Mejora de la usabilidad y la adaptabilidad mediante técnicas de minería de uso Web

Federico Botella
Enrique Lazcorreta
Centro de Investigación Operativa
Universidad Miguel Hernández de Elche
03202 Elche
{ federico, enrique }@umh.es

Pascual González
Antonio Fernández-Caballero
Instituto de Investigación en Informática
Universidad de Castilla-La Mancha
02071 Albacete
{ pgonzalez , caballer }@info-ab.uclm.es

Resumen

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

Palabras clave: Adaptabilidad, Usabilidad Sistemas sensibles al contexto.

1. Introducción

La aplicación de técnicas de Data Mining sobre los ficheros de log donde un servidor web guarda los accesos a un sitio web permite mejorar la facilidad de uso del sitio. Mediante este análisis se intenta adaptar el sitio web a las necesidades reales de sus usuarios. Esta adaptación se basa en el propio uso del sitio realizado por los usuarios, y no sólo en la opinión de los expertos, como se había abordado tradicionalmente. Serán los propios usuarios los que definan las partes del sitio web más útiles y así podremos adaptarles los enlaces a sus necesidades o preferencias sin necesidad de participación de los usuarios en el proceso a diferencia de otros sistemas [8] [13].

La búsqueda de técnicas de predicción sigue siendo un área en auge dentro del campo de la Web Mining. Cuando un usuario solicita una página, si el servidor dispone de información podría indicarle al usuario qué otras páginas suelen ser visitadas junto con la página solicitada. Esta información se puede obtener aplicando reglas de asociación al conjunto de todas las solicitudes que ha recibido el servidor por parte de todos los usuarios, conjunto que ha de ser convenientemente preprocesado para extraer información válida del mismo. El uso de técnicas de Data Mining viene motivado por el enorme tamaño de los ficheros de log que se deben procesar, debido al acceso de miles de usuarios distintos, que provocan el registro de millones de solicitudes en el servidor. La búsqueda de patrones que permitan agrupar a los usuarios permite reducir la información a tratar, obteniendo un tamaño aceptable para su procesamiento en tiempo real. El objetivo es ofrecer a todos los usuarios del sitio web la misma información.

Muchos usuarios visitan un sitio web en más de una ocasión, y muchas veces con el mismo objetivo: buscar cierta información u obtener el mismo servicio. En este trabajo veremos que es posible aplicar las mismas técnicas de predicción estudiando el comportamiento individual de cada usuario, de modo que reciba información más ajustada a su comportamiento como individuo. Además también se le mostrará al usuario información sobre el uso general del sitio web, lo que permitirá mejorar la usabilidad del sitio respecto de cada usuario, al mejorar la

satisfacción de cada usuario respecto del sistema [7].

El resto de este trabajo se organiza como sigue. En la sección 2 se exponen los conceptos básicos de los sistemas de predicción. En la sección 3 se muestra la metodología utilizada y las estructuras de datos empleadas. En la sección 4 se presentan las modificaciones realizadas al algoritmo *Apriori* para conseguir una mayor eficiencia. En la sección 5 presentaremos los resultados de la experimentación realizada con ficheros de log reales y que nos permiten concluir que nuestro método es aplicable en tiempo real. Por último, en la sección 6 se aportan las conclusiones del presente trabajo y las líneas de trabajo futuro.

