Efficient Discovery of Frequent Patterns using KFP-Tree from Web Logs

Shyam Sundar Meena
Department of Computer Science & Engineering
Swami Vivekanad College of Engineering
Indore, Madhya Pradesh, India

ABSTRACT
Frequent pattern discovery is a heavily focused area in data mining. Discovering concealed information from Web log data is called Web usage mining. Web usage mining discovers interesting and frequent user access patterns from web logs. This paper contains a novel approach, based on k-mean and frequent pattern tree (FP-tree), for frequent pattern mining from Weblog data.

Keywords
Web mining, Pattern discovery, k-mean and FP-tree

1. INTRODUCTION
The expansion of the World Wide Web has resulted in a large amount of data. Web mining discovers and extracts useful information from the World Wide Web (WWW) documents and services using the data mining techniques. Most users obtain WWW information using a combination of search engines and browsers; however these two types of retrieval mechanism do not address all of a user's information needs. The resulting growth in on-line information combined with the almost unstructured web data necessitates the development of computationally efficient web mining tools. Web Mining can be classified [1] as, web content mining, web structure mining and web usage mining. Web content mining means automatic search of information resources available online [2], in short, mining the data on the Web. Web structure mining means mining the web document's structure and links, in short, mining the Web structure data. Web usage mining includes the data from server access logs, user registration or profiles, user sessions or transactions, in short, mining the Web log data. Web mining subtasks are (a) resource finding and retrieving, (b) information selection and pre-processing, (c) patterns analysis and recognition, (d) validation and interpretation, and (e) visualization [3].
The majority of the web logs include information regarding fields: IP Address, User Name, Time Stamp, Access Request, Result Status, Byte Transferred, Referrer URL and User Agent. There are many efforts towards mining various patterns from Web logs [4] [9] [11].
Mined information of Web logs can be used for various purposes like: To improve the design of web sites used to gather business intelligence to improve sales and advertisement, analyzing system performance, building adaptive Web sites [7] [6] [10].
The organization of the paper is as follows. Section 2, discuss the related work. In section 3, proposed approach is discussed in detail. Results on the experiments conducted are discussed in section 4. Finally conclusion is discussed in section 5.

2. RELATED WORK
In Web usage mining several data mining techniques can be used. Association rules are used in order to discover the pages which are visited together even if they are not directly connected, which can reveal associations between groups of users with specific interest [12]. This information can be used for example for restructuring Web sites by adding links between those pages which are visited together. Association rules in Web logs are discovered in [14] [15] [16] [17] [18]. Sequence mining can be used for discover the Web pages which are accessed immediately after another. Using this knowledge the trends of the activity of the users can be determined and predictions to the next visited pages can be calculated. Sequence mining is accomplished in [13], where a so-called WAP-tree is used for storing the patterns efficiently. Tree-like topology patterns and frequent path traversals are searched by [14] [19] [20] [21].

3. PROPOSED METHODOLOGY
3.1 Preprocessing
During user visit to the web pages in a website, web log files are created in the Web Server. The Preprocessing includes the steps of Parsing, Cleaning and Session Identification. The preprocessing step is executed for each web log file at a time.
Actually the web log files are flat text files that contain many space or tab delimited fields. The important fields in any web log file are Data, time, Client IP address, Server IP address, Server Port, URL Visited and User Agent filed that gives details of the browser and operating system versions. In parsing step splits the text file is into specific fields and extracts the required fields into a database table. In this case, we need the fields, date, time, Client IP address, URL Visited and User Agent.
Once these fields are split, extracted and stored in a database table, the extracted records are then cleaned to remove the images, icons and unwanted requests. So delete all records that have .JPEG, .GIF and .CSS files in the URL Visited field. As a result the cleaned database with relevant records is obtained.
The next step in preprocessing is user session identification [13]. The log entries in the web log files are chronologically ordered based on the different user's requests from their client machine to the web server.

