Discovering Web Access Orders with Association Rules

Charles Lo[1] and Vincent Ng[1]

[1]Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

cschamlo@comp.polyu.edu.hk, cstyng@comp.polyu.edu.hk

ABSTRACT

In the past few years, the use of World-Wide-Web (WWW) has grown exponentially. It is important for companies to analyze the behaviors of their customers so as to have better profit and services. This paper explores the technique of association rules to discover the access patterns of WWW users. In order to support our proposed algorithms, there is a pre-mining phase which filters and transforms WWW access logs into a database of access transactions. Our first algorithm, Extended Apriori algorithm (EAA), is a variant of the infamous Apriori algorithm. This algorithm is modified to handle the item order in the counting and generation steps of the candidate access patterns. However, the EAA does not deal with the minimal viewing time constraint directly. In the OPM algorithm, we propose to consider how to reduce the number of database scans and exploit the minimal viewing time constraint during the candidate itemset generation. In our experiments, the results showed that the OPM algorithm took less than half of the time required by by the EAA [1].

1 INTRODUCTION

The problem of discovering association rules was first introduced in [1]. It is to find the possible associations between items when a user specifies a minimum support (s) and a minimum confidence (c). For example, one possible association rule can be "there are 10% of customers who buy bread and 20% of them also buy butter." Those rules are usually used to make better stocking and discounting decisions.

Recently, different knowledge discovery techniques have been attempted in the area of WWW because of its rapid growth and popularity. In [2], the original sequence of log data is converted into a set of maximal forward references. From these references, the full-scan and selective-scan algorithms are developed to determine the large traversal patterns of users. In the selective-scan algorithm, it can reduce the number of database scans by utilizing the candidate reference sequences. Borges and Levene proposed to use an incremental approach to find out the traversal patterns of users [3]. In their work, the concept of confidence and support for composite associate rules are redefined [3]. Two algorithms, Modified Depth-First Search Algorithm and Incremental Step Algorithm are proposed.

There may be many traversal patterns of our WWW users. Not all results are interesting. One suggestion in measuring a user's interest in the web pages is the viewing time of the pages [4]. Shahabi introduced a novel path clustering method based on the similarity of the history of user navigation. Therefore, instead of generating all the frequent patterns, we propose to allow users to define their own interests

- the number of web pages in a pattern, and
- the minimum viewing time of the pages in the pattern.

By exploiting, these information, it would improve the mining efficiency significantly.

The two new parameters above can be considered as new constraints in mining association rules. The first parameter is the size constraint which limits the number of items (pages) in each pattern. The second parameter is the viewing constraint which is an inequality of quantitative values associated with the items. Recently, Srikant et. al. [10] have proposed the *MultipleJoin* algorithm and the *Direct* algorithm to discover association rules which satisfy a given boolean expression over the items of a database. The expression is used to indicate the existence or non-existence of the items. Only association rules satisfying the expression are output. However, quantitative items are not supported. In [9], the CAP algorithm is developed to deal with the constrainted association queries. The algorithm shows how to optimize the mining process when the input constraints are either *anti-monotone* or *succinct* or both.

In this paper, we suggest two approaches in applying association mining to discover WWW access patterns. The first approach is to modify the infamous Apriori algorithm to handle the item order in the counting step and the candidate itemset generation step. However, the Extended Apriori algorithm (EAA) cannot deal with the minimal viewing constraint for the access patterns effectively. In the OPM algorithm, we propose to consider how to reduce the number of database scans and exploiting the minimal constraint during itemset generation.

