Índice general

Capítulo 6

MARCO TEORICO

6.1. El Proceso de Descubrimiento de Conocimiento en Bases de Datos - DCBD

El proceso de DCBD es el proceso que utiliza métodos de minería de datos (algoritmos) para extraer (identificar) patrones que evaluados e interpretados, de acuerdo a las especificaciones de medidas y umbrales, usando una base de datos con alguna selección, preprocesamiento, muestreo y transformación, se obtiene lo que se piensa es conocimiento [13].

El proceso de DCBD es interactivo e iterativo, involucra numerosos pasos con la intervención del usuario en la toma de muchas decisiones y se resumen en las siguientes etapas:

- Selección.
- Preprocesamiento / Data cleaning.
- Transformación / Reducción.
- Minería de Datos (Data Mining).
- Interpretación / evaluación.

6.2. Arquitecturas de Integración de las Herramientas DCBD con un SGBD

Las arquitecturas de integración de las herramientas DCBD con un SGBD se pueden ubicar en una de tres tipos: herramientas débilmente acopladas, medianamente acopladas y fuertemente acopladas con un SGBD [46].

Una arquitectura es débilmente acoplada cuando los algoritmos de Minería de Datos y demás componentes se encuentran en una capa externa al SGBD, por fuera del núcleo y su integración con este se hace a partir de una interfaz [46].

Una arquitectura es medianamente acoplada cuando ciertas tareas y algoritmos de descubrimiento de patrones se encuentran formando parte del SGBD mediante procedimientos almacenados o funciones definidas por el usuario [46].

Una arquitectura es fuertemente acoplada cuando la totalidad de las tareas y algoritmos de descubrimiento de patrones forman parte del SGBD como una operación primitiva, dotándolo de las capacidades de descubrimiento de conocimiento y posibilitándolo para desarrollar aplicaciones de este tipo [46].

Por otra parte, de acuerdo a las tareas que desarrollen, las herramientas DCBD se clasifican en tres grupos: herramientas genéricas de tareas sencillas, herramientas genéricas de tareas múltiples y herramientas de dominio específico.

Las Herramientas genéricas de tareas sencillas principalmente soportan solamente la etapa de minería de datos en el proceso de DCBD y requieren un pre y un post procesamiento de los datos. El usuario final de estas herramientas es típicamente un consultor o un desarrollador quien podría integrarlas con otros módulos como parte de una completa aplicación.

Las Herramientas genéricas de tareas múltiples realizan una variedad de tareas de descubrimiento, típicamente combinando clasificación, asociación, visualización, clustering, entre otros. Soportan diferentes etapas del proceso de DCBD. El usuario final de estas herramientas es un analista quien entiende la manipulación de los datos.

Finalmente, las Herramientas de dominio específico, soportan descubrimiento solamente en un dominio específico y hablan el lenguaje del usuario final, quien necesita conocer muy poco sobre el proceso de análisis.

6.3. Implementación de Herramientas DCBD débilmente acopladas con un SGBD

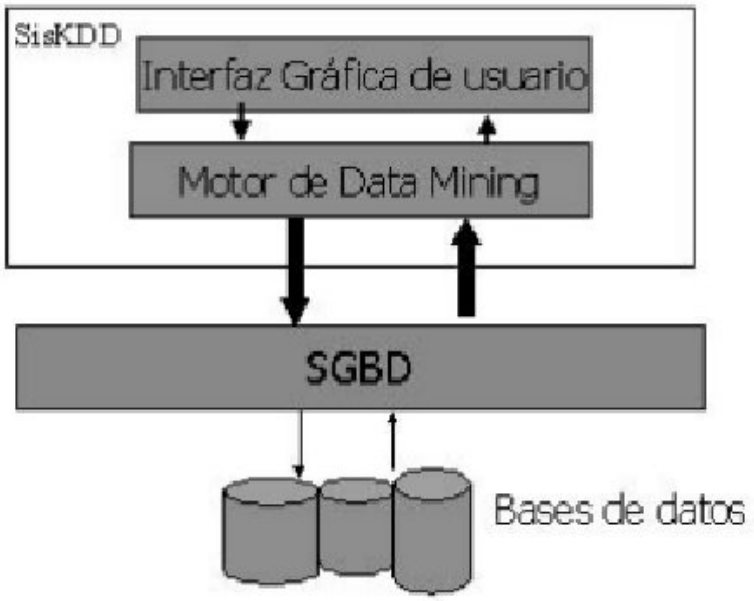

Figura 6.1: Arquitectura DCBD débilmente acoplada

La implementación de herramientas DCBD débilmente acopladas con un SGBD se hace a través de SQL embebido en el lenguaje anfitrión del motor de minería de datos [5]. Los datos residen en el SGBD y son leídos registro por registro a través de ODBC, JDBC o de una interfaz de cursores SQL. La ventaja de esta arquitectura es su portabilidad. Sus principales desventajas son la escalabilidad y el rendimiento. El problema de escalabilidad consiste en que las herramientas y aplicaciones bajo este tipo de arquitectura, cargan todo el conjunto de datos en memoria, lo que las limita para el manejo de grandes cantidades de datos. El bajo rendimiento se debe a que los registros son copiados uno por uno del espacio de direccionamiento de la base de datos al espacio de direccionamiento de la aplicación de minería de datos [8, 24] y estas operaciones de entrada/salida, cuando se manejan grandes volúmenes de datos, son bastante costosas, a pesar de la optimización de lectura por bloques presente en muchos SGBD (Oracle, DB2, Informix, PostgreSQL.) donde un bloque de tuplas puede ser leído al tiempo (figura 1).

6.4. Algoritmos implementados en TariyKDD

6.4.1. Apriori

La notación del algoritmo Apriori [6] es la siguiente:

Cuadro 6.1: Notación algoritmo Apriori

k-**itemset**	**Un itemset con** k **items**
L_k	Conjunto de itemsets frecuentes k (Aquellos con soporte mínimo).
C_k	Conjunto de itemsets candidatos k (Itemstes potencialmente frecuentes)

A continuación se muestra el algoritmo Apriori:

L_1 = {Conjunto de itemsets frecuentes 1 }
for ($k = 2$; $L_{k-1} \neq 0$; $k++$) do begin
    C_k = apriori-gen(L_{k-1}) // Nuevos candidatos
    forall *transacciones* $t \in D$ do begin
        C_t = subconjunto (C_k, t); // Candidatos en t
        forall *candidatos* $c \in C_t$ do
            $c.cont++$;
    end
    L_k = { $c \in C_k | c.count \geq minsup$ }
end Answer $\cup_k L_k$

La primera pasada del algoritmo cuenta las ocurrencias de los items en todo el conjunto de datos para determinar los itemsets frecuentes 1. Los subsecuentes pasos del algoritmo son basicamente dos, primero, los itemsets frecuentes L_{k-1} encontrados en la pasada $(k-1)$ son usados para generar los itemsets candidatos C_k, usando la función apriori-gen descrita en la siguiente subsección. Y segundo se cuenta el soporte de los itemsets candidatos C_k a través de un nuevo recorrido a la base de datos. Se realiza el mismo proceso hasta que no se encuentren más itemsets frecuentes.

