antes de comenzar a hacer la segunda lectura de \mathcal{D} , esta vez a través del *catálogo* comprimido.

En primer lugar no estamos interesados en crear un árbol \mathcal{L} completo, este era uno de los problemas que provocó la búsqueda de algoritmos eficientes para *ARM*. Sólo necesitamos expandir las primeras ramas del árbol, las correspondientes a los valores de la *clase*. Si necesitáramos cualquier otro valor del árbol completo podríamos reproducirlo con unos pocos cálculos aplicados sobre los datos que contiene nuestro \mathcal{L} reducido.

Es de destacar que al codificar los datos originales para construir un *catálogo* se comenzó con los valores de la *clase* y a continuación se añadieron los atributos, sin aparente orden. El orden en que estén estos atributos podría ser relevante en un análisis clásico de *Minería de Reglas de Asociación*, podría provocar muchas más búsquedas de las necesarias si la codificación fuera otra. En el caso de los *catálogos* comprimidos este hecho no afecta a la eficiencia del algoritmo ni a los recursos empleados. Desde el primer momento se reservará memoria para las primeras ramas del árbol \mathcal{L} completo y desde entonces no habrá reserva ni destrucción de memoria. El efecto inicial es un alto consumo de memoria que realmente no será utilizada (la que se evitaba reservar a la hora de generar candidatos por la propiedad *Apriori*). Sin embargo este consumo de memoria será

compensado por dos factores: si una hoja de nuestro árbol \mathcal{L} reducido queda con *soporte* nulo hemos descubierto una *Regla de Asociación Negativa*, y ahorramos millones de operaciones de búsqueda a cambio de operaciones de cálculo mucho más adaptadas al trabajo de un procesador.

En un fichero con n valores distintos distribuidos en A atributos se parte de un vector inicial de $n - A$ elementos, de los cuales sólo surgirán ramas de los $n_{A_1} - 1$ primeros si estamos ante un problema de *clasificación*.

3.1.5. Lectura de *datasets* comprimidos

ABIERTO

Para almacenar el contenido de un *dataset* de estas características se puede utilizar el árbol \mathcal{L} . Esto podría suponer mayor facilidad de transmisión de datos entre distintos dispositivos siempre que el nuevo formato ocupe menos espacio que el ocupado por el *dataset* original. También se puede utilizar $\mathcal{L}_{reducido}$ o $\mathcal{L}_{compacto}$, en función de sus tamaños y del uso que se quiera dar. Cuando se obtienen nuevos *registros* sin clasificar se puede utilizar cualquiera de estos formatos para contrastar la información que tenemos sobre el nuevo individuo en estudio, mientras utilicemos soporte unitario todos ellos son fácilmente escalables.

Ahora debería explicar qué formato es ideal para qué tarea o situación ¿No?

La lectura de un *dataset* comprimido es elemental si se ha optado por guardar $\mathcal{L}_{reducido}$ para la *clase*. Se dará prioridad al valor de la *clase* por lo que cada vez que se lea un *registro* comprimido se comprobará si el primer ítem se corresponde con un valor de la *clase* o si el valor de la *clase* se ha omitido, con lo que sabemos que la lectura se corresponde con el valor de la *clase* no utilizado en el *catálogo* comprimido. Los *registros clasificados* en la *clase* eliminada. . .

Cada *registro* aumentará el *soporte* de un máximo de M *itemsets*, buscados secuencialmente a partir del valor de la *clase*. Cada vez que se lee un valor del registro (tras comprobar si el primero es o no información sobre la *clase* y actuar en consecuencia) se incrementa el *soporte* del hijo con índice $itemLeido - itemAnterior - 1 - numHermanosItemAnterior$. De este modo con una única lectura de \mathcal{D} se completa $\mathcal{L}_{reducido}$ del que se puede obtener fácilmente cualquier *soporte* de \mathcal{L} .

Si se ha optado por el formato $\mathcal{L}_{compacto}$ se leen uno a uno los ítems del registro aumentando sólo un elemento de $\mathcal{L}_{compacto}$ tras cada lectura (al contrario que en *Apriori* que se busca en el resto de ramas en que esté guardado). Se lee el primer ítem y se incrementa su *soporte* en $\mathcal{L}_1[item1]$, se lee el segundo ítem y se incrementa su *soporte* en $\mathcal{L}_1[item1] - \mathcal{L}_2[item2]$. . . Con lo que no sólo se lee

una vez \mathcal{D} si no que se hace sólo una búsqueda por *transacción* utilizando índices directos en lugar de comparaciones recursivas...

El tamaño de $\mathcal{L}_{compacto}$ depende del número de atributos, M , del número de valores asignado a cada atributo, $n_i, i = 1 \dots M$ y de su ocurrencia en \mathcal{D} (en mushroom.dat el atributo TAL tiene TANTOS valores pero sólo aparece el valor TAL en el *catálogo*, lo que reduce el número de ítems a considerar en el análisis).

Para simplificar la notación en vez de hablar de *clase* y *atributos* consideraremos a la *clase* como un *atributo*. Será el primer *atributo* para simplificar la tarea de *clasificación* aunque podría ser cualquier otro *atributo*.

Sea $item_i \in \{1 \dots N\}$ un conjunto ordenado de los valores de M atributos. Sean $n_i, i = 1, \dots, M$ el número de valores del atributo A_i . Consideremos que en \mathcal{D} puede haber algún par atributo=valor que no aparezca por lo que su *soporte* sería 0 y no aportaría información al análisis, imaginemos que son N'' los que no aparecen en \mathcal{D} . Todos los atributos tienen un valor «más frecuente» (y si no lo tienen se asigna ese apelativo al primero de los «más frecuentes»), del que vamos a prescindir porque la información que proporcionan la podemos obtener analizando el resto de valores del atributo, esto nos proporciona M pares atributo=valor que no tendremos que considerar en el análisis. Luego nos quedan

$$N' = N - N'' - M$$

ítems que sí aportan información sobre \mathcal{D} . Los codificaremos usando los valores $0 \dots N' - 1$ con lo que $\mathcal{L}_1 compacto$ tendrá N' elementos, cada uno de ellos guardará un *soporte* y un vector con los ítems «menores» con que pueda estar relacionado, i.e., con los ítems representados por un número mayor pero que no sean valores del mismo atributo.

Listado 3.4: Librería $\mathcal{L}_{compacto}$

```
#define TItem int

class NODO
{
    int soporte;
    NODO *hijos;

public:
    NODO(TItem item,
         TItem N,
         TItem numHermanosMenores) : soporte(0),
                                     hijos(NULL)
    {
        TItem numHijos = N - item - numHermanosMenores;
        try
        {
            hijos = new (TItem *) [numHijos];
        }
    }
}
```

```

    catch (...)
    {
        //
    }
}

~NODO()
{
    delete hijos;
}

```

El árbol $\mathcal{L}_{compacto}$ se crea completo antes de leer un *dataset* comprimido, con lo que si hubieran problemas de memoria se detectarían antes de comenzar cualquier proceso. El acceso a esta estructura se realiza mediante su índice por lo que se ahorrarán millones de búsquedas en $\mathcal{L}_{compacto}$.

3.2. Minería de Reglas de Clasificación Asociativa

El problema de *Clasificación* tiene ya su propia notación, igual que el de *Minería de Reglas de Asociación* (definida en el capítulo 2), en esta sección intentaremos unificarla y discutiremos algunos aspectos de ambas disciplinas que no han sido considerados en la bibliografía usada para este trabajo, lo que nos llevará también a definir nuevos conceptos e intentar caracterizarlos.

Hernández León y col., 2010, utilizan las siguientes definiciones para modelizar las *Regla de Asociación de Clase* (que otros autores denominan *Reglas de Clasificación Asociativa*), las *reglas de asociación* adaptadas al problema de *Clasificación*.

Sean $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$ un conjunto de n ítems y \mathcal{T} un conjunto de transacciones. Cada *transacción* en \mathcal{T} está formada por un conjunto de ítems X tal que $X \in \mathcal{I}$.

Definición 27 (*Itemset*). El tamaño de un conjunto de ítems está dado por su cardinalidad; un conjunto de ítems de cardinalidad k se denomina k -itemset.

Definición 28 (*Soporte*). El soporte de un conjunto de ítems X , en adelante $Sop(X)$, se define como la fracción de transacciones en \mathcal{T} que contienen a X . El soporte toma valores en el intervalo $[0, 1]$.

Definición 29 (*minSup*). Sea $minSup$ un umbral previamente establecido, un conjunto de ítems X se denomina frecuente (*FI* por sus siglas en inglés) si $Sop(X) \geq minSup$.

Definición 30 (*Regla de Asociación*). Una Regla de Asociación (AR por sus siglas en inglés) sobre el conjunto de transacciones \mathcal{T} es una implicación $X \Rightarrow Y$ tal que $X \subseteq \mathcal{I}$, $Y \subseteq \mathcal{I}$ y $X \cap Y = \emptyset$.

Definición 31 (Especificidad). Dadas dos reglas $R_1 : X_1 \Rightarrow Y$ y $R_2 : X_2 \Rightarrow Y$, se dice que R_1 es más específica que R_2 si $X_2 \subset X_1$.

Las medidas más usadas en la literatura para evaluar la calidad de una Regla de Asociación son el soporte y la confianza.

Definición 32 (Soporte de una regla). El soporte de una regla de asociación $X \Rightarrow Y$ es igual a $\text{Sop}(X \cup Y)$.

Definición 33 (Confianza de una regla). La confianza de una regla de asociación $X \Rightarrow Y$, en adelante $\text{Conf}(X \Rightarrow Y)$ se define en función del soporte como $\frac{\text{Sop}(X \cup Y)}{\text{Sop}(X)}$. La confianza toma valores en el intervalo $[0, 1]$.

Es importante aclarar que cuando se haga referencia a un conjunto de ítems X se estará hablando de un subconjunto de \mathcal{I} y se supondrá, sin pérdida de generalidad, que existe un orden lexicográfico entre los ítems del conjunto \mathcal{I} .

Para extender las definiciones anteriores al problema de clasificación basada en CARs, además del conjunto \mathcal{I} , se tiene un conjunto de clases \mathcal{C} y un conjunto de transacciones etiquetadas $\mathcal{T}_{\mathcal{C}}$ (conjunto de entrenamiento). Las transacciones del conjunto $\mathcal{T}_{\mathcal{C}}$ están formadas por un conjunto de ítems X y una clase $c \in \mathcal{C}$. Esta extensión no afecta las definiciones de soporte y confianza enunciadas previamente.

Definición 34 (*Regla de Asociación de Clase*). Una Regla de Asociación de Clase (CAR) es una implicación $X \Rightarrow c$ tal que $X \subseteq \mathcal{I}$ y $c \in \mathcal{D}$. El soporte de una Regla de Asociación de Clase $X \Rightarrow c$ es igual a $\text{Sop}(X \cup \{c\})$ y la confianza es igual a $\frac{\text{Sop}(X \cup \{c\})}{\text{Sop}(X)}$.

Definición 35. Una Regla de Asociación de Clase $X \Rightarrow c$ ($X \subseteq \mathcal{I}$ y $c \in \mathcal{D}$) satisface o cubre a una transacción $t \subseteq \mathcal{I}$ si $X \subseteq t$.

Los clasificadores desarrollados basados en Reglas de Asociación de Clase seleccionan, para cada transacción t que se desee clasificar, el subconjunto de CARs que la cubren y con este subconjunto determinan la clase que se asignará a t .

Planteamiento del problema

Sea \mathcal{I} un conjunto de ítems, \mathcal{C} un conjunto de *clases*, $\mathcal{T}_{\mathcal{C}}$ un conjunto de transacciones de la forma $\{i_1, i_2, \dots, i_n, c\}$ tal que $\forall_{1 \leq k \leq n} [i_k \in \mathcal{I} \wedge c \in \mathcal{C}]$ (ver tabla 3.1), \mathcal{R} un conjunto ordenado de reglas $X \Rightarrow c$ tal que $X \subseteq \mathcal{I}$ y $c \in \mathcal{C}$, \mathcal{W} una función que asigna un peso a cada regla $r \in \mathcal{R}$ y D un criterio de decisión que utiliza a \mathcal{R} para asignar una *clase* a cada *transacción* t que se desee clasificar.

Cuadro 3.1: Representación general de un conjunto de transacciones

$\mathcal{T}_{\mathcal{C}}$	Ítems				<i>Clase</i>
t_1	i_{11}	i_{12}	\dots	i_{1k_1}	c_1
t_2	i_{21}	i_{22}	\dots	i_{2k_2}	c_2
			\dots		
t_n	i_{n1}	i_{n2}	\dots	i_{nk_n}	c_n

Dados \mathcal{I} , \mathcal{C} y $\mathcal{T}_{\mathcal{C}}$, construir un clasificador basado en CARs consiste en calcular \mathcal{R} , ordenar \mathcal{R} según la función de asignación de peso \mathcal{W} y definir el criterio de decisión D . El problema que se plantea en esta propuesta de tesis doctoral es la construcción de clasificadores basados en CARs.

A continuación presentan varios clasificadores basados en CARs (CBA (Classification Based on Associations), CMAR (Classification based on Multiple Association Rules), MMAC (Multi-class, Multi-label Associative Classification), MCAR (Multi-class Classification based on Association Rules), PRM (Predictive Rule Mining), CPAR (Classification based on Predictive Association Rules), TFPC (Total From Partial Classification)) y los suyos propios (CAR-NF) y una sección en la que discuten las limitaciones de la medida de calidad *Confianza* y discuten las propiedades que debe tener una medida de calidad para ser utilizada en la evaluación y ordenamiento de las CARs.

[...HLeon-
2010]

3.2.1. Adaptación de la notación

Para poder adaptar la notación de Hernández León necesitamos definir antes algunos conceptos.

