Frequent Pattern Mining of Web Log Files Working Principles

K. Suguna
Research Scholar,
Bharathiar University,
Assistant Professor,
Department of Computer Applications,
Dr.N.G.P Arts and Science College,
Coimbatore-641046,India.

K. Nandhini, PhD
Assistant Professor,
Department of Computer Science,
Chikkanna Government Arts College,
Tirupur-641602,India.

ABSTRACT
Frequent pattern mining plays a major role in mining of web log files. Web usage mining is the one of the web mining process that involves application of mining techniques to web server logs to extract the behavior of users. A web usage mining consists of three important phases: data preprocessing, patterns discovery and pattern analysis. In data preprocessing phase the unwanted data are removed and that are structured into necessary format for mining. It enables the user to translate the unprocessed data which is from server log files into useful data. The appropriate analysis of a web server log proves that the websites efficiently from the administrative and users’ prospective. Preprocessing results also more useful for the next phases of web usage mining.

General Terms
Data Preprocessing, pattern analysis, Apriori and FP Growth.

Keywords
World wide web, Preprocessing, Web usage mining and web server logs.

1. INTRODUCTION
The web is a way of retrieving information over the internet. It is the one of the model developed for an internet for sharing the information[1]. Web mining is the data mining technique used to mine the web data. The web mining can be classified into three Categories: Web content mining, Web structure mining and Web usage mining.

Web content mining is used to extracts useful information or knowledge from web documents. The web structure mining discovers useful knowledge from hyperlinks that describes the structure of the web.

Web usage mining refers to the discovery of user access patterns from web server logs, which maintains the record of every click made by each user[3]. In general, Web usage mining consists of three processes: data preprocessing, patterns discovery and patterns analysis.

2. WEB USAGE MINING
Web mining is the application of data mining techniques which is used to discover useful patterns from the web. The web mining can be divided into three categories: web usage mining, web content mining and web structure mining.

Figure 1: Web mining classification

Figure 2: Phases of Web Usage Mining
2.3 Pattern Analysis
In this phase, unwanted patterns are removed from the patterns recognized through pattern discovery phase.

3. DATA PREPROCESSING
Data preprocessing is one of the most complex phase of the Web Usage Mining process[13]. The data retrieved from the web log files are preprocessed to remove the unwanted and noisy data.

Data Preprocessing defines any type of processing made on raw data to organize it for another processing procedure[14].

The data that can be accessed through web is varied and partially planned or formless in nature. Due this improbability a web log file may consists of some unwanted log entries, whose presence does not matters from the web usage mining point of view[15].

This makes the preprocessing of log file an important requirement for discovering the knowledgeable patterns. The goal of preprocessing is to transform the unprocessed click stream data into a set of user profiles.

3.1 Data Cleaning
The purpose of data cleaning is to reduce irrelevant objects, and these kinds of techniques are of significance for any type of web log analysis. A web log file may consist of certain unnecessary data which has nothing to do with the mining procedure. So it is essential to remove those unrelated entries from the log file.

3.2 Path Completion
There are probability of missing pages after constructing transactions due to proxy servers and caching problems. In such a situation it becomes needs to identify the user’s access path and adding the missing paths.

3.3 Session Identification
The simplest method for identifying the session uses a timeout mechanism. The consequence of timeout method is that if the time between page requests exceeds a certain limit, signifies user is starting a new session.

3.4 User Identification
This is the next step after cleaning of data and most important task. Different users are identified, who contact web server, requesting for some resource on the web.

4. PATTERN DISCOVERY AND ANALYSIS
Pattern discovery is done after cleaning the data and after the identification of user transactions from the access logs. The analysis of the pre-processed data is very useful to all the organizations to carry out data extraction over the web[16]. The different kinds of mining algorithms that can be performed on the pre-processed data include Data Field Extraction Algorithm, Data storage Algorithm and Data Cleaning Algorithm.

4.1 Pattern Discovery
Pattern discovery is performed only after cleaning the data and after the identification of user transactions and sessions from the access logs.

4.1.1 Apriori Algorithm
Apriori is designed to operate on databases containing transactions. Apriori uses a "bottom up" approach, where the frequent subsets are extracted one item at a time and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Algorithm 1-Apriori Algorithm
The pseudocode for the algorithm is given below for a transaction database T, and a support threshold of \( \Sigma \). The set theoretic notation is employed, though note that T is a multiset. \( C_k \) is the candidate set for level k. Generate() algorithm is assumed to generate the candidate sets from the large itemsets of the preceding level, heeding the downward closure lemma. count[c] accesses a field of the data structure that represents candidate set \( c \), which is initially assumed to be zero.

Apriori(T,\( \varepsilon \))
L1<-{Large 1-itemsets}
K<2
whileLk-1\( \neq \phi \)
\( C_k \leftarrow \{c \in T \mid count[c] \geq \varepsilon \} \)
for transactions \( t \in T \)
\( C_t \leftarrow \{c \in \varepsilon \mid c \in c \} \)
For candidate \( c \in C_t \)
Count[c] \leftarrow Count[c]+1
\( L_k \leftarrow \{c \in \varepsilon \mid count[c] \geq \varepsilon \} \)
k \leftarrow k+1
U L_k
Return k

4.2. Pattern Analysis
The data is extracted from the preprocessed click stream data by using FP-Growth algorithm as follows:

4.2.1. Data Extraction using FP-Growth Algorithm
In this paper, we proposed FP-Growth Algorithm to extract the frequent patterns from the web log, the FP-Growth Algorithm is much efficient than the Data Field Extraction Algorithm.