Generación de candidatos en Apriori

La función apriori-gen toma como argumento L_{k-1}, o sea todos los itemsets frecuentes $(k-1)$. La función trabaja de la siguiente manera:

insert into C_k
select $p.item_1$, $p.item_2, ...,$ $p.item_{k-1}$, $q.item_{k-1}$
from L_{k-1} p, L_{k-1} q
where $p.item_1 = q.item_1, ...,$ $p.item_{k-2} = q.item_{k-2}$,
$p.item_{k-1} q.item_{k-1}$;

A continuación, en el paso de poda, se borran todos los itemsets $c \in C_k$ tal que algún subconjunto $(k-1)$ de c no este en L_{k-1}:

forall *itemsets* $c \in C_k$ do
forall subconjuntos $(k-1)$ s de c do
if $(s \notin L_{k-1})$ then
delete c from C_k;

6.4.2. FPGrowth

Como se puede ver en [24] la heurística utilizada por Apriori logra buenos resultados, ganados por (posiblemente) la reducción del tamaño del conjunto de candidatos. Sin embargo, en situaciones con largos patrones, soportes demasiado pequeños, un algoritmo tipo Apriori podría sufrir de dos costos no triviales:

- Es costoso administrar un gran número de conjuntos candidatos. Por ejemplo, si hay 10^4 Itemsets Frecuentes, el algoritmo Apriori necesitará generar más de 10^7 Itemsets Candidatos, así como acumular y probar su ocurrencia. Además, para descubrir un patrón frecuente de tamaño 100, como $a_1, ..., a_{100}$, se debe generar más de 2^{100} candidatos en total.
- Es una tarea demasiado tediosa el tener que repetidamente leer la base de datos para revisar un gran conjunto de candidatos.

El cuello de botella de Apriori es la generación de candidatos [24]. Este problema es atacado por los siguientes tres aspectos:

Primero, una innovadora y compacta estructura de datos llamada *Árbol de Patrones Frecuentes* o *FP-tree* por sus siglas en ingles (*Frequent Pattern Tree*), la cual es una estructura que almacena información crucial y cuantitativa acerca de

los patrones frecuentes. Únicamente los itemsets frecuentes 1 tendrán nodos en el árbol, el cual esta organizado de tal forma que los items más frecuentes de una transacción tendrán mayores oportunidades de compartir nodos en la estructura.

Segundo, un método de Minería de patrones crecientes basado en un FP-tree. Este comienza con un patrón frecuente tipo 1 (como patrón sufijo inicial), examina sus Patrones Condicionales Base (una "sub-base de datos" que consiste en el conjunto de items frecuentes, que se encuentran en patrón sufijo), construye su FP-tree (condicional) y dentro de este lleva a cabo recursivamente Minería. El patrón creciente es conseguido a través de la concatenación del patrón sufijo con los nuevos generados del FP-tree condicional. Un itemset frecuente en cualquier transacción siempre se encuentra en una ruta de los árboles de Patrones Frecuentes.

Tercero, la técnica de busqueda empleada en la Minería esta basada en particionamiento, con el método "divide y venceras". Esto reduce dramaticamente el tamaño del Patrón Condicional Base generado en el siguiente nivel de busqueda, así como el tamaño de su correspondiente FP-tree. Es más, en vez de buscar grandes patrones frecuentes, busca otros más pequeños y los concatena al sufijo. Todas estas técnicas reducen los costos de busqueda.

Diseño y construcción del Árbol de Patrones Frecuentes (FP-tree)

Sea $I = a_1, a_2, ..., a_m$ un conjunto de items, y $DB = T_1, T_2, ..., T_m$ una base de datos de transacciones, donde $T_i (i \in [1...n])$ es una transacción que contiene un conjunto de items en I. El soporte (u ocurrencia) de un patrón A o conjunto de items, es el número de veces que A esta contenida en DB. A es un patrón frecuente, si el soporte de A es mayor que el umbral o soporte mínimo, ξ.

Árbol de Patrones Frecuentes A través del siguiente ejemplo se examina como funciona el diseño de la estructura de datos para minar con eficiencia patrones frecuentes.

Ejemplo 1 Sea la base de datos de transacciones, cuadro 6.2 y $\xi = 3$. Una estructura de datos puede ser diseñada de acuerdo a las siguientes observaciones:

1. Aunque solo los items frecuentes jugarán un rol en la Minería de Patrones Frecuentes, es necesario leer la BD para identificar este conjunto de items.

2. Si se almacenan items frecuentes de cada transacción en una estructura compacta, se podría evitar el tener que leer repetidamente la BD.

3. Si múltiples transacciones comparten un conjunto idéntico de items frecuentes, estas pueden ser fusionadas en una sola, con el número de ocurrencias como contador.

4. Si dos transacciones comparten un prefijo común, las partes compartidas pueden ser unidas usando una estructura prefija. Si los items frecuentes son ordenados descendientemente, habrá mayor probabilidad de que los prefijos de las cadenas esten compartidos.

Cuadro 6.2: Base de Datos de transacciones

TID	**Items**	**Items frecuentes (ordenados)**
100	f,a,c,d,g,i,m,p	f,c,a,m,p
200	a,b,c,f,l,m,o	f,c,a,b,m
300	b,f,h,j,o	f,b
400	b,c,k,s,p	c,b,p
500	a,f,c,e,l,p,m,n	f,c,a,m,p

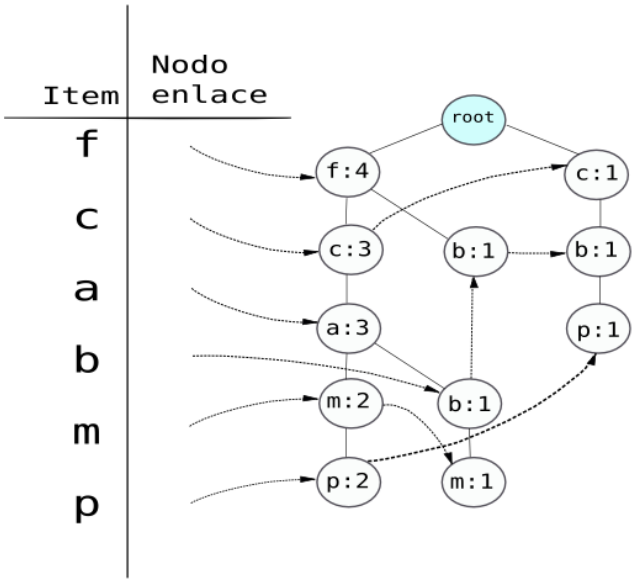

Figura 6.2: Tabla de cabeceras y Arbol FP-tree del ejemplo 1

Con estas observaciones se puede construir un Árbol de Patrones Frecuentes de la siguiente forma:

Primero, el leer DB genera una lista de items frecuentes, $\{(f : 4), (c : 4), (a : 3), (b : 3), (m : 3), (p : 3)\}$, (el número indica el soporte) ordenados descendientemente.

Segundo, crear la raíz del árbol con un *null*. Al leer la primera transacción se construye la primera rama del árbol $\{(f : 1), (c : 1), (a : 1), (m : 1), (p : 1)\}$. Para la segunda transacción ya que su lista de items frecuentes (f, c, a, b, m) comparte el prefijo común (f, c, a) con la rama existente (f, c, a, m, p), el conteo de cada nodo en el árbol prefijo es incrementado en 1, dos nuevos nodos son creados, $(b : 1)$ enlazado como hijo de $(a : 2)$ y $(m : 1)$ enlazado como hijo de $(b : 1)$. Para la tercera transacción como su lista de frecuentes solamente es (f, b) y el único prefijo compartido es (f), el soporte de (f) se incrementa en 1, y un nuevo nodo $(b : 1)$ es creado y enlazado como hijo de $(f : 3)$. La lectura de la cuarta transacción lleva a la construcción de la segunda rama del árbol, $\{(c : 1), (b : 1), (p : 1)\}$. Para la última transacción ya que su lista de items frecuentes (f, c, a, m, p) es identica a la primera, la ruta es compartida, incrementado el conteo de cada nodo en 1.