Definición 36 (Individuo). Sean $A_i, i = 1 \dots n$ un conjunto de atributos, siendo n_i el número de valores distintos que puede tomar el atributo A_i . Definimos un

individuo como una n -tupla, un conjunto de n valores obtenidos ordenadamente de los atributos A_i .

$$\text{individuo} = (a_1, a_2 \dots a_n), a_i \in A_i$$

Definición 37 (Registro). Sean $A_i, i = 1 \dots n$ un conjunto de atributos, siendo n_i el número de valores distintos que puede tomar el atributo A_i . Definimos un registro como una n -tupla, un conjunto de n valores obtenidos ordenadamente de los atributos A_i .

$$\text{registro} = (\text{item}_1, \text{item}_2 \dots \text{item}_n), \text{item}_i \in A_i$$

Definición 38 (Catálogo). Un catálogo es un conjunto de N registros diferentes.

$$\text{catálogo} = \{\text{registro}_i, i = 1 \dots N \mid \text{registro}_i \neq \text{registro}_j \forall i \neq j\}$$

3.3. Catálogo

Al revisar la documentación sobre mushroom.dat² pudimos plantear mejor el problema que queríamos resolver, encontrar todas las *Reglas de Asociación* presentes en un fichero «pequeño» que presenta estas características. En UCI - Machine Learning Repository³ describen su origen como

Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf

ABIERTO

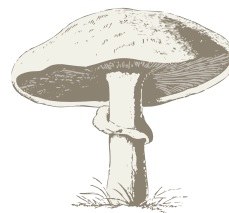


Figura 3.1: Seta (mushroom.dat)

²Ilustración obtenida en https://openclipart.org/detail/900/two-mushrooms-by-johnny_automatic

³<https://archive.ics.uci.edu/ml/datasets/Mushroom>

Describen brevemente el contenido del fichero, indicando que se trata de setas clasificadas como comestibles o venenosas (primer valor de cada fila).

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like “leaflets three, let it be” for Poisonous Oak and Ivy.

Y explican los valores que puede tomar cada uno de los atributos.

1. cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2. cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
3. cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. bruises?: bruises=t, no=f
5. odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6. gill-attachment: attached=a, descending=d, free=f, notched=n
7. gill-spacing: close=c, crowded=w, distant=d
8. gill-size: broad=b, narrow=n
9. gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
10. stalk-shape: enlarging=e, tapering=t
11. stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
12. stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s
13. stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s
14. stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
15. stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16. veil-type: partial=p, universal=u

- 17. veil-color: brown=n, orange=o, white=w, yellow=y
- 18. ring-number: none=n, one=o, two=t
- 19. ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
- 20. spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
- 21. population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
- 22. habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

El fichero `mushroom.dat`⁴ es un fichero de texto con 8 124 líneas, de las que muestro aquí las tres primeras:

```
p,x,s,n,t,p,f,c,n,k,e,e,s,w,w,p,w,o,p,k,s,u
e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g
e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m
```

El fichero `mushroom.dat` con el que llevo años trabajando está codificado de otro modo, conteniendo la misma información

```
1 3 9 13 23 25 34 36 38 40 52 54 59 63 67 76 85 86 90 93 98 107 113
2 3 9 14 23 26 34 36 39 40 52 55 59 63 67 76 85 86 90 93 99 108 114
2 4 9 15 23 27 34 36 39 41 52 55 59 63 67 76 85 86 90 93 99 108 115
```

Este modo de registrar información sobre los individuos de una población es muy habitual en todas las ciencias. Se determina qué *atributos* se pueden medir, codificándolos mediante un número reducido de valores distintos, se observa a un *individuo* y se mide el valor que toma en cada uno de los atributos seleccionados, obteniendo un vector de códigos.

Definición 39 (Individuo). Sean $A_i, i = 1 \dots n$ un conjunto de atributos, siendo n_i el número de valores distintos que puede tomar el atributo A_i . Definimos un individuo como una n -tupla, un conjunto de n valores obtenidos ordenadamente de los atributos A_i .

$$\text{individuo} = (a_1, a_2 \dots a_n), a_i \in A_i$$

⁴<https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data>

El propósito de recoger esta información suele ser el de *clasificación*, clasificar al *individuo* en función de los valores observados en los atributos en estudio. Esta asignación se hace en función de la información que tenemos sobre esta población, generalmente un almacén \mathcal{D} como mushroom.dat, que contiene datos sobre muchos individuos de esta población correctamente clasificados.

Aunque es un problema de *Clasificación* se trata usando *Minería de Reglas de Asociación* por la forma en que se han codificado los valores expuestos para crear el fichero mushroom.dat. Se han utilizado números enteros consecutivos, los dos primeros para determinar la *clase* (1 = *venenosa*, 2 = *comestible*), los siguientes para anotar el valor del atributo cap-shape (3 = *convex*, 4 = *bell*, 5 = *sunken*, 6 = *flat*, 7 = *knobbed*, 8 = *conical*), a continuación los valores del atributo cap-surface (9 = *smooth*, 10, 11 y 12)... Si encontramos *Reglas de Asociación* fuertes entre los atributos y la *clase* podemos intentar clasificar mejor a cualquier individuo de la población.

Esta estructura permite reducir notablemente la información a procesar para extraer reglas de asociación de estos ficheros, lo que convierte su estudio en una mera ejecución con *soporte* mínimo nulo (si fueran más grandes podríamos averiguar antes si no están afectados por el *dilema del ítem raro* como se ha mencionado antes). Analicemos primero la estructura y veremos después una serie de características. Para entenderlo mejor analizaremos mushroom.dat, que es quien nos puso sobre la pista de que algo se hacía mal con este tipo de almacenes \mathcal{D} .

1. Tiene 8 124 filas (a las que hemos llamado *transacciones* pero llamaremos *registro* a partir de ahora).
2. Cada registro tiene 23 elementos.
3. El primer valor de cada registro es un 1 o un 2.

El segundo valor es un 3, 4, 5, 6, 7 u 8.

...

Esto nos hace pensar que cada posición del registro representa a una variable categórica. Y descubrir que los valores utilizados para representar a una variable no coinciden con los utilizados por el resto de variables, y que estos valores son consecutivos, 1... 119.

En la primera lectura de \mathcal{D} , la que utilizamos para crear \mathcal{C}_1 , podemos anotar la longitud de todas las *transacciones*. De este modo será muy fácil comprobar si estamos trabajando con un *catálogo*.

Esto sólo lo detectamos porque se han codificado los valores de cada variable categórica de modo que no haya dos iguales y sean todos consecutivos. Basta con aplicar el algoritmo mostrado en el listado 3.5 para averiguar si \mathcal{D} contiene un *catálogo*, una vez leído y obtenido C_1 y M , el número de valores observado en todas las *transacciones*.

Listado 3.5: Comprobación sobre *catálogos*

```

Input:  $N$ ,  $M$  y  $C_1$ 

suma = 0;
for ( $i = 1 \dots M$ ) do
    suma +=  $C_1$ .soporte;
    if (suma >  $N$ )
        break;
    if (suma ==  $N$ )
        suma = 0;
end

return ((suma ==  $N$ ) || !suma);

```

Un *catálogo* tiene una estructura muy rígida. Si hay valores missing se puede añadir un valor al atributo en cuestión indicando esa circunstancia, o no utilizar la información del atributo en cuestión o incluso hacer una estimación del valor si se puede justificar.

En un *catálogo* todos los registros contienen n datos, y si forma parte de una tarea de *clasificación* uno o varios de esos datos representan una *clase*, generalmente el «*atributo*» cuyo valor queremos determinar a partir de la observación de los valores del resto de *atributos*.

Definición 40 (Registro). Sean $A_i, i = 1 \dots n$ un conjunto de atributos, siendo n_i el número de valores distintos que puede tomar el atributo A_i . Definimos un registro como una n -tupla, un conjunto de n valores obtenidos ordenadamente de los atributos A_i .

$$\text{registro} = (\text{item}_1, \text{item}_2 \dots \text{item}_n), \text{item}_i \in A_i$$

Definición 41 (*Catálogo*). Un catálogo es un conjunto de N registros diferentes.

$$\text{catálogo} = \{\text{registro}_i, i = 1 \dots N \mid \text{registro}_i \neq \text{registro}_j \forall i \neq j\}$$

El número de combinaciones entre todos los valores de los atributos en estudio es muy grande, sin embargo cuando se está haciendo un estudio real no se obtendrán todas las combinaciones posibles. En un *catálogo* sólo están las combinaciones que interesa estudiar, básicamente aquellas que sí se dan en la población en estudio. Es una selección de los registros que se podrían formar con todas las combinaciones de valores proporcionada por los atributos en estudio.

La esencia de un *catálogo* es que no tiene registros repetidos. Si tomamos una muestra de una población midiendo en cada individuo todos los valores de los atributos del estudio es muy probable que en la muestra existan registros repetidos. Las muestras de este tipo se verán a fondo en la sección 3.3.1, es importante estudiar sus diferencias con los *catálogos* pues contienen información sensiblemente diferente.

Un *catálogo* puede cambiarse fácilmente, basta con cambiar el número de valores que puede tomar una de las variables en estudio para que cambie por completo el *catálogo*. Una muestra no puede cambiarse, se obtiene de la realidad, se diseña previamente y si se quiere añadir a otra muestra en que se han utilizado otros valores debería rehacerse. . .

Si aprovechamos esta información podemos reducir notablemente los ítems a procesar sin perder la información que contiene el *catálogo* (o muestra) completo. . .

Los catálogos son colecciones de registros preparadas para resolver informáticamente un problema de clasificación. Y muchos investigadores de esta especialidad publican sus datos para que otros investigadores puedan hacer pruebas con las mismas condiciones de partida: una colección de datos con ciertas características. En UCI, KEEL, LUCS... encontraremos muchos catálogos entre los datasets que publican para resolver problemas de clasificación.

Cuando no sabíamos que esos ficheros contenían catálogos intentábamos aplicar bien conocidos algoritmos de ARM pero no podíamos extraer información que contienen los datos porque se desbordaba la RAM del equipo en que se está aplicando el algoritmo y se abortaba el proceso tras horas de cálculos que finalmente no obteníamos. Esto nos sorprendía porque el primer catálogo que intentamos analizar con Apriori sólo tiene 5 644 registros de 23 datos, no son

Acabo de descubrir LUCS, que discretiza las colecciones de UCI y me ofrece 97 valores distintos en adult, frente a los 27 245 que tiene el de UCI, he

números excesivos para un problema de Minería de Datos analizado con un ordenador de escritorio con cierta potencia y capacidad de RAM. Eso nos llevó a descubrir cómo se creó el catálogo a través de UCI/mushroom...

Los catálogos caracterizan un problema de clasificación concreto. Si queremos plantear otro problema de clasificación, bien etiquetando a los mismos individuos en otras *clases* o bien utilizando atributos diferentes no podemos utilizar directamente cualquier catálogo que tengamos sobre la misma población. Si los dos problemas usaran los mismos atributos pero diferentes *clases* y las *clases* en estudio son independientes no servirá de nada la información que tengamos sobre los catálogos completos del primer problema de clasificación si no sabemos analizar qué información puede ser relevante y cuál no, de hecho la información menos relevante en esta situación es la distribución de las *clases* en cada uno de los problemas de clasificación por lo que debemos huir de interpretaciones erróneas utilizando estos datos para estimar soportes o confianzas poblacionales.

De un catálogo se puede extraer información válida para otro problema de clasificación que utilice los mismos atributos ya que si en la muestra en que se basa el catálogo no presenta cierta relación entre los valores de los atributos YA SABEMOS QUE NO APARECERÁ ESA RELACIÓN AUNQUE CAMBIEMOS DE CLASES (siempre que el catálogo sea válido, aún tengo que hacer muchas definiciones sobre muestra, población, distribución de *clases*, problema de *clasificación*, *atributos*, *clases*, *catálogos*, catálogos completos, validez de un catálogo...).

Aunque la *ARM* busca cualquier relación entre cualquier par (o k -itemset) de valores de \mathcal{D} , el objetivo del problema de *clasificación* es siempre el mismo, etiquetar cada *registro* con una *clase* basándose en la información disponible sobre otros *registros* con valores idénticos en sus *atributos*.

3.3.1. Muestras

La importancia que se da en todos los estudios de *ARM* al *soporte* de una *regla de asociación* se justifica cuando se trabaja con muestras, que no contienen la misma información que un *catálogo*. Para obtener una muestra en un problema de *clasificación* se han de seleccionar individuos de una población, medir en cada uno de ellos el valor que toma cada uno de los *atributos* en estudio y averiguar la *clase* a la que pertenece. Al guardar todos los *registros* obtenidos de este modo es posible que existan *registros* repetidos, lo que aporta información sobre la distribución de la población en estudio con un elevado coste sobre los *catálogos*

ya que en estos no deberíamos repetir el registro si no añadir un número entero indicando el número de veces que aparece el *registro* en la muestra.

3.4. Catálogo Completo

Al seguir trabajando con *mushroom.dat* encontramos otra versión del mismo dataset en KEEL⁵, ésta con 5 644 registros debido a que se eliminaron los registros con datos missing. Este detalle nos hizo pensar en la información que puede aportar un dato desconocido: ninguna. El criterio de eliminación de registros con datos missing parece correcto pero ¿debemos prescindir de la información que puede aportar al estudio los valores recogidos en todos los registros eliminados? ¿Es posible que eliminando esos registros cambie la información que proporcionan los datos al estudio? En la sección 3.5 estudiaremos más a fondo esta circunstancia y ofreceremos resultados interesantes proporcionados por los datos que estamos manejando.

Para la siguiente definición necesito aclarar antes (con la notación de H. León si es posible) qué es un registro-tipo, luego indicar que el *catálogo* es *Catálogo Completo* si un registro-tipo apunta sólo a una *clase*.

Necesito
definir antes
conjunto-
DeValores-
DeAtributos,
quizá en otra
sección.

Definición 42 (*Catálogo Completo*). Un Catálogo Completo es un catálogo sin incertidumbre.

$$CC = \{r_i, i = 1 \dots N \mid r_i \neq r_j \forall i \neq j\}$$

En *mushroom.dat* han eliminado parte de la incertidumbre del problema de *clasificación* etiquetando como *venenosas* aquellas setas sobre las que no están seguros los investigadores (véase la descripción del *dataset* en pg. 132). Esto se podría haber resuelto añadiendo la *clase* «*sin-clasificar*» si el incremento de dimensiones no provoca problemas de memoria, de este modo se ganaría confianza en la clasificación de las setas *venenosas*.

