Técnica de minería de datos para la adaptación de sitios Web

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

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

Resumen

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

Palabras clave: Aplicaciones de minería de datos utilizando sistemas inteligentes, Computación inteligente en tiempo real para aplicaciones de Internet.

1. Introducción

Un sitio web con muchas páginas debería tener una estructura de navegación bien diseñada para que los usuarios pudieran obtener con facilidad las páginas de su interés. Este diseño suele ser realizado por expertos en contenidos o en diseño web, y a menudo no tienen en cuenta los aspectos de usabilidad y/o adaptabilidad de un sitio web [7]. El usuario real del sitio nunca interviene en este diseño; si deseara las páginas *A*, *F* y *H*, pero el experto decide que existe una relación más directa entre las páginas *A*, *B* y *C,* éstas serán las páginas con enlaces de navegación y no aquellas que realmente el usuario quería visitar. Esto ocasionará que el usuario emplee mucho tiempo en encontrar lo que busca o en el peor de los casos que no lo encuentre [3].

Un usuario no puede decidir sobre el diseño del sitio, pero muchos usuarios sí podrían decidir. Si un gran número de usuarios visitaran conjuntamente las páginas *A*, *F* y *H,* el webmaster podría definir los enlaces entre ellas, si conociera esta información, lo que mejoraría su rango de alcance. Una mejora en la facilidad de uso del sitio web que no requiere la intervención del webmaster es la predicción: presentar a cualquier usuario que visite la página *A,* enlaces a las páginas *F* y *H,* aunque realmente no se encuentren en el diseño inicial de la página *A*. Para lograrlo se debe estudiar previamente el uso del sitio, detectando si un porcentaje considerable de usuarios que solicitan el recurso *A* también solicitan los recursos *F* y *H* en la misma sesión [6].

Existen varios trabajos planteando diversas alternativas a esta solución ([5], [8] y [9]), todos ellos con un esquema común: considerar la información que aportan todos los usuarios, analizarla más o menos exhaustivamente para obtener patrones de comportamiento de los usuarios en general y resumirla de modo que con pocos recursos podamos ayudar al máximo al usuario anónimo, mediante la predicción del próximo recurso que solicitará.

El resto del trabajo se estructura como sigue. En la sección 2 se expondrá la metodología utilizada y las modificaciones realizadas al algoritmo *Apriori* para conseguir una mayor eficiencia. En la sección 3 presentaremos los resultados de la experimentación realizada con ficheros de log reales y que nos permitirán concluir que nuestro método es aplicable en tiempo real. Por último, en la sección 4 se aportan las conclusiones del presente trabajo y las líneas de trabajo futuro.

2. Metodología

La metodología propuesta se basa en el algoritmo *Apriori* [2] y en la búsqueda de reglas de asociación, por lo que definiremos una serie de conceptos previos y la implementación realizada del algoritmo para conseguir predicción en tiempo real.

2.1. Reglas de asociación y sesiones

La búsqueda de reglas de asociación se inicia con el estudio del problema de la cesta de la compra [1]. En el estudio de la navegación a través de un sitio web se puede tener en cuenta el orden en que se ha producido la navegación, lo cual no tiene importancia en el estudio del problema de la cesta de la compra. Los estudios más recientes llevan esta tendencia [4], sin embargo en este trabajo se realiza un enfoque más sencillo: estudiar el conjunto de páginas solicitadas por un mismo usuario en una misma sesión sin tener en cuenta el orden en que fueron accedidas.

