International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:4, 2007

Web Log Mining by an Improved AprioriAll Algorithm

WANG Tong[1] HE Pi-lian[2]

***Abstract*—**This paper sets forth the possibility and importance about applying Data Mining in Web logs mining and shows some problems in the conventional searching engines. Then it offers an improved algorithm based on the original AprioriAll algorithm which has been used in Web logs mining widely. The new algorithm adds the property of the UserID during the every step of producing the candidate set and every step of scanning the database by which to decide whether an item in the candidate set should be put into the large set which will be used to produce next candidate set. At the meantime, in order to reduce the number of the database scanning, the new algorithm, by using the property of the Apriori algorithm, limits the size of the candidate set in time whenever it is produced. Test results show the improved algorithm has a more lower complexity of time and space, better restrain noise and fit the capacity of memory.

***Keywords*—**Candidate Sets Pruning, Data Mining, Improved Algorithm, Noise Restrain, Web Log

I. Introduction

Web mining deals with the data related to the Web, they may be the data actually present in Web pages or the data concerning the Web activities. The Web can be viewed as the largest unstructured data source available, although the data on the Web sites, which composed them, is structured. This presents a challenging task for effective design of and access to Web pages. Web data can be classified into the following categories [1]:

1)Content of actual Web pages is that to be presented to the users.
2)Intrapage structure includes the HTML / XML code for the pages.
3)Interpage structure is the actual linkage structure between Web pages.
4)Data usage describes how Web pages are accessed.
5)User profiles include demographic and registration information obtained about users. These could also include information found in cookies.

This paper mainly to show the improved AprioriAll algorithm, which has lower complexity of time and space than that of the original one, can be used in Web Mining widely .

F. A. WangTong (1967-), male, lecturer, master, Ph.D candidate of the Electronic & Information College in TianJin University, now engaging in Data Mining, OLTP, OLAP, ActiveX, Web Service and COM+.
Phone: 086-022-81333525 Email: WangTong67123@eyou.com
S. B. HE Pi-lian (1943-), male, professor, director of Ph.D candidates of the Electronic & Information College in TianJin University

II. Mining with Web Logs

A. Web Logs

Web usage mining looks at logs of Web access. General access pattern tracking is a type of usage mining that looks at Web pages visit history [4]. Web Servers record access information as a click-stream-data into log files. Whenever a Web page is clicked, corresponding data will be generated and recorded. There are valuable information in the profile, such as the access patterns of users, the types of explores and operating system being used, as well as how long a user was on line. Parts of log file of Microsoft Internet Information Server 5.0 is shown as following and its format fits into the format addressed by W3C extension norm[3].

```
-----------------------------------------------------------------------
-
    2003-08-03 06:50:47 192.168.0.1 WORKGROUP\me
W3SVC1 Web01 192.168.0.2 80 GET/shop/Webpage.asp
lid=20&vid=1000&cat=books 0 4549 1141
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+5.0)
58C673C195B84D249FE0FB9DCCF02E9E
-----------------------------------------------------------------------
```

These logs can be examined from either a client perspective or a server perspective. When it is evaluated from a server side, mining uncovers information about the sites where the service resides. It can be used to improve the design of the sites. By evaluating a client's sequence of clicks, information about a user (or group of users) is detected. This could be used to perform prefetching and caching of pages.

B. The problems be suffer with the conventional search engines

Almost all Web sites have searching function, and the search engines which they are using are confronting with the following troubles[2]:

1) *Over abundance:* Most of the data on the Web are of no interest to most people. In other words, although there is a lot of data on the Web, an individual query will retrieve only a very small subset of it.
2) *Limited coverage:* Search engines often provide results from a subset of the Web pages. Because of the extreme size of the Web, it is impossible to search the entire Web at any time a query is submitted. Instead, most search engines create indices that are updated periodically. When a query is requested, only the index is directly accessed.

3) *Limited query:* Most search engines provide access based only on simple keyword-based searching. More advanced search engines may retrieve or order pages based on other properties such as popularity of pages.