3.4.1. Colecciones de Catálogos Completos

Todos los *Catálogos Completos* pueden tener un *Catálogo Completo* maximal y diversos *Catálogos Completos* de menores dimensiones. Volviendo al origen de esta investigación, disponemos de un dataset con registros clasificados en el que pueden haber datos missing...

⁵<http://sci2s.ugr.es/keel/dataset.php?cod=178>

3.5. Datos missing

mushroom.dat contiene 8 124 registros en su dataset original (UCI) sin embargo en el fichero KEEL contiene sólo 5 644 registros porque se han eliminado los registros con datos missing que contenía el fichero original.

ABIERTO

En un problema de *clasificación* los datos missing no deberían ser considerados. Hay autores que prefieren estimarlos [¿citas?] pero si no hay una justificación suficientemente poderosa no deberíamos «inventar» ningún valor cuando queremos clasificar correctamente a un individuo, la Minería de Datos nos puede dar información para reducir el espacio de búsqueda.

En un registro, la ausencia de un dato provocará tener un valor menos, lo que traducido a transacciones se convierte en una transacción de distinto tamaño que el resto de registros. Esto sería un problema para tratarlo como se ha propuesto en la sección 3.4 ya que el fichero \mathcal{D} no será considerado catálogo por no tener todos sus registros el mismo tamaño. Una solución consiste en considerar el valor missing de un atributo como un valor distinto al resto de los que realmente contiene, en el caso de mushroom.dat se denota con ?, al convertirlo en fichero \mathcal{D} se codificará ese valor como un valor distinto del atributo al que pertenezca y la única consecuencia es, aparentemente, que tenemos un ítem diferente más.

Los datos missing son difíciles de interpretar correctamente ya que para convertir los ficheros en formato adecuado para ARM se utiliza un código binario para indicar esta situación, i.e., si un registro no contiene información sobre un atributo se crea la categoría ficticia "dato desconocido."^{en} dicho atributo y se le asigna esa categoría, así podemos aplicar con cierta normalidad las técnicas de ARM, aunque la interpretación de las reglas que se pueden obtener es confusa:

Si no conocemos el valor del atributo \mathcal{A}_i podemos deducir que...

Eliminar los registros con datos missing parece una estrategia correcta para no tener que gestionar estas *reglas de asociación*. Sin embargo perdemos información sobre 2 480 registros, sin saber si la información que perdemos es relevante o no lo es.

Si analizamos el fichero mushroom.dat observaremos que sólo un atributo contiene datos missing, stalk-root, y ya hemos visto en la sección 3.4 que este atributo es prescindible, al menos en el dataset reducido de KEEL. Con lo que ahora sabemos sobre *catálogos* podemos diseñar otra estrategia, eliminar el atributo stalk-root del dataset original y comprobar si esos 2 480 registros ig-

norados por la estrategia utilizada contienen información que habíamos perdido irremediablemente.

En KEEL - Mushroom proporcionan dos ficheros mushroom.dat:

- por defecto el que han elaborado con su formato tras eliminar los registros con datos missing
- si lo buscamos nos ofrecen el dataset original en formato KEEL, con 8 124 registros.

Al analizar ambos ficheros se obtienen los siguientes resultados:

1. mushroom-original contiene un *Catálogo Completo* de 8 124 registros, con un atributo que contiene datos missing, stalk-root, que es prescindible y no aumenta el tamaño del catálogo.
2. mushroom contiene un *Catálogo Completo* de 5 644 registros, stalk-root es prescindible y no aumenta el tamaño del catálogo.

Añadir dataset a índice-alfabetico

Luego tenemos dos análisis sobre dos datasets que ofrecen datos similares pero deben contener diferente información. Analizar sus similitudes y diferencias nos dará más información que la que obteníamos con el primer fichero KEEL.

El primer dato relevante es que ambos ficheros son *Catálogos Completos*, si eliminamos en cualquiera de ellos el atributo stalk-root no se reduce el catálogo ni se pierde información. Si no sobran los 2 480 registros de más que tiene el catálogo original deberían de aportar algo de información que no está presente en el catálogo reducido.

Cualquier *regla de asociación* que incluya el ítem stalk-root=? en su antecedente debe ser ignorada ya que su interpretación sería «*Si no sé qué valor tiene stalk-root entonces...*».

Si eliminamos stalk-root de ambos ficheros obtendremos dos *Catálogos Completos* distintos, el primero con todos los registros-tipo del segundo y además 2 480 registros-tipo que no están en el segundo. Es evidente que el primero tiene información más precisa sobre el problema de *clasificación* que estamos analizando. Pero la información extra que contiene no es sobre el atributo stalk-root, ni tan siquiera se utiliza este atributo para obtener esa información, sólo depende de las co-ocurrencias entre valores del resto de atributos. Si eliminamos el atributo stalk-root de ambos ficheros obtendremos dos nuevos *Catálogos Completos*, esta vez sin datos missing, ambos orientados al mismo problema

de *clasificación* y sin ninguna discrepancia entre ellos (de hecho el reducido es una partición del original). Si pudiéramos utilizar cualquiera de ellos en nuestro próximo clasificador está claro que deberíamos usar el *Catálogo Completo* que ese obtiene eliminando el atributo stalk-root al dataset original.

El *Catálogo Completo* mayor contiene al menor pero el menor se ha obtenido a partir de la información proporcionada por otro atributo (que ahora ya no está). Ambos contienen algo de información distinta para el mismo problema de *clasificación*. En el menor hay más ARN_2 , que desaparecen al añadir nuevos registros-tipo para formar el mayor, es lo que podría ocurrir si seguimos tomando muestras y aparecen nuevos registros-tipo que no presenten incertidumbre, el *Catálogo Completo* va mejorando y nos informa que los datos que conocemos garantizan una buena clasificación (no es necesario medir otro atributo porque no tenemos incertidumbre y cada vez sabemos más sobre los registros-tipo de este problema de clasificación).

No todos los *catálogos* con datos missing tendrán las mismas características que mushroom.dat, pero sí se puede dar en muchos estudios que sigamos añadiendo registros a un dataset pero sin incluir el valor de uno de sus atributos, si el resto de atributos es suficiente para clasificar correctamente al individuo tendremos una situación similar a la analizada en esta sección. Si estamos trabajando con un *Catálogo Completo* completo y válido es muy posible que no necesitemos nunca más medir el valor de ese atributo, pero no deberíamos borrarlo del dataset original porque alguna vez podría aparecer algo de incertidumbre en el catálogo y sería útil poder añadir toda la información que ya se recogió en su día sobre cualquier otro atributo medible en los individuos en estudio.

3.6. Publicaciones

En Interacción'12⁶ comenzamos a publicar nuestros resultados sobre las colecciones de *transacciones* estructuradas mediante *datasets* comprimidos. Los mayores avances los hemos obtenido este último año y estamos en un proceso actual de publicación de resultados en diversas revistas y un nuevo congreso internacional.

⁶<http://interaccion2012.umh.es/>

Efficient analysis of transactions to improve web recommendations, 2012

Lazcorreta Puigmartí, E., Botella Beviá, F. y Fernández-Caballero, A. Efficient analysis of transactions to improve web recommendations. *Actas del XIII Congreso Internacional de Interacción Persona-Ordenador*, ACM International Conference Proceeding Series, 2012.

Resumen

When we deal with big repositories to extract relevant information in a short period of time, pattern extraction using data mining can be employed. One of the most used patterns employed are Association Rules, which can measure item co-occurrence inside large set of transactions. We have discovered a certain type of transactions that can be employed more efficiently that have been used until today. In this work we have applied a new methodology to this type of transactions, and thus we have obtained execution times much faster and more information than that obtained with classical algorithms of Association Rule Mining. In this way we are trying to improve the response time of a recommendation web system in order to offer better responses to our users in less time. Copyright 2012 ACM.

3.6.1. [...]

En la actualidad estamos completando la redacción de un artículo sobre esta fase de nuestra investigación, los *Catálogos Completos*, que será enviada en breve para su revisión y posible publicación en revista. En él se exponen la teoría y experimentos que forman la sección 3.4.

⁷ https://www.researchgate.net/publication/266652926_Efficient_analysis_of_transactions_to_improve_Web_Recommendations

⁸ <https://www.deepdyve.com/lp/association-for-computing-machinery/efficient-analysis-of-transactions-to-improve-web-recommendations-mBuBBNIv7c>

⁹ https://github.com/EnriqueLazcorreta/tesis-0/blob/master/bib/nuestros/LazcorretaBotellaFCaballero_EfficientAnalysisOfTransactionsWebRecommendations_2012.pdf

¹⁰ https://github.com/EnriqueLazcorreta/tesis-0/blob/master/bib/nuestros/LazcorretaBotellaFCaballero_AnalisisEficienteDeTransacciones_12_Presentacion.pdf

3.6.2. [...]

La Interacción Persona-Ordenador es una disciplina en la que se analizan muchos catálogos por lo que presentaremos nuestra aportación a esta rama de la ciencia en el Congreso Internacional Interacción'16¹¹, que se celebrará en septiembre de 2016 en Salamanca.

¹¹<http://interaccion2016.usal.es/>

Conclusiones y Trabajo Futuro

Las mejores ideas expuestas en esta tesis son muy simples. Desde el primer algoritmo que entra en juego, Apriori, hasta la elaboración de catálogos completos ínfimos son ideas muy simples que implementadas de forma eficiente pueden hacer lo que se le pide a la Minería de Datos: buscar una aguja en un pajar.

Los catálogos completos tienen un potencial fácil de descubrir mediante sencillas técnicas informáticas de Minería de Datos. Este trabajo presenta una teoría en torno a un tipo de datos muy utilizado que posibilita la obtención extrema de la información que contienen grandes colecciones de datos utilizando la tecnología actual en tiempo real.

Los datos bien recogidos reflejan el estado actual del mundo que nos rodea, por eso es importante poder analizarlos rápidamente utilizando en algunos casos información histórica sobre el mismo problema o bien partiendo de un nuevo problema y analizando rápidamente las características de los datos que proporciona su estudio. Si sabemos qué puede descubrir la Minería de Datos a partir de la observación de los datos que hemos recogido podremos crear algoritmos que descubran lo que estamos buscando en tiempo real y con un uso aceptable de recursos de un servidor dedicado. Pero si no entendemos bien el problema que queremos resolver con los datos que tenemos es difícil que lo podamos explicar de una forma genérica a una máquina por muchos recursos y capacidad que tenga.

Este trabajo presenta unos antecedentes que encaminan al investigador a descubrir, quizá por casualidad, las características especiales de un modelo mate-

mático de almacenamiento de información y el uso que se está dando a estas colecciones de datos por parte de especialistas en el problema de *clasificación*. La aparición del problema del *Minería de Reglas de Clasificación Asociativa* en [...] era previsible, todas las reglas de asociación tienen un aspecto muy simple que sugiere a cualquier investigador que puede ser utilizado en el problema de clasificación. El hecho de que yo, especializado en el problema de asociación, observara los mismos datos que los especialistas en clasificación tendría que llevarnos al mismo resultado si ellos habían alcanzado el óptimo o a un mejor resultado si yo era capaz de aportar ideas sobre cómo utilizar los elementos de ARM.

El primer descubrimiento simple y útil de esta tesis son las *Transacciones Estructuradas* expuestas en la sección 3.1. Con ellos descubrí que el modo de aplicar técnicas de ARM en los artículos que consultaba no era del todo correcto [...]. No soy especialista todavía en el Problema de Clasificación por lo que algunas conclusiones de esos artículos y, sobre todo, las pruebas de eficiencia de los algoritmos que proponían, estaban fuera de mi alcance. Se me ocurrió incorporar las restricciones iniciales del problema de clasificación a un problema general de asociación. Los problemas de asociación se resuelven mediante la fuerza bruta leyendo todos los datos que tenemos y mirándolos desde distintas perspectivas, si quiero resolver un problema distinto, un problema de clasificación, usando técnicas de minería de reglas de asociación debería aprovechar, al menos, la rígida estructura de los datasets usados para clasificación (en asociación sólo hay una norma: en un registro no se cuentan los datos repetidos, lo que hace que el número de reglas de asociación que se puede buscar sea tan grande que provoque desbordamiento de memoria en los programas que intentan analizar grandes colecciones de datos). Se me ocurrió que si todos los registros han de tener un valor para cada uno de los atributos en estudio podía reducir el número de datos a procesar y las dimensiones del dataset eliminando únicamente un valor de cada atributo en todo el dataset. Al hacerlo y comprobar que la nueva colección de datos, compresión sin pérdidas de la colección original, sí se podía analizar utilizando el clásico Apriori y obtener todas las reglas de asociación que contenía empecé a asimilar mejor las características de un catálogo.

Primero descubrí características matemáticas, restricciones teóricas que me permitían reducir las dimensiones del problema original y, usando muchos recursos, obtener toda la información que contienen esas pequeñas colecciones de datos en cuanto a reglas de asociación se refiere. Pero tenía que haber algo más, las características matemáticas que utilicé en 3.6 me exigían usar muchos recursos

y no me ofrecían información demasiado relevante, además seguía necesitando mucha RAM para trabajar con colecciones pequeñas de datos, a pesar de que ya sabía que contenían muchísima información. Quería encontrar mejor información en menos tiempo y usando menos RAM por lo que introduje la STL a mi desarrollo y comprobé en la primera aplicación que la teoría de conjuntos tenía mucho que aportar al análisis de catálogos.

Tantos años de trabajo han dado lugar a muchas ideas teóricas sobre la aplicación de técnicas de *DM* por lo que quedan abiertas muchas líneas de investigación que podrían ser continuación de este trabajo. Como *trabajar a nivel de bits* buscando la máxima eficiencia en el uso informático de grandes colecciones de datos, o *profundizar en el desarrollo de Clasificadores*, de *lógica difusa para agrupación de valores en atributos numéricos o de amplios rangos* y de tantas otras cosas que han ido apareciendo en el estado del arte de esta tesis y que no he podido abarcar para centrarme en obtener algo tangible mediante el método científico.