El empleo de reglas de asociación en el estudio del uso de un sitio web se fundamenta en el siguiente razonamiento: si cada vez que un usuario navega por nuestro sitio sólo visita páginas que traten sobre el mismo "tema", después de varias visitas podemos estudiar qué páginas son visitadas en una misma sesión y tendremos bien diferenciados los distintos "temas" que conforman el sitio web. Por ello, un buen método para agrupar las páginas del sitio sería analizar cómo las agrupan los usuarios en sus distintas sesiones. Una vez que el usuario solicita una página que está perfectamente ubicada en un pequeño conjunto de páginas, el servidor podría informar al usuario sobre el resto de páginas que pertenecen al mismo grupo, y con qué frecuencia son visitadas en conjunción con la solicitada. La realidad no es tan perfecta, pues en cada sesión se suele visitar páginas de "temas" muy diversos, por lo que el análisis propuesto no es tan inmediato.

El principal problema al analizar la posible relación entre las páginas de un sitio web real es el elevado número de páginas distintas que tiene el sitio. Para analizar las relaciones directas entre cada par de páginas de un sitio web con 2.000 páginas necesitaríamos guardar y analizar más de 2.001.000 datos en la memoria del ordenador. Si queremos profundizar mínimamente en el análisis y estudiar la relación entre ternas de páginas necesitaremos del orden de $2x10^{12}$ datos diferentes en memoria, lo cual suele provocar bastantes problemas de desbordamiento de memoria en la implementación de los algoritmos. Por este motivo se introducen los conceptos de soporte y confianza en el estudio de reglas de asociación.

El *soporte* de una página (o grupo de páginas) es la frecuencia relativa de dicha página en el conjunto de solicitudes analizadas: si la página *A* aparece en 200 de las 1.000 sesiones analizadas se dice que A tiene un soporte del 20%. Si se fija un soporte mínimo razonable para que una página sea estudiada, se consigue un análisis con las páginas que son accedidas con más frecuencia, obteniendo tiempos de computación aceptables para realizar el análisis en tiempo real.

La *confianza* de una regla de asociación es la frecuencia relativa con que ocurre. Si la página *B* aparece en 20 de las 200 sesiones en las que está *A*, la confianza de la regla "*La página A está relacionada con la página B*" será del 10% (conf(A→B) = 10%). La confianza no cumple la propiedad conmutativa: en el ejemplo anterior, si *B* aparece en 40 de las sesiones donde está *D*, entonces la confianza de la regla B→A será del 50%.

El uso de estos dos parámetros en el análisis del uso del sitio web permitirá realizar un análisis de un sitio web completo en un tiempo razonable.

Si un usuario accede a una página que no tenga soporte mínimo no recibirá información sobre posibles enlaces a otras páginas que posiblemente desea o suele visitar. ¿Cómo podríamos proporcionar esta información al usuario? La solución viene dada por el uso del concepto de sesión. Las sesiones no son más que subconjuntos ordenados de páginas solicitadas al servidor. En el análisis de uso solo se estudian los datos disponibles sobre las sesiones de este usuario, y no el conjunto global de páginas visitadas, lo que reduce en el 99% de los casos el número total de páginas a estudiar, sin renunciar al estudio de todas las páginas visitadas por el usuario.

¿Cómo podemos averiguar la identidad de un usuario para individualizar su análisis si se trabaja con usuarios anónimos? Solo se puede conocer la dirección IP desde la cual un usuario anónimo accede al sitio web, pero esto permitirá agrupar las

páginas visitadas por un usuario en una misma sesión.

2.2. Implementación del algoritmo

El método propuesto requiere aplicar un 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; el algoritmo *Apriori* parecía el apropiado. Se eligió una estructura de datos eficiente para implementar el algoritmo, puesto que el uso de matrices dinámicas suele ocasionar la copia de la matriz completa al ejecutar las operaciones de inserción y eliminación de un elemento de la matriz en memoria, lo que aumentaría el tiempo final del análisis.

2.2.1. Notación empleada

Se ha mantenido la notación original del algoritmo *Apriori* en la medida de lo posible. 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 *items*, que se denotarán por α_j. De este modo una transacción puede verse como un conjunto ordenado de items

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

siendo n_i el número de items 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* items que aparecen simultáneamente en alguna transacción.

