Improved Apriori Algorithm for Web Log Data

Santosh Shakya*, Susheel Gupta** and Gopal Patidar***
*-***Truba Institute of Engineering and Information Technology Bhopal, India

Abstract: Web log mining technique is the very useful technique of extracting useful information from server logs that is use Web usage mining is the process of finding out what users are looking for on the Internet. The mining frequent patterns from web log data can help to optimize the structure of a web site and improve the performance of web servers. Users can also benefit from these frequent patterns from Web. Many efforts have been done to mine frequent patterns efficiently. Apriori and its variants and pattern-growth approach are the two representative frequent pattern mining approaches for candidate-generation-and-test approach. In this article we have conducted extensive experiments on real world web log data to analyze the characteristics of web logs and the behaviors of these two approaches on web log data. In this paper we propose a new Apriori algorithm for Frequent Pattern Mining for web log data. Our experimental results show that proposed algorithm can significantly improve the performance on frequent pattern mining on web log data.

Keywords: Web Mining, Web Usage Mining, Web Log, Apriori Algorithm.

Introduction

Web content used to be generated by a well-known source like a company, a service provider or a private person and presented to the users of the internet communication. It has changed, such that now, 'well-known' sources, which are a variety of open source and commercial organizations, provide tools to make use of the described paradigm by allowing users to generate content and at the same time providing a web-portal or something similar, in which this content may be presented by the creators to other users. So these platforms have reached a very high popularity and are heavily used, the video platform You-tube, the social network Face-book and the weblog platform Word-press just being among the most famous and intensely used ones. This thesis focuses on the latter type of Web 2.0 applications. Weblog is not a tool or application as such: It is a website with separate articles, displayed in reverse timely order. The weblog portals platforms equip users with the ability to run such websites without having the necessary technical skills to do so by them. This opportunity has caused a massive use of blogs by all types of people and for all kinds of topics. The use of blogs has increased so largely, that Techno ratio recently determined the total number of new blog articles to an astounding value of ten new articles per second [1-2]. In this paper we designed a Web log mining system for frequent item set mining and its visualization using modified Apriori hash tree algorithm.

Background Techniques

Web Usage Mining

Web usage mining is the process of extracting useful information from server logs whose use Web usage mining is the process of finding out what users are looking for on the Internet. Many users might be looking at only textual data and some others might be interested in multimedia data. The Web Usage Mining is the application of data mining techniques to discover interesting usage patterns from Web data in order to understand and better serve the needs of Web-based applications. The usage data captures the identity or origin of Web users along with their browsing behavior at a Web site. The web usage mining itself can be classified further depending on the kind of usage data considered:

1. Web Server Data: The user logs are collected by the Web server. Data includes IP address and page reference and access time.
2. Application Server Data: Commercial application servers have significant features to enable e-commerce applications to be built on top of them with little effort. The key feature is the ability to track various kinds of business events and log them in application server logs.
3. Application Level Data: New kinds of events can be defined in an application; also logging can be turned on for them thus generating histories of these specially defined events and It must be noted, however, that many end applications require a combination of one or more of the techniques applied [3-4].

Web Structure Mining

Web structure mining is the process of using graph theory to analyze the node and connection structure of a web site. The type of web structural data and web structure mining can be divided into two kinds:

1. Extracting patterns from hyperlinks in the web: a hyperlink is a structural component that connects the web page to a different location.
2. Mining the document structure: analysis of the tree-like structure of page structures to describe HTML or XML tag usage.

Event Logs

In general an event log records the events that occur in a certain process for a certain case. Many of the information systems as discussed in the previous section log events in an event log. The process definition specifies which activities should be executed in structure. When a new case is started a new instance of the process is generated which is called a process instance. The process instance keeps track of the current status of the case in the process. This process instance might leave a trace of events that are executed for that case in the event log. Each event is an instance of a certain activity as defined within the process definition. Furthermore, events are ordered to indicate in which sequence activities have occurred. In most cases this order is defined by the date and time or timestamp attribute of the event. Sometimes the start and stop information is recorded of a single activity. This is recorded in the event type attribute of the event. Other common attribute is the resource that executed the event which can be a user of the system, this system itself or an external system. Orther attributes can be stored within the event log related to the event for example the data attributes added or changed Unfortunately, every information system logs not events in the described way. Information about the relation between events and activities or even between traces and process instances is often not recorded. Main reason for this is that the event log is not seen as a recording of the execution of a process by system designers. Some systems the event log is used for debugging errors encountered in the system [3-6].