La investigación mostrada en el último capítulo de esta tesis está avalada por su implementación en el campo de la Minería de Datos utilizando la tecnología actual. El preproceso de cualquier catálogo permite crear colecciones de catálogos que pueden ser utilizadas en tiempo real en grandes problemas de clasificación que pueden ser escalados sin tener que renunciar en cada nuevo estudio a todo el conocimiento adquirido en estudios sobre las mismas *clases*. En este trabajo se ha demostrado que cualquier subconjunto de un catálogo completo puede ser tratado como catálogo completo considerando siempre la incertidumbre que puede contener, si quisiéramos utilizar los datos de un problema de clasificación en otro problema de clasificación con otras *clases* podríamos comenzar con los registros-tipo del primer catálogo, todos los que puedan ser clasificados en el segundo problema se incorporan al catálogo del segundo problema pero así no voy bien, lo que quería decir es que si empezamos con el menor de todos los catálogos ínfimos y vamos catalogando en la segunda *clase* todos sus registros podemos llegar a no tener incertidumbre (caso ideal y poco probable si la segunda *clase* es independiente de la primera, dato interesante) pero al menos si tenemos incertidumbre es posible que sea poca, si hacemos lo mismo con otros catálogos ínfimos podríamos descubrir qué atributos aportan más determinación al segundo problema y plantear un catálogo inicial para el segundo problema.

También queda para el futuro la agrupación de valores en los atributos numéricos. Hay ya muchas investigaciones en torno a este campo y creo que con los primeros análisis hechos a un dataset se puede obtener información que pueda

Es evidente que tengo que reescribir este párrafo. La idea es interesante pero...

ayudar al investigador a hacer las agrupaciones de modo que se pueda seguir trabajando con catálogos completos ya que el agrupamiento puede generar incertidumbre. Este aspecto es muy importante pero es mucho lo que hay que investigar para llegar a conclusiones y resultados útiles, como en “Using Conjunction of Attribute Values for Classification”.

4.1. Flujos de Datos

Fernández Mena, Hernández León y Palancar, 2013, presentan un estado del arte actualizado sobre el uso de *Reglas de Asociación de Clase* sobre Flujos de Datos, que está estrechamente relacionado con las propuestas hechas en esta sección ya que centran el foco en el *clasificador* obtenido tras las primeras lecturas de los datasets disponibles para el estudio de *clasificación* en curso.

Los *clasificadores* para flujos de datos, a diferencia de los *clasificadores* para datos estáticos, no disponen de todo el conjunto de transacciones etiquetadas (conjunto de entrenamiento) a priori, sino que estas arriban de forma incremental a lo largo del tiempo. Por otro lado, dado que los flujos de datos son, en teoría, infinitos, resulta imposible cargar todas las transacciones en memoria y se requiere de un algoritmo incremental que actualice el clasificador con la información de las nuevas transacciones, reutilizando la información extraída de las transacciones previas. En general, los clasificadores para flujos de datos [1,8,23] deben garantizar que:

1. Cada *transacción* del flujo de datos sea procesada a lo sumo una vez.
2. Los resultados de *clasificación* estén disponibles en todo momento.
3. El modelo resultante sea consistente con las nuevas *transacciones* que llegan, pues los datos pueden variar a lo largo del tiempo debido a cambios en su distribución de probabilidad.

La *clasificación* en flujos de datos tiene diversas aplicaciones como: detección de software malicioso [46], *clasificación* de paquetes en el

área de las telecomunicaciones [53], monitoreo de procesos industriales [4,37], monitoreo de datos de navegación en vehículos automotrices [38], detección de fraudes en transacciones bancarias [65], entre otros.

Varios han sido los *clasificadores* adaptados al entorno de flujos de datos. Entre los más reportados se encuentran los árboles de decisión [19,25,26,34,36,59,61] y los métodos de ensamble de *clasificadores* [9,39,47,51,57]. Otros *clasificadores* que han recibido menor atención son los basados en el vecino más cercano [5,54], las máquinas de vectores de soporte [43,66] y los basados en reglas. Particularmente, los *clasificadores* basados en *Reglas de Asociación de Clase*, son preferidos por muchos especialistas debido a su interpretabilidad, aspecto que los hace más expresivos y fáciles de comprender. Su interpretabilidad permite a los especialistas modificar las reglas con base en su experiencia y así mejorar la eficacia del clasificador. Además de los clasificadores basados en CARs, los árboles de decisión también generan reglas comprensibles. Para construir un clasificador utilizando árboles de decisión se sigue una estrategia voraz seleccionando en cada momento la característica que mejor separa las *clases*. Sin embargo, esta estrategia voraz puede podar reglas interesantes. En [55], los autores probaron que las reglas obtenidas de los árboles de decisión son un subconjunto de las reglas generadas por los clasificadores basados en CARs, asumiendo un umbral relativamente bajo de concurrencia (Soporte) de los elementos que componen la regla.

Los *catálogos* consiguen reducir el número de objetos a utilizar en memoria RAM, de modo que el *clasificador* puede encontrar una *regla determinista basada en los datos conocidos*.

Añadir *regla determinista basada en los datos conocidos a definiciones e índice.*



Notación

La notación usada en este informe se ha intentado ajustar a la más utilizada en la bibliografía revisada a lo largo de estos años de investigación. Por este motivo no es uniforme en los tres capítulos de investigación en que se divide esta tesis.

A.1. Sistemas de Recomendación Web

En este capítulo...

A.2. Minería de Reglas de Asociación

En este capítulo...

A.3. Catálogos

En este capítulo...



Código

Se ha desarrollado mucho código para poder comprobar todo lo que se afirma en esta tesis. Se ha trabajado del modo más estándar posible para conseguir un código eficiente y que pueda ser incorporado a otras investigaciones. Se publicará como código abierto bajo la licencia... en...

B.1. Sistemas de Recomendación Web

En este capítulo...

B.2. Minería de Reglas de Asociación

En este capítulo...

B.3. Catálogos

En este capítulo...

Listado B.1: Cabecera para lectura de ficheros KEEL

```
#ifndef TFICHEROKEEL_H
#define TFICHEROKEEL_H

#include "defs.h" //(* #include ...
```

```
#include <fstream>
#include <stdio.h>
#include <iostream>
using std::cout;
using std::endl;

// #include <forward_list>
// using std::forward_list;
#include <list>
using std::list;
#include <vector>
using std::vector;
#include <string>
using std::string;
#include <map>
using std::map;
// *)

/** class TFicheroKEEL
 *
 * Esta clase es una interfaz para utilizar los
 * ficheros que pone a nuestra
 * disposición el proyecto @link
 * http://sci2s.ugr.es/keel/index.php KEEL @endlink
 *
 * Extrae la información de los metadatos del fichero ,
 * lee la colección de datos y
 * crea un fichero con el formato que necesita mi
 * aplicación para gestionarlo con
 * eficiencia:
 *
 * — Se codifican los distintos valores de la clase con
 * los códigos 0, 1...
 * — Se codifican el resto de valores mediante números
 * enteros consecutivos sin dejar
 * ninguno reduciendo las necesidades de RAM de los
 * algoritmos utilizados.
 *
 * También crea el fichero D comprimido optimizando los
 * códigos usados. Se guardan
 * también las codificaciones hechas.
```

```

*
*  Guarda también todos los datos descriptivos del
*    fichero , que ayudan a la toma de
*  decisiones del analista y a la elaboración de
*    informes para las pruebas que se
*  hagan sobre estos ficheros .
*
*  Se leen líneas de un máximo de 4096 caracteres , si
*    el fichero tuviera líneas más
*  largas no será correcta la lectura y se podrán
*    obtener resultados inesperados .
*
*  @todo Mayor control sobre capacidad_linea_ y
*    capacidad_separador_ para no usar
*    linea_ y posicion_separador_ fuera de su
*    alcance .
*/
class TInfoFicheroKEEL;
class TFicheroKEEL
{
public:
    static bool CompruebaSiEsKEEL(const string
        &nombre_fichero_datos)
    {
        FILE *fichero =
            fopen(nombre_fichero_datos.c_str() , "rt");
        if (!fichero)
        {
            cout << "No se ha podido abrir el fichero "
                << nombre_fichero_datos
                << " (Abortada la lectura de fichero
                    KEEL)";
            fclose(fichero);
            return false;
        }

        // Busco @data, leyendo sólo las 500 primeras
        // líneas
        int caracter = fgetc(fichero),
            num_linea = 0;
        while (caracter != EOF && num_linea < 500)
        {

```

```

        num_linea++;
        while (caracter != EOF && caracter != '\n' &&
               caracter != '@')
            caracter = fgetc(fichero);
        if (caracter == '@')
        {
            caracter = fgetc(fichero);
            if (caracter == 'd') caracter =
                fgetc(fichero); else continue;
            if (caracter == 'a') caracter =
                fgetc(fichero); else continue;
            if (caracter == 't') caracter =
                fgetc(fichero); else continue;
            if (caracter == 'a') caracter =
                fgetc(fichero); else continue;
            // Si lee @data termina el bucle y la
            búsqueda
            break;
        }
        caracter = fgetc(fichero);
    }
    fclose(fichero);
    return (caracter != EOF && num_linea < 500);
}

private:
    TFicheroKEEL(const string &ruta_ficheros_OUT, const
                  string &nombre_fichero_KEEL);
    virtual ~TFicheroKEEL();

    bool LeeMetadatos(); //(* Métodos de lectura del
                          fichero KEEL
    bool LeeEtiqueta();
    bool LeeNombre();
    bool LeeTipoYDominio();
    bool LeeMetadato();
    bool LeeAtributo();
    bool LeeInputOutput();

    bool LeeDatos();
    bool LeeRegistro(); //*)

```

```

//      unsigned long GetNumRegistros() { return
num_registros_; }
      unsigned long GetNumVariables() { return
      nombre_variables_.size(); }
      const int GetNumClases() const { return
      num_clases_; }
//      unsigned long GetNumValores() { return
num_valores_; }
      vector<string> *GetNombreVariables() { return
      &nombre_variables_; }

      unsigned long GuardaD();
      unsigned long GuardaDComprimido(const string
      &nombre_fichero_D);
//      unsigned long GuardaCl();

      const bool Codificado() const { return codificado_;
      }

      void MuestraElRestoDeLinea();
      int SaltaEspaciosYComas();

      void ReorganizaVariables(); //!< Coloca las clases
      en primer lugar

      unsigned long Codifica();
      int BuscaDatos();
      int LeeYGuardaRegistro(std::ofstream &fichero_OUT);

private:
      /*(* Miembros privados
      string carpeta_proyecto_; //!< Donde guardar
      ficheros auxiliares
      string nombre_fichero_KEEL_; //!< Nombre y
      ubicación del fichero
      FILE *fichero_; //!< Fichero KEEL

      int num_variables_,
      num_clases_;
      unsigned long num_registros_; //!< Número de
      registros del fichero
      unsigned long num_valores_; //!< Número de valores

```

```
        distintos en el fichero

    /// @todo Aclarar si uso list o vector en TODOS los
    /// miembros.
    /// @todo Sustituir por TAttributo
    vector<string> nombre_variables_; //!< Nombres de
        las variables
    vector< vector<string> > dominio_variables_; //!<
        Dominio teórico de variables
    vector<char> tipo_variables_; //!< Real, entero o
        categórico

    map<string , unsigned long> **valores_; //!< Valores
        leídos en el fichero

    vector<string> input_ , //!< Nombre de los atributos
        output_ ; //!< Nombres de las clases

    string nombre_coleccion_;

    char tipo_metadato_;

    string *codigo_2_valor_;
    map<string , int> **valor_2_codigo_;
    bool codificado_ ; //!< *)

    friend class TInfoFicheroKEEL;
};

#endif // TFICHEROKEEL_H
```



Datos utilizados

Para llevar a cabo las pruebas de rendimiento y aplicabilidad de nuestras propuestas se han usado datos propios y datos procedentes de diferentes repositorios públicos como UCI, KEEL, LUCS-KDD...

C.1. Sistemas de Recomendación Web

En este capítulo usamos datos de un servidor propio con la intención de poder utilizar las recomendaciones sugeridas por nuestra metodología en el servidor del que se obtuvieron. Son los ficheros TAL y CUAL que no publicaremos por carecer de interés su contenido, ya que el servidor del que se obtuvieron ya no está disponible y no se podría dar ninguna interpretación a los resultados obtenidos...

También usamos TAL...

C.2. Minería de Reglas de Asociación

En este capítulo seguimos trabajando con los mismos datos que en el anterior e incorporamos...

C.3. Catálogos

En este capítulo es en el que más opciones hemos tenido a la hora de seleccionar datos y probar la eficiencia y posibilidades de nuestros desarrollos. Existen

muchos repositorios públicos bien documentados sobre el diseño y contenido de estos datasets y es una información que enriquece mucho la investigación. . .

