Comparative Study of Techniques to Discover Frequent Patterns of Web Usage Mining

Mona S. Kamat[1], J. W. Bakal[2] & Madhu Nashipudi[3]

[1,3]Information Technology Department, Pillai Institute Of Information Technology(PIIT)
Panvel, Navi Mumbai, India
[*2] Shivajirao S. Jondhale College Of Engineering(SSJCOE), Dombivali, Thane, India
Email : [1]monakamat@gmail.com, [2]bakaljw@gmail.com & [3]madhu.nashipudi@yahoo.in

***Abstract* – This topic mainly briefs about the Web Usage Mining process, the stages involved in it followed by detailed study of three algorithms to discover frequent patterns namely FAP(Frequent Access Pattern) mining, GSP(Generalized Sequential Pattern) and DFS(Depth First Search). Performance of DFS and FAP mining is better than GSP since it requires repeated session scan to fins the pattern. Since DFS and FAP mining depend on the construction of pattern lattice and FAP tree it is more effective. Few applications of Web usage mining are briefed in the last section. This paper gives an overview of Web Usage mining process but mainly focuses on the comparative study of three algorithms GSP, FAP mining and DFS and their performances.**

Keywords— DFS(Depth First Search), FAP(Frequent Access Pattern), Frequent Patterns, GSP(Generalized Sequential Pattern)

I. INTRODUCTION

This Web Usage Mining is the process in which user access patterns are discovered and analyzed by mining the log files and related data associated with a certain website. It is a kind of web mining which automatically discovers user usage patterns and is helpful in studying and analyzing user interests. Web usage mining consists of mainly three stages, namely data preprocessing, pattern discovery and pattern analysis as shown in fig[1].

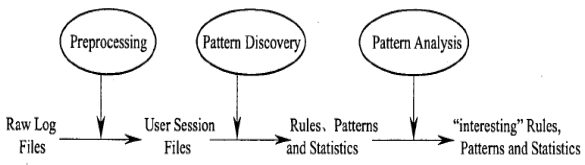

Fig.1. Stages in Web Usage Mining

Data for web mining is collected from multiple sources. Data is in different format following different conventions with many duplicates, inconsistencies, sometimes incomplete. Hence data preprocessing is very vital and is also most complex of all stages. It reduces the data size radically thus improving the efficiency of mining. Pattern Discovery applies various techniques on the preprocessed data to discover frequent patterns like statistics analysis, clustering, association rule mining, sequential pattern, classification and so on. In the next stage, Pattern analysis, all the patterns discovered in the previous phase are analysed to choose only the interesting patterns and rules sieving the useless patterns and rules.

This paper gives a brief overview of web usage mining process and explains few mining algorithms to find frequent patterns which constitute the basic information source for intelligent web-based systems also making a comparative study of their performances.

II. LITERATURE SURVEY

An Xidong Wang et al propose a method that can discover users' frequent access patterns underlying users' browsing web behaviours. They introduce the concept of access pattern according to a user's access path, and then based on it put forward a revised algorithm (FAP-Mining) based on the FP-tree algorithm to mine frequent access patterns. The new algorithm first constructs a frequent access pattern tree and then mines users 'frequent access patterns on the tree. The algorithm is accurate and scalable for mining frequent access patterns with different lengths.[7]

Murat Ali Bayir et al have explained web mining and its categories. Further explaining in brief about the data preprocessing, they have presented few algorithms

for searching frequent patterns and a few algorithms based on construction of pattern lattice. Also they have presented the results of the experiments they carried out to observe and analyse performances of the algorithms they have listed[5].

Theint Theint Aye explains the importance of data preprocessing to improve ease and efficiency of mining process. It mainly focuses on data preprocessing stages and explained algorithms for field extraction and data cleaning algorithms that performs the process of separating fields from the single line of the log file and eliminates inconsistent or unnecessary items in the analysed data respectively.[1]

Robert Cooley et al. have presented a survey of the research in the area of Web usage mining and have briefly discussed the pattern discovery techniques.