2. Trabajo relacionado

El procedimiento general seguido por los investigadores en la predicción es el de preprocesar los datos disponibles para convertirlos en nuevos datos que, convenientemente estudiados, puedan darnos el conocimiento que estamos buscando [6]. Básicamente los únicos datos de que se dispone sobre el uso de un sitio web son los que guarda el servidor en los ficheros de log. Estos ficheros de log son ficheros de texto plano que contienen una línea por cada solicitud del usuario al servidor, anotando el instante de la solicitud, la dirección IP del usuario, el recurso solicitado, el estado de la solicitud (si se sirvió correctamente el recurso o no) y otros datos que no son de interés en el presente trabajo. Las características de la WWW hacen que una página esté formada por muchos recursos (imágenes, texto, etc), por lo que cuando un usuario solicita una página al servidor realmente está solicitando muchos recursos del mismo y todos ellos serán anotados en el fichero de log. Esto provoca el primer planteamiento a la hora de estudiar un fichero de log: hay que eliminar todos los recursos que realmente no sean objetivo del usuario, además de aquellos que no reconoce el servidor como recursos del sitio (generalmente errores en las solicitudes). Existen otros problemas en el uso de ficheros de log, como la existencia de páginas cargadas en la caché del servidor y cuyas solicitudes no aparecerán en él [14]. Algunos autores proponen resolver con el uso de “huellas” [17]. Otro problema es la existencia de agentes en la red que navegan automáticamente por todo un sitio web y cuyo comportamiento no debería ser considerado en el estudio.

Una vez eliminados los recursos que no son de nuestro interés hemos de organizar los datos del fichero de log de modo que puedan reflejar el uso del sitio web. La solución más aceptada para organizar estos datos es la creación de sesiones: conjunto de recursos que, presumiblemente, provienen de un mismo usuario en una visita a nuestro sitio. El problema que presenta la obtención de sesiones es que no suele haber información precisa sobre las mismas en el fichero de log; los usuarios de un sitio web son en su mayoría anónimos y de ellos sólo sabemos la dirección IP desde la que acceden y el navegador que han usado para ello. Al respecto hay muy variados estudios, véase [9], [12], o [11]. En este trabajo hemos considerado sesión aquel conjunto de recursos solicitados desde la misma dirección IP de modo que entre una solicitud y la siguiente no hayan transcurrido más de 10 minutos y entre la primera y la última solicitud no transcurran más de 4 horas.

Las sesiones son los datos de los que vamos a extraer el conocimiento. Para tener un rápido acceso a ellas lo conveniente es guardarlas en una base de datos, codificadas de modo que en lugar de anotar los recursos con largas cadenas de texto se guarde un valor único que los identifique unívocamente y que sea fácil de utilizar por un ordenador: un número es una buena elección. El éxito en la predicción dependerá tanto del preproceso aplicado a los ficheros de log como del criterio utilizado para construir las sesiones como del análisis que se haga sobre éstas. Los sitios web que buscan adaptar sus contenidos a los usuarios son sitios muy visitados, por lo que la base de datos de sesiones que proporcionan a este estudio puede contener millones de registros, de lo que se deriva la necesidad de recurrir a técnicas de data mining para proceder a su análisis.

El análisis de la base de datos de sesiones puede estar dirigido a diferentes objetivos, interesándonos en este artículo la predicción de solicitudes de los usuarios anónimos. En esta línea se han propuesto muy diversos planteamientos sobre cómo sintetizar la información que contiene la (generalmente enorme) base de datos; entre otras soluciones se propone el uso de Hypertext Probabilistic Grammar[4]; el uso de Programación Genética Basada en Gramática [10]; el sistema de

predicción WhatNext, basado en n-gram [15]; o el modelo multi-step dynamic n-gram [16].

Todas las técnicas proponen algún algoritmo que utiliza toda la información disponible para crear alguna estructura de datos que sea manejable por la memoria principal del ordenador. Una vez aplicado el algoritmo se pueden extraer del resultado reglas de asociación que permitirán, entre otras cosas, clasificar los ítems en diferentes clusters de modo que si un usuario accede a un recurso podemos buscar el cluster al que pertenece y hacer predicciones en tiempo real basándonos en la información ya preprocesada de dicho cluster. Un problema en la obtención de estas reglas de asociación es la actualidad de los datos, pues el planteamiento suele pasar por no obtener información más que de los ítems más frecuentes, por lo que la escalabilidad de la estructura en que se guarden los resultados está limitada a los ítems que son frecuentes en el preproceso de la base de datos, no admitiendo en general la introducción de nuevos ítems sin tener que proceder a la ejecución del algoritmo sobre la base de datos completa. Esto limita el uso de estos algoritmos en tiempo real.