2 ACCESS TRANSACTIONS

For most WWW servers, there are access logs to keep track of their utilizations. An example of such as a log

[1]The work of the authors were supported in part by the Hong Kong RGC Grant: Q120.

```
158.132.24.196 - - [07/Oct:12:37:34 ] "GET /icons/blank.gif HTTP/1.0" 200 148
158.132.24.196 - - [07/Oct:12:37:34 ] "GET /icons/back.gif HTTP/1.0" 200 216
158.132.24.196 - - [07/Oct:12:37:34 ] "GET /icons/folder.gif HTTP/1.0" 200 225
158.132.24.196 - - [07/Oct:12:37:34 ] "GET /icons/image2.gif HTTP/1.0" 200 309
158.132.24.196 - - [07/Oct:12:37:34 ] "GET /icons/text.gif HTTP/1.0" 200 229
158.132.24.196 - - [07/Oct:12:42:43 ] "GET /yuen.htm HTTP/1.0" 403 288
158.132.24.196 - - [07/Oct:12:42:57 ] "GET /index.html HTTP/1.0" 403 290
158.132.24.196 - - [07/Oct:12:45:12 ] "GET /index.html HTTP/1.0" 403 290
158.132.24.196 - - [07/Oct:12:45:15 ] "GET /yuen.htm HTTP/1.0" 403 288
158.132.24.187 - - [07/Oct:14:12:59 ] "GET /_vti_inf.html HTTP/1.0" 403 293
158.132.24.187 - - [07/Oct:14:13:00 ] "GET /_vti_inf.html HTTP/1.0" 403 293
158.132.24.187 - - [07/Oct:14:13:00 ] "GET / HTTP/1.0" 403 280
158.132.24.196 - - [07/Oct:14:28:38 ] "GET / HTTP/1.0" 200 7122
158.132.24.196 - - [07/Oct:14:28:39 ] "GET /Images/news.jpg HTTP/1.0" 200 3507
```

Figure 1: WWW access log.

is shown in Figure 1. It is difficult to apply any mining technique directly on these logs as their formats may vary. More importantly, each line in the log file only indicates one access of a particular Web page. This is only interesting from a single server view, but not for individual users. It is more useful if we can identify Web pages that are accessed by an individual user, or even a session done by an individual user. A normal WWW access log records the information of client IP address, name of html pages and the access time on those pages. Based on these information, our identification method is based on the followings assumptions

- The server can be a WWW server or a proxy server as long as access logs can be recorded.
- Only one user is active on a single IP address at a time.
- When a user has been idle for a long time (say 1 hour), the current session ends.
- Local cache in the client machines have been disable and every Web page access will need to be served by the server.

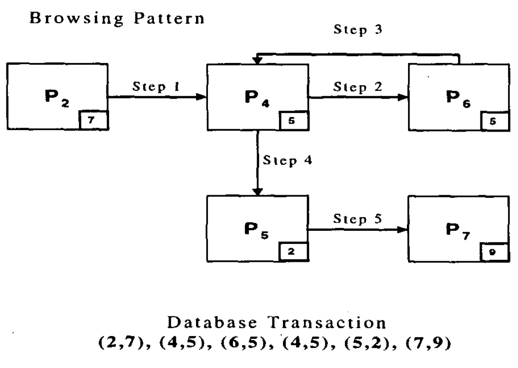

Figure 2: An access pattern.

Each session of a user is transformed into an *access transaction*. For example, in Figure 2, User A accessed the web pages in the sequence of $P_2 \rightarrow P_4 \rightarrow P_6 \rightarrow P_4 \rightarrow P_5 \rightarrow P_7$. The access sequence forms a list of order pairs in the format of (P_i, d_{ij}) where P_i is the page label (number) and d_{ij} the viewing time of the page. Each order pair of T_i is represented as W_{ij}. For the example in Figure 2, the sequence of the access transaction is {(2,7), (4,5), (6,5), (4,5), (5,2), (7,9)}.

From the access log, we identify the sessions, transform them into access transactions and then prepare for mining of user behaviors. Suppose that there are two access transactions, T_i and T_l, and S_k^i and S_k^l are two sub-sequences of size k in the transactions, respectively. We can then have the following definition.

Definition 1 *S_k^i and S_k^l are are access equivalent iff $\exists x, y$ such that $P_{\theta(i,x+a)} = P_{\theta(l,y+a)}$ where $a = 0, 1, 2, \ldots, k-1$ and $\theta(i,x)$ represent the page label for W_{ix}.*

In Figure 3, we define the terminologies which will be used in later sections.