Server Log

A server log is a log file or several files automatically created and maintained by a server of activity performed by it. An example is a web server log which maintains a history of page requests. The W3C maintains a standard format means the Common Log Format for web server log files, on other hand proprietary formats exist. Most recent entries are typically appended to the end of the file. The request information about, including client IP address, and request date/time, bytes served, user agent, page requested, HTTP code, and referrer are typically added. The data can be combined into a single file, separated into distinct logs, such as an error log, access log, or referrer log. Server logs typically do not collect user-specific information.

These files are usually not accessible to general Internet users, only to the webmaster or other administrative user. Statistical analysis of the server log may be used to examine traffic patterns by time of day, day of week, referrer, or user agent. The efficient web site administration, adequate hosting resources and fine tuning of sales efforts can be aided by analysis of the web server logs. The marketing departments of any organization that owns a website should be trained to understand these powerful tools [5-6].

Frequent Itemset Mining

Use the term frequent itemset for a set S that appears in at least fraction s of the baskets," where is some chosen constant, typically 0.01 or 1%. We assume data is too large to fit in main memory. Either it is stored in a RDB, say as a relation Baskets (BID; item) or as a at file of records of the form (BID; item1; item2... item n). When evaluating the running time of algorithms:

Count the number of passes through the data. Since the principal cost is often the time it takes to read data from disk, the number of times we need to read each datum is often the best measure of running time of the algorithm. There is a key principle, called mono-tonicity or the a-priori trick that helps us find frequent itemsets:

If a set of items S is frequent (i.e., appears in at least fraction s of the baskets), then every subset of S is also frequent [7-8].

Mining closed and maximal frequent itemsets

A major challenge in mining frequent patterns from a large data set is the fact that such mining often generates a huge number of patterns satisfying the min-sup threshold, especially when min-sup is set low. This is because if a pattern is frequent, each of its sub-patterns is frequent as well. Large pattern will contain an exponential number of smaller and frequent sub-patterns. For overcome this problem, closed frequent pattern mining and maximal frequent pattern mining was proposed.

A pattern α is a closed frequent pattern in a data set D if α is frequent in D; there exists no proper super-pattern β such that β has the same support as α in D. A pattern α is a maximal frequent pattern (or max-pattern) in set D if α is frequent, and there exists no pattern β such that $\alpha \subset \beta$ and β is frequent in D. For the same min-sup threshold, the set of closed frequent patterns contains the complete information regarding to its corresponding frequent patterns; whereas the set of max-patterns, though more compact, usually have not contain the complete support information regarding to its corresponding frequent patterns [8-9].

Association Rules and Frequent Itemsets
The market-basket problem assumes we have some large number of items. Customers will their market baskets with some subset of the items, we get to know what items people buy together, even if we don't know who they are marketers use this information to position items, and control the way a typical customer traverses the store.
In addition to the marketing usage, the same sort of question has the following uses:
1. Baskets = documents; items = words. Words appearing frequently together in documents may represent phrases. It can be used for intelligence gathering.
2. Baskets = sentences, items = documents. Two documents with many of the same sentences could represent plagiarism or mirror sites on the Web [8-10].

Proposed Techniques

Web Log Mining System
We designed a Web log mining system for frequent item set mining and its visualization. The input and output of the system is Web log files as well as visualized patterns or text reports. The whole system includes:

Generation of Log file: The quality of the patterns discovered in web usage mining process highly depends on the quality of the data used in the mining processes. When the web browser traces the web pages and stores the Server log file. The web usage data contains information about the Internet addresses of web users with their navigational behavior the basic information source for web usage.

Web Server Data: When any user agent hits an URL in a domain, the information related to that operation is recorded in an access log file. The data processing task, the web log data can be preprocessed in order to obtain session information for all users. Access log file on the server side contains log information of user that opened a session. These records have seven common fields, which are: a. User's IP address, b. Access date and time, c. Request method (GET or POST), d. URL of the page accessed, e. Transfer protocol (HTTP 1.0, HTTP 1.1,), f. Success of return code. g. Number of bytes transmitted.