Otro problema fundamental de este tipo de estudios se debe al hecho de trabajar con enormes bases de datos, con lo que cualquier estudio que se haga sobre las mismas conllevará mucho tiempo de acceso al disco en que se encuentre la base de datos; de ahí que las propuestas que encontramos hagan mucho énfasis en minimizar el número de accesos al disco proponiendo algoritmos capaces de extraer información con pocas lecturas de la base de datos. Éste es el principal problema del algoritmo Apriori [2] para encontrar reglas de dependencia entre los ítems (recursos en nuestro caso) presentes en grandes base de datos de transacciones (sesiones). Sin embargo es un algoritmo muy eficiente, creándose a partir de él una familia de algoritmos que pretenden aligerar el número de lecturas de la base de datos.

En este trabajo utilizamos este algoritmo, implementándolo eficientemente para trabajar con colecciones “pequeñas” de sesiones, las que puede proporcionar un único usuario. En [5] proponemos un método para hacer predicciones individualizadas, en tiempo real, basadas en la información que tenemos en la base de datos sobre los accesos del usuario (al que identificamos con la IP desde la que accede a nuestro sitio web). No tiene sentido guardar en disco la información preprocesada de todas las sesiones de cada usuario; serían muchos los datos a procesar y guardar y no estarían actualizados cuando quisiéramos usarlos. Lo que sí guardaremos en otra base de datos más pequeña es el número de registro de cada una de las sesiones provenientes de una misma IP, con lo que la lectura de todas las sesiones de un mismo usuario es rápida y, sobre todo, requiere de un espacio de memoria RAM manejable por cualquier servidor de hoy en día. Por este motivo no debemos preocuparnos más de los temidos accesos a disco, sólo hemos de preocuparnos de dotar al algoritmo de la máxima eficiencia pues no se aplicará una sola vez sobre muchos datos, sino que deberá ejecutarse cada vez que un usuario del que tengamos cierto número de sesiones guardadas acceda a nuestro sitio web.

3. Metodología propuesta

El método propuesto requiere aplicar el algoritmo de búsqueda de reglas de asociación cada vez que un usuario inicie sesión en nuestro sitio web. Por ello se debía elegir un algoritmo rápido y que consumiera pocos recursos de memoria, por lo que el algoritmo *Apriori* parecía el apropiado. La elección de la estructura de datos que necesita el algoritmo se realizó en base a la eficiencia, puesto que el empleo de matrices dinámicas suele conllevar que las operaciones de inserción y eliminación de un elemento de la matriz en memoria puede suponer la copia de la matriz completa, lo que ralentiza el proceso y aumenta el tiempo final del análisis.

Para aplicar el algoritmo se parte de un conjunto de transacciones que se denota por *D*; este conjunto se obtiene filtrando el conjunto de sesiones del usuario. De cada sesión se obtiene un conjunto ordenado lexicográficamente y sin repeticiones de las páginas que forman la sesión. A este conjunto se denomina *transacción* y se denota por t_i, $i=1,\ldots,m$, siendo m el número total de transacciones que forman D. A partir de ahora nos referiremos a las páginas como *ítems*, que se denotarán por α_j. De este modo una transacción puede verse como un conjunto ordenado de ítems

$$t_i = \{\alpha_{i1},\ldots,\alpha_{in_i}\}$$

siendo n_i el número de ítems de t_i. Una vez aplicado el algoritmo a *D* se obtiene el conjunto de todos los *k-itemsets frecuentes*, siendo un *k-itemset* un conjunto de *k* ítems que aparecen simultáneamente en alguna transacción.