Índice de figuras

1.	Proceso <i>KDD</i>	8
1.1.	Fases del proceso de <i>KDD</i> [Fayyad et al. (1996)]	14
1.2.	Selección de datos	23
1.3.	Preproceso	24
1.4.	Estructura del sitio web	28
1.5.	Transformación	32
1.6.	<i>Sesión de navegación</i>	42
1.7.	<i>Sesión de navegación ponderada</i>	43
1.8.	<i>MNW</i>	44
1.9.	Obtención del <i>MPNW</i>	45
2.1.	Tiempos de ejecución con la colección BMS-POS.dat	94
2.2.	Tiempos de ejecución con la colección BMS-View1.dat	95
2.3.	Tiempos de ejecución con la colección de datos propios	95
2.4.	genrules() original vs. nuestra modificación	101
2.5.	\mathcal{D} vs \mathcal{R}	104
3.1.	Seta (mushroom.dat)	131

Índice de tablas

1.1.	Resumen de tiempos de ejecución del algoritmo modificado . .	64
1.2.	Resumen de tiempos de ejecución del algoritmo modificado . .	64
2.1.	Reglas analizadas y encontradas (foodmart)	100
2.2.	Reglas analizadas y encontradas (T40I10D100K)	101
2.3.	\mathcal{D} vs \mathcal{R}	104
3.1.	Representación general de un conjunto de transacciones	130

Índice de definiciones

1.	Definición (<i>Páginas</i> del sitio web)	27
2.	Definición (<i>Enlaces internos</i> del sitio web)	27
3.	Definición (<i>Sesión de navegación</i>)	30
4.	Definición (Conjunto de <i>sesiones de navegación</i>)	31
5.	Definición (<i>Páginas visitadas</i> del sitio web)	33
6.	Definición (Tiempo de permanencia en las <i>páginas visitadas</i>) . .	34
7.	Definición (<i>Enlaces internos utilizados</i> en el sitio web)	34
8.	Definición (<i>Mapa de Navegación Web</i>)	44
9.	Definición (<i>Mapa Personal de Navegación Web</i>)	44
10.	Definición (<i>Mapa de Uso del Sitio Web</i>)	45
11.	Definición (Población de ítems)	48
12.	Definición (<i>Transacción</i>)	49
13.	Definición (Almacén \mathcal{D})	49
14.	Definición (<i>Regla de Asociación</i>)	50
15.	Definición (Minería de Reglas de Asociación)	50
16.	Definición (<i>Soporte</i> de un <i>itemset</i>)	52
17.	Definición (<i>Soporte mínimo</i>)	53
18.	Definición (<i>Itemset</i> frecuente)	53
19.	Definición (Conjunto de <i>itemsets</i> frecuentes, \mathcal{L})	53
20.	Definición (<i>Soporte</i> de una <i>Regla de Asociación</i>)	53

21.	Definición (<i>Confianza de una Regla de Asociación</i>)	53
22.	Definición (<i>Confianza mínima</i>)	54
23.	Definición (Conjunto de candidatos a k -itemsets frecuentes, C_k)	55
24.	Definición (<i>lift</i>)	60
1.	Lema	87
2.	Lema	87
25.	Definición (<i>Registro</i>)	122
26.	Definición (<i>Registro Clasificado</i>)	122
3.	Lema	122
4.	Lema	123
27.	Definición (<i>Itemset</i>)	128
28.	Definición (<i>Soporte</i>)	128
29.	Definición (<i>minSup</i>)	128
30.	Definición (<i>Regla de Asociación</i>)	129
31.	Definición (<i>Especificidad</i>)	129
32.	Definición (<i>Soporte de una regla</i>)	129
33.	Definición (<i>Confianza de una regla</i>)	129
34.	Definición (<i>Regla de Asociación de Clase</i>)	129
35.	Definición	129
36.	Definición (<i>Individuo</i>)	130
37.	Definición (<i>Registro</i>)	131
38.	Definición (<i>Catálogo</i>)	131
39.	Definición (<i>Individuo</i>)	133
40.	Definición (<i>Registro</i>)	135
41.	Definición (<i>Catálogo</i>)	135
42.	Definición (<i>Catálogo Completo</i>)	138

Índice de listados

1.1. Algoritmo de obtención de <i>sesiones de navegación</i>	31
1.2. Función <i>CreaSesion()</i>	32
1.3. Función <i>CierraSesion()</i>	32
1.4. Algoritmo <i>Apriori</i>	55
1.5. Función <i>apriori – gen</i> : unión	56
1.6. Función <i>apriori – gen</i> : poda	56
1.7. Función <i>genrules()</i>	57
1.8. Algoritmo de obtención de \mathcal{D}	59
1.9. Función <i>apriori – gen</i> : unión y poda	62
1.10. Función <i>apriori – gen</i> : poda relajada	62
1.11. Función <i>apriori – gen</i> : unión y poda relajada	63
2.1. Algoritmo AIS, 1993	74
2.2. Algoritmo SETM, 1993	75
2.3. Algoritmo SETM, función merge-scan	76
2.4. Algoritmo SETM, selección de candidatos	76
2.5. Algoritmo SETM, <i>itemsets</i> extendibles	76
2.6. Algoritmo AprioriTID, 1994	81
2.7. Algoritmo DHP (fase 1), 1995	82
2.8. Algoritmo DHP (fase 2), 1995	82
2.9. Algoritmo DHP (fase 3), 1995	83
2.10. Algoritmo Partition, 1995	84

2.11. Algoritmo Basic, 1997	87
2.12. Algoritmo Cumulate, 1997	88
2.13. Algoritmo EstMerge, 1997	89
2.14. Función <i>genrules()</i> modificada	99
3.1. \mathcal{L} completo	123
3.2. \mathcal{L} reducido	124
3.3. \mathcal{L} compacto	124
3.4. Librería \mathcal{L} _compacto	127
3.5. Comprobación sobre <i>catálogos</i>	135
B.1. Cabecera para lectura de ficheros KEEL	153

Índice alfabético

- Bases de Datos (DB), 13, 14, 21–23, 25, 37, 49, 71, 73–75, 77, 83–86, 88, 89, 108, 109
 - Gestor de Bases de Datos (DBMS), 74, 77
- C, 93, 95, 96
- C++, 72, 92, 93, 96
- Clasificación, 11, 35, 36, 43, 48, 49, 51, 77, 105, 115–120, 122, 126, 128, 129, 133, 135–138, 140, 144, 146
 - \mathcal{A} , atributo, 116, 120–124, 126, 133, 135–137, 139
- Catálogo, 11, 124–126, 130, 131, 134–141, 147
 - Catálogo Completo, 11, 137–140, 142
 - mushroom.dat, 11, 115–120, 126, 131–133, 137–140
- Clase, 122–126, 129, 130, 133, 135–137, 145, 147
 - Clasificador, 146, 147
 - \mathcal{C} , conjunto de clases, 129, 130
 - \mathcal{R} , conjunto de reglas, 102–106, 129
 - Registro, 121–124, 137
 - Registro Clasificado, 122
 - Reglas de Asociación de Clase, 128, 129, 146, 147
- Clustering, 26, 27, 35, 36, 46, 70
- Experiencia de Usuario (UX), 15, 16, 20, 36
- Fichero de log (*logfile*), 16, 19, 22, 25, 28–31, 33, 58, 59, 62, 63, 66, 67, 93
 - Extended Log File Format, 22
- Grafo, 9, 28, 37, 42, 43, 45, 46, 58
- Ítem Raro, 70, 108–111, 114, 117, 118

- Dilema del Ítem Raro, 10, 11, 61, 70, 108, 111, 114–118, 133
- Itemset, 48–50, 52–58, 60, 63, 71–77, 79–92, 94, 96–99, 102, 103, 107, 115, 116, 118, 119, 122–124, 126, 128
- Knowledge Discovery in Databases (KDD), 8, 9, 13, 14, 16, 17, 19, 21–25, 27, 35, 38, 40, 43, 49, 69, 77
- Mapa de Navegación Web (MNW), 41, 44
- Mapa de Uso del Sitio Web (MUSW), 45, 46
- Mapa Personal de Navegación Web (MPNW), 44–46
- Minería de Datos (DM), 8–11, 13, 19, 20, 24, 26, 27, 30, 32–36, 40, 41, 43, 47, 49, 59, 67, 69, 77, 93, 94, 116, 145
- Minería de Itemsets Frecuentes (FIM), 54, 55, 61, 78, 79, 86, 90–92, 96, 115, 117
- Minería de Reglas de Asociación (ARM), 16, 36, 38, 47, 49–55, 58, 59, 61, 69–72, 76–81, 83–85, 87–92, 95, 108, 109, 114–123, 125, 128, 133, 137
- AIS, 54, 73, 75, 81
- Apriori, 9, 10, 38, 47, 54, 55, 61, 66, 70, 79–82, 88, 89, 91–93, 96, 97, 102, 103, 106, 112, 113, 125, 126
- Apriori-TID, 54
- AprioriHybrid, 54, 80
- AprioriTID, 80
- Basic, 54, 86
- BitTableFI, 72
- Candidate Distribution, 54, 85, 86
- Clique, 54, 88
- Count Distribution, 54, 85, 86
- Cumulate, 54, 86
- Data Distribution, 54, 85, 86
- DHP, 54, 82, 89–91
- DIC, 54, 86
- Direct, 87
- Eclat, 54, 88, 89
- EstMerge, 54, 87
- FP-Growth, 54, 89
- IHP, 91
- M-Apriori, 89
- Max-Miner, 88
- MaxClique, 54, 88
- MaxEclat, 54, 88
- MaxMiner, 54
- Multipass DHP (M-DHP, 89
- Multipass-Apriori, 89
- MultipleJoins, 87
- OCD, 81
- Partition, 54, 84, 89
- PHP, 90
- Pincer-Search, 90
- PincerSearch, 54
- RangeApriori, 54
- Reorder, 87
- SEAR, 84
- SETM, 54, 74
- SPEAR, 84
- SPINC, 84
- SPTID, 84
- Minería de Reglas de Clasifica-

- ción Asociativa (CARM), 128, 144
- Minería Web (WM), 8, 16, 21
 - Minería de Contenido Web (WCM), 16, 21
 - Minería de Estructura Web (WSM), 16, 21, 28
 - Minería de Uso Web (WUM), 8, 9, 16, 17, 21–23, 25, 27, 32–34, 37, 39–41, 48, 49, 51, 69
- Patrón, 8, 10, 13–15, 17, 20, 27, 30, 36–39, 41, 49, 62, 77, 85
- Personalización de la Web, 9
- Predicción, 24, 35–37, 39, 54, 66, 67
- Reglas de Asociación, 10, 11, 47, 48, 50–55, 57–61, 69–72, 77–81, 83–87, 91, 96, 97, 102, 107, 109–111, 114–116, 118, 128, 131, 133, 137, 139, 140
- Antecedente, 50, 53, 59, 71, 72, 86, 97, 110
- Apriori
 - \mathcal{C} , Conjunto de Candidatos, 55–57, 61, 62, 73–76, 81–84, 87–89, 125, 134
 - \mathcal{L} , Conjunto de Itemsets Frecuentes, 53, 55–57, 61, 62, 73, 74, 81–84, 86–89, 96–99, 115, 123–127, 166
- Confianza, 52–55, 57, 59, 60, 63, 86, 96, 97, 99–101, 103, 105, 107, 118, 128–130
- Consecuente, 50, 53, 59, 60, 71, 86, 110
- \mathcal{D} , dataset o almacén de Transacciones, 49, 50, 52–60, 64, 70–96, 99, 100, 102–105, 109–111, 115–117, 119–123, 125–127, 129, 133, 134, 137, 138
- \mathcal{I} , conjunto de ítems, 48, 49, 52, 64, 71, 72, 76–78, 80, 82, 86, 91, 120–123, 128, 129
- lift*, índice de estímulo, 60, 61, 63, 118
- Reglas de Oportunidad, 10, 70, 108, 110, 111, 114
- Soporte, 10, 52–57, 59–61, 63, 64, 71, 73–77, 81–84, 87–91, 94–100, 103, 105–111, 115–119, 123, 125–129, 133, 137
- TID, Identificador de Transacción, 73, 75, 91
- Transacción, 9–11, 38, 41, 47–54, 57, 58, 60, 63, 71–75, 78, 80, 81, 85, 91, 102, 108, 110, 114, 116–122, 126, 128–130, 134, 141, 146
- Regresión, 26, 35
- Secuencia, 9, 29, 37, 38, 41, 42, 63, 77
- Sesión de navegación, 9, 10, 29–34, 37, 38, 41–44, 46, 47, 49, 51, 58–60, 63, 64, 108, 119, 121
- Sistema de Recomendación (RS), 14–16, 70, 106, 108, 110, 120, 121
- Sistema de Recomendación Web (WRS), 9, 10, 13, 15–17, 20, 40, 51, 58–61, 69, 70, 102
- visualización, 35

World Wide Web (WWW), 7, 8, 16,
18, 19, 21, 30, 69
Portal web, 17–20, 102, 107, 108,
114





Sobre la bibliografía

Todas las citas bibliográficas que figuran en este informe, tanto de trabajos propios como de otros investigadores, contienen enlaces externos que han sido revisados entre noviembre y diciembre de 2015. Para no perder las direcciones utilizadas en la versión impresa se han añadido en forma de notas a pie de página, aunque algunas URLs sean difíciles de reproducir a mano.

ABIERTO

Se han utilizado diferente imágenes¹ para indicar los recursos adicionales sobre la bibliografía utilizada que he ido recopilando a través del proceso de investigación. En la versión impresa aparecen a modo informativo, describiendo visualmente la información adicional que contiene cada elemento bibliográfico.

 indica que hay una copia del artículo, resumen o presentación en el CD que complementa esta tesis, en formato .pdf, en alguna de las carpetas de bib. Sólo es útil si se está revisando este documento desde el CD adjunto o desde una copia de este documento colocada junto a una copia de la carpeta bib del CD.

 enlaza a la URL desde la que se puede descargar el artículo.

 enlaza a la URL con información sobre la referencia.

Scopus enlaza a la URL de <http://www.scopus.com> con información sobre la

¹Obtenidas de <http://sourceforge.net/projects/openiconlibrary/>, una completa librería de iconos e imágenes de código abierto.


referencia. Requiere acceso identificado, en mi caso proporcionado por mi universidad.


☞ enlaza a la URL con el documento publicado como *Working Paper* para el Instituto Universitario Centro de Investigación Operativa de la Universidad Miguel Hernández de Elche.

🔗 enlaza a la URL en que he publicado este informe o la parte de él que se cita.










También están en el CD adjunto las tres bases bibliográficas que he utilizado en este informe, todas ellas en la carpeta bib y con formato BIBTEX. En ellas encontrará el lector más información que la expuesta en este informe. He utilizado los gestores bibliográficos JabRef² y BibDesk³ pero por tratarse de ficheros de texto plano pueden abrirse con múltiples aplicaciones. Se incluye también la base bibliográfica tesis.bib, en la que se encuentran la mayoría de artículos que he utilizado a lo largo de estos años para documentarme sobre el estado del arte de la materia tratada. No se han incluido en este informe todas estas referencias pero no quería que todos estos autores y artículos se olvidaran pues de ellos extraje mucho conocimiento en su día.

Todos los artículos incluidos en formato .pdf han sido descargados sin usar servicios de pago por lo que asumo que su utilización a través de la edición digital de este informe no infringe ninguna licencia de uso. Personalmente me ha resultado muy productivo tener preparados estos enlaces para elaborar un informe de mayor calidad y espero que al lector le sea útil para tener a mano mucha más información sobre los temas tratados en esta tesis.