Data Preprocessing: This is the phase where data are cleaned from noise by overcoming the difficulty of recognizing different users and sessions, in order to be used as input to the next phase of pattern discovery. Data preprocessing phase always involves data cleaning and user identification and session identification.

1) Data Cleaning. In our system, we use server logs in Common Log Format. We examine Web logs and remove irrelevant or redundant items like image, sound, video files which could be downloaded without an explicit user request. Other removal items include HTTP errors, records created by crawlers, etc., which can't truly reflect users' behavior.
2) User Identification. To identify the users, the simple method is requiring the users to identify themselves, by logging in before using the web-site or system. Other approach is to use cookies for identifying the visitors of a web-site by storing a unique ID. However, these two methods are not general enough because they depend on the application domain and the quality of the source data, for this in our system we only set them as an option. Detail should be implemented according to different application domains.
We have implemented a more general method to identify user. We have three criteria:
(1) A new IP indicates a new user.
(2) Same IP but different Web browsers, or different operating systems, in terms of type and version, means a new user.
(3) Suppose the topology of a site is available, if a request for a page originates from the same IP address like other already visited pages, and indirect hyperlink exists between the pages, it indicates a new user.
3) Session Identification. To identify the user sessions is also very important because it will largely affects the quality of pattern discovery result. A user session can be defined as a set of pages visited by the same user within the duration of one particular visit to a web-site.

Pattern Mining: While various mining algorithms could be incorporated into the system to mine different types of patterns, currently, we only implemented sequential pattern mining on Web log data. We plan to add other part in future work.

Frequent Itemset Analysis: In this phase, the mined patterns which in great numbers need to be evaluated by end users in an easy and interactive way using improved Apriori algorithm

Proposed Improved Apriori Algorithm
The proposed algorithm is based on the Hash tree Algorithm steps of frequent item sets and rule generation phases. Frequent item sets are generated in two steps. The first step all possible combination of items, called the candidate item set (Ck) is

generated. The second step, support of each candidate item set is counted and those item sets that have support values greater than the user-specified minimum support from the frequent item set (Fk). In this algorithm the database is scanned multiple times and the number of scans cannot be determined in advance.
Suppose one of the large item sets is Lk, Lk = {I1, I2, …, Ik}, association rules with this item sets are generated in the following way the first rule is {I1, I2, … , Ik-1} {Ik}, by checking the confidence this rule can be determined as interesting or not. Next the other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, after then the confidences of the new rules are checked to determine the processes iterated until the antecedent becomes empty. So the second sub problem is quite straight forward, many of the researches focus on the first sub problem. Apriori algorithm finds the frequent sets L in Database D. A k-item set is an item set with exactly k item in it.
An association rule is about the relationship between two disjoint item sets, X and Y. It is denoted as X => Y. It presents the pattern → When X occurs, and Y also occurs.
Association rules not represent any sort of causality, correlation between the two items sets.
X => Y does not mean that X causes Y. There is no causality.
X => Y can imply different meaning than Y=>X, unlike correlation
Support for an item set X in a transactional database D is defined as count (X) / |D|
For an association rule X =>Y, we can calculate
Support(X => Y) = support (XY) = support(X union Y).
Confidence(X => Y) = support (XY) / support(X).
Support (S) and Confidence (C) can also be related to join probabilistic and conditional probabilities as follows
Support(X =>Y) = P(XY). Confidence (X=>Y)
SET k = 1;
Find frequent item set, from the set of all candidate item sets;
Ck: Candidate item set of size, Lk: frequent item set of size k;
Scan D and count each item set in C, if the count is greater than min-Supp, and then add that itemset to Lk
For k = 1, C1 = all item sets of length = 1
For k > 1, generate Ck from Lk-1 as follows:

The join step
Ck = k-2 way join of Lk-1 with itself.
If both {a1,..,ak-2, ak-1} & {a1 ,.., ak-2, ak} are in Lk-1, then add {a1 ,..,ak-2, ak-1, ak} to Ck
Remove {a1 , …,ak-2, ak-1, ak}, if it contains a non-frequent (k-1) subset.
For every non-empty subset A of X
Let B = X - A.
A => B is an association rule if Confidence (A => B) >= min-Conf.
Where, confidence (A => B) = support (AB) / support (A), and
Support (A =>) B .) =Support (AB)
To overcome boundary problem, Find out the min support, Scan D and count each itemset in Ck,
if the count is greater than minSupp, then add that itemset to Lk
For k = 1, C1 = all item sets of length = 1, For k > 1, generate Ck from Lk-1 as follows:
Ck = k-2 way join of Lk-1 with itself.
If both {a1,..,ak-2, ak-1} & {a1,.., ak-2, ak} are in Lk-1, then add {a1,..,ak-2, ak-1, ak} to Ck
The items are always stored in the sorted order.