3.1. La estructura de datos

La estructura elegida para implementar el algoritmo pretende reducir al máximo el número de elementos a almacenar y ser ágil en su recorrido. Para ello se ha optado por guardar los conjuntos L_k y C_k en un árbol de ítems, denotando con L'_k el *k*-ésimo nivel del árbol partiendo de su raíz. De este modo, en L'_1 se guardarán todos los ítems que sean frecuentes en la primera lectura de *D* ($L'_1=L_1$, es realmente el conjunto de 1-itemsets frecuentes, pero esta relación no volverá a repetirse: $L'_k \neq L_k \ \forall k \neq 1$, pues L'_k sólo contiene ítems, no *k-itemsets*). Por ejemplo, para conocer L_3 tendremos que recorrer los tres primeros niveles del árbol, L'_1, L'_2 y L'_3. Llamaremos *L* a este árbol de *k-itemsets* frecuentes.

Este modo de almacenar los datos aprovecha que las transacciones estén ordenadas y respeta ese orden, con lo que el primer ítem frecuente encontrado sólo aparecerá una vez en *L*, el segundo ítem frecuente encontrado aparecerá a lo sumo dos veces y así sucesivamente.

Aunque nos refiramos a C'_k como el conjunto de candidatos a ser *k-itemsets* frecuentes, realmente es un conjunto de ítems, similar a los conjuntos L'_k y que sólo tiene sentido si se añade como último nivel del árbol formado por L'_1, L'_2, …, L'_{k-1}. Esta elección se realizó de nuevo por el ahorro de espacio en memoria dentro de una estructura fácilmente accesible. Gran parte del tiempo de ejecución del algoritmo se empleará en la búsqueda en *L* de los *k-itemsets* presentes en *D;* con nuestra implementación se reduce considerablemente este tiempo de búsqueda pues no se buscan *k-itemsets* sino ítems. Inicialmente se busca en L'_1 el primer ítem del *k-itemset* en estudio, si no se encuentra ya se sabe que ese *k-itemset*, y todos los que se pudieran obtener de la misma transacción que comiencen con el mismo ítem, no son candidatos. Si se encuentra, se buscará sólo en el bloque de L'_2 descendiente del ítem encontrado el segundo ítem del *k-itemset*. Este bloque tiene siempre menos elementos que L'_1, por lo que esta segunda búsqueda será más rápida. Si se encuentra, se continúa con el bloque de ítems de L'_3 descendientes de este segundo ítem; de nuevo con menos elementos que el bloque anterior.

4. Algoritmo Apriori con relajación de poda

Utilizaremos el pseudocódigo original de Agrawal pues nuestro propósito no es mejorar el método sino encontrar una implementación eficiente del mismo. De hecho el cuerpo principal del algoritmo es el mismo:

1) $L_1 = \{\text{large } 1\text{-itemsets}\}$;
2) for $(k=2;\ L_{k-1} \neq \phi;\ k{+}{+})$ do begin
3) $C_k = apriori-gen(L_{k-1})$;
4) forall transactions $t \in D$ do begin
5) $C_t = subset(C_k, t)$;
6) forall candidates $c \in C_t$ do
7) $c.count{+}{+}$;
8) end
9) $L_k = \{c \in C_k \ / \ c.count \geq minsup\}$
10) end
11) Answer = $\bigcup_k L_k$

Algoritmo 1. Algoritmo *Apriori*

La función *apriori-gen* recibe como argumento L_{k-1}, el conjunto de (k-1)-*itemsets* frecuentes. Devuelve un superconjunto del conjunto de todos los k-itemsets candidatos a ser frecuentes. La función original está dividida en dos procesos: unión y poda.

La *unión* consiste en crear un conjunto de "posibles" candidatos a partir del conjunto de *(k-1)-itemsets* frecuentes L_{k-1}. En este proceso se aprovecha el orden de los elementos de *L* y, en lugar de hacer el producto cartesiano $L_{k-1}xL_{k-1}$ se considerarán sólo aquellos *k*-itemsets cuyos primeros *(k-2)* ítems formen parte de dos *(k-1)-itemset* frecuentes distintos:

```
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};
```

Algoritmo 2. apriori-gen: unión

La *poda* consiste en eliminar del conjunto de posibles candidatos aquellos que contengan algún *(k-1)*-itemset que no sea frecuente pues, a priori, no será tampoco frecuente:

```
forall itemsets c ∈ C_k do
  forall (k−1)−subsets s of c do
    if (s ∉ L_{k-1}) then
      delete c from C_k;
```