El principal problema que presenta la aplicación de este algoritmo reside en el número de accesos a *D*. La búsqueda de reglas de asociación suele realizarse sobre grandes bases de datos, por lo que en la aplicación real del algoritmo, el conjunto *D* no puede alojarse en la memoria principal debido a su tamaño. Se han abordado muchas soluciones para reducir el número de veces que se accede a *D*. En el método aquí propuesto, veremos que el conjunto *D* no tiene un tamaño excesivo y puede almacenarse en memoria principal.

A partir de la primera lectura de *D* se crea la primera lista de *candidatos*, *C[1]* ó C_1, que contendrá cada uno de los ítems presentes en *D* junto al número de veces que aparece. A continuación se eliminan de C_1 todos los ítems que no superen el soporte mínimo fijado para el análisis, llamando al conjunto obtenido *conjunto de 1-itemsets frecuentes,* y denotándolo por *L[1]* ó L_1.

Una vez conocido *L[1]*, si contiene algún elemento, se procede a obtener *C[2]*, el conjunto de candidatos a ser 2-*itemsets* frecuentes, con la idea que da nombre al algoritmo: si un ítem no está en L_1 por no ser frecuente, no es candidato a formar un 2-*itemset* frecuente. Aprovechando el orden lexicográfico de los ítems se puede ahorrar recursos: si se anota en *C[2]* el candidato *AB*, no será necesario anotar el candidato *BA*, pues la posible relación entre ambos ya se estudia con el primer candidato. Una vez reservada la memoria con *C[2]* para estudiar todos los posibles 2-*itemsets* frecuentes, se procede a una segunda lectura de *D*, extrayendo todos los subconjuntos de 2 elementos de cada transacción. Cuando se trate de un candidato, se incrementará su frecuencia. Una vez terminada la lectura de *D* se guardan en L_2 sólo los 2-*itemsets* que son frecuentes (los que tienen soporte mínimo).

Si L_2 contiene algún elemento se procede a la obtención de C_3 de forma análoga a como se obtuvo C_2: sólo serán candidatos a 3-*itemsets* frecuentes aquellos 3-*itemsets* que puedan derivarse de L_2 x L_2. Aprovechando el orden en que están escritos se genera un conjunto de candidatos bastante ajustado (se genera un 3-*itemset* por cada par de 2-*itemsets* cuyo primer elemento coincida), y después se eliminan del conjunto aquellos que contenga algún 2-*itemset* no frecuente. Una vez generados los candidatos se vuelve a leer *D*, se cuentan las apariciones de todos los candidatos y se guardan los candidatos frecuentes en L_3. Los niveles posteriores se tratan del mismo modo, hasta encontrar un nivel L_k sin elementos, lo cual indica la finalización del algoritmo.

Las reglas de asociación se obtienen a partir de los conjuntos de *k-itemsets* frecuentes, aplicando el concepto de confianza mínima: una

regla será "fuerte"' si supera la confianza mínima.

2.2.2. La estructura de datos

La estructura elegida para implementar el algoritmo pretende reducir al máximo el número de elementos a almacenar. Por ello se guardan 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.

Este modo de almacenar los datos aprovecha y respeta el orden de las transacciones, 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 $L'_1, L'_2, \ldots, L'_{k-1}$.

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.

2.2.3. El algoritmo

Utilizaremos el algoritmo *Apriori* original [2] sin modificación alguna:

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

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)* items formen parte de dos *(k-1)-itemset* frecuentes distintos:

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

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 do
  forall (k-1)-subsets s of c do
    if ($s \notin L_{k-1}$) then
      delete c from C_k