The prune step
Remove {a1, …,ak-2, ak-1, ak}, if it contains a non-frequent (k-1) subset. For every non-empty subset A of X
Let B = X - A.
A => B is an association rule if Confidence (A => B) >= min-Conf.
Where, confidence (A => B) = support (AB) / support (A), and Support (A =>B.) = Support
(AB)
One way to improve efficiency of the APRIORI would be to Prune without checking all k-1 subsets
Join without looping over the entire set, Lk-1. One way to improve efficiency of the Apriori would be to prune without checking all k-1 subsets
Join without looping over the entire set, Lk-1.

Improved Apriori Algorithm
One way to improve efficiency of the Apriori as follows
Stepp1. Prune without checking all k-1 subsets.

Step2. Join without looping over the entire set, Lk-1.
Step3. *Speed up searching and matching*.
Step4. *Reduce the number of transactions* a kind of instance selection.
Step5. Reduce the number of passes over data on disk.
Step6. Reduce number of subsets per transaction that must be considered.
Step7. Reduce number of candidates this can be done by using *hash trees*
A *Hash tree* stores all candidate k-item sets and their counts. Root is empty and its children are the frequent 1-itemsets. Any node at depth = k will denote and frequent k-itemset. An example for an hash tree for C2 = 11, 14, 16, 26, 27, 33 is shown below

```
{}                                          **root
/1          |2          \3                **edge and label
/2  |3  \5  /3  \5  /5
[11:][14:][16:] [26:][27:] [33:]            **leaves
```

An internal node v at level n contains, bucket pointers. From these we tell which branch is the next one to be traversed. Hash of the mth item is used to decide this.

Join step using Hash Tree
The frequent k-1 item sets have common parents, should be considered for the joining step and checking all k-1 item sets in Lk-1is avoided.

Prune step using Hash Tree
To determine if a k-1 itemset is frequent, we look only for those item sets that have common parents, and thus avoid going through all k-1 item sets in Lk-1.To overcome crisp boundary problem, find out the min-supp and Scan D and count each itemset in Ck, if the count is greater than min Supp, after then add that itemset to Lk
For k = 1, C1 = all item sets of length = 1, For k > 1, generate Ck from Lk-1 as follows:
The join step:
Ck = k-2 way join of Lk-1 with itself. If both {a1,..,ak-2, ak-1} & {a1,.., ak-2, ak} are in Lk-1, then add{a1,..,ak-2, ak-1, ak} to Ck The items are mainly stored in the sorted order.
The prune step:
Remove {a1, …,ak-2, ak-1, ak}, if it contains a non-frequent (k-1) subset. For every non-empty
subset A of X Let B = X - A.
A => B is an association rule if Confidence (A => B) >= min-Conf. Where, confidence (A => B) >=support (AB) / support (A), and Support (A =>)B .)=Support (AB) the way to improve efficiency of the APRIORI would be to Prune not checking all k-1 subsets and Join without looping over the entire set, Lk-1.
Now the minimum support value will have the crisp boundary problem that is the output value will not be optimized one and the efficiency will be low to make it optimized and to improve the efficiency we have done the following modifications ,i.e. from the minimum support value from the Apriori hash tree, divide the minimum support by 50% of the total item set, since we calculate the min support from Apriori hash tree the result in ascending so the optimized result will not be behind the 50% region. Now we have obtained the optimized value compare to the previous Apriori algorithm. Thus we can overcome the crisp boundary problem by our modified algorithm and we have improved the efficiency by our algorithm.

Results Analysis

There are many major contributions that are involved in this work with respect to Information retrieval from the web log. First, this work focuses on link filtering and content filtering to eliminate the duplicate items from the search results. It has knowledge based summarizer on keywords and synonyms and provides a back link reference for tracking the facts of the summary. The quick browse our modified algorithm proposed in this work helps in faster access to the relevant information in the web mining search. Finally, the whole system has been developed using knowledge based intelligent components with rules so that it can be embedded in a collaborative environment for personalization and effective information retrieval.
Figure 1 represents the efficiency of the apriori hash tree algorithms (blue line) and proposed modified apriori hash tree with algorithm (red line) comparison graph. Our proposed techniques are better efficient than existing algorithm.