T	all access transaction in the database
T_i	i^{th} access transaction
W_{ij}	entries of T_i as $(P_{\theta(i,j)}, d_{ij})$
$\theta(i,j)$	function to find the page label of the j^{th} element of transaction T_i
$P_{\theta(i,j)}$	the j^{th}-page accessed by T_i
$d_{i,j}$	viewing time of the page in W_{ij}
S_k^i	a subsequence of size k in T_i
$AP(S_k^i)$	set of access patterns (page labels) equivalent to S_k^i
A_k	an acess pattern of size k
L_k	set of frequent patterns of size k
C_k	set of candidate patterns of size k

Figure 3: Some definitions.

3 PROBLEM DESCRIPTION

The support count of an access pattern is the number of the access equivalent patterns in the access transactions. As in [1], when the count exceeds the minimum support (s), it is called as a *frequent access pattern.* That is,

Definition 2 *$AP_k \equiv AP(S_k^i)$ is frequent if* $\mid AP(S_k^i) \mid > s \times \mid T \mid$

Next, we formally define our first problem statement.
Problem 1 *Consider a database of access transactions as $T_1, T_2, \ldots, \ldots$. We would like to extract all the frequent access patterns AP_k for all k's.*

As discussed in the Introduction section, not all access patterns are interesting. Our second problem statement is to allow a user in specifying two constraints. The first one is the number of web pages in a pattern, and the other is the minimum viewing time of the pages in the pattern. With a given size of a pattern, it allows us to avoid the generation of non-interesting itemset. Hence, there will be less database scans. The viewing time constraint reveals the interest of users in the pages and allow us to prune away unnecessary candidate sets. With these constraints, we define our second problem below.

Problem 2 *As in the first problem statement, consider the following access transactions*

$$T_i = W_{i1}, \ldots, W_{im}$$
$$W_{ij} = (P_{\theta(i,j)}, d_{ij})$$

in a database. We would like to extract all the frequent patterns of size k (AP_k's) where for each equivalent access pattern S_k^i of a AP_k, we have

$$\sum d_{ij} \geq F.$$

F is a threshold value representing the minimal viewing time of the pattern.

In fact, Problem 2 is the same as Problem 1 except with the introduction of the viewing time constraint.

4 EXTENDED APRIORI ALGORITHM

We first present the Extended Apriori Algorithm (EAA) for finding all frequent access patterns. It is a modified version of Apriori which considers the ordering of the pages during the mining. The viewing constraint will only be applied after all the frequent pattern are generated. The EAA has two phases.

- Find all the access patterns S's whose supports are greater than the minimum support s and these patterns are called frequent access pattern.
- Evaluate the total viewing time of the mined patterns and check if they satisfy the viewing constraint.

Obviously, the first phase is the core of the algorithm. In order to generate the large patterns, L_k, we need to consider the order of pages (items) in generating the candidate set C_k. In the original Apriori algorithm, the order of items within a itemset is ignored. The candidate set generation part of the algorithm is modified as in Figure 4. In the procedure, it ensures that C_k is the superset of L_k with the consideration of the pages ordering within the pattern.

After C_{k+1} is generated, we have to prune those pattern whose support is less than the minimum requirements. So, we have to scan the database to count the number of support of each pattern in C_{k+1} by storing them into a hash tree. After the hashing, we get the desired pattern in L_{k+1}. The algorithm stops when reaching the size constraint set by the user. The second phase is simply done by checking the $\sum d_{i,j}$ of each pattern S_k in L_{k+1} if they exceed F.

```
Given a set of $L_k$, we want to form the candidate itemset $C_{k+1}$.

1. For each pair of access patterns in $L_k$ as $A_k$ and $A_k$' where
    • $A_k = P_{11}, \ldots P_{1k}$, and
    • $A_k' = P_{21}, \ldots P_{2k}$.
    • if $((P_{11} = P_{21})$ and $\ldots (P_{1(k-1)} = P_{2(k-1)}))$ then
        – C' = $(P_{11}, \ldots, P_{1(k-1)}, P_{1k}, P_{2k})$
        – C" = $(P_{11}, \ldots, P_{1(k-1)}, P_{2k}, P_{1k})$
        – $C_{k+1} = C_{k+1} \cup$ C' $\cup$ C"
```

Figure 4: Candidate pattern generation

5 OPM ALGORITHM

In our work, only patterns of size k which is specified by users and their viewing times longer than F are useful. We try to exploit these two constraints to achieve better efficiency. This is done by looking into the following aspects.