Algoritmo 3. apriori-gen: poda

En nuestra implementación mezclamos ambos pasos en uno solo, de nuevo aludiendo a la ganancia de tiempo pues es muy costoso, computacionalmente hablando, borrar un elemento de un vector. El pseudocódigo que reúne a los dos anteriores en uno solo es fácil de escribir:

```
insert into C_k
select c = {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})
      and
      ( forall (k−1)−subsets s of c do
          if (s ∉ L_{k-1}) then
            return false;
        return true; );
```

Algoritmo 4. apriori-gen: unión y poda

La primera condición, con nuestra estructura *L*, es de inmediata aplicación: dos itemsets *p* y *q* tendrán sus *(k-2)* primeros ítems coincidentes si parten del mismo ítem del conjunto L'_{k-2}; el ítem *(k-1)* tendremos que buscarlo sólo entre los descendientes en L'_{k-1} del ítem en cuestión, que también están ordenados por lo que, si hay descendientes, sólo por su posición ya sabemos si $p.item_{k-1}$ es menor que $q.item_{k-1}$. Luego para el paso *unión* sólo hemos de recorrer los ítems de L'_{k-2} y copiar ordenadamente en C'_k los ítems del bloque apropiado de L'_{k-1}.

Aunque la estructura de los datos permita búsquedas rápidas de itemsets, en la poda se deben hacer hasta *k* búsquedas en *L* de *(k-1)*-itemsets. Esto supone *k·(k-1)* búsquedas de ítems a lo largo de *L*, y para valores grandes de *k* puede suponer un gran consumo de tiempo. Por este motivo *proponemos una relajación de la poda*: consideraremos los candidatos formados por *p* y *q* que, además de contener los mismos *(k-2)* primeros ítems, verifiquen que el 2-itemset $(p.item_{k-1}, q.item_{k-1})$ es frecuente. El pseudocódigo anterior se modificará con:

```
(…)
where (…) and
      ((p.item_{k-1}, q.item_{k-1}) ∈ L_2);
```

Algoritmo 5. relajación de la poda

y nuestra implementación se limitará a buscar $p.item_{k-1}$ en L'_1 y $q.item_{k-1}$ en los descendientes del ítem anterior. Es evidente que esta relajación proporcionará más candidatos a C'_k, con un mayor consumo de memoria, pero se trata de un vector temporal que sólo contiene ítems por lo que no es de esperar un desbordamiento de la memoria del computador. Dado que la forma de contar en una transacción los *k*-itemsets frecuentes que contiene, una vez localizados en *L* los *(k-1)* primeros ítems del itemset en estudio consiste únicamente en comprobar si el siguiente ítem pertenece a C'_k, aunque esta relajación pueda proporcionar más candidatos a considerar y "buscar" en *D*, el tiempo empleado en estas búsquedas extras no será comparable al tiempo ahorrado en la creación de C'_k.

5. Resultados obtenidos

Hemos aplicado el algoritmo a distintas colecciones de sesiones utilizando un equipo con un procesador AMD Athlon XP 2000+ y 768MB de RAM, corriendo bajo Microsoft Windows XP Profesional SP2. El fichero de log usado guarda en 219,81MB las solicitudes hechas al servidor del sitio web durante 5 días del mes de junio del 2004, recogiendo 1.158.341 solicitudes, resultando válidas para el análisis 125.755 de ellas (el resto son errores del cliente o del servidor (18.283 líneas con estados 4xx y 5xx) o solicitudes de recursos auxiliares (imágenes, sonidos, ficheros multimedia, hojas de estilo, etc), con 1.533 recursos distintos. Fijando el tiempo máximo permitido entre dos solicitudes de un mismo usuario para no iniciar sesión en 10 minutos, y un máximo de 4 horas por sesión obtenemos un total de 9.041 sesiones distintas, hechas por 4.061 usuarios (direcciones IP).