Algoritmo 3. apriori-gen: poda

La implementación realizada del algoritmo conlleva ciertas modificaciones que no producen ningún cambio en el resultado final. Sólo se modifica el número de candidatos producido; podría ocurrir que se obtuvieran más candidatos a cambio de ahorrarnos la mayoría de cálculos que propone la poda. Unir en un solo paso la unión y la poda ha mejorado la eficiencia de la implementación, disminuyendo el número de inserciones y eliminaciones de candidatos. La función apriori-gen implementada tiene el siguiente pseudocódigo:

insert into C_k

select $c = \{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}\}$

from $L_{k-1}\ p, L_{k-1}\ q$

where $(p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2},$
$p.item_{k-1} < q.item_{k-1})$

and

$(p.item_{k-1}, q.item_{k-1}) \in L_2$;

Algoritmo 4. apriori-gen: unión y poda relajada

3. *Apriori* con relajación de poda

En un primer estudio teórico se previó que el número de candidatos obtenidos con el algoritmo modificado sería muy superior al obtenido utilizando el algoritmo original. Sin embargo, al experimentar con datos de ficheros de log reales se observó que las sesiones producidas por un mismo usuario de nuestro sitio web presentaban patrones muy marcados y eran muy homogéneas, en cuanto a las páginas que contenían. Esto permitió realizar un estudio completo de todas las páginas, es decir, sin tener que considerar soporte mínimo ni confianza y permitió generar predicciones para cualquier página visitada anteriormente por el usuario.

Se obtuvieron tiempos inferiores a los 30 milisegundos en todas las ejecuciones del algoritmo modificado, salvo en el análisis de las sesiones realizadas por un agente que realizó una visita al sitio web (que duró 1.5 segundos) y el análisis de seis direcciones IP correspondientes a ordenadores públicos, con sesiones muy heterogéneas, cuyos análisis se realizaron en tiempos de entre 30 y 60 milisegundos.

3.1. Datos utilizados

Para probar la eficiencia del algoritmo modificado se preprocesaron las solicitudes hechas a un servidor web durante los 22 últimos días del mes de julio del 2004. En la lectura del fichero de log se observaron 2.977.380 solicitudes a recursos, 94.442 de las cuales no pudieron ser servidas (errores en la petición y/o peticiones a páginas no existentes). 2.437.378 peticiones fueron realizadas a recursos auxiliares a las páginas solicitadas por los usuarios (imágenes, ficheros de estilo, archivos multimedia, etc.) En el análisis se emplearon 445.560 solicitudes, conteniendo 2.158 recursos diferentes.

Para este estudio se definió una sesión de usuario como la secuencia de solicitudes realizada desde la misma dirección IP, de modo que entre una petición y la siguiente no transcurrieran más de 15 minutos, y entre la primera petición y la última de una sesión no transcurrieran más de 8 horas. Así se obtuvieron un total de 30.559 sesiones realizadas desde 11.394 direcciones IP; obteniendo así información sobre los accesos de 11.394 usuarios distintos.

3.2. Resultados obtenidos

Solo se sugerirán al usuario aquellos enlaces que mantengan una relación más fuerte con la página solicitada. En principio no se consideraron los casos en que el usuario había solicitado alguna otra página en la misma sesión. Por ese motivo sólo es necesario aplicar el algoritmo hasta el nivel $k=2$ y así podemos sugerir al usuario un enlace hacia las páginas que muestren mayor confianza en el árbol L.

Para dar información sobre todas las páginas visitadas por el usuario en otras sesiones, se fijó el soporte mínimo en 1. En la Tabla 1 se muestra el tiempo necesario para ejecutar el algoritmo completo y presentar un conjunto de enlaces sugeridos utilizando un ordenador de sobremesa convencional (Pentium 4, 1.7GHz, 512MB, Windows XP Pro).

Tiempo (milisegundos)	**Frecuencia**	**(%)**
$t<15$	9.079	79.7
$15 \le t<30$	2.305	20.2
$30 \le t<45$	4	0.0
$45 \le t<60$	2	0.0
$t \ge 60$	1	0.0

Tabla 1. Tiempos ejecución algoritmo modificado

Al iniciar el algoritmo las sesiones ya estaban codificadas y guardadas en el orden temporal de los accesos. Al no estar ordenadas lexicográficamente como requiere el algoritmo, primero se copia el conjunto de sesiones en D. El tiempo empleado en esta reordenación de los datos se muestra en la tabla 2.