Para facilitar más el funcionamiento del árbol una Tabla de Cabeceras es construida, en la cual cada item apunta a su ocurrencia en el árbol. Los nodos con el mismo nombre son enlazados en secuencia y después de leer todas las transacciones, se puede ver el árbol resultante en la figura 6.2.

Por tanto un **Árbol de Patrones Frecuentes (FP-tree)** es una estructura como se define a continuación:

- Consiste de una raiz etiquetada como *null*, un conjunto de árboles hijos de la raiz y una Tabla de Cabeceras de items frecuentes.
- Los nodos del Árbol de Patrones Frecuentes o FP-tree tienen tres campos: nombre del item, contador y enlaces a los demás nodos. El nombre del item registra que item este nodo representa, el contador registra el número de transacciones representadas por la porción de la ruta que alcanzan a este nodo y los enlaces llevan al siguiente nodo en el FP-tree, que tiene el mismo nombre o a *null* si no hay nada.
- Cada entrada en la Tabla de Cabeceras de items frecuentes tiene dos campos, nombre del item y el primer nodo enlazado, al cual apunta la cabecera con

el mismo nombre.

Minando Patrones Frecuentes con FP-tree

Existen ciertas propiedades del Árbol de Patrones Frecuentes que facilitarán la tarea de Minería de Patrones Frecuentes:

1. **Propiedad de nodos enlazados**. Para cualquier nodo frecuente a_i, todos los posibles Patrones Frecuentes que contenga a_i pueden ser obtenidos siguiendo los nodos enlazados de a_i, comenzando desde a_i en la Tabla de Cabeceras de items frecuentes.

Ejemplo 2 El siguiente es el proceso de Minería basado en el FP-tree de la figura 6.2. De acuerdo a la propiedad de nodos enlazados para obtener todos los Patrones Frecuentes de un nodo a_i, se comienza desde la cabeza de a_i (en la Tabla de Cabeceras).

Comenzando por los nodos enlazados de p su Patrón Frecuente resultante es $(p:3)$ y sus dos rutas en el FP-tree son $(f:4, c:3, a:3, m:2, p:2)$ y $(c:1, b:1, p:1)$. La primera ruta indica que la cadena "(f, c, a, m, p)" aparece dos veces en la base de datos. Se puede observar que "(f, c, a)" aparece tres veces y "(f)" se encuentra cuatro veces, pero con p solo aparecen dos veces. Además para estudiar que cadena aparece con p, unicamente cuenta el prefijo de p, $(f:4, c:3, a:3, m:2)$. Similarmente, la segunda ruta indica que la cadena "(c, b, p)" aparece solo una vez en el conjunto de transacciones de DB, o que el prefijo de p es $(c:1, b:1)$. Estos dos prefijos de p, $(f:2, c:2, a:2, m:2)$ y $(c:1, b:1)$, forman los Sub-Patrones Base de p o llamados Patrones Condicionales Base. La construcción de un FP-tree sobre este Patrón Condicional Base lleva a unicamente una rama $(c:3)$. Así que solo existe un Patrón Frecuente $(cp:3)$.

Para el nodo m, se obtiene el Patrón Frecuente $(m:3)$ y las rutas $(f:4, c:3, a:3, m:2)$ y $(f:4, c:3, a:3, b:1, m:1)$. Al igual que en el análisis anterior se obtiene los Patrones Condicionales Base de m, que son $(f:2, c:2, a:2)$ y $(f:1, c:1, a:1, b:1)$. Al construir un FP-tree sobre estos, se obtiene que el FP-tree condicional de m es $(f:3, c:3, a:3)$. Después se podría llamar recursivamente a una función de Minería basada en un FP-tree $(\text{mine}((f:3, c:3, a:3)|m))$.

La figura 6.3 muestra como funciona $(\text{mine}((f:3, c:3, a:3)|m))$ y que incluye minar tres items a, c, f. La primera deriva en un Patrón Frecuente $(am:3)$,

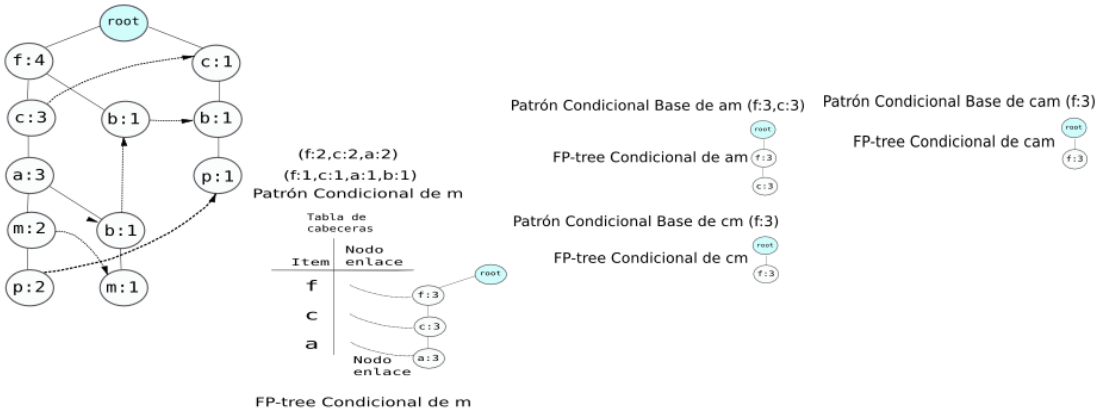

Figura 6.3: FP-tree condicional para m

y una llamada a (mine$((f : 3, c : 3)|am)$); la segunda deriva en un Patrón Frecuente $(cm : 3)$, y una llamada a (mine$((f : 3)|cm)$); y la tercera deriva en unicamente el Patrón Frecuente $(fm : 3)$. Con adicionales llamadas recursivas a (mine$((f : 3, c : 3)|am)$) se obtiene $(cam : 3)$, $(fam : 3)$ y con una llamada a (mine$((f : 3)|cam)$), se obtendrá el patrón más largo $(fcam : 3)$. Similarmente con la llamada de (mine$((f : 3)|cm)$), se obtiene el patrón $(fcm : 3)$. Además todo el conjunto de Patrones Frecuentes que tienen a m es $\{(m : 3), (am : 3), (cm : 3), (fm : 3), (cam : 3), (fam : 3), (fcam : 3), (fcm : 3)\}$ o que explicado de otra forma, por ejemplo para m los Patrones Frecuentes son obtenidos combinando a m con todos sus Patrones Condicionales Base $(f : 3, c : 3, a : 3)$, entonces sus Patrones Frecuentes serían los ya mencionados $\{(m : 3), (am : 3), (cm : 3), (fm : 3), (cam : 3), (fam : 3), (fcam : 3), (fcm : 3)\}$. .

Así mismo, con el nodo b se obtiene $(b : 3)$ y tres rutas: $(f : 4, c : 3, a : 3, b : 1)$, $(f : 4, b : 1)$ y $(c : 1, b : 1)$. Dado que los Patrones Condicionales Base de b: $(f : 1, c : 1, a : 1)$, $(f : 1)$ y $(c : 1)$ no generan items frecuentes, el proceso de Minería termina. Con el nodo a solo se obtiene un Patrón Frecuente $\{(a : 3)\}$ y un Patrón Condicional Base $\{(f : 3, c : 3)\}$. Además su conjunto de Patrones Frecuentes puede ser generado a partir de sus combinaciones. Concatenandolas con $(a : 3)$, obtenemos $\{(fa : 3), (ca : 3), (fca : 3), \}$. Del nodo c se deriva $(c : 4)$ y un Patrón Condicional Base $\{(f : 3)\}$, y el conjunto de Patrones Frecuentes asociados con $(c : 3)$ es $\{(fc : 3)\}$. Con el nodo f solo se obtiene $(f : 4)$ sin Patrones Condicionales Base.