4) *Limited customization:* Query results are often determined only by the query itself. However, as with traditional IR systems, the desired results are often dependent on the background and knowledge of the user as well.

III. The Improvement of Sequential Patterns Algorithm Based On Users

A. *The base principle*

The effectiveness of a set of Web pages depends not only on the content of individual Web pages, but also on the structure of the pages and their ease of use. The most common data mining technique used on click-stream-data is that of uncovering traversal patterns. A traversal pattern is a set of pages visited by a user in a session. Knowledge of frequently references of contiguous pages can be useful to predict future references and thus for prefetching and caching purposes. Following it, knowledge of backward traversals can be used to improve the design of a set of Web pages by adding new links to shorten future traversals. The use of such performance improvements as user side caching actually alter the sequences visited by a user and impact any mining of the web logs data at the server side. The maximal profit is used primarily to reduce the number of meaningful patterns discovered. That is the rules, which have been expressed by other more complex rules, can be deleted. The problem of the click-stream-data is that the data of Web logs is asynchronous. Although a record will be written into a Web log file whenever a user click a Web site, monitoring the access pattern of a user is not a easy thing, since the information in a Web log file is recorded in the time order. Because users click the Web page at different time, in order to traversal users click-stream, we can not analyze the Web logs on time order only, users property should be taken into account too, that means we should traversal the Web logs according to the order of timestamp as well as that of UserID.

A sequential pattern (as applied to Web usage mining) is defined as an ordered set of pages that satisfies a given *support* and is maximal (i.e., it has no subsequence that is also frequent). *Support* is defined not as the percentage of sessions with the pattern, but rather the percentage of the customers who have the pattern. Since a user may have many sessions, it is possible that a sequential pattern should span a lot of sessions. It also needs not be contiguously accessed pages. A k-sequence is a sequence of length k (i.e., is it has k pages in it).

B. *The original AprioriAll algorithm*

The first step is sort step to put the data in the correct order, that is ordered by UserID and timestamp, the remaining steps are somewhat similar to those of the Apriori algorithm. The difference between the Apriori and the AprioriAll is that when generate the candidate sets, the AprioriAll use full join, that means all the items in L_{k-1} can be joined and put into C_k. On the contrary, in the Apriori, all the items in L_{k-1} can only be forth joined. As for a Web user can navigate back or forth, the AprioriAll but not the Apriori be used on Web log mining. The sort step creates the actual customer sequences, which are the complete reference sequences from one user (across transactions). During the first scan it finds large 1-itemset. Obviously, a frequent 1-itemset is the same as a frequent 1-sequence. In subsequent scans, candidates are generated from the large itemsets of the previous scans and then are counted. In counting the candidates, however, the modified definition of *support* must be used. In the algorithm we show that AprioriAll is used to perform this step.

The original algorithm[1]

Input

```
D={S_1,S_2,…,S_k}     //Database of sessions
S          //Support
```

Output: sequential patterns

Sequential patterns algorithm

```
D=sort D on UserID and time of first page reference in each session;
Find L_1 in D;
L=AprioriAll (D, S, L_1);
Find maximal reference sequences from L;
```

The following is an example to show the processing steps.

EXAMPLE The XYZ Corporation maintains a set of five Web pages:{A B C D E}.The following sessions (listed in timestamp order) have been created: D={S_1={U_1,<A,B,C>} S_2={U_2,<A,C>} S_3={U_1,<B,C,E>} S_4={U_3,<A,C,D,C,E>}}. Here we have added to each session the UserID. Suppose the *support* threshold is 30%.