1. Reduce the number of scanning the entire database (size constraint), and
2. Reduce the number of candidate pattern (viewing time constraint).

With the new constraints, we developed the *generating sequence* and *max_min pruning* techniques, respectively. These techniques are embedded in our second algorithm, *Order Pattern Mining* (OPM).

5.1 Generating Sequence

The idea is to generate C_k from $L_{k/2}$ if ever possible. With this approach, we can skip the generation steps of $L_{k/2+1}, L_{k/2+2}, \ldots$, and L_{k-1}. In the best case, it takes only $\log_2 k$ steps in finding L_k instead of the k steps in Apriori. When k is not a multiple of 2, we first find out the L's up to L_w where $2^w < k < 2^{w+1}$. We then find L_k by utilizing L_w, $L_{w/2}$, $L_{w/4}$, …, etc. For example, if k is 31, instead of taking 31 steps to find out L_{31}, we will only take 9 steps,

$$L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow L_6 \rightarrow L_7 \rightarrow$$
$$L_{14} \rightarrow L_{15} \rightarrow L_{30} \rightarrow L_{31}$$

```
Given the generating sequence $GS : gs_1, gs_2, \ldots$, and
let $L_{s_i}$ be the current large itemset.

• if $((gs_i - gs_{i-1}) \neq 1)$ then
    – For any two patterns $A_{s_i}$, $A_{s_i}' \in L_{s_i}$
        * if $((A_{s_i} \cap A_{s_i}') = \phi)$ then
          { $c = A_{s_i} \cup A_{s_i}'$; Insert $c$ into $C_{s_{i+1}}$; }
  else
    – Apply the candidate generation procedure as
      in Figure 4.
```

Figure 5: Mining under generating sequence GS.

For a given integer value k, the generating sequence GS: $(gs_1, gs_2, \ldots)$ is obtained by the following formula

$$gs_{i-1} = \begin{cases} gs_{i/2} & \text{if } gs_i \text{ is even} \\ gs_{i-1} & otherwise \end{cases} \quad (1)$$

where gs_1 equals to 1 initially. The resultant sequence GS is then used to guide the generation of the candidate pattern. The algorithm which make use of the generate sequence is shown in Figure 5.

5.2 Max_Min Pruning

The performance of the Apriori algorithm is mainly depended on two integers X and Y, where X is the number of the database scanning and Y is the number of candidate itemsets in each iteration. In the previous section, we have discussed how to reduce the value of X rapidly.

Here, we will discuss how to reduce the value of Y. There are two pruning possibilities in reducing the size of the candidate pattern. The first situation is when C_k is generated from L_{k-1}. In this case, for any candidate pattern c if one of its $(k-1)$-subsets is not in L_{k-1}, c should be removed from C_k. The second situation is when C_k is generated from $L_{k/2}$. Similarly, for any pattern that some of its $k/2$-subsets is not in $L_{k/2}$, c should be removed also. There is a trade off when C_{2k} is generated by L_k. In general, the number of candidate patterns is larger when compare to the case that C_{2k} is generated by L_{2k-1}. To offset the problem, we can utilize the viewing time constraint in our pruning.

