

Hola, tesis doctoral en informática !!!

Índice general

1. Sistema de Recomendación Web	7
1.1. Personalización de la Web	7
1.2. Minería de Uso Web	7
1.2.1. DATOS	7
1.2.2. Selección	7
1.2.3. Preproceso	7
1.2.4. Transformación	7
1.2.5. Minería de Datos (DM)	17
1.2.6. Evaluación e Integración	17
1.2.7. CONOCIMIENTO	17
1.3. Minería de Datos	17
1.3.1. Mapas de Navegación Web	17
1.3.2. Reglas de Asociación	17
1.4. Publicaciones	28
1.4.1. Actas de HCII'05	28
1.4.2. Actas de Interacción'05	28
1.4.3. Actas de SICO'05	28
2. Minería de Reglas de Asociación	29
2.1. Conceptos básicos	31

2.1.1.	Tipo de Datos	31
2.1.2.	Primeros algoritmos	31
2.1.3.	Formato de D	31
2.1.4.	Fases de ARM	31
2.2.	<i>Minería de Itemsets Frecuentes</i>	31
2.2.1.	Algoritmos y estructuras	31
2.2.2.	Evaluación de diferentes implementaciones	31
2.3.	Generación de <i>reglas de asociación</i>	31
2.3.1.	genrules()	31
2.3.2.	Apriori2	32
2.4.	El <i>Ítem Raro</i>	32
2.4.1.	Estudio de <i>Ítems Raros</i>	32
2.4.2.	<i>Reglas de Oportunidad</i>	32
2.5.	Publicaciones	32
2.5.1.	HCII'07	32
2.5.2.	ESWA, vol. 35(3) 2008	32
2.5.3.	Interacción'10	32
3.	<i>ARM para Clasificación</i>	33
3.1.	Conceptos básicos	35
3.1.1.	Tal	36
3.2.	<i>Catálogo</i>	36
3.3.	<i>Catálogo Comprimido</i>	37
3.3.1.	Lectura de catálogos comprimidos	38
3.4.	<i>Catálogo Completo</i>	38
3.4.1.	Colecciones de <i>Catálogos Completos</i>	38
3.5.	Datos missing	38
3.6.	Publicaciones	41
3.6.1.	[...]	42
3.6.2.	[...]	42
4.	Conclusiones y Trabajo Futuro	43
A.	Notación	47
A.1.	Sistemas de Recomendación Web	47
A.2.	Minería de Reglas de Asociación	47
A.3.	Catálogos	47

	5
B. Código	49
B.1. Sistemas de Recomendación Web	49
B.2. Minería de Reglas de Asociación	49
B.3. Catálogos	49
C. Datos utilizados	53
C.1. Sistemas de Recomendación Web	53
C.2. Minería de Reglas de Asociación	53
C.3. Catálogos	53
Índice de figuras	55
Índice de cuadros	57
List of Theorems	59
Índice de definiciones	60
Índice de listados	61
Índice alfabético	63
D. Sobre la bibliografía	65
Bibliografía	71

Motivación

Este...

SRW

Los SRW surgen de...

Catálogos

Un catálogo es...

Sistema de Recomendación Web

En ...

1.1. Personalización de la Web

1.2. Minería de Uso Web

1.2.1. DATOS

1.2.2. Selección

1.2.3. Preproceso

1.2.4. Transformación

Una vez obtenidos los llamados Preprocessed Data en la figura ?? 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 múltiples 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 inter-

vienen 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 1.1.

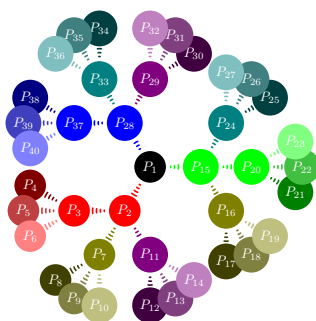


Figura 1.1: Estructura del sitio web

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 1.1 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, siendo 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 hemos 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()*

```

 $t_0 = t_{fin} = t_i$ ;
insert (IPi,  $t_0$ ,  $t_{fin}$ ,  $p_i$ ) into sesionesabiertas;