²  <http://jabref.sourceforge.net/>

³  <http://bibdesk.sourceforge.net/>

Bibliografía

- Agrawal, Rakesh, Tomasz Imielinski y Arun Swami (1993a). «Database Mining: A Performance Perspective». En: *IEEE Transactions on Knowledge and Data Engineering* 5.  ¹, págs. 914-925 (vid. pág. 77).
- (1993b). «Mining Association Rules between Sets of Items in Large Databases». En: *Proc. of the 1993 ACM SIGMOD International Conference on Management of data*. Ed. por P. Bunemann y S. Jajodia.  ². Washington, D.C., United States, págs. 207-216 (vid. págs. 36, 47, 54, 71, 77, 78, 118).
- Agrawal, Rakesh, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen y A. Inkeri Verkamo (1996). «Advances in knowledge discovery and data mining». En: ed. por Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth y Ramasamy Uthurusamy.  ³ ⁴. Menlo Park, CA, USA: American Association for Artificial Intelligence. Cap. Fast discovery of association rules, págs. 307-328 (vid. pág. 81).
- Agrawal, Rakesh y John C. Shafer (1996). «Parallel Mining of Association Rules». En: *IEEE Trans. on Knowl. and Data Eng.* 8.6.  ⁵, págs. 962-969 (vid. págs. 54, 85).

















¹<http://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/tkde93.pdf>

²<http://www.almaden.ibm.com/cs/quest/papers/sigmod93.pdf>

³<http://cs-people.bu.edu/evimaria/cs565/advances.pdf>

⁴<http://portal.acm.org/citation.cfm?id=257938.257975>

⁵<http://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/rj10004.pdf>

- Agrawal, Rakesh y Ramakrishnan Srikant (1994a). «Fast Algorithms for Mining Association Rules». En: *Proc. of the 20th Very Large Data Bases Conference*. Ed. por Jorge B. Bocca, Matthias Jarke y Carlo Zaniolo.  ⁶. VLDB. Morgan Kaufmann Publishers Inc., págs. 487-499 (vid. págs. 9, 47, 54, 80, 81, 97, 118).
- (1994b). «Fast Algorithms for Mining Association Rules (EXTENDIDO)».  ⁷ (vid. págs. 47, 54, 57, 70).
- (1995). «Mining Sequential Patterns». En: *Eleventh International Conference on Data Engineering*. Ed. por Philip S. Yu y Arbee S. P. Chen.  ⁸. Taipei, Taiwan: IEEE Computer Society Press, págs. 3-14 (vid. págs. 37, 41).
- An, Aijun y Nick Cercone (2001). «Rule Quality Measures for Rule Induction Systems: Description and Evaluation». En: *Computational Intelligence*.  ⁹, págs. 409-424 (vid. pág. 61).
- Anderson, Corin R. (2002). «A Machine Learning Approach to Web Personalization».  ¹⁰. Tesis doct. Department of Computer Science & Engineering, University of Washington (vid. pág. 9).
- Baeza-Yates, Ricardo y Cuauhtémoc Rivera Loaiza (2003). «Ubicuidad y Usabilidad en la Web». En: *Revista de Gerencia Tecnológica Informática* 2.2. ¹¹ (vid. pág. 20).
- Bayardo, Roberto J. (1998). «Efficiently mining long patterns from databases». En: *Proc. of the 1998 ACM-SIGMOD Int. Conf. on Management of Data*.  ¹², págs. 85-93 (vid. págs. 54, 88, 116).
- Bodon, Ferenc (2003). «A fast APRIORI implementation». En: *Proceedings of the 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI'03)*. Vol. 90.  ¹³ ¹⁴ (vid. pág. 92).
- (2004). «Surprising results of trie-based FIM algorithms». En: *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'04)*. Ed. por Bart Goethals, Mohammed Javeed Zaki y Roberto Ba-

⁶ <http://www.vldb.org/conf/1994/P487.PDF>

⁷ <http://rakesh.agrawal-family.com/papers/vldb94apriori-rj.pdf>

⁸ <http://rakesh.agrawal-family.com/papers/icde95seq.pdf>

⁹ <http://www.cse.yorku.ca/~aan/research/paper/ci01.pdf>





















¹⁰ www.the4cs.com/~corin/research/pubs/thesis.ps.gz

¹¹ <http://users.dcc.uchile.cl/~rbaeza/inf/usabilidad.html>

¹² <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.107.1120&rep=rep1&type=pdf>

¹³ http://www.cs.bme.hu/~bodon/kozos/papers/bodon_trie.pdf

¹⁴ <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-90/bodon.pdf>

- yardo. Vol. 126. CEUR Workshop Proceedings.  ¹⁵. Brighton, UK (vid. págs. 61, 92).
- (2005). «A trie-based APRIORI implementation for mining frequent item sequences». En: *Proceedings of the 1st International Workshop on Open Source Sata Mining (OSDM)*.  ¹⁶ ¹⁷. Chicago, Illinois: ACM, págs. 56-65 (vid. págs. 61, 92).
- (2006). *A Survey on Frequent Itemset Mining*.  ¹⁸ (vid. pág. 92).
- Borgelt, Christian (2004). «Efficient implementations of Apriori and Eclat». En: *Proc. of the Workshop on Frequent Itemset Mining Implementations*.  ¹⁹. FIMI'04 (vid. págs. 61, 116).
- Borges, José y Mark Levene (1999). «Data Mining of User Navigation Patterns». En: *Revised Papers from the International Workshop on Web Usage Analysis and User Profiling - WEBKDD 1836*.  ²⁰, págs. 92-11 (vid. págs. 10, 37, 41).
- Brin, Sergey, Rajeev Motwani, Jeffrey D. Ullman y Shalom Tsur (1997). «Dynamic Itemset Counting and Implication Rules for Market Basket Data». En: *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. Ed. por Joan Peckham.  ²¹  ²². ACM Press, págs. 255-264 (vid. págs. 54, 86).
- Bucklin, Randolph E. y Catarina Sismeiro (2001). «A Model of Web Site Browsing Behavior Estimated on Clickstream Data». En: *Journal of Marketing Research* 40.3.  ²³, págs. 249-267 (vid. pág. 28).
- Carmona, C.J., S. Ramírez-Gallego, F. Torres, E. Bernal, M.J. del Jesus y S. García (2012). «Web usage mining to improve the design of an e-commerce website: OrOliveSur.com». En: *Expert Systems with Applications* 39. ²⁴, págs. 11243-11249 (vid. pág. 121).
- Chen, Xin y Xiaodong Zhang (2003). «A Popularity-Based Prediction Model for Web Prefetching». En: *IEEE Computer*.  ²⁵, págs. 63-70 (vid. pág. 39).

¹⁵ <http://www.cs.bme.hu/~bodon/en/apriori/bodon.pdf>

¹⁶ <http://www.cs.bme.hu/~bodon/kozso/papers/p56-bodon.pdf>

¹⁷ http://www.cs.bme.hu/~bodon/kozso/papers/bodon_osdm05_slide.pdf

¹⁸ <http://www.cs.bme.hu/~bodon/kozso/papers/fim-survey.pdf>

¹⁹ www.borgelt.net/papers/fimi_03.ps.gz

²⁰ http://www.dcs.bbk.ac.uk/~mark/download/web_mining.pdf















²¹ <http://www.cs.ucla.edu/~zaniolo/czdemo/tsur/Papers/dic-final.ps>

²² <http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput695-00/papers/dic.pdf>

²³ <http://cobweb.cs.uga.edu/~eileen/WebEffectiveness/Papers/bucklinandsismeiro2003.pdf>

²⁴ <http://www.scopus.com/inward/record.url?eid=2-s2.0-84861193731&partnerID=40&md5=5ab85a7f28c008bd2af1bf062088ab6>

²⁵ <http://web.cse.ohio-state.edu/hpcs/WWW/HTML/publications/papers/TR-03-1.pdf>

- Cheung, William y Osmar R. Zaiane (2003). *Incremental Mining of Frequent Patterns Without Candidate Generation Or Support*.  ²⁶ (vid. pág. 91).
- Cho, Yoon Ho, Jae Kyeong Kim y Soung Hie Kim (2002). «A personalized recommender system based on web usage mining and decision tree induction». En: *Expert Systems with Applications* 23.3. ²⁷, págs. 329-342 (vid. pág. 16).
- Constantine, Larry L. y Lucy A. D. Lockwood (1999). *Software for Use: A Practical Guide to the Models and Methods of Usage-centered Design*. ²⁸. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. (vid. pág. 18).
- Cooley, Robert, Bamshad Mobasher y Jaideep Srivastava (1997). «Web Mining: Information and Pattern Discovery on the World Wide Web». En: *Proc. of ICTAI'97*.  ²⁹ (vid. pág. 21).
- (1999). «Data Preparation for Mining World Wide Web Browsing Patterns». En: *Knowledge and Information Systems*.  ³⁰, págs. 5-32 (vid. pág. 25).
- De Bra, Paul, Lora Aroyo y Vadim Chepegin (2004). «The Next Big Thing: Adaptive Web-Based Systems». En: *Journal of Digital Information* 5.1. ³¹ (vid. pág. 21).
- Dong, Jie y Min Han (2007). «BitTableFI: An efficient mining frequent itemsets algorithm». En: *Knowledge-Based Systems* 20.4. ³², págs. 329-335 (vid. pág. 72).
- Duyn, Douglas K. Van, James Landay y Jason I. Hong (2002). *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. ³³. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. (vid. pág. 18).
- Etzioni, Oren (1996). «The World-Wide Web: Quagmire or Gold Mine?». En: *Communications of the ACM* 39.11.  ³⁴, págs. 65-68 (vid. págs. 8, 16).
- Fayyad, Usama, Gregory Piatetsky-Shapiro y Padhraic Smyth (1996). «From Data Mining to Knowledge Discovery in Databases». En: *AI Magazine* 17.3. ³⁵, págs. 37-54 (vid. págs. 8, 13).

²⁶ <http://webdocs.cs.ualberta.ca/~zaiane/postscript/ideas03-2.pdf>

²⁷ <http://www.sciencedirect.com/science/article/pii/S0957417402000520>

²⁸ <http://dl.acm.org/citation.cfm?id=301248>

²⁹ http://dmr.cs.umn.edu/Papers/P1997_5.pdf

³⁰ <http://facweb.cs.depaul.edu/mobasher/classes/ect584/papers/cms-kais.pdf>


















³¹ <http://wwwis.win.tue.nl/~debra/jodi2004/>

³² <http://www.sciencedirect.com/science/article/pii/S095705106001493>

³³ <http://dl.acm.org/citation.cfm?id=548998>

³⁴ <ftp://ftp.cs.washington.edu/homes/etzioni/softbots/cacm96.ps>

³⁵ <http://www.csd.uwo.ca/faculty/ling/cs435/fayyad.pdf>

- Fernández Mena, Yaniela, Raudel Hernández León y José Hernández Palancar (2013). *Clasificación basada en reglas de asociación de clase para flujos de datos, un estado del arte*. Inf. téc. Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV) (vid. pág. 148).
- Ferré, Xavier, Natalia Juristo, Helmut Windl y Larry Constantine (2001). «Usability Basics for Software Developers». En: *IEEE Software* 18.1.  ³⁶, págs. 22-29 (vid. pág. 20).
- Fink, Josef, Alfred Kobsa y Andreas Nill (1996). «User-Oriented Adaptivity and Adaptability in the AVANTI Project». En: *Proceedings of the Conference Designing for the Web: Empirical Studies*.  ³⁷ (vid. pág. 20).
- Frawley, William J., Gregory Piatetsky-Shapiro y Christopher J. Matheus (1992). «Knowledge Discovery in Databases: An Overview». En: *AI Magazine* 13.3.  ³⁸  ³⁹, págs. 57-70 (vid. pág. 15).
- Friedman, Jerome H. (1997). «Data mining and statistics: What's the connection?» En: *Proceedings of the 29th Symposium on the Interface*.  ⁴⁰ (vid. pág. 14).
- Goethals, Bart (2003). *Survey on Frequent Pattern Mining*. Inf. téc.  ⁴¹. HIIT Basic Research Unit; Department of Computer Science; University of Helsinki, pág. 43 (vid. págs. 35, 61, 69).
- Groth, Dennis P. y Edward L. Robertson (2001). «Discovering Frequent Itemsets in the Presence of Highly Frequent Items». En: *Web Knowledge Management and Decision Support: 14th International Conference on Applications of Prolog*. Vol. 2543. Lecture Notes in Computer Science.  ⁴². INAP 2001. Springer-Verlag GmbH, págs. 251-264 (vid. pág. 54).
- Han, Jiawei y Yongjian Fu (1995). «Discovery of Multiple-Level Association Rules from Large Databases». En: *Proc. of the Int. Conf. on Very Large Data Bases*.  ⁴³, págs. 420-431 (vid. págs. 83, 109).
- Han, Jiawei, Micheline Kamber y Jian Pei (2011). *Data Mining: Concepts and Techniques*. 3rd. ⁴⁴. Elsevier Ltd, Oxford (vid. pág. 41).