III. STAGES IN WEB USAGE MINING

Briefly, the stages are noted as follows:

i. Obtain data from various sources

ii. Data preprocessing

iii. Pattern Discovery

iv. Pattern Analysis

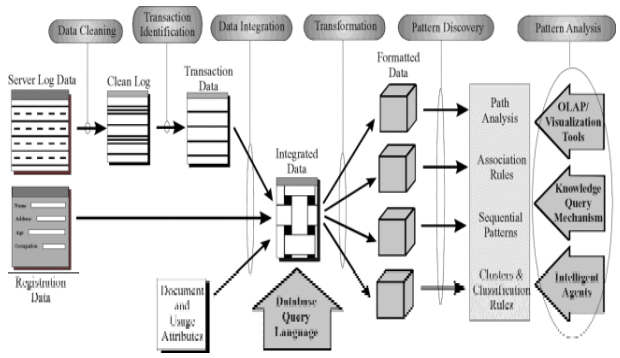

Fig.2. General Architecture of Web Usage Mining[14]

A. *Data Preprocessing*

The data should be preprocessed to improve the efficiency and ease of the mining process. The main task of data preprocessing is to prune noisy and irrelevant data, and to reduce data volume for the pattern discovery phase. Field Extraction and data cleaning algorithms parse the web log records separating the fields and purging. Covering

B. *Pattern discovery*

Few techniques to discover patterns from preprocessed data are listed like converting IP addresses to domain names, filtering, dynamic site analysis, cookies, path analysis, association rules, sequential patterns, clustering, decision trees etc.

C. *Pattern Analysis*

Following statistics are a few listed ones which are the end products of analysis such as the frequency of visits per document, most recent visit per document, who is visiting which documents, frequency of use of each hyperlink, and most recent use of each hyperlink. The common techniques used for pattern analysis are visualization techniques, OLAP techniques, Data & Knowledge Querying, Usability Analysis.

IV. TECHNIQUES TO DISCOVER FREQUENT PATTERNS

A. *Frequent Access Pattern Mining(FAP Mining)*

FAP algorithm is rooted on FP Growth Algorithm and it also constructs a FAP tree like FP tree. FAP algorithm is divided into two steps. In the first step, it constructs frequent access pattern tree (FAP tree) as per the access paths obtained from the user sessions and records the access counts of each page. Next step is where the function of FAP-growth is used to mine both long and short access patterns on the FAP tree.

FP Growth when used in mining association rules and sequential patterns shows good functionality. But there is no sequence among those elements of an item during mining association rules, whereas access pattern mining requires sequential page access. Thus the Fp-growth has been revised by X. Wang et al in [7]

Table 1. Episode of user access path in users session files

User Name	Session Name	Access Path
B Userid	S1	A−B−C−D−B−E
Userid	S2	A−B−C−D−C−B−E−G−E−C−H
Userid	S3	A−B−A−C−I
Userid	S4	C−I−G−I−K−I−D

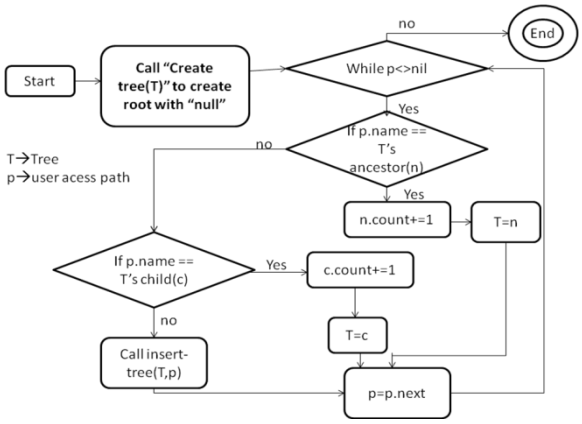

Fig.3. Flowchart for algorithm to construct frequent acess pattern

The Construction of FAP-Tree

Algorithm: FAP_Tree(tree, p). Construct frequent Access Pattern tree [b]

Input: The set of user access path p.

Output: The set of use access pattern.

The flowchart in fig 3 shows the construction of FAP tree algorithm. To facilitate frequent access pattern generation and FAP tree traversal, a page header table is built and is associated to the tree in such a way that each page points to its position in the tree and the table entries are sorted ascending in order of the page count. Table 1 shows an episode of access path of certain user contained in the user session file. Accordingly the function of FAP-Tree constructs frequent access tree as in fig.4

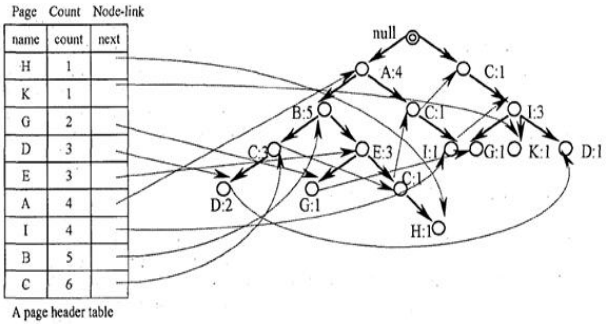

Fig 4. FAP tree[7]