Al iniciar el algoritmo las sesiones están ya codificadas y guardadas respetando el orden temporal de los accesos, con repetición de recursos si la hubiera en la sesión real. Al no estar ordenadas lexicográficamente como requiere el algoritmo lo primero que haremos será copiar el conjunto de sesiones en *D*, conjunto de transacciones en las que sus ítems están ordenados lexicográficamente y no se repiten. El tiempo empleado en esta reordenación de los datos se incluye en los tiempos expuestos en la tabla 1.

Nº transac-ciones	Nº items	Transacción más larga	Soporte mínimo (frecuencia)	Tiempo (ms)
3	11	11	1	1
22	584	298	1	47
77	21	11	1	1
140	519	80	1	31
158	526	39	1	46
182	1.157	394	1	1.281
			5	218
			10	187

Tabla 1. Tiempos de ejecución del algoritmo

Cuando un usuario accede al sitio web se comprueba si existe al menos una sesión de este usuario en la base de datos de usuarios/ Idsesiones; si es así, se leen los registros que contienen las sesiones que interesan de la base de datos de sesiones, se copian en *D* (en memoria principal) y se procede a analizarlas. En la tabla 1 se muestra un resumen de los resultados obtenidos sobre conjuntos de 3 a 182 sesiones. En la primera columna se muestra el número de sesiones guardadas de anteriores visitas del usuario al sitio web. La segunda columna muestra el número de ítems distintos presentes en *D*, el número total de recursos visitados por el usuario en todo su historial. La tercera columna muestra la longitud de la sesión con más accesos. En la cuarta columna se indica el soporte mínimo que usa el algoritmo, i.e., el número de veces que debe aparecer un k-*itemset* para ser considerado en el estudio. Por último, en la quinta columna se muestra el tiempo empleado desde que se conoce al usuario hasta obtener la estructura de datos *L*, a partir de la cual se obtendrán las predicciones.

De esta tabla 1 extraemos la información que necesita nuestro modelo para actuar sin intervención del webmaster: Es factible elaborar el árbol *L* anotando todos los ítems de todas las transacciones (soporte mínimo = 1) en la mayoría de los casos, pero se puede observar en la sexta fila de la tabla 1 un proceso que, al ejecutar el algoritmo sobre estos datos y con soporte mínimo unitario tienen ocupado al servidor algo más de 1 segundo, sólo en la creación de *L;* además de requerir un gran espacio de memoria sólo para guardar y manipular los datos de un único usuario ¿Podemos saber si un estudio será lento antes de completarlo? Si lo supiéramos al iniciar la ejecución del algoritmo podríamos detenerla inmediatamente y proceder con una modificación automática de los parámetros considerados, por ejemplo, aumentando el soporte mínimo si el problema es debido al alto número de ítems visitado por el usuario. En las líneas 7 y 8 de la tabla 1, se observa el efecto de reducir el soporte mínimo a 5 o a 10.

Los usuarios no se comportan igual en todos los sitios web. La estructura del sitio web es determinante en el modo en que es usado. Hay sitios web muy bien diseñados en que los usuarios encuentran fácilmente lo que buscan y, por tanto, pueden presentar muchas sesiones semejantes entre sí. En estos sitios será fácil estudiar su uso con soporte mínimo unitario. Otros sitios web son más caóticos, sus sesiones muy heterogéneas y el estudio de su uso será, pues, más complejo. Al iniciar el estudio del uso de un sitio web es

conveniente averiguar si podremos aplicar en tiempo real nuestro algoritmo a todos los usuarios. Para ello se realiza un estudio preliminar con los ficheros de log disponibles, aplicando el algoritmo a todos los usuarios que ya han pasado por el sitio web. Un análisis de regresión múltiple entre los datos de la tabla 1, considerando como variable dependiente los tiempos obtenidos en la ejecución del algoritmo, nos puede orientar al respecto. El análisis realizado en este trabajo nos indica que, en el sitio web estudiado, las variables que más influyen en el tiempo de proceso son el número de transacciones estudiadas, el número de items diferentes de *D* y el tamaño de L'_2.