³⁶ http://lucio.ls.fi.upm.es/xavier/papers/usability_b.pdf

³⁷ <http://www.ics.uci.edu/~kobsa/papers/1996-designing-web-kobsa.pdf>

³⁸ <http://aaai.org/ocs/index.php/aimagazine/article/viewFile/1011/929>

³⁹ https://mitpress.mit.edu/sites/default/files/titles/content/9780262660709_sch_0001.pdf

















⁴⁰ <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.6162&rep=rep1&type=pdf>

⁴¹ <http://adrem.ua.ac.be/~goethals/software/survey.pdf> <http://adrem.ua.ac.be/~goethals/software/survey.pdf>

⁴² https://www.researchgate.net/publication/2884014_Discovering_Frequent_Itemsets_in_the_Presence_of_Highly_Frequent_Items

⁴³ <http://www.vldb.org/conf/1995/P420.PDF>

⁴⁴ <http://www.sciencedirect.com/science/book/9780123814791>

- Han, Jiawei, Jian Pei y Yiwen Yin (2000). «Mining frequent patterns without candidate generation». En: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. SIGMOD '00.  ⁴⁵. Dallas, Texas, United States: ACM, págs. 1-12 (vid. págs. 54, 89).
- Han, Jiawei, Jian Pei, Yiwen Yin y Runying Mao (2004). *Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*.  ⁴⁶ (vid. pág. 54).
- He, Daqing y Ayse Göker (2000). «Detecting Session Boundaries from Web User Logs». En: *Proc. of of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*.  ⁴⁷ (vid. pág. 30).
- Hernández León, Raudel, Jesús A. Carrasco, José Hernández Palancar y J. Fco. Martínez Trinidad (2010). *Desarrollo de Clasificadores basados en Reglas de Asociación*. Inf. téc.  ⁴⁸. Coordinación de Ciencias Computacionales (INAOE) (vid. pág. 128).
- Hervás-Martínez, César, Cristóbal Romero Morales y Sebastián Ventura Soto (2004a). «Comparación de medidas de evaluación de reglas de asociación». En: *Tercer Congreso de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB'04*.  ⁴⁹ (vid. pág. 37).
- (2004b). «Selección de medidas de evaluación de reglas obtenidas mediante programación genética basada en gramática». En:  ⁵⁰. Tendencias de la Minería de Datos en España, Red Española de Minería de Datos. Cap. 23 (vid. pág. 37).
- Hipp, Jochen, Ulrich Güntzer y Gholamreza Nakhaeizadeh (2000). «Algorithms for Association Rule Mining - A General Survey and Comparison». En: *SIGKDD Explorations* 2.1.  ⁵¹, págs. 58-64 (vid. págs. 69, 89).
- Holsheimer, Marcel, Martin Kersten, Heikki Mannila y Hannu Toivonen (1995). «A Perspective on Databases and Data Mining». En: *In 1st Intl. Conf. Knowledge Discovery and Data Mining*.  ⁵², págs. 150-155 (vid. pág. 77).
- Holt, John D. y Soon M. Chung (1999). «Efficient mining of association rules in text databases». En: *Proceedings of the eighth international conference on*

⁴⁵ http://cs.sungshin.ac.kr/~jpark/HOME/References/han_sigmod00.pdf

⁴⁶ http://www.cs.sfu.ca/~jpei/publications/dami03_fpgrowth.pdf

⁴⁷ <http://www.pitt.edu/~dah44/docs/he00detecting.pdf>















⁴⁸ <http://ccc.inaoep.mx/portalfiles/file/CCC-10-002.pdf>

⁴⁹ <http://sci2s.ugr.es/keel/publication.php>

⁵⁰ <http://www.lsi.us.es/redmidas/Capitulos/LMD23.pdf>

⁵¹ <http://www.kdd.org/sites/default/files/issues/2-1-2000-06/hipp.pdf>

⁵² http://pdf.aminer.org/000/472/799/a_perspective_on_databases_and_data_mining.pdf

- Information and knowledge management*. CIKM '99.  ⁵³. Kansas City, Missouri, United States: ACM, págs. 234-242 (vid. pág. 89).
- (2002). «Mining association rules using inverted hashing and pruning». En: *Information Processing Letters* 83 (4). ⁵⁴, págs. 211-220 (vid. pág. 91).
- Hong, Tzung-Pei, Chan-Sheng Kuo y Shyue-Liang Wang (2004). «A fuzzy Apriori-Tid mining algorithm with reduced computational time». En: *Applied Soft Computing* 5.1. ⁵⁵, págs. 1-10 (vid. pág. 91).
- Houtsma, Maurice A. W. y Arun N. Swami (1993). *Set-Oriented Mining of Association Rules in Relational Databases*. Inf. téc.  ⁵⁶. IBM Almaden Research Center (vid. págs. 54, 74, 77).
- (1995). «Set-Oriented Mining for Association Rules in Relational Databases». En: *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*.  ⁵⁷. Washington, DC, USA: IEEE Computer Society, págs. 25-33 (vid. pág. 77).
- Hu, Ya-Han y Yen-Liang Chen (2006). «Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism». En: *Decision Support Systems* 42.1. Referenciado en ESWA-Apriori2, págs. 1-24 (vid. pág. 118).
- Huang, Xiangji, Fuchun Peng, Aijun An y Dale Schuurmans (2004a). «Dynamic Web Log Session Boundary Detection with Statistical Language Modeling». En: *Journal of the American Society for Information Science and Technology (JASIST)* 55.14.  ⁵⁸, págs. 1290-1303 (vid. pág. 30).
- (2004b). *Dynamic Web Log Session Identification with Statistical Language Models*.  ⁵⁹ (vid. pág. 30).
- Huang, Xiangji, Fuchun Peng, Aijun An, Dale Schuurmans y Nick Cercone (2003). «Session Boundary Detection for Association Rule Learning Using n -Gram Language Models». En: *Proc. of the 16th Canadian Conference on Artificial Intelligence*.  ⁶⁰ (vid. pág. 30).

⁵³ http://pdf.aminer.org/000/095/434/efficient_mining_of_association_rules_in_text_databases.pdf

⁵⁴ <http://portal.acm.org/citation.cfm?id=608109.608115>

⁵⁵ <http://dx.doi.org/10.1016/j.asoc.2004.03.009>















⁵⁶ <http://doc.utwente.nl/19441/1/00380413.pdf>

⁵⁷ <http://doc.utwente.nl/19441/1/00380413.pdf>

⁵⁸ <http://www.yorku.ca/jhuang/paper/jasist04.pdf>

⁵⁹ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.9.6602&rep=rep1&type=pdf>

⁶⁰ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.12.8127&rep=rep1&type=pdf>

- Joachims, Thorsten, Dayne Freitag y Tom Mitchell (1997). «WebWatcher: A Tour Guide for the World Wide Web». En: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.  ⁶¹ (vid. pág. 20).
- Kahramanli, Humar y Novruz Allahverdi (2009). «A new Method for Composing Classification Rules: AR+OPTBP». En: *Proceedings of The 5th International Advanced Technologies Symposium (IATS'09)*.  ⁶², págs. 1-6 (vid. págs. 36, 51).
- Kim, Kyung-Joong y Sung-Bae Cho (2007). «Personalized mining of web documents using link structures and fuzzy concept networks». En: *Applied Soft Computing* 7.1. ⁶³, págs. 398-410 (vid. pág. 19).
- Kiran, R. Uday y P. Krishna Reddy (2009). *An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules*. Inf. téc. 2009 IEEE Symposium on Computational Intelligence y Data Mining (IEEE CIDM 2009) (vid. págs. 109, 118).
- Kohavi, Ron y Ross Quinlan (1999). *Decision Tree Discovery*.  ⁶⁴ (vid. págs. 35, 36).
- Kosala, Raymond y Hendrik Blockeel (2000). «Web Mining Research: A Survey». En: *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM* 2.1.  ⁶⁵, págs. 1-15 (vid. pág. 21).
- Kouris, Ioannis N., Christos H. Makris y Athanasios K. Tsakalidis (2005). «Using information retrieval techniques for supporting data mining». En: *Data & Knowledge Engineering* 52.3. ⁶⁶, págs. 353-383 (vid. pág. 15).
- Lam, Shyong K “Tony”, Dan Frankowski y John Riedl (2006). «Do You Trust Your Recommendations? An Exploration Of Security and Privacy Issues in Recommender Systems». En: *Lecture Notes in Computer Science* 3995.3. 
⁶⁷, págs. 14-29 (vid. págs. 19, 23).
- Li, Haifeng y Ning Zhang (2010). «Mining maximal frequent itemsets on graphics processors». En: *FSKD*. ⁶⁸ ⁶⁹, págs. 1461-1464 (vid. pág. 116).

⁶¹ http://www.cs.cornell.edu/people/tj/publications/joachims_etal_97b.pdf

⁶² http://iats09.karabuk.edu.tr/press/bildiriler_pdf/IATS09_01-01_105.pdf

⁶³ <http://www.sciencedirect.com/science/article/pii/S1568494605000815>

⁶⁴ <http://robotics.stanford.edu/users/ronnyk/treesHB.ps>
















⁶⁵ <http://arxiv.org/pdf/cs.LG/0011033.pdf>

⁶⁶ <http://www.sciencedirect.com/science/article/pii/S0169023X04001272>

⁶⁷ <http://files.grouplens.org/papers/lam-etrics2006-security.pdf>

⁶⁸ <http://dx.doi.org/10.1109/FSKD.2010.5569206>

⁶⁹ [http://lmgtfy.com/?q="Miningmaximalfrequentitemsetsongraphicsprocessors"](http://lmgtfy.com/?q=)

- Lichman, M. (2013). *UCI Machine Learning Repository*. ⁷⁰ (vid. pág. 117).
- Lin, Dao-I y Zvi M. Kedem (2002). «Pincer-search: An efficient algorithm for discovering the maximum frequent set». En: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* 14.3. ⁷¹, págs. 553-566 (vid. págs. 54, 90, 91).
- Liu, Bing y Wynne Hsu (1996). «Post-Analysis of Learned Rules». En: *AAAI/IAAI, Vol. 1*.  ⁷², págs. 828-834 (vid. págs. 35, 38).
- Liu, Bing, Wynne Hsu y Yiming Ma (1998). «Integrating Classification and Association Rule Mining». En: *Knowledge Discovery and Data Mining*.  ⁷³, págs. 80-86 (vid. págs. 35, 36, 51, 116).
- (1999). «Mining association rules with multiple minimum supports». En: *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*.  ⁷⁴. San Diego, California, United States: ACM Press, págs. 337-341 (vid. págs. 109, 118).
- Liu, Guimei (2005). «Supporting Efficient and Scalable Frequent Pattern Mining».  ⁷⁵. Tesis doct. The Hong Kong University of Science y Technology (vid. pág. 91).
- Liu, Guimei, Hongjun Lu y Jeffrey Xu Yu (2007). «CFP-tree: A compact disk-based structure for storing and querying frequent itemsets». En: *Information Systems* 32.2. ⁷⁶, págs. 295-319 (vid. pág. 91).
- Liu, Li, Eric Li, Yimin Zhang y Zhizhong Tang (2007). «Optimization of frequent itemset mining on multiple-core processor». En: *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*.  ⁷⁷. Vienna, Austria: VLDB Endowment, págs. 1275-1285 (vid. pág. 91).
- Malik, Kuldeep Singh y Neeraj Raheja (2013). «Improving performance of Frequent Itemset algorithm». En: *International Journal of Research in Engineering & Applied Sciences* 3.3.  ⁷⁸, págs. 168-177 (vid. pág. 116).
- Mannila, Heikki, Hannu Toivonen y A. Inkeri Verkamo (1994). «Efficient algorithms for discovering association rules». En: *AAAI Workshop on Knowledge Discovery in Databases (KDD'94)*. Ed. por Usama M. Fayyad y Ramasamy

⁷⁰ <http://archive.ics.uci.edu/ml>

⁷¹ <http://dl.acm.org/citation.cfm?id=628231>

⁷² <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.42.7715&rep=rep1&type=pdf>

⁷³ <http://www.cs.uiuc.edu/class/fa05/cs591han/papers/bliu98.pdf>

















⁷⁴ <http://www.philippe-fournier-viger.com/spmf/MISApriori.pdf>

⁷⁵ <http://lbezone.ust.hk/bib/b863997>

⁷⁶ <http://www.sciencedirect.com/science/article/pii/S0306437905000906>

⁷⁷ <http://www.vldb.org/conf/2007/papers/industrial/p1275-liu.pdf>

⁷⁸ <http://www.euroasiapub.org/IJREAS/mar2013/18.pdf>

- Uthurusamy.  ⁷⁹. Seattle, Washington: AAAI Press, págs. 181-192 (vid. pág. 80).
- Mobasher, Bamshad, Robert Cooley y Jaideep Srivastava (2000). «Automatic Personalization Based on Web Usage Mining». En: *Communications of the ACM* 43.8.  ⁸⁰, págs. 142-151 (vid. pág. 21).
- Mueller, Andreas (1995). *Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison*. Inf. téc.  ⁸¹. University of Maryland at College Park (vid. pág. 84).
- Ng, Raymond T. y Jiawei Han (1994). «Efficient and Effective Clustering Methods for Spatial Data Mining». En: *Proc. of the 20th International Conference on Very Large Data Bases*.  ⁸². VLBD'05. Los Altos, CA 94022, USA: Morgan Kaufmann Publishers, págs. 144-155 (vid. págs. 35, 39, 46).
- Nielsen, Jacob (1998). *Personalization is Over-Rated*. Inglés. WWW.  ⁸³. NN/g Nielsen Norman Group (vid. págs. 17, 18).
- (2000). *Designing Web Usability*. Designing Web Usability n.º 667. ⁸⁴. New Riders (vid. pág. 18).
- Özel, S. Ayşe y H. Altay Güvenir (2001). «An Algorithm for Mining Association Rules Using Perfect Hashing and Database Pruning». En: *Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'01)*. Ed. por A. Acan, I. Aybay y M. Salamah.  ⁸⁵. Gazimagusa, T.R.N.C., págs. 257-264 (vid. pág. 90).
- Ozmutlu, H. Cenk, Amanda Spink y Seda Ozmutlu (2002). «Analysis of large data logs: an application of Poisson sampling on excite web queries». En: *Information Processing & Management* 38.4. ⁸⁶, págs. 473-490 (vid. pág. 36).
- Palshikar, Girish K., Mandar S. Kale y Manoj M. Apte (2007). «Association rules mining using heavy itemsets». En: *Data & Knowledge Engineering* 61.1.  ⁸⁷, págs. 93-113 (vid. pág. 109).