```

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  $t_0$  and  $t_{fin}$  from sesion;
insert sesion into sesiones;
delete sesion from sesionesabiertas;

```

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 1.2: 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 esta-

mos 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 pagina 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)**1.2.6. Evaluación e Integración****1.2.7. CONOCIMIENTO****1.3. Minería de Datos****1.3.1. Mapas de Navegación Web****1.3.2. Reglas de Asociación**

Las *Reglas de Asociación* fueron introducidas por Agrawal, Imielinski y A. Swami (1993) 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 8 (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.*

$$\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_M\} \quad (1.8)$$

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. 9).

Definición 9 (Transacción). *Una transacción \mathcal{T} es un itemset, un subconjunto de \mathcal{I} , obtenido en determinadas circunstancias.*

$$\mathcal{T} \subseteq \mathcal{I} = \{I_j \mid I_j \in \mathcal{I}, j = 1 \dots k\} \quad (1.9)$$

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

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 informa-

ción 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 11 (*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 ($a_i \cap c_i = \emptyset$).*

$$R_i = \{a_i \rightarrow c_i\} \quad (1.10)$$

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 12 (*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* (Liu, 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 13 (*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.*

$$\text{soporte}(X) = |\mathcal{T}_X| = n_X, \text{ siendo } \mathcal{T}_X = \{\mathcal{T} \in \mathcal{D} / X \in \mathcal{T}\} \quad (1.11)$$

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.

$$\text{soporte}_{\text{relativo}}(X) = \frac{n_X}{|\mathcal{D}|} \quad (1.12)$$

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 14 (*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 15 (*Itemset frecuente*). Un itemset frecuente es un itemset con soporte en \mathcal{D} igual o superior a $\min\text{Sup}$.

Definición 16 (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} .

$$\mathcal{L} = \cup_k \mathcal{L}_k \quad (1.13)$$

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 17 (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} .

$$\text{soporte}(X \rightarrow Y) = \text{soporte}(X \cup Y) = n_{XY} \quad (1.14)$$

$$\text{soporte}_{\text{relativo}}(X \rightarrow Y) = \frac{\text{soporte}(X \cup Y)}{|\mathcal{D}|} = \frac{n_{XY}}{|\mathcal{D}|} \quad (1.15)$$

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 18 (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 .

$$\text{confianza}(X \rightarrow Y) = \frac{\text{soporte}(X \cup Y)}{\text{soporte}(X)} = \frac{n_{XY}}{n_X} \quad (1.16)$$

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 19 (*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, 1993), *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* (Lin y Kedem, 2002) o *FP-Growth* (Han, Pei y Yin, 2000; 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 *item-*

sets 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 20 (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
     $\mathcal{C}_k = apriori - gen(\mathcal{L}_{k-1});$ 
    forall transactions  $t \in \mathcal{D}$  do begin
         $\mathcal{C}_t = subset(\mathcal{C}_k, t);$ 
        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 *apriori - 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 *apriori – 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 *apriori – 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 $\mathcal{L}_{k-1}\mathcal{C}_k$ 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 $\mathcal{L}_{k-1}\mathcal{C}_k$ 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

11) `end`

Técnicas de *ARM* aplicadas a un *Sistema de Recomendación Web*

1.4. Publicaciones

1.4.1. Actas de HCII'05

1.4.2. Actas de Interacción'05

1.4.3. Actas de SICO'05

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

2.1.1. Tipo de Datos

...

2.1.2. Primeros algoritmos

...

2.1.3. Formato de D