Nº transac-ciones	Nº items	Transacción más larga	Soporte mínimo (frecuencia)	Tiempo (ms)
3	11	11	1	1
11	13	10	1	1
20	51	26	1	1
30	139	57	1	49
51	127	21	1	16
86	602	222	1	1,540
			5	467
			10	187
275	73	52	1	30
364	114	22	1	16

Tabla 2. Tiempos ejecución 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, en cuyo caso se leen los registros que contienen las sesiones en estudio, se copian adecuadamente en *D* (en memoria principal) y se procede a analizarlas. En la tabla 2 se muestra un resumen de los resultados obtenidos sobre conjuntos de 3 a 364 sesiones, a partir de los cuales se han ajustado ciertos parámetros del algoritmo modificado. En la primera columna se muestra el número de sesiones guardadas de anteriores visitas del usuario al sitio web. Este dato es conocido antes de aplicar el algoritmo. 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. 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.

Se puede elaborar el árbol *L* anotando todos los ítems de todas las transacciones (soporte mínimo = 1), pero se puede observar en la sexta fila de la tabla 2 un proceso que, al ejecutar el algoritmo sobre estos datos y con soporte mínimo unitario tienen ocupado al servidor algo más de 1.5 segundos, sólo en la creación de *L*. En la líneas 7 y 8, se observa el efecto de reducir el soporte mínimo a 5 o a 10.

4. Conclusiones

El algoritmo propuesto se puede utilizar en un sitio web en tiempo real, pues los tiempos de respuesta para proporcionar una sugerencia de página a un usuario son aceptables para cualquier usuario web, tendiendo en cuenta las peculiaridades de estos entornos.

La relajación de la poda permite disminuir el 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, realizándose hasta un 66% menos de operaciones de comprobación.

La modificación propuesta para el algoritmo *Apriori* consistente en una relajación de la poda permite obtener resultados de predicciones en tiempo real, para personalizar y adaptar la navegación anónima de los sitios web. Cualquier usuario que solicite la página *A* de un sitio web, recibirá, en una sesión posterior, sugerencias de enlaces a páginas que, con mayor frecuencia, también visitó junto a *A*.

Si un usuario solicita primero la página *A* y luego solicita la página *B*, el algoritmo deberá informarle qué páginas acompañaron con más frecuencia al par *AB* en sus sesiones previas, y no sólo las páginas que guardan relación con *B*. En esta línea es donde actualmente estamos mejorando el algoritmo para dotarle de la capacidad para ajustar ciertos parámetros de forma automática, por ejemplo, aumentando el soporte mínimo, con objeto de mantener la eficiencia en tiempo real del mismo.

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 *Proceedings 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] 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.

[4] Bucklin, R. E. and Sismeiro, C., "A Model of Web Site Browsing Behavior Estimated on Clickstream Data," *Journal of Marketing Research*, vol. 40, no. 3, pp. 249-267, June2001.

[5] Chen, X. and Zhang, X., "A Popularity-Based Prediction Model for Web Prefetching," *IEEE Computer*, pp. 63-70, Mar.2003.

[6] Ng, R. T. and Han, J., Efficient and Effective Clustering Methods for Spatial Data Mining, in *20th International Conference on Very Large Data Bases,* Santiago, Chile: 1994, pp. 144-155.

[7] Peñarrubia, A., Fernández-Caballero, A., and González, P., Portales Web Adaptativos: Una propuesta de futuro, in *V Congreso Interacción Persona-Ordenador, Interaccion 2004,* Lerida: 2004, pp. 314-320.

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

[9] 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.