⁷⁹ <http://www.cs.helsinki.fi/TR/0/0-1994-11.ps.gz>

⁸⁰ http://dmr.cs.umn.edu/Papers/P2000_10.pdf

⁸¹ <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.947>

⁸² <http://www.vldb.org/conf/1994/P144.PDF>


















⁸³ <http://www.nngroup.com/articles/personalization-is-over-rated/>

⁸⁴ <http://www.nngroup.com/books/designing-web-usability/>

⁸⁵ <http://www.cs.bilkent.edu.tr/~guvenir/publications/TAINN01-AOAG.pdf>

⁸⁶ <http://www.sciencedirect.com/science/article/pii/S0306457301000437>

⁸⁷ <http://dl.acm.org/citation.cfm?id=1228174>

- Park, Jong Soo, Ming-Syan Chen y Philip S. Yu (1995). «An effective hash-based algorithm for mining association rules». En: *ACM SIGMOD Record* 24.2. 
⁸⁸, págs. 175-186 (vid. págs. 54, 81).
- (1997). «Using a Hash-Based Method with Transaction Trimming for Mining Association Rules». En: *Knowledge and Data Engineering* 9.5. 
⁸⁹, págs. 813-825 (vid. págs. 54, 118).
- Peñarrubia, Arturo, Antonio Fernández-Caballero y Pascual González (2004). «Portales Web Adaptativos: Una Propuesta de Futuro». En: *Actas del V Congreso Internacional Interacción Persona-Ordenador*. 
⁹⁰ (vid. pág. 40).
- Perkowitz, Mike y Oren Etzioni (1997a). *Adaptive Sites: Automatically Learning from User Access Patterns*. Inf. téc. TR-97-03-01. 
⁹¹. University of Washington Dept. of Computer Science (vid. pág. 17).
- (1997b). «Adaptive Web Sites: an AI Challenge». En: *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*. 
⁹², págs. 16-23 (vid. pág. 17).
- (1998). «Adaptive Web Sites: Automatically Synthesizing Web Pages». En: *Proc. of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI'98)*. 
⁹³. Madison, Wisconsin, United States: American Association for Artificial Intelligence, págs. 727-732 (vid. págs. 17, 18, 36).
- (1999a). «Adaptive Web Sites: Conceptual Cluster Mining». En: *IJCAI*. 
⁹⁴, págs. 264-269 (vid. págs. 18, 35, 36).
- (1999b). «Towards Adaptive Web Sites: Conceptual Framework and Case Study». En: *Computer Networks* 31.11-16. 
⁹⁵, págs. 1245-1258 (vid. págs. 18, 36).
- (2000a). «Adaptive Web Sites». En: *Communications of the ACM* 43.8. ⁹⁶, págs. 152-158 (vid. pág. 17).

⁸⁸ <http://user.it.uu.se/~kostis/Teaching/DM-01/Handouts/PCY.pdf>

⁸⁹ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.35.5196&rep=rep1&type=pdf>

⁹⁰ <http://aipo.es/articulos/3/45.pdf>

⁹¹ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.29.4426&rep=rep1&type=pdf>















⁹² <http://www.cs.washington.edu/research/adaptive/papers/ijcai97.pdf>

⁹³ <http://homes.cs.washington.edu/~etzioni/papers/aaai98.pdf>

⁹⁴ <https://www.cs.washington.edu/research/adaptive/papers/ijcai99.pdf>

⁹⁵ http://homes.cs.washington.edu/~etzioni/papers/perkowitz_www8.pdf

⁹⁶ <http://cacm.acm.org/magazines/2000/8/7597-adaptive-web-sites/abstract>

- Perkowitz, Mike y Oren Etzioni (2000b). «Towards Adaptive Web Sites: Conceptual Framework and Case Study». En: *Artificial Intelligence* 118.1-2. ⁹⁷, págs. 245-275 (vid. págs. 18, 35, 36).
- Pitkow, James (1997). «In search of reliable usage data on the WWW». En: *Computer Networks and ISDN Systems* 29.8-13. ⁹⁸, págs. 1343-1355 (vid. págs. 21, 25).
- Rácz, Balázs, Ferenc Bodon y Lars Schmidt-Thieme (2005). «On benchmarking frequent itemset mining algorithms: from measurement to analysis». En: *OSDM '05: Proceedings of the 1st international workshop on open source data mining*. ⁹⁹. Chicago, Illinois: ACM, págs. 36-45 (vid. pág. 92).
- Ritu y Jitender Arora (2014). «Intensification of Execution of Frequent Item-Set Algorithms». En: *International Journal of Recent Development in Engineering and Technology (IJRDET)* 2 (6). ¹⁰⁰ (vid. pág. 116).
- Rokach, Lior y Oded Maimon (2014). *Data Mining with Decision Trees: Theory and Applications*. 2nd. Vol. 81. Series in Machine Perception and Artificial Intelligence. ¹⁰¹. World Scientific Publishing Co. Pte. Ltd. (vid. págs. 14, 35).
- Rozenberg, Boris y Ehud Gudes (2006). «Association rules mining in vertically partitioned databases». En: *Data & Knowledge Engineering* 59.2. ¹⁰², págs. 378-396 (vid. págs. 19, 23).
- Sadhasivam, Kanimozhi Selvi Chenniangirivalsu y Tamilarasi Angamuthu (2011). «Mining Rare Itemset with Automated Support Thresholds». En: *Journal of Computer Science* 7.3. ¹⁰³, págs. 394-399 (vid. pág. 115).
- Saglam, Burcu, F. Sibel Salman, Serpil Sayin y Metin Türkay (2006). «A mixed-integer programming approach to the clustering problem with an application in customer segmentation». En: *European Journal of Operational Research* 173.3. ¹⁰⁴, págs. 866-879 (vid. pág. 35).
- Sahoo, Jayakrushna, Ashok Kumar Das y A. Goswami (2014). «An Algorithm for Mining High Utility Closed Itemsets and Generators». En: *ArXiv e-prints*. ¹⁰⁵ (vid. pág. 116).

⁹⁷ <http://www.sciencedirect.com/science/article/pii/S0004370299000983>

⁹⁸ <https://www.ischool.utexas.edu/~i385df04/readings/pitkow%281997%29-search.pdf>

⁹⁹ <http://www.cs.bme.hu/~bodon/kozok/papers/racz05benchmarking.pdf>

¹⁰⁰ http://www.ijrdet.com/files/Volume2Issue6/IJRDET_0614_08.pdf
















¹⁰¹ <http://www.worldscientific.com/worldscibooks/10.1142/9097>

¹⁰² <http://www.sciencedirect.com/science/article/pii/S0169023X05001461>

¹⁰³ <http://thescpub.com/PDF/jccsp.2011.394.399.pdf>

¹⁰⁴ <http://www.sciencedirect.com/science/article/pii/S0377221705006879>

¹⁰⁵ <http://arxiv.org/pdf/1410.2988v1.pdf>

- Savasere, Ashok, Edward Omiecinski y Shamkant B. Navathe (1995). «An Efficient Algorithm for Mining Association Rules in Large Databases». En: *The VLDB Journal*. Ed. por Umeshwar Dayal, Peter M. D. Gray y Shojiro Nishio.  ¹⁰⁶. Morgan Kaufmann, págs. 432-444 (vid. págs. 54, 83).
- Schafer, J. Ben, Joseph A. Konstan y John Riedl (2001). «E-Commerce Recommendation Applications». En: *Data Mining and Knowledge Discovery* 5.1/2.  ¹⁰⁷, págs. 115-153 (vid. págs. 16, 36).
- Scime, Anthony (2004). *Web Mining: Applications and Techniques*. ¹⁰⁸. IGI Global (vid. pág. 21).
- Srikant, Ramakrishnan y Rakesh Agrawal (1996). «Mining Sequential Patterns: Generalizations and Performance Improvements». En: *Proceedings of the 5th International Conference on Extending Database Technology*.  ¹⁰⁹, págs. 3-17 (vid. págs. 37, 41).
- (1997). «Mining generalized association rules». En: *Future Generation Computer Systems* 13.2-3.  ¹¹⁰, págs. 161-180 (vid. págs. 54, 86).
- Srikant, Ramakrishnan, Quoc Vu y Rakesh Agrawal (1997). «Mining Association Rules with Item Constraints». En: *Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining*.  ¹¹¹ (vid. pág. 87).
- Srivastava, Jaideep, Robert Cooley, Mukund Deshpande y Pang-Ning Tan (2000). «Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data». En: *SIGKDD Explorations* 1.2.  ¹¹², págs. 12-23 (vid. pág. 21).
- Su, Zhong, Qiang Yang, Ye Lu y Hong-Jiang Zhang (2000). «WhatNext: A Prediction System for Web Requests using *N*-gram Sequence Models». En: *Proc. of the First International Conference on Web Information Systems Engineering (WISE'00)*.  ¹¹³. Washington, DC, USA: IEEE Computer Society, págs. 200-207 (vid. págs. 37, 39).
- Sun, Xiaoming, Zheng Chen, Liu Wenyin y Wei-Ying Ma (2002). «Intention Modeling for Web Navigation». En: *Proc. of International Conference on Intelli-*

¹⁰⁶ <http://www.vldb.org/conf/1995/P432.PDF>

¹⁰⁷ <http://www.cs.umd.edu/~samir/498/schafer01ecommerce.pdf>

¹⁰⁸ <http://www.igi-global.com/book/web-mining-applications-techniques/1045>













¹⁰⁹ <http://dm.kaist.ac.kr/kse525/resources/papers/edbt96gsp.pdf>

¹¹⁰ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.40.7602&rep=rep1&type=pdf>

¹¹¹ http://www.rsrikant.com/papers/kdd97_const.pdf

¹¹² <http://nlp.uned.es/WebMining/Tema5.Usos/srivastava2000.pdf>

¹¹³ http://research.microsoft.com/pubs/68790/prediction_web_requests.pdf

- gent Information Technology.  ¹¹⁴. ICIT2002. Beijing, China, págs. 107-112 (vid. págs. 37, 39).
- Suzuki, Einoshin (2004). «Discovering interesting exception rules with rule pair». En: *Proceedings of the ECML/PKDD Workshop on Advances in Inductive Rule Learning*. Ed. por J. Fuernkranz.  ¹¹⁵, págs. 163-178 (vid. pág. 116).
- Thabtah, Fadi, Peter Cowling y Suhel Hamoud (2006). «Improving rule sorting, predictive accuracy and training time in associative classification». En: *Expert Systems with Applications* 31.2.  ¹¹⁶ ¹¹⁷, págs. 414-426 (vid. págs. 36, 51, 116).
- Toivonen, Hannu (1996). «Sampling Large Databases for Association Rules». En: *In Proc. 1996 Int. Conf. Very Large Data Bases*. Ed. por T. M. Vijayaragaman, Alejandro P. Buchmann, C. Mohan y Nandlal L. Sarda.  ¹¹⁸. Morgan Kaufman, págs. 134-145 (vid. pág. 84).
- Tsay, Yuh-Jiuan, Tain-Jung Hsu y Jing-Rung Yu (2009). «FIUT: A new method for mining frequent itemsets». En: *Information Sciences* 179.11. ¹¹⁹, págs. 1724-1737 (vid. págs. 35, 92).
- Tseng, Ming-Cheng y Wen-Yang Lin (2007). «Efficient mining of generalized association rules with non-uniform minimum support». En: *Data & Knowledge Engineering* 62.1, págs. 41-64 (vid. pág. 118).
- Villena, Julio, José Carlos González, Emma Barceló y Juan Ramón Velasco (2002). «Minería de uso de la web mediante huellas y sesiones». En: *Actas VIIth Iberoamerican Conference on Artificial Intelligence*.  ¹²⁰. Iberamia. España, págs. 69-78 (vid. pág. 25).
- W3CLog (1996). *Extended Log File Format*. Inglés. W3C (vid. págs. 19, 22).
- W3CP3P (2002). *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. Inglés. W3C (vid. pág. 19).
- Wang, YanboJ., Qin Xin y Frans Coenen (2008). «Mining Efficiently Significant Classification Association Rules». English. En: *Data Mining: Foundations and Practice*. Ed. por TsauYoung Lin, Ying Xie, Anita Wasilewska y Churn-

¹¹⁴ <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=E0FE28C2CB01B68C1D811E4044CEE9C8?doi=10.1.1.12.9782&rep=rep1&type=pdf>

¹¹⁵ <http://www.ke.tu-darmstadt.de/events/ECML-PKDD-04-WS/Proceedings/suzuki.pdf>














¹¹⁶ <http://scim.brad.ac.uk/staff/pdf/picowlin/ThabtahCowlingHamoud2006.pdf>

¹¹⁷ <http://www.sciencedirect.com/science/article/pii/S0957417405002290>

¹¹⁸ <http://www.vldb.org/conf/1996/P134.PDF>

¹¹⁹ <http://www.sciencedirect.com/science/article/B6V0C-4VDS8JY-1/2/4832c16c758c95802af049f13cb21478>

¹²⁰ <http://www.lsi.us.es/iberamia2002/confman/SUBMISSIONS/184-u1aaded11e.pdf>

- Jung Liau. Vol. 118. *Studies in Computational Intelligence*.  ¹²¹. Springer Berlin Heidelberg, págs. 443-467 (vid. pág. 116).
- Wu, Cheng Wei, Bai-En Shie, Vincent S. Tseng y Philip S. Yu (2012). «Mining top-K High Utility Itemsets». En: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12.  ¹²². Beijing, China: ACM, págs. 78-86 (vid. pág. 116).
- Yun, Hyunyon, Danshim Ha, Buhyun Hwang y Keun Ho Ryu (2003). «Mining association rules on significant rare data using relative support». En: *J. Syst. Softw.* 67.3. ¹²³, págs. 181-191 (vid. pág. 118).
- Zaki, Mohammed Javeed y Bart Goethals, eds. (2003). *Proceedings of FIMI'03 Workshop on Frequent Itemset Mining Implementations*.  ¹²⁴ (vid. pág. 70).
- Zaki, Mohammed Javeed, Srinivasan Parthasarathy, Mitsunori Ogihara y Wei Li (1997). «New Algorithms for Fast Discovery of Association Rules». En: *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*. Ed. por David Heckerman, Heikki Mannila, Daryl Pregibon, Ramasamy Uthurusamy y Menlo Park.  ¹²⁵  ¹²⁶. Rochester, NY, USA: AAAI Press, págs. 283-286 (vid. págs. 54, 88).
- Zhao, Qiankun y Sourav S. Bhowmick (2003). *Association Rule Mining: A Survey*. Inf. téc. TR116.  ¹²⁷. School of Computer Engineering. University of Nanyang Technological (vid. pág. 69).

¹²¹ <http://www.ii.uib.no/~xin/dmBookChap2007.pdf>

¹²² <http://wan.poly.edu/KDD2012/docs/p78.pdf>

¹²³ <http://www.sciencedirect.com/science/article/pii/S0164121202001280>

¹²⁴ https://www.academia.edu/2617201/Proceedings_of_FIMI03_Workshop_on_Frequent_Itemset_Mining_Implementations

¹²⁵ <http://web.cse.ohio-state.edu/dmrl/papers/kdd97.pdf>

¹²⁶ <http://www.cs.rpi.edu/~zaki/PaperDir/URTR651.pdf>

¹²⁷ <https://www.lri.fr/~antoine/Courses/Master-ISI/Regle-association.pdf>