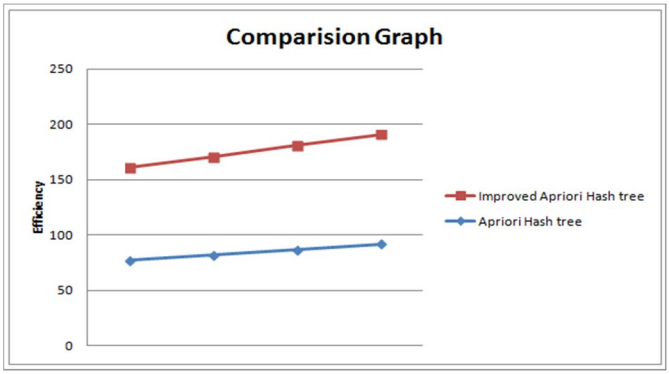

Fig 1 Efficiency of two algorithms

Conclusion

Our proposed algorithm concept is very simple in comparison of existing web miner which is used to produce web-log. Due to simplicity our proposed technique is high efficient. In the Proposed technique apply Hash based A-priori with some advance features like Bit shift operator on web-log for finding from web-log .The proposed method gives efficient results in comparison to existing algorithm. With the advances in technology it is of vital importance that our proposed system is robust enough to withstand the advances in technology. The more an A-priori technique relies on mathematics, the less the robustness. The time efficiency of our proposed technique measures in ms to finds URL from Web-log it is very good in comparison of existing algorithm.

References

[1] An Efficient Web Mining Algorithm for Web Log Analysis: E-Web Miner, M.P. Yadav, P.Keserwani, S.Ghosh978-1-4577-0697-4/12/$26.00 ©2012 IEEE.

[2] Tong, Wang and Pi-lian , He, Web Log Mining by an Improved AprioriAll Algorithm World Academy of Science, Engineering and Technology, Vol 4 2005 pp 97-100.

[3] Wen-Hai Gao Research On Client Behaviour Pattern Recognition System Based On Web Log Mining, Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao, 11-14 July 2010, DO1, 978-1-4244-6527-9/10/$26.00 ©2010, IEEE, pp 466-470

[4] Levi Albert,and Koç Çetin Kaya CONSEPP: CONvenient and Secure Electronic Payment Protocol Based on X9.59 Proceedings, The 17th Annual Computer Security Applications Conference, pages 286- 295, New Orleans, Louisiana, IEEE Computer Society Press, Los Alamitos, California, December 10-14, 2001

[5] Guo, Di, Collector Engine System: A Web Mining Tool for E Commerce, Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06) DOIcomputer society 0-7695-2616-0/06 2006 IEEE Yuewen, LI, Research on E-Commerce Secure Technology DOI 978-1-4244-3709- 2/10 2010 IEEE

[6] Mei Li and Cheng Feng, Overview of WEB Mining Technology and Its Application in E-commerce, 2010 2nd International Conference on Computer Engineering and Technology ,Volume 7.DOI 978-1-4244- 6349-7/10 .2010 IEEE pp V7-277-V7-280

[7] Gregory Buehrer, Srinivasan Parthasarathy, and Amol Ghoting. Out-ofcore frequent pattern mining on a commodity pc. In KDD ’06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, 2006, pp 86–95.

[8] DPVG06] Nele Dexters, Paul W. Purdom, and Dirk Van Gucht. A probability analysis for candidate-based frequent itemset algorithms. In SAC ’06: Proceedings of the 2006 ACM symposium on Applied computing, New York, NY, USA, 2006. ACM, pp541–545.

[9] Amol Ghoting, Gregory Buehrer, Srinivasan Parthasarathy, Daehyun Kim, Anthony Nguyen, Yen-Kuang Chen, and Pradeep Dubey. Cacheconscious frequent pattern mining on a modern processor. In Klemens B¨ohm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Ake Larson, and Beng Chin Ooi, editors, VLDB ACM, 2005, pp 577–588.

[10] C. K. Chui, B. Kao, and E. Hung. Mining frequent itemsets from uncertain data. In 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2007, Nanjing, China, pages 47–58, 2007.