3.2 The K-Means Method
The data is clustered using the Standard K-means algorithm [12] [13] which is a multi-pass technique. The k-means algorithm takes the input parameter, k, and partitions a set of n objects into K clusters so that the resulting intra cluster similarity is high but the inter cluster similarity is low. Cluster similarity is measured in regard to the mean value of

the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity. The algorithm iterates between the following steps till convergence:

1. Initialize K centroids at random for K clusters and assign each vector to the closest cluster centroid.
2. Compute the centroids of all current clusters.
3. Generate a new partition by assigning each item to the closest cluster centroid.
4. If cluster memberships change compared to the last iteration, go to step 2, else stop

3.3 Frequent Pattern Tree Basics

Let I = {I_1, I_2, I_3…I_n} be a set of items, and a transaction database T = {t_1, t_2, t_3…t_m} where t_i (i ε [1: m]) is a transaction which contains a set of items in I. The support (absolute occurrence of frequency, not the relative one as in some literature) of a pattern A, which is a set of items, is the number of transactions containing A in T. A, is a frequent pattern if A's support is no less than a predefined, minimum support threshold, ξ.

Given a transaction database T and a minimum support threshold, ξ, the problem of finding the complete set of frequent patterns is called the frequent pattern mining problem.

A frequent pattern tree (or FP-tree in short) is a tree structure consists of one root labeled as "null', a set of item prefix sub-trees as the children of the root, and a frequent-item header table. Each node in the item prefix sub-tree consists of three fields: item-name, count, and node-link, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none. Each entry in the frequent-item header table consists of two fields, (1) item-name and (2) head of node-link, which points to the first node in the FP-tree carrying the item-name. Based on this definition, we have the following FP-tree construction algorithm.

Let's first examine an example in [5] using the frequent-pattern growth approach. Suppose we have transaction data base shown in Figure 1 with minimum sup-port count be 2.

The first scan of the database derives the set of frequent items (1-itemsets) and their support counts (frequencies). The set of frequent items is sorted in the order of descending support count. Thus, we get the set L= {{I2: 7}, {I1: 6}, {I3: 6}, {I4: 2}, {I5: 2}}. An FP-tree is then constructed using the algorithm in [5]. First, create the root of the tree, labeled with "null." Scan database a second time. The items in each transaction are processed in L order and a branch is created for each transaction. For example, the scan of the first transaction, "T1: I1, I2, I5," which contains three items (I2, I1, I5 in L order), leads to the construction of the first branch of the tree with three nodes, {I2: 1}, {I1:1}, and {I5: 1}, where I2 is linked as a child of the root, I1 is linked to I2, and I5 is linked to I1. The second transaction, T2, contains the items I2 and I4 in L order, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common prefix, I2, with the existing path for T100. Therefore, we instead increment the count of the I2 node by 1, and create a new node, {I4: 1}, which is linked as a child of {I2: 2}. In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.

TID	List of Item_IDs
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1I2,,I3

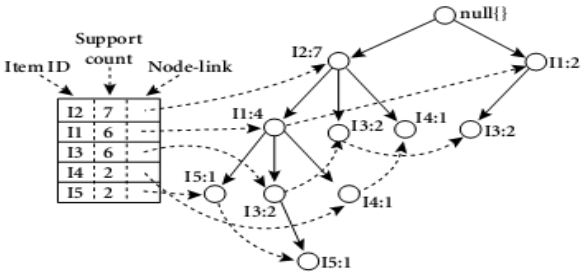

Figure 1: Database and corresponding FP-tree

To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links. The tree obtained after scanning all of the transactions is shown in Figure 1 with the associated node-links. In this way, the problem of mining frequent patterns in databases is transformed to that of mining the FP-tree.

3.4 Mining FP-tree

Start from each frequent length-1 pattern, construct its conditional pattern base, then construct its (conditional) FP-tree, and perform mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

The FP-growth method transforms the problem of finding long frequent patterns to searching for shorter ones recursively and then concatenating the suffix. It uses the least frequent items as a suffix, offering good selectivity. The method substantially reduces the search costs. Mining the FP-tree by creating conditional pattern bases is described in the figure 2.