...

2.1.4. Fases de ARM

...

2.2. *Minería de Itemsets Frecuentes*

Minería de Itemsets Frecuentes ...

2.2.1. Algoritmos y estructuras

...

2.2.2. Evaluación de diferentes implementaciones

...

2.3. Generación de *reglas de asociación*

...

2.3.1. `genrules()`

...

2.3.2. Apriori2

...

2.4. El *Ítem Raro*

...

2.4.1. Estudio de *Ítems Raros*

...

2.4.2. *Reglas de Oportunidad*

...

2.5. Publicaciones

...

2.5.1. HCII'07

...

2.5.2. ESWA, vol. 35(3) 2008

...

2.5.3. Interacción'10

...

ARM para Clasificación

ABIERTO

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 *soprote* propuesto por Sadhasivam y Angamuthu (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 *soprote* 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.

Esta reflexión nos llamó la atención en ficheros tan «pequeños» como mushroom.dat o chess.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 de *clasificación*.

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

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) (Liu, Hsu y Ma, 1998; Bayardo, 1998; 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; Wang, Xin y Coenen, 2008; Li y 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 *transacciones* dedujimos que el 2 aparecería en las restantes 8 124 - X XXX. 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 $Y\ YYY$ transacciones, en $Z\ ZZZ$ ocasiones junto al ítem 1, luego en las restantes $Y\ YYY - Z\ ZZZ$ 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. Conceptos básicos

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.

Definición 21 (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 -túpla, un conjunto de n valores obtenidos ordenadamente

de los atributos A_i .

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

3.1.1. Tal

Definición 22 (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 -túpla, 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 23 (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 -túpla, 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 24 (Catálogo). Un catálogo es un conjunto de N registros diferentes.

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

3.2. Catálogo

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.

Acabo de descubrir LUCS, que discretiza las colecciones de UCI y

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 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 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. *Catálogo Comprimido*

Aprovechando las restricciones implícitas de los catálogos como mushroom...

3.3.1. Lectura de catálogos comprimidos

3.4. Catálogo Completo

ABIERTO

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

Al seguir trabajando con `mushroom.dat` encontramos otra versión del mismo dataset en KEEL

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

Definición 25 (*Catálogo Completo*). Un Catálogo Completo es un catálogo sin incertidumbre.

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

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. . .

3.5. Datos missing

ABIERTO

`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.

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 3.3 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 ignorados 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.

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'12 comenzamos a publicar nuestros resultados sobre las colecciones de *transacciones* estructuradas mediante *catálogos*. 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.

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.

  Scopus  2  3  4  5

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

² https://www.researchgate.net/publication/266652926_Efficient_analysis_of_transactions_to_improve_Web_Recommendations Revisado en diciembre de 2015.

³ <https://www.deepdyve.com/lp/association-for-computing-machinery/efficient-analysis-of-transactions-to-improve-web-recommendations-mBUBBN> Revisado en diciembre de 2015.

⁴ https://github.com/EnriqueLazcorreta/tesis-0/blob/master/bib/nuestros/LazcorretaBotellaFCaballero_EfficientAnalysisOfTransactionsWebRecommendations_2012.pdf Revisado en diciembre de 2015.