En la figura 1 se observa una doble tendencia, lo que hace pensar que conocer el valor de esta variable es importante para predecir el tiempo que empleará el algoritmo, pero hay alguna otra variable que modifica la forma en que el número de transacciones influye sobre el tiempo.

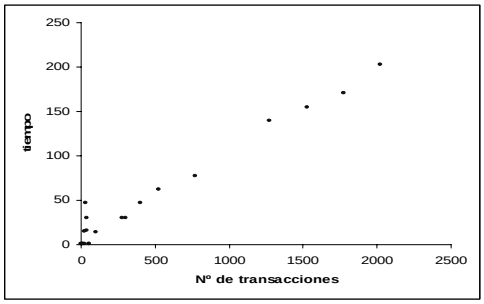

Figura 1. Transacciones frente a tiempo empleado

En la Figura 2 se muestra cómo influye el número de elementos obtenidos en L'_2 sobre el tiempo de ejecución del algoritmo.

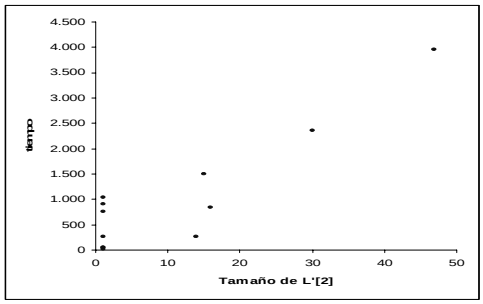

Figura 2. $|L'_2|$ frente a tiempo empleado

El número de transacciones es conocido antes de aplicar el algoritmo. El número de ítems diferentes se conoce cuando se realiza la primera lectura de *D*, i.e., muy pronto. El tamaño de L'_2 puede estimarse a partir de la fórmula:

$$|L'_2| < |C'_2| = \frac{(1+|L'_1|)\cdot|L'_1|}{2}$$

pero realmente no se conoce hasta el momento de obtener L'_2. Al aplicar el algoritmo se deben considerar estos valores para reajustar los parámetros del algoritmo, si es necesario, y conseguir así que se ejecute sin un uso excesivo de tiempo o de memoria.

Es notorio que el algoritmo es aplicable en tiempo real dado el poco tiempo que, en general, consume para poner a disposición del servidor los datos necesarios para realizar las predicciones. La relajación de la poda permite ahorrar tiempo de proceso; El estudio de los *k-itemsets* hasta el tercer nivel del árbol no supone ninguna diferencia con respecto al número de candidatos proporcionado por la poda original del algoritmo, haciendo hasta un 66% menos de operaciones de comprobación.

6. Conclusiones

Nuestra propuesta no sustituye a la de la mayoría: aprovechar el momento del día de menor actividad en el servidor para preprocesar toda la base de datos y obtener clusters, para ayudar con ello en la navegación del sitio web a un usuario anónimo. Nuestra propuesta pretende complementar el procedimiento general, dando un grado de personalización mayor sin pérdida del anonimato por parte del usuario (presenta el problema del posible cambio de IP de un usuario), pero tiene fácil solución si el usuario decide voluntariamente registrarse cada vez que entra al sitio. Al tratar al usuario como "persona" en lugar de cómo "miembro de un grupo" (como hacen la mayoria de propuestas), podremos hacer recomendaciones de páginas, aunque solo se hayan visitado una vez.

La utilización de este método permitirá al webmaster analizar el comportamiento de los usuarios del sitio de forma permanente, lo que le permitirá rediseñar en todo o en parte el sitio basándose en el Diseño Centrado en el Usuario: El diseño de la funcionalidad de un sitio debe estar dirigido por y para los usuarios [3].

En la actualidad estamos procesando nuevos ficheros de log con el objetivo de profundizar en el estudio de los parámetros del modelo, de modo que podamos dotar al algoritmo de argumentos para "protegerse" de cálculos exhaustivos. Asi mismo, se pretende automatizar el proceso de ayuda en el diseño del sitio, de modo que sea transparente para el webmaster.