En la siguiente tabla se muestran los Patrones Condicionales Base y los FP-trees generados.

Cuadro 6.3: Patrones Condicionales Base y FP-trees Condicionales

Item	**Patrón Condicional Base**	**FP-tree condicional**
p	$\{(f:2, c:2, a:2, m:2), (c:1, b:1)\}$	$\{(c:3)\}\|p$
m	$\{(f:2, c:2, a:2), (f:1, c:1, a:1, b:1)\}$	$\{(f:3, c:3, a:3)\}\|m$
b	$\{(f:1, c:1, a:1), (f:1), (c:1)\}$	ϕ
a	$\{(f:3, c:3)\}$	$\{(f:3, c:3)\}\|a$
c	$\{(f:3)\}$	$\{(f:3)\}\|c$
f	ϕ	ϕ

Como se dijo anteriormente los Patrones Frecuentes se obtienen a partir de las combinaciones de cada uno de los items con sus Patrones Condicionales Base. Por ejemplo para m, sus Patrones Frecuentes $\{(m:3), (am:3), (cm:3), (fm:3), (cam:3), (fam:3), (fcam:3), (fcm:3)\}$, son obtenidos de combinar a m con cada uno de sus Patrones Condicionales Base $\{(f:2, c:2, a:2), (f:1, c:1, a:1, b:1)\}$.

6.4.3. EquipAsso

1. Nuevos Operadores Del Algebra Relacional Para Asociación.

   a) Operador Associator (α)
   Associator(α) es un operador algebraico unario que al contrario del operador Selección o Restricción (σ), aumenta la cardinalidad o el tamaño de una relación ya que genera a partir de cada tupla de una relación, todas las posibles combinaciones de los valores de sus atributos, como tuplas de una nueva relación conservando el mismo esquema. Por esta razón esta operación, debe ser posterior a la mayoría de operaciones en el proceso de optimización de una consulta.

   Su sintaxis es la siguiente:

   $$\alpha_{tam_inicial, tam_final}(\mathrm{R})$$

   El operador Associator genera, por cada tupla de la relación R, todos sus posibles subconjuntos (itemsets) de diferente tamaño. Associator

toma cada tupla t de R y dos parámetros: $tam\_inicial$ y tam_final como entrada, y retorna, por cada tupla t, las diferentes combinaciones de atributos X_i, de tamaño $tam\_inicial$ hasta tamaño tam_final, como tuplas en una nueva relación. El orden de los atributos en el esquema de R determina los atributos en los subconjuntos con valores, el resto se hacen nulos. El tamaño máximo de un itemset y por consiguiente el tamaño final máximo (tam_final) que se puede tomar como entrada es el correspondiente al valor del grado de la relación.
Formalmente, sea $A = \{A_1, ..., A_n\}$ el conjunto de atributos de la relación R de grado n y cardinalidad m, IS y ES el tamaño inicial y final respectivamente de los subconjuntos a obtener. El operador α aplicado a R.

$$\alpha_{IS,ES}(R) = \{\cup_{all} \quad X_i \quad | \quad X_i \subseteq t_i, \quad t_i \in R, \forall_i \forall_k (X_i = v_i(A_1), v_i(A_2), null, ..., v_i(A_k), null\ , v_i(A_k)\ null), (i = (2^n - 1) * m), (k = IS, ..., ES), \quad A_1 A_2 ... A_k, \quad IS = 1, ES = n\}$$

produce una nueva relación cuyo esquema R(A) es el mismo de R de grado n y cardinalidad $m' = (2^n - 1) * m$ y cuya extensión r(A) está formada por todos los subconjuntos X_i generados a partir de todas las combinaciones posibles de los valores no nulos $v_i(A_k)$ de los atributos de cada tupla t_i de R. En cada tupla X_i únicamente un grupo de atributos mayor o igual que IS y menor o igual que ES tienen valores, los demás atributos se hacen nulos.

Ejemplo 1. Sea la relación R(A,B,C,D):

A	**B**	**C**	**D**
a1	b1	c1	d1
a1	b2	c1	d2

El resultado de $R1 = \alpha_{2,4}(R)$, se puede observar en la figura 6.4.

b) Operador Equikeep (χ)
Equikeep (χ) es un operador unario que restringe los valores de los atributos de cada una de las tuplas de la relación R a únicamente los valores de los atributos que satisfacen una expresión lógica.

Su sintaxis es la siguiente:

$\chi_{expresion\_logica}(R)$

Cuadro 6.4: Resultado de $R1 = \alpha_{2,4}(R)$

A	**B**	**C**	**D**
a1	b1	null	null
a1	null	c1	null
a1	null	null	d1
null	b1	c1	null
null	b1	null	d1
null	null	c1	d1
a1	b1	c1	null
a1	b1	null	d1
a1	null	c1	d1
null	b1	c1	d1
a1	b1	c1	d1
a1	b2	null	null
a1	null	c1	null
a1	null	null	d2
null	b2	c1	null
null	b2	null	d2
null	null	c1	d2
a1	b2	c1	null
a1	b2	null	d2
a1	null	c1	d2
null	b2	c1	d2
a1	b2	c1	d2

El operador EquiKeep restringe los valores de los atributos de cada una de las tuplas de la relación R a únicamente los valores de los atributos que satisfacen una expresión lógica *expr_log* , la cual esta formada por un conjunto de cláusulas de la forma Atributo=Valor, y operaciones lógicas AND, OR y NOT. En cada tupla, los valores de los atributos que no cumplen la condición *expr_log* se hacen nulos. EquiKeep elimina las tuplas vacías, i.e. las tuplas con todos los valores de sus atributos nulos.

Formalmente, sea $A = A_1, ..., A_n$ el conjunto de atributos de la relación R de esquema R(A), de grado n y cardinalidad m. Sea p una expresión lógica integrada por cláusulas de la forma $A_i = const$ unidas por los operadores booleanos AND ($\hat{}$), OR (v), NOT ($\neg$). El operador χ aplicado

a la relación R con la expresión lógica p:

$$\chi_p(R) = \{t_i(A) | \forall_i \forall_j (p(v_i(A_j)) = v_i(A_j) \quad si \quad p = true \quad y \quad p(v_i(A_j)) = null \quad si \quad p = false), i = 1...m', j = 1...n, \quad m' \leq m\}$$

produce una relación de igual esquema R(A) de grado n y cardinalidad m', donde $m' \leq m$. En su extensión, cada n-tupla t_i, esta formada por los valores de los atributos de R, $v_i(A_j)$, que cumplan la expresión lógica p, es decir $p(v_i(Y_j))$ es verdadero, y por valores nulos si $p(v_i(Y_j))$ es falso.