⁵ https://github.com/EnriqueLazcorreta/tesis-0/blob/master/bib/nuestros/LazcorretaBotellaFCaballero_AnalisisEficienteDeTransacciones_12_Presentacion.pdf Revisado en diciembre de 2015.

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, 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.

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'16, que se celebrará en septiembre de 2016 en Salamanca.

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 los *catálogos comprimidos* expuestos en la sección ???. 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 ?? me exigían usar muchos recur-

sos 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".



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 GuardaC1();

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

1.1. Estructura del sitio web	10
1.2. Transformación	14

Índice de cuadros

List of Theorems

1.	Definición (<i>Páginas</i> del sitio web)	9
2.	Definición (<i>Enlaces internos</i> del sitio web)	9
3.	Definición (<i>Sesión de navegación</i>)	12
4.	Definición (Conjunto de <i>sesiones de navegación</i>)	13
5.	Definición (<i>Páginas visitadas</i> del sitio web)	15
6.	Definición (Tiempo de permanencia en las <i>páginas visitadas</i>) . .	15
7.	Definición (<i>Enlaces internos utilizados</i> en el sitio web)	16
8.	Definición (Población de ítems)	18
9.	Definición (<i>Transacción</i>)	18
10.	Definición (Almacén \mathcal{D})	19
11.	Definición (<i>Regla de Asociación</i>)	20
12.	Definición (Minería de Reglas de Asociación)	20
13.	Definición (<i>Soporte</i> de un <i>itemset</i>)	21
14.	Definición (<i>Soporte mínimo</i>)	22
15.	Definición (<i>Itemset</i> frecuente)	23
16.	Definición (Conjunto de <i>itemsets</i> frecuentes, \mathcal{L})	23
17.	Definición (<i>Soporte</i> de una <i>Regla de Asociación</i>)	23
18.	Definición (<i>Confianza</i> de una <i>Regla de Asociación</i>)	23
19.	Definición (<i>Confianza mínima</i>)	24
20.	Definición (Conjunto de candidatos a k - <i>itemsets</i> frecuentes, \mathcal{C}_k)	25

21.	Definición (Individuo)	35
22.	Definición (Individuo)	36
23.	Definición (Registro)	36
24.	Definición (<i>Catálogo</i>)	36
25.	Definición (<i>Catálogo Completo</i>)	38

Índice de listados

1.1.	Algoritmo de obtención de <i>sesiones de navegación</i>	13
1.2.	Función <i>CreaSesion()</i>	14
1.3.	Función <i>CierraSesion()</i>	14
1.4.	Algoritmo <i>Apriori</i>	25
1.5.	Función <i>apriori – gen</i> : unión	26
1.6.	Función <i>apriori – gen</i> : poda	26
1.7.	Función <i>genrules()</i>	27
B.1.	Cabecera para lectura de ficheros KEEL	49

Índice alfabético

- Bases de Datos (DB), 19
- Clasificación, 18, 19, 21, 33–35, 38, 40, 44
 - \mathcal{A} , atributo, 34, 39
 - Catálogo, 36–41
 - Catálogo Completo, 38–41
 - mushroom.dat, 33–35, 38–40
- Clustering, 8, 9, 30
- Fichero de log (*logfile*), 10–13, 15
- Grafo, 9, 10
- Ítem Raro, 30, 32, 35
 - Dilema del Ítem Raro, 30, 33–35
- Itemset, 18, 20–27, 33, 34
- Knowledge Discovery in Databases (KDD), 9, 19, 29
- Minería de Datos (DM), 8, 11, 14–16, 19, 29, 34
- Minería de Itemsets Frecuentes (FIM), 24, 31, 33, 35
- Minería de Reglas de Asociación (ARM), 17, 19, 20, 22, 24, 28–30, 33–35
 - AIS, 24
 - Apriori, 17, 24, 25, 30
 - Apriori-TID, 24
 - AprioriHybrid, 24
 - Basic, 24
 - Candidate Distribution, 24
 - Clique, 24
 - Count Distribution, 24
 - Cumulate, 24
 - Data Distribution, 24
 - DHP, 24
 - DIC, 24
 - Eclat, 24
 - EstMerge, 24
 - FP-Growth, 24

- MaxClique, 24
- MaxEclat, 24
- MaxMiner, 24
- Partition, 24
- PincerSearch, 24
- RangeApriori, 24
- SETM, 24
- Minería de Reglas de Clasificación Asociativa (CARM), 44
- Minería Web (WM)
 - Minería de Estructura Web (WSM), 10
 - Minería de Uso Web (WUM), 9, 14, 15, 18, 19, 21, 29
- Patrón, 9, 11, 19
- Predicción, 24
- Reglas de Asociación, 17–25, 27, 29–31, 33, 34, 39, 40
 - Antecedente, 20, 23
 - Apriori
 - \mathcal{C} , Conjunto de Candidatos, 25, 26
 - \mathcal{L} , Conjunto de Itemsets Frecuentes, 23, 25–27, 33
 - Confianza, 21, 23–25, 27
 - Consecuente, 20, 23
 - \mathcal{D} , almacén de Transacciones, 19–27, 30, 33–35, 38, 39
 - \mathcal{I} , conjunto de ítems, 18, 22
 - Reglas de Oportunidad, 30, 32
 - Soporte, 21–27, 33, 34
 - Transacción, 17–23, 26, 34, 35, 41
- Regresión, 8
- Secuencia, 10
- Sesión de navegación, 10–16, 19, 21
- Sistema de Recomendación (RS), 30
 - Sistema de Recomendación Web (WRS), 7, 21, 28–30