Agradecimientos

Este trabajo está financiado, en parte, por el proyecto de la Junta de Comunidades de Castilla-La Mancha PBC-03-003 y por la Ayuda de la Universidad Miguel Hernández-Bancaja a cargo del proyecto RR. 1256/04.

Referencias

[1] Agrawal, R., Imielinski, T., and Swami, A., Mining association rules between sets of items in large databases, in Proc. of the 1993 ACM SIGMOD international conference on Management of data, 1993, pp. 207-216.

[2] Agrawal, R. and Srikant, R., Fast Algorithms for Mining Association Rules, in Proc.20th Int.Conf.Very Large Data Bases, {VLDB}, Santiago, Chile, 1994, pp. 487-499.

[3] Baeza-Yates, R. and Rivera, C., "Ubicuidad y Usabilidad en la Web," Revista de gerencia tecnológica informática, vol. 1, no. 2 2003.

[4] Borges, J. and Levene, M., "Data Mining of User Navigation Patterns," WEBKDD, pp. 92-111, 1999.

[5] Botella, F., Lazcorreta, E., Fernández-Caballero, A., and González, P., Personalization through Inferring User Navigation Maps from Web Log Files, in Human-Computer Interaction International, HCII 2005, Las Vegas, 2005.

[6] Cooley, R., Mobasher, B., and Srivastava, J., Web Mining: Information and Pattern Discovery on the World Wide Web, in Proc. of 9th IEEE International Conference on Tools with Artificial Intelligence, Newport Beach, CA, IEEE Computer Society, 1997, pp. 558-567.

[7] Ferr, X., Juristo, N., Windl, H., and Constantine, L. L., "Usability Basics for Software Developers," IEEE Software, vol. 18, no. 1, pp. 22-29, 2001.

[8] Fink, J., Kobsa, A., and Nill, A., User-oriented Adaptivity and Adaptability in the AVANTI project, in Designing for the Web: Empirical Studies, Redmond, WA, 1996.

[9] He, D. and Göker, A., Detecting Session Boundaries from Web User Logs, in 22nd Annual Colloquium on IR Research, Cambridge, UK, 2000, pp. 57-66.

[10] Hervás-Martínez, C., Romero, C., and Ventura, S., Comparación de medidas de evaluación de reglas de asociación, in III Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'04), Cordoba, Spain, 2004, pp. 126-133.

[11] Huang, X., Peng, F., An, A., and Schuurmans, D., "Dynamic Web Log Session Identification with Statistical Language Models," Journal of the American Society for Information Science and Technology (JASIST), vol. 55, no. 14, pp. 1290-1303, 2004.

[12] Huang, X., Peng, F., An, A., Schuurmans, D., and Cercone, N., Session Boundary Detection for Association Rule Learning Using n-gram Language Models, in 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2003, Halifax, Canada, Springer, 2003, pp. 237-251.

[13] Joachims, T., Freitag, D., and Mitchell, T., WebWatcher: A Tour Guide for the World Wide Web, in Proc. of the International Joint Conferenceon Artificial Intelligence, 1997.

[14] Pitkow, J., "In search of reliable usage data on the WWW," Sixth international conference on World Wide Web, pp. 1343-1355, 1997.

[15] Su, Z., Qiang, Y. L., and Zhang, H. J., WhatNext: A Prediction System for Web Requests using *N*-gram Sequence Models, in Proc. of the First International Conference on Web Information Systems Engineering (WISE'00), IEEE Computer Society, 2000, pp. 214-221.

[16] Sun, X., Chen, Z., Wenyin, L., and Ma, W. Y., Intention Modeling for Web Navigation, in Proc.of International Conference on Intelligent Information Technology (ICIIT-02), Beijing, China, 2002, pp. 107-112.

[17] Villena, J., González, J. C., Barceló, E., and Velasco, J. R., Minería de uso de la web mediante huellas y sesiones, Sevilla, 2002.