In the first database scanning to find L_1, we can obtain the maximum viewing time of individual pages and form the list $maxlst_1$. After we obtained L_2, the maximum viewing time of each pattern is the maximum sum of the viewing times of these two pages in different transactions. The list $maxlst_2$ $(\{V_{2_1}, V_{2_2}, \ldots\})$ is then re-sorted in a descending order according to the maximum values. This procedure is repeated at the end of each iteration and forms one $maxlst_i$ everytime. The idea of Max_Min pruning is to allow us to remove a candidate access pattern whose maximum is less than the minimum viewing time set (F), even before the database scanning. The pruning procedure of Max_Min is shown in Figure 6.

```
Given $maxlst_i$ as { $V_{i_1}, V_{i_2}, \ldots$ }.

1. if the sum of the $V_{i_1} + \ldots + V_{i_k}$ is greater than $F$
   then
     • set counter $c = (\lceil (k-i)/i \rceil) + 1$
     • set $maxsum_i$ equal to the sum of $V_{i_1} + \ldots +$
       $V_{i_{c-1}}$ items
     • Repeat
         – increase $c$ by 1;
           until $V_{i_c} + maxsum_i \leq F$

         – remove all the items whose index is larger
           than $c$ from $L_i$ and form $L_i$'
```

Figure 6: Min_pruning for L_i at iteration i

6 EXPERIMENTS

In this section, we compare the two proposed algorithms, EAA and OPM. We notice that it is not sufficient to determine the efficiency of the algorithms by only counting the number of steps in generating the candidate itemsets. For the OPM, it takes fewer steps to find L_ki, but at the same time, it takes extra effort for Max_Min pruning process of the large itemsets at each step. For example, in pruning L_k, we need to verify if all of the (k/2)-subsets for an element in C_k are in $L_{k/2}$. If items are totally associated with each other, the number of comparisons will be ${}_kC_{k/2} \times_n C_{k/2}$ at each mining step. On the other hand, for EAA, there will only be ${}_kC_{k-1} \times_n C_{k-1}$ comparisons at each step. However, EAA generates the frequent patterns incrementally, and a post-mining process for the evaluating of the viewing time inequality is required. Obviously, the OPM algorithm will out-perform EAA by exploiting the two new constraints.

In order to assess the performance of the algorithms, we have implemented the EAA and the OPM algorithm in C++. We ran our preliminary experiments on a SUN SPARC workstation with 32 Mbytes of main memory. There are two sets of experiments. The first set is based on a fixed number of transactions (1000) generated by a program modified from the pseudo-data generation program in [1]. The specific pattern size is 8, the minimum threshold of the viewing time is 30 minutes and support is set to be 5%. We are interested in measuring the time to discover the constrainted access pattern under different numbers of pages among the transactions. The results are shown in Figure 7.

The second set of experiments is similar to the first

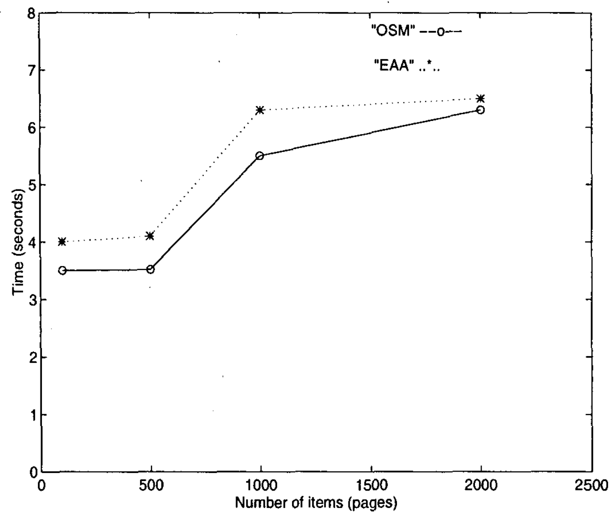

Figure 7: Discovery time for a database of 10^4 access transactions but varying number of pages.