In this example, user U_1 actually has two transactions (sessions). To find his sequential patterns, We must think of his sequence as the actual concatenation of those pages in S_1 and S_3. Also, since *support* is measured not by transactions but by users, a sequence is large if it is contained in at least one customer's sequence. After the sort step, we have that D={S_1={U_1,<A,B,C>} S_3={U_1,<B,C,E>} S_2={U_2,<A,C>} S_4={U_3,<A,C,D,C,E>}}. We find L_1={<A> <B> <C> <D> <E>} since each page is referenced by at least one customer. The following table outlines the steps taken by AprioriAll:

C_1={<A> <B> <C> <D> <E>} ***There are 5 members.*** L_1={<A> <B> <C> <D> <E>}
C_2={<A,B> <A,C> <A,D> <A,E> <B,A> <B,C> <B,D> <B,E> <C,A> <C,B> <C,D> <C,E> <D,A> <D,B> <D,C> <D,E> <E,A> <E,B> <E,C> <E,D>} ***There are 20 members.*** L_2={<A,B> <A,C> <A,D> <A,E> <B,C> <B,E> <C,B> <C,D> <C,E> <D,C> <D,E>}
C_3={<A,B,C> <A,B,D> <A,B,E> <A,C,B> <A,C,D> <A,C,E> <A,D,B> <A,D,C> <A,D,E> <A,E,B> <A,E,C> <A,E,D> <B,C,A> <B,C,E> <B,C,D> <B,E,A> <B,E,C> <B,E,D> <C,B,A> <C,B,E> <C,B,D> <C,D,A> <C,D,B> <C,D,E> <C,E,A> <C,E,B> <C,E,D> <D,C,A> <D,C,B> <D,C,E> <D,E,A> <D,E,C>} ***There are 32 members.***

L_3={<A,B,C> <A,B,E> <A,C,B> <A,C,D> <A,C,E> <A,D,C> <A,D,E> <B,C,E> <C,B,E> <C,D,E> <D,C,E>}

C_4={<A,B,C,E> <A,B,C,D> <A,B,E,C> <A,B,E,D> <A,C,B,E> <A,C,B,D> <A,C,D,B> <A,C,D,E> <A,C,E,B> <A,C,E,D> <A,D,C,B> <A,D,C,E> <A,D,EB> <A,D,E,C> <B,C,E,A> <B,C,E,D> <C,B,E,A> <C,B,E,D> <C,D,E,A> <C,D,E,B> <D,C,E,A> <D,C,E,B>}

There are 22 members.

L_4={<A,B,C,E> <A,C,B,E> <A,C,D,E> <A,D,C,E>}

C5=φ

C. The improved algorithm

The goal of the improvement is to reduce the size of the candidate sets. By this way, the number of scanning database can be reduced when generating the large set, which increase efficiency. By watching every candidate set, there is only a few turples is really large while many others are not. The reason for this, is that, firstly this algorithm only takes time order into account during a candidate generated and does not consider the user property, and secondly, for no method be used to pruning the candidate set, it makes a lot of candidates which have no relationship with a certain user, and so the cost of time and space consuming are high. The property of the Apriori algorithm, which is that *every element's subset of the items in L_k must be in the L_{k-1},* can be used to prune.

In the following improved algorithm, on the one hand, the user property is forced to take into account, every element in the candidate sets and large sets refers to the UserID. In every step to generate a candidate set, the elements in the previous large set which have the same UserID should be crossed each other, and the others should not be account. On the other hand, the candidate sets generated in every step is pruned in the light of the property of Apriori and the result is called C'_k. By these way, we can reduce the size of every candidate set sharply, as well as the number of scanning database, so the complexity of the time and space can be reduced. The following is the description of this improved algorithm.

The improved algorithm

Input

```
  U={ U1,U2,…,Ui }      // The set of users
  D={t1,t2,…,tk}        //Database of sessions with UserID
  S                     //Support
```