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I1	{{I2,I1:1},{I2,I1,I3:1}}	{I2:2,I1:2}	{I2,I5:2},{I1,I5:2},{I2,I1,I5:2}
I4	{{I2,I1:1},{I2:1}}	{I2:2}	{I2,I4:2}
I3	{{I2,I1:1},{I2:2},{I1:2}}	{I2:4,I1:2},{I1:2}	{I2,I3:4},{I1,I3:4},{I2,I1,I3:2}
I1	{{I2:4}}	{I2:4}	{I2,I1:4}

Figure 2: Mining the FP-tree by creating conditional (sub-) pattern bases

3.5 KFP-tree algorithm

Input: Web log File
Output: Frequent Patterns
Steps

A. Preprocessing

B. Clustering

a) Arbitrarily choose k objects from D as the initial cluster centers.
b) Repeat
c) (Re) assign each object to the cluster, to which the object is the most similar, based on the mean value of the objects in the cluster;
d) Update the cluster means (It mean calculate the mean value of the objects for each cluster)
e) Until no change

C. Mining frequent patterns

1. FP-tree construction

Scan the transaction database D once. Collect F, the set of frequent items, and their support counts and sort F in support count descending order as L, the list of frequent items. Create the root of an FP-tree, and label it as "null." For each transaction Trans in D do the following. Select and sort the frequent items in Trans according to the order of L. Let the sorted frequent item list in Trans be [p|P], where p is the first element and P is the remaining list and call insert tree ([p|P], T), which is performed as follows. If T has a child N such that N.item-name=p.item-name, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree (P, N) recursively.

2. Mining the FP-tree

The FP-tree is mined by calling FP-growth (FP-tree, α)
FP-growth (Tree, α)

a) if Tree contains a single path P
b) then for each combination (denoted as β) of the nodes in the path P do
c) generate pattern $\beta \cup \alpha$ with support = minimum support of nodes in β;
d) else for each a in the header of Tree do {
e) generate pattern β = ai ∪ α with support = ai .support;
f) construct β's conditional pattern base and then β 's conditional FP-tree Treeβ;
g) if Treeβ =6 0
h) then call FP-growth (Treeβ , β)
i) Clustering groups the similar web access records from the weblogs. Using the similarity property of the records in the clusters, KFP-tree algorithm generates the frequently accessed web pages efficiently.

4. EXPERIMENTAL RESULT

The problem I have solved to mine frequent patterns in weblogs. As web logs are often very large in size buts parse in density, the efficiency of frequent pattern mining algorithm is important. In this section, I present a performance comparison of FP-growth with the recently proposed efficient method KFP-tree.

All the experiments are performed on a 2.40-GHz Pentium PC machine with 1 Gigabytes main memory, running on Microsoft Windows XP. All the programs are written in java. Please also note that run time used here means the total execution time, i.e., the period between input and output, instead of CPU time measured in the experiments in some literature. The synthetic data sets which we used for our experiments were generated using the procedure described in [8].

The results in figure 3 proved that the proposed algorithm is seen to perform better in aspect of computation time.

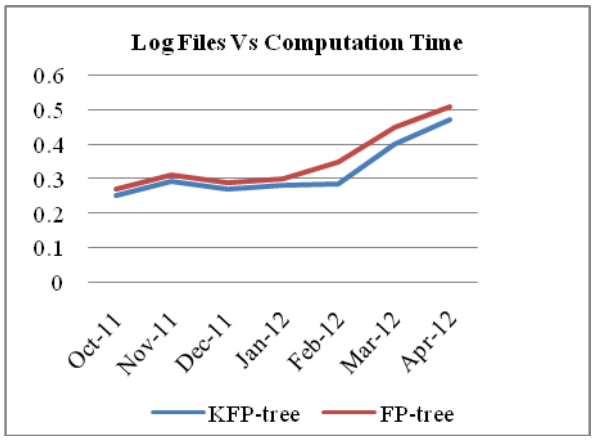

Figure 3: Comparing the Run Times of the FP-tree and KFP-tree algorithms

Thus KFP-tree algorithm takes lesser run time and prunes more rules than the traditional frequent pattern analysis, FPA approach. Thus this is a hierarchical frequent pattern mining approach that is found suitable for analyzing web log data and to predict useful information from the analyzed data.