Ejemplo 2. Sea la relación R(A,B,C,D) y la operación $\chi_{A=a1 \vee B=b1 \vee C=c2 \vee D=d1}(R)$:

A	**B**	**C**	**D**
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b1	c1	d1
a2	b2	c1	d2
a1	b2	c2	d1

El resultado de esta operación es la siguiente:

A	**B**	**C**	**D**
a1	b1	null	d1
a1	null	null	null
null	null	c2	null
null	b1	null	d1
null	null	null	null
a1	null	c2	d1

2. Algoritmo EquipAsso
   El primer paso del algoritmo simplemente cuenta el número de ocurrencias de cada item para determinar los 1-itemsets frecuentes. En el subsiguiente paso, con el operador EquiKeep se extraen de todas las transacciones los itemsets frecuentes tamaño 1 haciendo nulos el resto de valores. Luego se aplica el operador Associator desde $I_s = 2$ hasta el grado n. A continuación se muestra el algoritmo Equipasso:

   $L1 = \{1 - itemsets \quad frecuentes\}$;
   forall $transacciones \quad t \in D$ do begin
   $R = \chi_{L1}(D)$
   $K = 2$

$g = grado(R)$
$R' = \alpha_{k,g}(R)$
end
$L_k = \{count(R')|c.count \geq minsup\}$
Respuesta $= \cup L_k$;

6.5. Estado del Arte

En el desarrollo de nuestro trabajo de grado realizamos un estudio de herramientas de minería de datos elaboradas por otras universidades y centros de investigación que nos permitieron ver el estado actual de este tipo de aplicaciones. A continuación se describen las herramientas estudiadas.

6.5.1. WEKA - Waikato Environment for Knowledge Analysis

WEKA es una herramienta libre de minería de Datos realizada en el departamento de Ciencias de la Computación de la Universidad de Waikato en Hamilton, Nueva Zelanda. Los principales gestores de este proyecto son Ian H. Waitten y Eibe Frank autores del libro Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations [52] cuyo octavo capítulo sirve como tutorial de Weka y es libremente distribuido junto con el ejecutable y el código fuente. Las ultimas versiones estables de Weka pueden ser descargadas de la página oficial del Proyecto [17].

La implementación de la herramienta fue hecha bajo la plataforma Java utilizando la versión 1.4.2 de su máquina virtual. Al ser desarrollada en Java, Weka posee todas las características de una herramienta orientada a objetos con la ventaja de ser multiplataforma, la ultima versión (3.4) ha sido evaluada bajo ambientes GNU/Linux, Macintosh y Windows siguiendo una arquitectura Cliente/Servidor lo que permite ejecutar ciertas tareas de manera distribuida.

La conexión a la fuente de datos puede hacerse directamente hacia un archivo plano, considerando varios formatos como C4.5 y CVS aunque el formato oficial es el ARFF que se explica más adelante, o a través de un driver JDBC hacia diferentes Sistemas Gestores de Bases de Datos(SGBD).

Entre las principales características de Weka esta su modularidad la cual se fundamenta en un estricto y estandarizado formato de entrada de datos que de-

nominan ARFF (Atributte - Relation File Format), a partir de este tipo de archivo todos los algoritmos de minería implementados en Weka son trabajados. Nuevas implementaciones y adiciones deben ajustarse a este formato.

El formato ARFF consiste en una archivo sencillo de texto que funciona a partir de etiquetas, similar a XML, y que debe cumplir con dos secciones: una cabecera y un conjunto de datos. La cabecera contiene las etiquetas del nombre de la relación (@relation) y los atributos (@atributte), que describen los tipos y el orden de los datos. La sección datos (@data) en un archivo ARFF contiene todos los datos del conjunto que se quiere evaluar separados por comas y en el mismo orden en el que aparecen en la sección de atributos [15].

La conexión al SGBD se hace en primera instancia a través de una interfaz gráfica de usuario construyendo una sentencia SQL siguiendo un modelo relacional pero a partir de construida la tabla a minar se trabaja en adelante fuera de línea a través de flujos (streams) que se comunican con archivos en disco duro.

WEKA cubre gran parte del proceso de Descubrimiento de Conocimiento, las características especificas de minería de datos que se pueden ver son preprocesamiento y análisis de datos, clasificación, clustering, asociación y visualización de resultados.

WEKA posee una rica colección de algoritmos que soportan el proceso de minería que aplican diversas metodologías como arboles de decisión, conjuntos difusos, reglas de inducción, métodos estadísticos y redes bayesianas.

Por poseer diferentes modos de interfaces gráficas, WEKA se puede considerar una herramienta orientada al usuario aunque cabe aclarar que exige un buen dominio de conceptos de minería de datos y del objeto de análisis. Muchas tareas se encuentran soportadas aunque no del todo automatizadas por lo que el proceso de descubrimiento debe ser guiado aún por un analista.

Un análisis completo de las características de WEKA con respecto a otras aplicaciones presentes en el mercado se puede encontrar en [14].

6.5.2. ADaM - Algorithm Development and Mining System

El proyecto ADaM es desarrollado por el Centro de Sistemas y Tecnologías de la Información de la Universidad de Alabama en Huntsville, Estados Unidos. Este sistema es usado principalmente para aplicar técnicas de minería a datos científicos obtenidos de sensores remotos [45].

ADaM es un conjunto de herramientas de minería y procesamiento de imágenes que consta de varios componentes interoperables que pueden usarse en conjunto para realizar aplicaciones en la solución de diversos problemas. En la actualidad, ADaM (en su versión 4.0.2) cubre cerca de 120 componentes [44] que pueden ser configurados para crear procesos de minería personalizados. Nuevos componentes pueden fácilmente integrarse a un sistema o proceso existente.

ADaM 4.0.2 provee soporte a través del uso de componentes autónomos dentro de una arquitectura distribuida. Cada componente esta desarrollado en C, C++ u otra interfaz de programación de aplicaciones y entrega un ejecutable a modo de script donde cada componente recibe sus parámetros a través de línea de comandos y arroja los resultados en ficheros que pueden ser utilizados a su vez por otros componentes ADaM. Eventualmente se ofrece Servicios Web de algunos componentes por lo que se puede acceder a ellos a través de la Web.

Lastimosamente el acceso a fuentes de datos es limitada no ofreciendo conexión directa hacia un sistema gestor de bases de datos. ADaM trabaja generalmente con ficheros ARFF [15] que son generados por sus componentes.

Dentro de las herramientas ofrecidas por ADaM encontramos soporte al preprocesamiento de datos y de imágenes, clasificación, clustering y asociación. La visualización y análisis de resultados se deja para ser implementado por el sistema que invoca a los componentes. ADaM 4.0.2 ofrece un amplio conjunto de herramienta que implementa diversos metodologías dentro del área del descubrimiento de conocimiento. Existen módulos que implementan árboles de decisión, reglas de asociación, métodos estadísticos, algoritmos genéticos y redes bayesianas.

ADaM 4.0.2 no soporta una interfaz gráfica de usuario, se limita a ofrecer un conjunto de herramientas para ser utilizadas en la construcción de sistemas que cubran diferentes ámbitos. Por tal motivo, es necesario un buen conocimiento de los conceptos de minería de datos, a parte de fundamentos en el análisis y procesamiento de imágenes. No obstante, ADaM ofrece un muy buen soporte a sistemas

donde se busque descubrimiento de conocimiento, un ejemplo de la implementación de ADaM puede verse en [2] donde se utilizan componentes ADaM en el análisis e interpretación de imágenes satelitales de ciclones para estimar la máxima velocidad de los vientos.

6.5.3. Orange - Data Mining Fruitful and Fun

Construído en el Laborátorio de Inteligencia Artifícial de la Fácultad de Computación y Ciencias de la Información de la Universidad de Liubliana, en Eslovenia y ya que fué liberado bajo la Licencia Pública General (GPL) puede ser descargado desde su sitio oficial [33].