Output: sequential patterns

SEQUENTIAL PATTERNS ALGORITHM

```
D=sort D on UserID and time of first page reference in each session;
    L1 with UserID={large 1-itemsets};
    For (k=2; Lk-1≠null; k++) do
     Begin
       Ck=Apriori-gen(Lk-1,U);//new candidate set
       For all transaction ti   D do
        begin
          Ci=subset(Ck, ti);
          For all candidate c   Ci do
             c.count++;
        end
        Lk={c   Ck|c.count>S};//S:support
      End
    Find maximal reference sequences from L;

  Procedure Apriori-gen(Lk-1,S,U)
  Ck=null;
  For each itemset Li   Lk-1
      For each itemset Lj   Lk-1
         Begin
           If Li and Lj has same U
             begin
             C=Li join Lj;
             If has infreqyent-subset(c,Lk-1)
                Delete c;
             Else
                Add c to Ck;
             end
         End
  Return Ck;

 Procedure has infreqyent-subset(c,Lk-1)
 For each (k-1) subset s of c
     If s   Lk-1 then return False;
     Else True;
```

The following table outlines the steps taken by the improved algorithm:

C_1={<A>$_{1,2,3}$, <B>$_1$, <C>$_{1,2,3}$, <D>$_3$, <E>$_{1,3}$} **There are 5 members.**

L_1={<A>$_{1,2,3}$, <B>1, <C>$_{1,2,3}$, <D>$_3$, <E>$_{1,3}$}

C_2={<A,B>$_1$, <A,C>$_{1,2,3}$, <A,D>$_3$, <A,E>$_{1,3}$, <B,A>$_1$, <B,C>$_1$, <B,E>$_1$, <C,A>$_{1,2,3}$, <C,B>$_1$, <C,D>$_3$, <C,E>$_{1,3}$, <D,A>$_3$, <D,C>$_3$, <D,E>$_3$} **There are14 members.**

L_2={<A,B>$_1$, <A,C>$_{1,2,3}$, <A,D>$_3$, <A,E>$_{1,3}$, <B,C>$_1$, <B,E>$_1$, <C,B>$_1$, <C,D>$_3$, <C,E>$_{1,3}$, <D,C>$_3$, <D,E>$_3$}

C_3={<A,B,C>$_1$, <A,B,E>$_1$, <A,C,B>$_1$, <A,C,D>$_3$, <A,C,E>$_{1,3}$, <A,D,C>$_3$, <A,D,E>$_3$, <A,E,B>$_1$, <A,E,C>$_{1,3}$, <A,E,D>$_3$, <B,C,A>$_1$, <B,C,E>$_1$, <B,E,A>$_1$, <B,E,C>$_1$, <C,B,A>$_1$, <C,B,E>$_1$, <C,D,A>$_3$, <C,D,E>$_3$, <C,E,A>$_{1,3}$, <C,E,B>$_1$, <C,E,D>$_3$, <D,C,A>$_3$, <D,C,E>$_3$, <D,E,A>$_3$, <D,E,C>$_3$}

There are 25 members.

C'_3={<A,B,C>$_1$, <A,B,E>$_1$, <A,C,B>$_1$, <A,C,D>$_3$, <A,C,E>$_{1,3}$, <A,D,C>$_3$, <A,D,E>$_3$, <B,C,E>$_1$, <C,B,E>$_1$, <C,D,E>$_3$, <D,C,E>$_3$} **There are 11 members.**

L_3={<A,B,C>$_1$, <A,B,E>$_1$, <A,C,B>$_1$, <A,C,D>$_3$, <A,C,E>$_{1,3}$, <A,D,C>$_3$, <A,D,E>$_3$, <B,C,E>$_1$, <C,B,E>$_1$, <C,D,E>$_3$, <D,C,E>$_3$}

C_4={<A,B,C,E>$_{1,3}$, <A,B,E,C>$_{1,3}$, <A,C,B,E>$_{1,3}$, <A,C,D,E>$_3$, <A,C,E,B>$_1$, <A,C,E,D>$_3$, <A,D,C,E>$_3$, <A,D,E,C>$_3$, <B,C,E,A>$_1$, <C,B,E,A>$_1$, <C,D,E,A>$_3$, <D,C,E,A>$_3$}

There are 12 members.

C'_4={<A,B,C,E>$_1$, <A,C,B,E>$_1$, <A,C,D,E>$_3$, <A,D,E,C>$_3$}

There are 4 members.