5. CONCLUSION

I have proposed a novel approach, based on k-mean and frequent pattern tree (FP-tree), for frequent pattern mining from Weblog data. Weblog databases are homogenous databases in binary format. Initially the binary data is clustered using the multi-pass K-means algorithm. The similar groups of data or clusters are used in FP-tree procedure for frequent item-set generation. Experiments are performed using real and synthetic data, and found KFP-tree algorithm is more efficient compared to FP-tree algorithm. The future work regarding this paper is to apply the same for different web sites to confirm the results.

6. ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers for their thorough reviews, and constructive suggestions which significantly enhance the presentation of the paper.

7. REFERENCES

[1] Raymond Kosala and Hendrik Blockeel. 2000, "Web Mining Research: A Survey", ACM SIGKDD.

[2] Sanjay Kumar Madria, Sourav S Bhowmick, Ng W.K. and Lim E.P. 1999, "Research Issues in Web Data Mining", Springer.

[3] Qingyu Zhang and Richard S. Segall. 2008, "Web Mining: A Survey Of Current Research, Techniques, And Software", In International Journal of Information Technology and Decision Making, Volume: 07, Issue: 04, pp. 683-720.

[4] R. Cooley, B. Mobasher, and J. Sriv astava. Data preparation for mining World Wide Web browsing patterns. In Journal of Knowledge & Information Systems, Vol. 1, No. 1, 1999.

[5] Han,Kamber, "Data Mining Concepts & Techniques", M.Kaufman.

[6] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on Web usage mining. In Communications of the ACM, (43) 8, August 2000.

[7] M. Perkowitz and O. Etzioni. Adaptive Sites: Automatically learning from user access patterns. In Proc. 6th Int'l World Wide Web Conf., Santa Clara, California, April 1997.

[8] R. Agrawal and R. Srikant. Fast algorithms formining association rules. In VLDB'94, pp. 487-499.

[9] M. Spiliopoulou and L. Faulstich. WUM: A tool for Web utilization analysis. In Proc. 6th Int'l Conf. on Extending Database Technology (EDBT'98), Valencia, Spain, March 1998.

[10] L. Tauscher and S. Greeberg. How people revisit Web pages: Empirical findings and implications for the de sign of history systems. In Int'l Journal of Juman Computer Studies, Special Issue on World Wide Web Usability, 47:97-138, 1997.

[11] O. Zaiane, M. Xin, and J. Han. Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs. In Proc. Advances in Digital Libraries Conf. (ADL'98), Melbourne, Australia, pages 1244-158, April 1998.

[12] M. Eirinaki and M. Vazirgiannis, "Web mining for web personalization," ACM Trans. Inter. Tech., Vol. 3, No. 1, pp. 1-27, 2003.

[13] J. Pei, J. Han, B. Mortazavi-A sl, and H. Zhu, "Mining access patterns efficiently from web logs," in PADKK '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications. London, UK: Springer-Verlag, 2000, pp. 396-407.

[14] M. S. Chen, J. S. Park, and P. S. Yu, "Data mining for path traversal patterns in a web environment," in Sixteenth International Conference on Distributed Computing Systems, 1996, pp. 385-392.

[15] J. Punin, M. Krishnamoorthy, and M. Zaki, "Web usage mining: Languages and algorithms," in Studies in Classification, Data Analysis, and Knowledge Organization. Springer-Verlag, 2001.

[16] P. Batista, M. ario, and J. Silva, "Mining web access logs of an on-line newspaper," 2002.

[17] O. R. Zaiane, M. Xin, and J. Han, "Discovering web access patterns and trends by applying olap and data mining technology on web logs," in ADL '98: Proceedings of the Advances in Digital Libraries Conference. Washington, DC, USA: IEEE Computer Society, 1998, pp. 1-19.

[18] J. F. F. M. V. M. Li Shen, Ling Cheng and T. Steinberg, "Mining the most interesting web access associations," in WebNet 2000-World Conference on the WWW and Internet, 2000, pp. 489-494