- World Wide Web (WWW), 12, 29





Sobre la bibliografía

ABIERTO

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 2014. 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.

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 JabRef ² y BibDesk ³ 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 (1993). «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.  ¹. Washington, D.C., United States, págs. 207-216 (vid. págs. 17, 24).
- Agrawal, Rakesh y John C. Shafer (1996). «Parallel Mining of Association Rules». En: *IEEE Trans. on Knowl. and Data Eng.* 8.6.  ², págs. 962-969 (vid. pág. 24).
- 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.  ³. VLDB. Morgan Kaufmann Publishers Inc., págs. 487-499 (vid. págs. 17, 24).
- (1994b). «Fast Algorithms for Mining Association Rules (EXTENDIDO)».  ⁴ (vid. págs. 17, 24, 27, 30).
- Bayardo, Roberto J. (1998). «Efficiently mining long patterns from databases». En: *Proc. of the 1998 ACM-SIGMOD Int. Conf. on Management of Data*.  ⁵, págs. 85-93 (vid. págs. 24, 34).


















¹<http://www.almaden.ibm.com/cs/quest/papers/sigmod93.pdf>

²<http://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/rj10004.pdf>

³<http://www.vldb.org/conf/1994/P487.PDF>

⁴http://rakesh.agrawal-family.com/papers/vldb94apriori_rj.pdf

⁵<http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.107.1120&rep=rep1&type=pdf>

- Borgelt, Christian (2004). «Efficient implementations of Apriori and Eclat». En: *Proc. of the Workshop on Frequent Itemset Mining Implementations*.  ⁶. FIMI'04 (vid. pág. 34).
- 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.  ⁷  ⁸. ACM Press, págs. 255-264 (vid. pág. 24).
- 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.  ⁹, págs. 249-267 (vid. pág. 10).
- Goethals, Bart (2003). *Survey on Frequent Pattern Mining*. Inf. téc.  ¹⁰. HIIT Basic Research Unit; Department of Computer Science; University of Helsinki, pág. 43 (vid. pág. 29).
- 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.  ¹¹. INAP 2001. Springer-Verlag GmbH, págs. 251-264 (vid. pág. 24).
- 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.  ¹². Dallas, Texas, United States: ACM, págs. 1-12 (vid. pág. 24).
- Han, Jiawei, Jian Pei, Yiwen Yin y Runying Mao (2004). *Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*. ¹³ (vid. pág. 24).
- He, Daqing y Ayse Göker (2000). «Detecting Session Boundaries from Web User Logs». En: *Proc. of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*.  ¹⁴ (vid. pág. 12).

⁶ www.borgelt.net/papers/fimi_03.ps.gz

⁷ <http://www.cs.ucla.edu/~zaniolo/czdemo/tsur/Papers/dic-final.ps>

⁸ <http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput695-00/papers/dic.pdf>

⁹ <http://cobweb.cs.uga.edu/~eileen/WebEffectiveness/Papers/bucklinandsismeiro2003.pdf>

















¹⁰ <http://adrem.ua.ac.be/~goethals/software/survey.pdf> <http://adrem.ua.ac.be/~goethals/software/survey.pdf>

¹¹ <http://www.informatics.indiana.edu/dgroth/research/publications/groth-inap01.pdf>

¹² http://cs.sungshin.ac.kr/~jpark/HOME/References/han_sigmod00.pdf

¹³ http://www.cs.sfu.ca/~jpei/publications/dami03_fpgrowth.pdf

¹⁴ <http://www.pitt.edu/~dah44//docs/he00detecting.pdf>

- Hipp, Jochen, Ulrich Güntzer y Gholamreza Nakhaeizadeh (2000). «Algorithms for Association Rule Mining - A General Survey and Comparison». En: *SIGKDD Explorations* 2.1.  ¹⁵, págs. 58-64 (vid. pág. 29).
- Houtsma, Maurice A. W. y Arun N. Swami (1993). *Set-Oriented Mining of Association Rules in Relational Databases*. Inf. téc.  ¹⁶. IBM Almaden Research Center (vid. pág. 24).
- 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.  ¹⁷, págs. 1290-1303 (vid. pág. 12).
- (2004b). *Dynamic Web Log Session Identification with Statistical Language Models*.  ¹⁸ (vid. pág. 12).
- 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*.  ¹⁹ (vid. pág. 12).
- 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)*.  ²⁰, págs. 1-6 (vid. pág. 21).
- Li, Haifeng y Ning Zhang (2010). «Mining maximal frequent itemsets on graphics processors». En: *FSKD*. ²¹ ²², págs. 1461-1464 (vid. pág. 34).
- Lichman, M. (2013). *UCI Machine Learning Repository*. ²³ (vid. pág. 35).
- 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. ²⁴, págs. 553-566 (vid. pág. 24).