L_4={<A,B,C,E>$_1$, <A,C,B,E>$_1$, <A,C,D,E>$_3$, <A,D,C,E>$_3$}

C_5=φ

Obviously, the number of elements in every candidate set here is smaller than that of the original algorithm.

IV. TEST AND RESULT

In order to compare the improved algorithm to the original one, we used a piece of Web logs of www.baidu.com.cn, which about 2.5G, to do the test. The result showing in TABLE I:

TABLE I
TEST RESULT

	CPU	Memory 1	Time cost 1	Memory 2	Time cost 2
Original algorithm	PIII 1G	512K	75’ 35’’	1G	55’ 20’’
improved algorithm	PIII 1G	512K	15’ 28’’	1G	13’16’’

The principle for improving is to reduce the number of elements in every candidate set, but any change of the large sets is not allowed. There are three aspects to make this algorithm better than the original one. Firstly, when the candidates are being produced, instead of dealing with all the items of the previous large set, only the elements which having the same UserID are crossed. Secondly, by pruning, the number of elements in the candidate sets is decreased once more. Thirdly, while deciding whether an element is large or not, it only need to scan the turples in the database with the same UserID, by this way, there are only a few records in the database being scanned.

Although the cost of time and space consuming are reduced greatly, but in the real processing, the capacity of the memory should be taken into account either. For the sake of the huge number of the Web logs, there is no memory in the world has enough space to be loaded. This has become one of the reasons for losing. The huge number of the Web logs records produces great press on the CPU too. By thinking of no relationship between the patterns of different users, three methods can be used. The first one is loading all Web log records into SQL Server database during preprocessing, and using the power of the DBMS to store, group, sort and query the data. TABLE II and TABLE III show the Data Directory.

TABLE II
THE USERS TABLE

Fields Name	Data Type	Length	Note
OrderNum1	long	4	PK
IP	Varchar	50	Not null
UserID	Varchar	50	Not null

TABLE III
THE VISITED PAGES TABLE

Fields Name	Data Type	Length	Note
OrderNum2	long	4	PK
OrderNum1	long	4	FK
Url	Varchar	100	Not null
DuringTime	number	9	Not null

The second one is that, in order to match the capacity of the memory, the data of only one user are loaded for each time. The third one is that, multi-computers can be connected into a net, in which, every computer only process the data of one user on each time and these computers are running on parallel. By these measures, not only the neck of the memory problem, but also the problem of the huge data of Web logs has been ruled out. When a computer is dealing with the data of one user, a small *support* is used, in order not to lose any patterns of users. But on this way, some noise maybe introduced too. On the next step, all of the result of the previous step are collected together into a new data set D_g, and use the original *support* to match the all users’ patterns and lead to the finial result. For the *support* return to the original one, the noise which introduced by the previous step are all ruled out. Moreover, in the case of new data be appended into Web logs, only the new data should be dealt singly and whose result can be appended into D_g. Next time, we need only to scan the D_g again. By this way, the result produced by previous processing can be used either, this means time be saved.

REFERENCES

[1] Margaret H Dunham. Data Mining Introductory and Advanced Topics [M]. Beijing: Tsinghua University Press, 2003, p195-220.
[2] Lin jie bin, Liu ming de, Chen xiang. Data mining and OLAP Theory & Practice [M]. Beijing: Tsinghua University Press, 2003, p194-244.
[3] Tony Bain. SQL Server 2000 Data Warehouse and Analysis Services [M]. Beijing: China Electric Power Press, 2003, p443-470.
[4] Jiawei Han, Micheline Kamber. Data Mining Concepts and Techniques [M]. Beijing: China Machine Press, 2001, p290-297.
[5] Claude Seidman. Data Mining With SQL Server 2000 Technical Reference [M]. Beijing: China Machine Press, 2002, p100-118.
[6] Lan H Witten, Eibe Frank. Practical Machine Learning Tools and Techniques with JAVA Implementations [M]. Beijing: China Machine Press, 2003, p77-118.