Orange [12] en su núcleo es una librería de objetos C++ y rutinas que incluyen entre otros, algoritmos estandar y no estandar de Minería de Datos y Aprendizaje Maquinal, además de rutinas para la entrada y manipulación de datos. Orange provee un ambiente para que el usuario final pueda acceder a la herramienta a través de scripts hechos en Python y que se encuentran un nivel por encima del núcleo en C++. Entonces el usuario puede incorporar nuevos algoritmos o utilizar código ya elaborado y que le permiten cargar, limpiar y minar datos, así como también imprimir árboles de desición y reglas de asociación.

Otra característica de Orange es la inclusión de un conjunto de Widgets gráficos que usan métodos y módulos del núcleo central (C++) brindando una intefáz agradable e intuitiva al usuario.

Los Widgets de la interfaz gráfica y los módulos en Python incluyen tareas de Minería de Datos desde preprocesamiento hasta modelamiento y evaluación. Entre otras funciones estos componentes poseen técnicas para:

Entrada de datos Orange proporciona soporte para varios formatos populares de datos. Como por ejemplo:

- **.tab** Formato nativo de Orange. La primera línea tiene los nombres de los atributos, la segunda línea dice cual es el tipo de datos de cada columna (discreto o continuo) y en adelante se encuentran los datos separados a través de tabuladores.
- **.c45** Estos archivos están compuestos por dos archivos uno con extención .names que tiene los nombres de las columnas separados por comas y otro .data con los datos separados también con comas.

Manipulación de datos y preprocesamiento Entre las tareas que Orange incluye en este apartado tenemos visualización gráfica y estadística de datos, procesamiento de filtros, discretización y construcción de nuevos atributos entre otros métodos.

Minería de Datos y Aprendizaje Máquinal Dentro de esta rama Orange incluye variedad de algoritmos de asociación, clasificación, regresión logistica, regresión lineal, árboles de regresión y acercamientos basados en instancias (instance-based approaches).

Contenedores Para la calibración de la predicción de probabilidades de modelos de clasificación.

Métodos de evaluación Que ayudan a medir la exactiud de un clasificador.

Podría catalogarse como una falencia el hecho de que Orange no incluya un modulo para la conexión a un Sistema Gestor de Bases de Datos, pero si se revisa bien la documentación de los modulos se encuentra que ha sido desarrollado uno para la conexión a MySQL (orngMySQL [34]). El modulo provee una entrada a MySQL a través de este modulo y de sencillos comandos en Python los datos de las tablas pueden transferidos desde y hacia MySQL. Así mismo programadores independientes han desarrollado varios modulos entre los que se destaca uno para algoritmos de Inteligencia Artificial.

Dentro de las herramientas de Minería de Datos, Orange podría catalogarse como una Herramienta Genérica Multitarea. Estas herramientas realizan una variedad de tareas de descubrimiento, típicamente combinando clasificación, asociación , visualización, clustering, entre otros. Soportan diferentes etapas del proceso de DCBD. El usuario final de estas herramientas es un analista quien entiende la manipulación de los datos.

Orange funciona en varias plataformas como Linux, Mac y Windows y desde su sitio web [33] se puede descargar toda la documentación disponible para su instalación. Al instalar Orange el usuario puede acceder a una completa información de la herramienta, así como a prácticos tutoriales y manuales que permiten familiarizarse con la misma. Su sitio web [33] incluye tutoriales básicos sobre Python y Orange, manuales para desarrolladores más avanzados y además tener acceso a los foros de Orange en donde se despejan todos los interrogantes sobre esta herramienta.

En si Orange esta hecho para usuarios experimentados e investigadores en aprendizaje maquinal con la ventaja de que el software fue liberado con licencia GPL, por tanto cualquier persona es libre de desarrollar y probar sus propios algoritmos reusando tanto código como sea posible.

6.5.4. TANAGRA - A Free Software for Research and Academic Purposes

TANAGRA [39] es software de Minería de Datos con própositos acádemicos e investigativos, desarrollado por Ricco Rakotomalala, miembro del Equipo de Investigación en Ingeniería del Conocimiento (ERIC - Equipe de Recherche en Ingénierie des Connaissances [1]) de la Universidad de Lyon, Francia. Conjuga varios métodos de Minería de Datos como análisis exploratorio de datos, aprendizaje estadístico y aprendizaje maquinal.

Este proyecto es el sucesor de SIPINA, el cuál implementa varios algoritmos de aprendizaje supervisado, especialmente la construcción visual de árboles de desición. TANAGRA es más potente, contiene además de algunos paradigmas supervisados, clustering, análisis factorial, estadística paramétrica y no-paramétrica, reglas de asociación, selección de características y construcción de algoritmos.

TANAGRA es un proyecto open source, así que cualquier investigador puede acceder al código fuente y añadir sus propios algoritmos, en cuanto este de acuerdo con la licencia de distribución.

El principal próposito de TANAGRA es proponer a los investigadores y estudiantes una arquitectura de software para Minería de Datos que permita el análisis de datos tanto reales como sintéticos.

El segundo próposito de TANAGRA es proponer a los investigadores una arquitectura que les permita añadir sus propios algoritmos y métodos, para así comparar sus rendimientos. TANAGRA es más una plataforma experimental, que permite ir a lo esencial y obviarse la programación del manejo de los datos.

El tercero y último próposito va dirigido a desarrolladores novatos y consiste en difundir una metodología para construir este tipo de software con la ventaja de tener acceso al código fuente, de esta forma un desarrollador puede ver como ha sido construido el software, cuales son los problemas a evadir, cuales son los principales pasos del proyecto y que herramientas y librerías usar.

Lo que no incluye TANAGRA es lo que constituye la fuerza del software comercial en el campo de la Minería de Datos: Grandes fuentes de datos, acceso directo a bodegas y bases de datos (Data Warehouses y Data Bases) así como la aplicación de data cleaning.

Para realizar trabajos de Minería de Datos con Tanagra, se deben crear esquemas y diagramas de flujo y utilizar sus diferentes componentes que van desde la visualización de datos, estadísticas, construcción y selección de instancias, regresión y asociación entre otros.

El formato del conjunto de datos de Tanagra es un archivo plano (.txt) separado por tabulaciones, en la primera línea tiene un encabezado con el nombre de los atributos y de la clase, a continuación vienen los datos con el mismo formato de separación con tabuladores. Tanagra también acepta conjuntos de datos generados en Excel y uno de los estandares en Minería de Datos, el formato de WEKA (archivos .arff). Tanagra permite cargar un solo conjunto de datos.

A medida que se trabaja con TANAGRA y se van generando reportes de resultados, existe la posibilidad de guardarlos en formato HTML.

La paleta de componentes de Tanagra tiene implementaciones de tareas de Minería de Datos que van desde preprocesamiento hasta modelamiento y evaluación. Entre otras TANAGRA permite usar técnicas para:

Entrada de datos Con soporte para varios formatos populares de datos.

Manipulación de datos y preprocesamiento Como muestreo, filtrado, discretización y construcción de nuevos atributos entre otros métodos.

Construcción de Modelos Métodos para la construcción de modelos de clasificación, que incluyen árboles de clasificación, clasificadores Naive-Bayes y regresión logística.

Métodos de regresión Como regresión multiple lineal.

Métodos de evaluación Utilizados para medir la exactitud de un clasificador.

Al ser TANAGRA un proyecto Open Source es facilmente descargable desde su sitio web [40], el cual contiene tutoriales, referencias y conjuntos de datos.

6.5.5. AlphaMiner

AlphaMiner es desarrollado por el E-Business Technology Institute (ETI) de la Universidad de Hong Kong [28] bajo el apoyo del Fondo para la Innovación y la Tecnología (ITF) [35] del Gobierno de la Región Especial Administrativa de Hong Kong (HKSAR).