4.2.1.1 Frequent Pattern Tree
In FP mining perform one scan of transaction database to identify the set of frequent items[4] To avoid repeatedly scanning the original transaction database, each transaction can be stored in some compact structure[5].

To merge the shared set with number of occurrence registered as count, if multiple transactions are shared a set of frequent items, so it is easy to check whether two sets are identical[6].

The original algorithm to construct the FP-Tree defined by Han is presented in Algorithm 2.
**Algorithm 2: FP-tree construction**

**Input:** A transaction database DB and a minimum support threshold \( \xi \).

**Output:** FP-tree, the frequent-pattern tree of DB.

**Method:** The FP-tree is constructed as follows.

1. Scan the transaction database DB once. Collect \( F \), the set of frequent items, and the support of each frequent item. Sort \( F \) in support-descending order as FLList, the list of frequent items.
2. Create the root of an FP-tree, \( T \), and label it as “null”. For each transaction Trans in DB do the following:
   - Select the frequent items in Trans and sort them according to the order of FLList. Let the sorted frequent-item list in Trans be \( [ p | P] \), where \( p \) is the first element and \( P \) is the remaining list. Call insert tree\(( [ p | P], T \)\).
   - \( T \) is performed as follows. If \( T \) has the function insert tree\(([ p | P], \text{a child } N \text{ such that } N\text{-item-name} = p\text{-item-name}) \); increment \( N \)’s count by \( 1 \); else create a new node \( N \), with its count initialized to \( 1 \), its parent link linked to \( T \), and its node-link linked to the nodes with the same item-name via the node-link structure. If \( P \) is nonempty, call insert tree\((P, N)\) recursively.

By using this algorithm, the FP-tree is constructed in two scans of the database. The first scan collects and sorts the set of frequent items, and the second constructs the FP-Tree[7].

**Table 1. Transaction Dataset**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{a,b}</td>
</tr>
<tr>
<td>2</td>
<td>{f,c,b}</td>
</tr>
<tr>
<td>3</td>
<td>{f,c,d,e}</td>
</tr>
<tr>
<td>4</td>
<td>{a,b,f,c}</td>
</tr>
<tr>
<td>5</td>
<td>{f,d,c,e}</td>
</tr>
<tr>
<td>6</td>
<td>{a,d,e}</td>
</tr>
</tbody>
</table>

(i) After reading TID = 1

(ii) After reading TID = 6

**4.2.1.2 Frequent Pattern Growth Algorithm**

The FP-Growth Algorithm, proposed by Han, is an efficient and accessible method for mining the whole set of frequent patterns by pattern fragment growth, using an widespread prefix-tree structure for storing compressed and crucial information about frequent patterns simply known as frequent-pattern tree (FP-tree)[9]. The following algorithm for mining frequent patterns using FP-tree.

**Algorithm 3-FP-growth: Mining frequent patterns with FP-tree**

**Input:** A database DB, represented by FP-tree constructed according to Algorithm 1 and a minimum support threshold \( \xi \).

**Output:** The complete set of frequent patterns.

**Method:** call FP-growth(FP-tree, null).

Procedure FP-growth\((\text{Tree}, \alpha)\) {

(1) if \( \text{Tree} \) contains a single prefix path Then, let \( P \) be the single prefix-path part of \( \text{Tree} \); Let \( Q \) be the multipath part with the top branching node replaced by a null root; For each combination (denoted as \( \beta \)) of the nodes in the path \( P \) do Generate pattern \( \beta \cup \alpha \) with support = minimum support of nodes in \( \beta \); Let freq pattern set\((P)\) be the set of patterns so generated;

(2) else let \( Q \) be \( \text{Tree} \); For each item ai in Q do \{Generate pattern \( \beta = ai \cup \alpha \) with support = ai.Construct \( \beta \)’s conditional pattern-base and then \( \beta \)’s conditional FP-tree Treeβ if Treeβ = \( \emptyset \) then call FP-growth(Tree, \( \beta \)); Let freq pattern set\((Q)\) be the set of patterns so generated; \} return(freq pattern set\((P)\cup \text{freq pattern set}(Q))\}

**Table 2. Header Table**

<table>
<thead>
<tr>
<th>Item</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
</tr>
</tbody>
</table>
Step 1: Construct conditional Pattern base for each item in the header table

<table>
<thead>
<tr>
<th>Item</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
</tbody>
</table>

### Step 2: Construct the conditional FP-Tree for the frequent items of the pattern base

<table>
<thead>
<tr>
<th>Item</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
</tr>
</tbody>
</table>

### Step 3: Recursively mine the Conditional FP-Tree

- **Conditional FP-tree of “e”: (fca:3)**
  - { }  
  - | f : 3 |
  - | c : 3 |
  - | a : 3 |

- **Conditional FP-tree of “ac”: (fc:3)**
  - { }  
  - | f : 3 |
  - | c : 3 |

- **Conditional FP-tree of “cae”: (f:3)**
  - { }  
  - | f : 3 |

The frequent pattern “fcae” is found after step 3 using FP growth algorithm.

### 5. CONCLUSION

In this paper we defined a framework for data preprocessing and pattern analysis using Apriori and FP-Growth algorithms. The Apriori algorithm preprocesses the data from the web log files. The FP-Growth algorithm extracts the frequent data from cleaned data. The appropriate analysis of a web server log proves that the websites works efficiently. This research work will focus on graph based approach in future.

### 6. REFERENCES


[13] Jiawei Han, Ian Pei, Yiwen Tin, Runying Mao, “Mining Frequent Pattern without Candidate Generation: A Frequent Pattern Tree Approach”, Volume-8.

[14] Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Hua Zhu, “Mining Access Pattern Efficient from Web Logs”