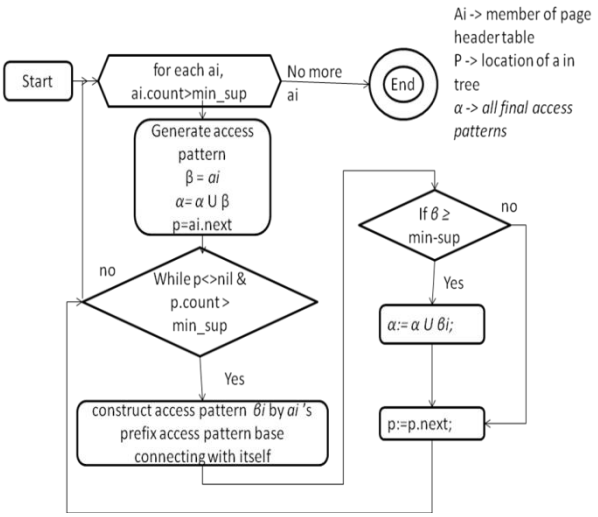

Fig.5.Flowchart of FAP-growth algorithm

FAP Growth

Algorithm: FAP-growth(tree, α), mine frequent access pattern Input: FAP tree, min_support

Output: α the set of all the access patterns

Table 2. Mining result of the FAP Tree

a_i	Set of Prefix Access Pattern Base	Frequent access pattern generated(User name omitted)
G	{}	{G}:2
D	{{C}:3、{B,C}:3、{A,B,C}:3}	{D}:2、{C,D}:2、{B,C,D}:2、{A,B,C,D}:2
E	{B}:5、{A,B}:4	{E}:3、{B,E}:3、{A,B,E}:3
A	{}	{A}:4
I	{C}:1	{I}:3
B	{A}:4	{B}:5、{A,B}:4
C	{B}:5、{A,B}:4	{C}:3、{B,C}:3、{A,B,C}:3

B. Generalised Sequential Pattern(GSP)

R.Srikant and R. Agrawal proposed the basic structure of GSP algorithm for finding sequential patterns. The algorithm makes multiple passes over the session set. At the end of the first pass the algorithm yields frequent items, 1-element frequent itemset. The following pass begins with the frequent itemset yielded in the previous pass to generate potential itemsets called candidates. The candidates are then pruned based on the minimum support which again gives the set of frequent sets of n-element itemsets in pass n. The algorithm is iterative and ends when there no more frequent sets. The three main steps carried out in each pass are as shown in fig. 6.

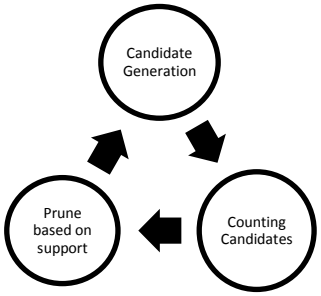

Fig.6. Steps in each pass of GSP algorithm

Candidate Generation: It generates candidate sequences by joining Ck-1 with Ck-1. Sequence s1 joins with s2 if by dropping the first item of s1 and the last item of s2, same subsequence is obtained.[12]This creates candidates for the current pass.

Counting Candidates: In each pass, one log-sequence is read at a time and we increment the access count of candidates contained in the log-sequence

Pruning: Only those candidates having their access count equal or higher than the mininmum support will b filtered and passed as the frequent itemset in the iteration

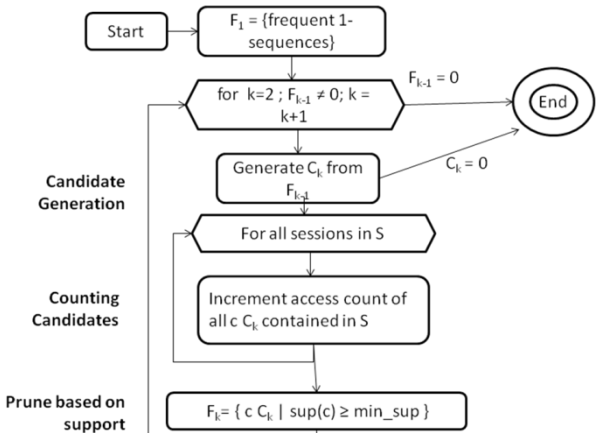

Fig.7. Flowchart for GSP algorithm

GSP Algorithm

Input: S(sessions), F1(Frequent 1-sequences), min_sup(the minimal access count that satisfies the support threshold).