AlphaMiner es un sistema de Minería de Datos, Open Source, desarrollado en java de propósito general pero más enfocado al ambiente empresarial. Implementa algoritmos de asociación, clasificación, clustering y regresión logística. Posee una gama amplia de funcionalidad para el usuario, al realizar cualquier proceso minero dando la posibilidad al usuario de escoger los pasos del proceso KDD que más se ajusten a sus necesidades, por medio de nodos que el analista integra a un árbol KDD siguiendo la metodología Drag & Drop. Es por eso que el proceso KDD en AlphaMiner no sigue un orden estructurado, sino que sigue la secuencia brindada por el propósito u objetivo del analista, ademas brinda la visualicion estadística de distintas maneras, permitiendo un analisis más preciso por parte del analista.

El principal objetivo de AlphaMiner es la inteligencia Comercial Económica o Inteligencia de Negocios (BI - Business Inteligence) es uno de los medios más importantes para que las compañías tomen las decisiones comerciales más acertadas. Las soluciones de BI son costosas y sólo empresas grandes pueden permitirse el lujo de tenerlas. Por tanto las compañías pequeñas tienen una gran desventaja. AlphaMiner proporciona las tecnologías de BI de forma económica para dichas empresas para que den soporte a sus decisiones en el ambiente de negocio cambiante rápido.

AlphaMiner tiene dos componentes principales:

1. Una base de conocimiento.
2. Un árbol KDD, que permite integrar nodos artefacto que proporcionan varias funciones para crear, revisar, anular, interpretar y argumentar los distintos análisis de datos

AlphaMiner implementa los siguientes pasos del proceso KDD:

1. Acceso de diferentes formas a las fuentes datos.
2. Exploración de datos de diferentes maneras.
3. Preparación de datos.

4. Vinculación de los distintos modelos mineros.

5. Análisis a partir de los modelos.

6. Despliegue de modelos al ambiente empresarial.

La característica más importante de AlphaMiner, es su capacidad para almacenar los datos después del núcleo minero en una base de conocimiento, que puede ser reutilizada. Esta función aumenta su utilidad significativamente, y brinda un gran apoyo logístico al nivel estratégico de una empresa. Aventajando cualquier sistema tradicional de manipulación de datos. AlphaMiner proporciona una funcionalidad adicional para construir los modelos de Minería de Datos, conformando una sinergia entre los distintos algoritmos de minería.

AlphaMiner se asemeja a Tariy en varias características, por un lado el lenguaje de programación en el que fue implementado es JAVA, por otro lado es Open Source, tiene conectividad JDBC a los distintos gestores de bases de datos, además de conectividad con archivos de Excel. Otra semejanza es el tipo de formato que presenta para el manejo de tablas unívaluadas en algoritmos de asociación, específicamente en bases de datos de supermercados, ya que después de escoger los campos de dicha base de datos se la lleva a un formato de dos columnas una para la correspondiente transacción y otra para el ítem.

6.5.6. YALE - Yet Another Learning Environment

YALE [53, 32] es un ambiente para la realización de experimentos de aprendizaje maquinal y Minería de Datos. A través de un gran número de operadores se pueden crear los experimentos, los cuales pueden ser anidados arbitrariamente y cuya configuración es descrita a través de archivos XML que pueden ser fácilmente creados por medio de una interfaz gráfica de usuario. Las aplicaciones de YALE cubren tanto investigación como tareas de Minería de Datos del mundo real.

Desde el año 2001 YALE ha sido desarrollado en la Unidad de Inteligencia Artificial de la Universidad de Dortmund [36] en conjunto con el Centro de Investigación Colaborativa en Inteligencia Computacional (Sonderforschungsbereich 531).

El concepto de operador modular permite el diseño de cadenas complejas de operadores anidados para el desarrollo de un gran número de problemas de aprendizaje. El manejo de los datos es transparente para los operadores. Ellos no tienen nada que ver con el formato de datos o con las diferentes presentaciones de los

mismos. El kernel de Yale se hace a cargo de las transformaciones necesarias. Yale es ampliamente usada por investigadores y compañías de Minería de Datos.

Modelando Procesos De Descubrimiento De Conocimiento Como Arboles De Operadores

Los procesos de descubrimiento de conocimiento son vistos frecuentemente como invocaciones secuenciales de métodos simples. Por ejemplo, después de cargar datos, uno podría aplicar un paso de preprocesamiento seguido de una invocación a un método de clasificación. El resultado en este caso es modelo aprendido, el cual puede ser aplicado a datos nuevos o no revisados todavía. Una posible abstracción de estos métodos simples es el concepto de operadores. Cada operador recibe su entrada, desempeña una determinada función y entrega una salida. Desde ahi, el método secuencial de invocaciones corresponde a una cadena de operadores. Aunque este modelo de cadena es suficiente para la realización de muchas tareas básicas de descubrimiento de conocimiento, estas cadenas planas son a menudo insuficientes para el modelado de procesos de descubrimiento de conocimiento más complejas.

Una aproximación común para la realización del diseño de experimentos más complejos es diseñar las combinaciones de operadores como un gráfico direccionado. Cada vertice del gráfico corresponde a un operdor simple. Si dos operadores son conectados, la salida del primer operador será usada como entrada en el segundo operador. Por un lado, diseñar procesos de descubrimiento de conocimiento con la ayuda de gráficosdireccionados es muy poderoso. Por el otro lado, existe una desventaja principal: debido a la pérdida de restricciones y la necesidad de ordenar topológicamente el diseño de experimentos es menudo poco intuitivo y las validaciones automáticas son difíciles de hacer.

Yale ofrece un compromiso entre la simplicidad de cadenas de operadores y la potencia de los gráficos direccionados a través del modelamiento de procesos de descubrimiento de conocimiento por medio de árboles de operadores. Al igual que los lenguajes de programación , el uso de árboles de operadores permite el uso de conceptos como ciclos, condiciones, u otros esquemas de aplicación. Las hojas en el árbol de operadores corresponden a pasos sencillos en el proceso modelado como el aprendizaje de un modelo de predicción ó la aplicación de un filtro de preprocesamiento. Los nodos interiores del árbol corresponden a más complejos o abstractos pasos en el proceso. Este es a menudo necesario si los hijos deben ser aplicados varias veces como, por ejemplo, los ciclos. En general, los nodos de los operadores internos definen el flujo de datos a través de sus hijos. La raiz del árbol

corresponde al experimento completo.

Qué Puede Hacer YALE?

YALE provee mas de 200 operadores incluyendo:

Algoritmos de aprendizaje maquinal Un gran número de esquemas de aprendizaje para tareas de regresión y clasificación incluyendo Máquinas para el Soporte de Vectores (SVM), árboles de decisión e inductores de reglas, operadores Lazy, operadores Bayesianos y operadores Logísticos. Varios operadores para la minería de reglas de asociación y Clustering son también parte de YALE. Además, se adicionaron varios esquemas de Meta Aprendizaje.

Operadores WEKA Todos los esquemas de aprendizaje y evaluadores de atributos del ambiente de aprendizaje WEKA también están disponibles y pueden ser usados como cualquier otro operador de YALE.

Bibliografía

[1] Université Lumière Lyon 2. Eric equipe de recherche en ingénierie des conaissances. http://chirouble.univ-lyon2.fr, 2006.

[2] NASA Ñational Aeronautics and Space Administration. Tropical cyclone windspeed indicator. http://pm-esip.nsstc.nasa.gov/cyclone/.

[3] R. Agrawal, T. Imielinski, and A. Swami. *Mining Association Rules between Sets of Items in Large Databases*. ACM SIGMOD, 1993.