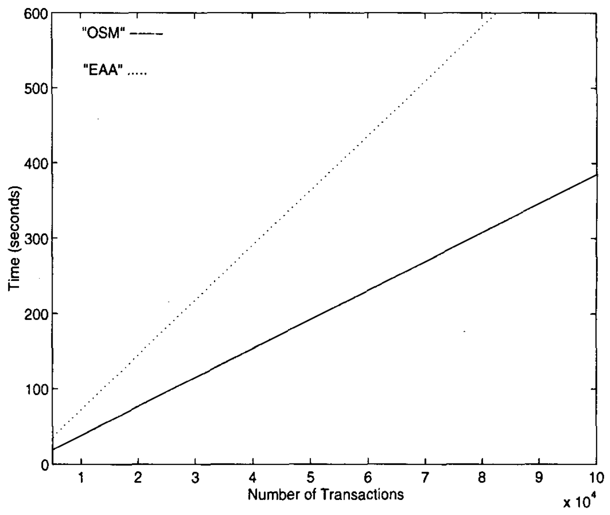

Figure 8: Discovery time for different number of database transaction.

set, except that we have fixed the number of pages and experimented with different database sizes. The corresponding results is shown in Figure 8. The results show that the OPM algorithm are more efficient and also the number of transactions has a larger impact than the number of pages in the access log database. This is expected since we have to scan the whole database everytime for each iteration. The OPM algorithm skips many unnecessary iterations. Therefore, the more transactions in the database, the better the algorithm can save the computational effort. This can easily observed in Figure 8, and the OPM algorithm generally speeds up the mining process for more 100% when compared with OAA.

7 CONCLUSIONS

In this paper, we present two algorithms for the mining of access patterns in WWW pages. Before the mining process, there is a pre-mining phase which filters and transforms WWW access logs into a database of access transactions. Our first approach is to modify the infamous Apriori algorithm to handle the item order in the counting and generation steps of the candidate access patterns. However, the Extended Apriori algorithm does not deal with the minimal time constraint for each access pattern effectively. In the OPM algorithm, we propose to consider how to reduce the number of database scans and exploiting the minimal constraint during itemset generation. In our experiments, the results showed the OPM algorithm took less than half of the time required by by the EAA

REFERENCES

[1] R. Agrawal, R Srikant. *Fast Algorithm for Mining Association Rules in Large Databases*, In Proc. VLDB'94, Santiago, Chile, September 1994.

[2] Ming-Syan Chen, Jong Soo Oark, Philip S.Yu. *Data Mining for path Trversal Patterns in a Web Environment*, In Proc. ICDCS'96, pp. 385 - 392.

[3] Jose Borges, Mark Levene. *Mining Association Rules in Hypertext Databases*, In KDD'98, pp. 149 - 153.

[4] Cyrus Shahabi, Amir M.Zarkesh, Jafar Adibi, Vishal Shah. *Knowledge Discovery from Web-Page Navigation*, In IEEE RIDE'1997.

[5] Roberto J. Bayardo Jr. *Efficiently Mining Long Patterns from Databases*, In Proc. of the ACM-SIGMOD

[6] R. Agrawal, R Srikant. *Mining Sequential Patterns,* In Proc. International Conference on Data Engineering, 1995, pp. 3-14.

[7] R. Agrawal, T Imielinski, A Swami. *Mining Association Rules between Sets of Items in Large Databases*, In Proc. ACM-SIGMOD International Conference on Management of Data, 1993, pp. 207-216.

[8] RJ Miller and Y Yang. *Association Rules over Interval Data*, In Proc. ACM-SIGMOD 1997, pp. 452-461.

[9] R Ng, L Lakshmanan, J Han, A Pang. *Exploratory Mining and Pruning Optimizations of Constrained Association Rules*, In Proc. ACM-SIGMOD 1998.

[10] R Srikant, Q Vu and R Agrawal. *Mining Association Rules with Item Constraints*, KDD'97, pp. 67-73.

[11] R Srikant and R Agrawal. *Mining Quantitative Association Rules in Large Relational Tables*, In Proc. ACM-SIGMOD 1996.

[12] T Zhang, R Ramakrishnan and M Livny. *BIRCH: An Efficient Data Clustering Method for Very Large Databases*, In Proc. of the ACM-SIGMOD, Montreal, Canada, 1996.