¹⁵ <http://www.kdd.org/sites/default/files/issues/2-1-2000-06/hipp.pdf>

¹⁶ <http://doc.utwente.nl/19441/1/00380413.pdf>

¹⁷ <http://www.yorku.ca/jhuang/paper/jasist04.pdf>

¹⁸ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.9.6602&rep=rep1&type=pdf>

¹⁹ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.12.8127&rep=rep1&type=pdf>

²⁰ http://iats09.karabuk.edu.tr/press/bildiriler_pdf/IATS09_01-01_105.pdf

²¹ <http://dx.doi.org/10.1109/FSKD.2010.5569206>

²² [http://lmgfry.com/?q="Miningmaximalfrequentitemsetsongraphicsprocessors"](http://lmgfry.com/?q=) Revisado en diciembre de 2015.

²³ <http://archive.ics.uci.edu/ml>

²⁴ <http://dl.acm.org/citation.cfm?id=628231>

- Liu, Bing, Wynne Hsu y Yiming Ma (1998). «Integrating Classification and Association Rule Mining». En: *Knowledge Discovery and Data Mining*.  ²⁵, págs. 80-86 (vid. págs. 21, 34).
- Malik, Kuldeep Singh y Neeraj Raheja (2013). «Improving performance of Frequent Itemset algorithm». En: *International Journal of Research in Engineering & Applied Sciences* 3.3.  ²⁶, págs. 168-177 (vid. pág. 34).
- 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.  ²⁷, págs. 175-186 (vid. pág. 24).
- (1997). «Using a Hash-Based Method with Transaction Trimming for Mining Association Rules». En: *Knowledge and Data Engineering* 9.5.  ²⁸, págs. 813-825 (vid. pág. 24).
- 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).  ²⁹ (vid. pág. 34).
- Sadhasivam, Kanimozhi Selvi Chenniangirivalasu y Tamilarasi Angamuthu (2011). «Mining Rare Itemset with Automated Support Thresholds». En: *Journal of Computer Science* 7.3.  ³⁰, págs. 394-399 (vid. pág. 33).
- Sahoo, Jayakrushna, Ashok Kumar Das y A. Goswami (2014). «An Algorithm for Mining High Utility Closed Itemsets and Generators». En: *ArXiv e-prints*.  ³¹ (vid. pág. 34).
- 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.  ³². Morgan Kaufmann, págs. 432-444 (vid. pág. 24).
- Srikant, Ramakrishnan y Rakesh Agrawal (1997). «Mining generalized association rules». En: *Future Generation Computer Systems* 13.2–3.  ³³, págs. 161-180 (vid. pág. 24).

²⁵ <http://www.cs.uiuc.edu/class/fa05/cs591han/papers/bliu98.pdf>

²⁶ <http://www.euroasiapub.org/IJREAS/mar2013/18.pdf>

²⁷ <http://user.it.uu.se/~kostis/Teaching/DM-01/Handouts/PCY.pdf>

²⁸ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.35.5196&rep=rep1&type=pdf>
















²⁹ http://www.ijrdet.com/files/Volume2Issue6/IJRDET_0614_08.pdf

³⁰ <http://thescipub.com/PDF/jcssp.2011.394.399.pdf>

³¹ <http://arxiv.org/pdf/1410.2988v1.pdf>

³² <http://www.vldb.org/conf/1995/P432.PDF>

³³ <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.40.7602&rep=rep1&type=pdf>

- 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.  ³⁴, págs. 163-178 (vid. pág. 34).
- 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.  ³⁵ ³⁶, págs. 414-426 (vid. págs. 21, 34).
- 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-Jung Liao. Vol. 118. Studies in Computational Intelligence.  ³⁷. Springer Berlin Heidelberg, págs. 443-467 (vid. pág. 34).
- 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.  ³⁸. Beijing, China: ACM, págs. 78-86 (vid. pág. 34).
- Zaki, Mohammed Javeed y Bart Goethals, eds. (2003). *Proceedings of FIMI'03 Workshop on Frequent Itemset Mining Implementations*.  ³⁹ (vid. pág. 30).
- 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.  ⁴⁰  ⁴¹. Rochester, NY, USA: AAAI Press, págs. 283-286 (vid. pág. 24).
- Zhao, Qiankun y Sourav S. Bhowmick (2003). *Association Rule Mining: A Survey*. Inf. téc. TR116.  ⁴². School of Computer Engineering. University of Nanyang Technological (vid. pág. 29).

³⁴ <http://www.ke.tu-darmstadt.de/events/ECML-PKDD-04-WS/Proceedings/suzuki.pdf>

³⁵ <http://scim.brad.ac.uk/staff/pdf/picowlin/ThabtahCowlingHamoud2006.pdf>

³⁶ <http://www.sciencedirect.com/science/article/pii/S0957417405002290>

³⁷ <http://www.ii.uib.no/~xin/dmBookChap2007.pdf>

³⁸ <http://wan.poly.edu/KDD2012/docs/p78.pdf>

³⁹ https://www.academia.edu/2617201/Proceedings_of_FIMI03_Workshop_on_Frequent_Itemset_Mining_Implementations

⁴⁰ <http://web.cse.ohio-state.edu/dmrl/papers/kdd97.pdf>

⁴¹ <http://www.cs.rpi.edu/~zaki/PaperDir/URTR651.pdf>

⁴² <https://www.lri.fr/~antoine/Courses/Master-ISI/Regle-association.pdf>