[4] R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, and T. Bollinger. The quest data mining system. In *2Âº Conference KDD y Data Mining*, Portland, Oregon, 1996.

[5] R. Agrawal and K. Shim. Developing tightly-coupled data mining applications on a relational database system. In *The Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, 1996.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB Conference*, Santiago, Chile, 1994.

[7] R. Brachman and T. Anand. *The Process of Knowledge Discovery in Databases: A First Sketch, Workshop on Knowledge Discovery in Databases*. 1994.

[8] S. Chaudhuri. Data mining and database systems: Where is the intersection? In *Bulletin of the Technical Committee on Data Engineering*, volume 21, Marzo 1998.

[9] M. Chen, J. Han, and P. Yu. Data mining: An overview from database perspective. In *IEEE Transactions on Knowledgeand Data Engineering*, 1996.

[10] Quadrillion Corp. Q-yield. http://www.quadrillion.com/qyield.shtm, 2001.

[11] IBM Corporation. Intelligent miner. http://www-4.ibm.com/software/data/iminer, 2001.

[12] J Demsar and B Zupan. Orange: From experimental machine learning to interactive data mining. Technical report, Faculty of Computer and Information Science, University of Liubliana, Slovenia, http://www.ailab.si/orange/wp/orange.pdf, 2004.

[13] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview, in advances in knowledge discovery and data mining. In *AAAI Pres / The MIT Press*, 1996.

[14] M. Goebel and L. Gruenwald. A survey of data mining and knowledge discovery software tools. In *SIGKDD Explorations*, volume 1 of *1*, June 1999.

[15] Waikato ML Group. Attribute-relation file format (arff). http://www.cs.waikato.ac.nz/ ml/weka/arff.html.

[16] Waikato ML Group. Collections of datasets. *http* : $//www.cs.waikato.ac.nz/ml/weka/index_datasets.html$.

[17] Waikato ML Group. The waikato environment for knowledge analisys. http://www.cs.waikato.ac.nz/ml/weka.

[18] J. Han, J. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, K. Koperski, G. Liu, Y. Lu, N. Stefanovic, L. Winstone, B. Xia, O. Zaiane, S. Zhang, and H. Zhu. Dbminer: A system for data mining in relational databases and data warehouses. In *CASCON: Meeting of Minds*.

[19] J. Han, Y. Fu, and S. Tang. Advances of the dblearn system for knowledge discovery in large databases. In *International Joint Conference on Artificial Intelligence IJCAI*, Montreal, Canada, 1995.

[20] J. Han, Y. Fu, W. Wang, J. Chiang, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, and O. Zaiane. Dbminer: A system for mining knowledge in large relational databases. In *The second International Conference on Knowledge Discovery & Data Mining*.

[21] J. Han, Y. Fu, W. Wang, J. Chiang, O. Zaiane, and K. Koperski. *DBMiner: Interactive Mining of Multiple-Level Knowledge in Relational Databases*. ACM SIGMOD, Montreal, Canada, 1996.

[22] J. Han and M. Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[23] J. Han and J. Pei. Mining frequent patterns by pattern-growth: Methodology and implications. In *SIGKDD Explorations*, volume 2:14-20, 2000.

[24] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD*, Dallas, TX, 2000.

[25] Java Hispano. El abc de jdbc. http://javahispano.org/tutorials.type.action?type=j2se, 2004.

[26] T. Imielnski and H. Mannila. A database perspective on knowledge discovery. In *Communications of the ACM*.

[27] RuleQuest Research Inc. C5.0. http://www.rulequest.com, 2001.

[28] E-Business Technology Institute. E-business technology institute, the university of hong kong. http://www.eti.hku.hk, 2005.

[29] Quinlan J.R. *C4.5: Programs for Machine Leraning*. Morgan Kaufmann Publishers, 1993.

[30] Kdnuggets. http://www.kdnuggets.com/software, 2001.

[31] C. Matheus, P. Chang, and G. Piatetsky-Shapiro. Systems for knowledge discovery in databases. In *IEEE Transactions on Knowledge and Data Engineering*, volume 5, 1993.

[32] I Mierswa, M Wurst, R Klinkenberg, M Scholz, and T Euler. Yale: Rapid prototyping for complex data mining tasks. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006.

[33] Faculty of Computer and Slovenia Information Science, University of Liubliana. Orange, fruitful and fun. http://www.ailab.si/orange, 2006.

[34] Faculty of Computer and Slovenia Information Science, University of Liubliana. Orange's interface to mysql. http://www.ailab.si/orange/doc/modules/orngMySQL.htm, 2006.

[35] Goverment of Hong Kong. Innovation and technology fund. http://www.itf.gov.hk, 2006.

[36] Artificial Inteligence Unit of the University of Dortmund. Artificial inteligence unit of the university of dortmund. http://www-ai.cs.uni-dortmund.de, 2006.

[37] G. Piatetsky-Shapiro, R. Brachman, and T. Khabaza. *An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications*. 1996.

[38] J.R. Quinlan. *Induction of decision trees. Machine Learning.* 1986.

[39] R Rakotomalala. Tanagra. In *TANAGRA: a free software for research and academic purposes*, volume 2, pages 697–702. EGC'2005, 2005.

[40] R Rakotomalala. Tanagra project. http://chirouble.univ-lyon2.fr/ ricco/tanagra/en/tanagra.html, 2006.

[41] Isoft S.A. Alice. http://www.alice-soft.com/html/prod_alice.htm, 2001.

[42] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In *ACM SIGMOD*, 1998.

[43] SPSS. Clementine. http://www.spss.com/clementine, 2001.

[44] Information Technology The University of Alabama in Huntsville and Systems Center. Adam 4.0.2 components. http://datamining.itsc.uah.edu/adam/documentation.html.

[45] Information Technology The University of Alabama in Huntsville and Systems Center. Algorithm development and mining system. http://datamining.itsc.uah.edu/adam/index.html.

[46] R. Timarán. Arquitecturas de integración del proceso de descubrimiento de conocimiento con sistemas de gestión de bases de datos: un estado del arte, en revista ingeniería y competitividad. *Revista de Ingeniería y Competitividad, Universidad del Valle*, 3(2), Diciembre 2001.

[47] R. Timarán. Descubrimiento de conocimiento en bases de datos: Una visión general. In *Primer Congreso Nacional de Investigación y Tecnología en Ingeniería de Sistemas*, Universidad del Quindío, Armenia, Octubre 2002.

[48] R. Timarán. *Nuevas Primitivas SQL para el Descubrimiento de Conocimiento en Arquitecturas Fuertemente Acopladas con un Sistema Gestor de Bases de Datos.* PhD thesis, Universidad del Valle, 2005.

[49] R. Timarán and M. Millán. Equipasso: an algorithm based on new relational algebraic operators for association rules discovery. In *Fourth IASTED International Conference on Computational Intelligence*, Calgary, Alberta, Canada, July 2005.

[50] R. Timarán and M. Millán. Equipasso: un algoritmo para el descubrimiento de reglas de asociación basado en operadores algebraicos. In *4Âª Conferencia Iberoamericana en Sistemas, Cibernética e Informática CICI 2005*, Orlando, Florida, EE.UU., Julio 2005.

[51] R. Timarán, M. Millán, and F. Machuca. New algebraic operators and sql primitives for mining association rules. In *IASTED International Conference Neural Networks and Computational Intelligence*, Cancun, Mexico, 2003.

[52] I. Waitten and F. Eibe. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. 2001.

[53] YALE. Yale - yet another learning environment. http://rapid-i.com, 2006.