Output: the set of all the access patterns: F

Table: 3 Mining result for GSP algorithm

1-Frequent access pattern generated	*2-Frequent access pattern generated*	*3-Frequent access pattern generated*	*4-Frequent access pattern generated*
{A}:4	*{AB}:3*	*{ABC}:2*	*{ABCD}:2*
{B}:5	*{BC}:3*	*{ABE}:2*	
{C}:6	*{BE}:2*	*{BCD}:2*	
{D}:3	*{CD}:3*		
{E}:3			
{G}:2			
{I}:4			

C. *Depth First Search(DFS)*

In this algorithm, the patterns are categorized according to the length executed on lattice model. Patterns will form a lattice based on the pattern-length and pattern-frequency. And using this lattice, frequent patterns are searched depth first.

Lattice Construction: The basic element of the lattice is an atom i.e. single page. Each atom or page stands for length-1 prefix equivalence class. Beginning from bottom elements the frequency of upper elements with length n can be calculated by using two n-1 length patterns belonging to the same class[5]. Example: Consider three atoms corresponding to three web pages P1, P2 and P3. Sample lattice up to some length-3 patterns and all length-2 patterns shown in fig.8.

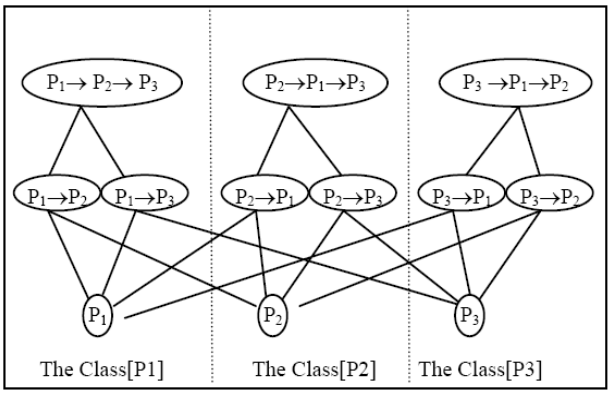

Fig.8. Pattern Lattice[5]

Session id-timestamp list: After data preprocessing, a series of web pages visited in each session is obtained. Session id timestamp list is a list which keeps session id and timestamp information for any patterns in all sessions. The timestamp information keeps the timestamp value of last atom for patterns with length > 1. Example: 4 pages and 3 sessions given below.

S1 = Page1 →Page2 → Page4 → Page1 → Page3

S2 = Page4 → **Page3** → **Page1** → Page2

S3 = **Page3** → Page4 → **Page1**

Table 4. Session id-timestamp list

$Page_1$		$Page_2$		$Page_3$		$Page_4$	
S.id	TS	S.id	TS	S.id	TS	S.id	TS
1	1	1	2	1	5	1	3
1	4	2	4	2	2	2	1
2	3			3	1	3	2
3	3						

Table 5. Session id-timestamp list for Page3→Page1

Session	Timestamp
2	3
3	3

The count for pattern Page3→Page1 is 2/3 since it occurs twice in three sessions. They are then pruned based on the minimum support. Fig.9. shows the flowchart depicting the working of DFS algorithm.

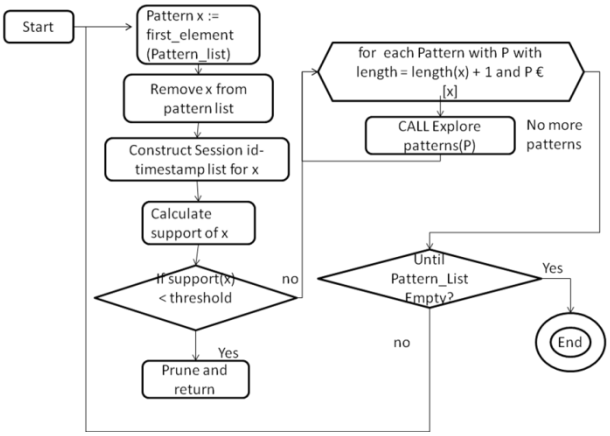

Fig.9. Flowchart for DFS algorithm

II. PERFORMANCE COMPARISON

FAP mining algorithm mines the complete set of frequent itemsets without generating the candidate set. FAP-growth works in a divide-and-conquer way. The first pass of the database derives a list of frequent items arranged in the descending order of frequency to generate a frequent-access-pattern tree, or FAP-tree. The algorithm searches for shorter ones recursively and then concatenates the suffix finding the long frequent patterns. Performance studies demonstrate that the method substantially reduces search time.[6]

GSP can suffer from two-nontrivial costs: (1) generating a huge number of candidate sets, and (2) repeatedly scanning the database and checking the candidates by pattern matching.

DFS gives a good performance since it eliminates infrequent patterns at each level and in the memory it keeps fewer patterns at each step.

Murat Ali Bayir et al. have performed experimental results on the web logs of the departmental web server on GSP and DFS algorithm. The log files of the web server at Computer department are used. (www.ceng.metu.edu.tr).

In the experiments GSP has given the worst results because it does not use pattern lattice structure and at each step it has to perform a session scan. DFS is better because it eliminates infrequent patterns at each level and in the memory it keeps fewer patterns at each step. Session-id timestamp list structure prevents unnecessary database scans for evaluating frequency of length 1 and length 2 patterns. Hence DFS gives a better performance. Fig.10. compares discovery time of all frequent patterns of different lengths for DFS and GSP algorithms.

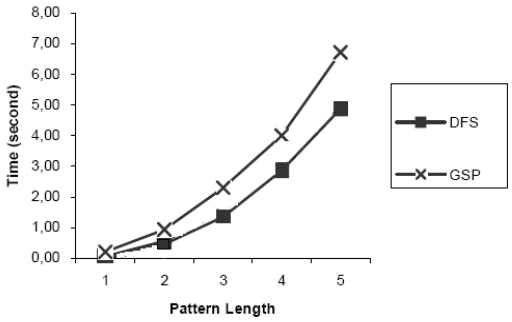

Fig.10. Comparison of performance of GSP and DFS[5]

Comprehensive Study Comparison of FAP-growth, DFS and GSP Algorithms:

GSP algorithm is based on apriori property: if an itemset α is not frequent, then any of its superset cannot not be frequent either. GSP uses a bottom-up search implying that in order to produce a frequent sequence of length n, all 2^n subsequences have to be generated. Therefore this exponential complexity limits it to discover only short patterns, since it prunes any candidate sequence in which there is a subsequence which is infrequent. Table 4. gives a comparison between the three algorithms based on various factors.

Table 4. Performance comparison of FAP, GSP and DFS

	FAP-growth	**GSP**	**DFS**
Algorithm	. Constructs FP tree .finds long frequent patterns by searching shorter ones and concatenating suffix	Candidate Generation Count Prune	.Construct Pattern Lattice .Search Patterns by creating session Id-Timestamp List .Prune
Candidate Generation	No	Yes	No
Based on	FP tree	Candidate Generation	Pattern Lattice
Scan entire dataset	Only twice to construct FP tree and later works on FP tree	Scan repeatedly for pattern matching	Scans to create Session Id-Timestamp List
Memory	Holds FP-tree in memory	All candidates generated as well as datasets	Very few patterns as patterns are pruned at each level
When large dataset/ low support	Bushy FP tree may not fit in the main memory	Exponential number of candidates	Pattern lattice becomes huge
Scalable	Not when support is very low else yes	Only when support is fairly high	Not when support is very low else yes
Execution time	Lower	Fairly high	Lower

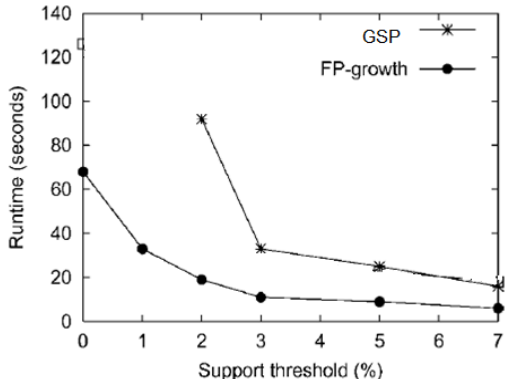

Fig.11. Scalability with threshold[8]

Table 4. gives a comprehensive comparison between the three algorithms, GSP, FAP-Growth and DFS algorithm. Fig.11 shows the scalability in GSP and FP-growth algorithms.

FP-growth works in a *divide-and-conquer* way. The first scan of the database derives a list of frequent items in which items are ordered by frequency descending order. According to the list, the database is represented as frequent-pattern tree, or F*P-tree*, which shows the association between items. The FP-tree starts with frequent length-1 pattern (suffix pattern), constructing its *conditional pattern base* (a "subdatabase", consisting the prefix paths in the FP-tree containing the suffix pattern), then forming conditional FP-tree, and performing mining recursively on this tree. It uses the least frequent items as a suffix, offering good selectivity. Performance studies demonstrate that the method substantially reduces search time.

DFS is algorithms take a more incremental approach as it generates possible frequent sequences and uses a divide-and-conquer approach. This algorithm mainly makes an attempt to lessen the search space.

III. ACKNOWLEDGMENT

I would like to thank Dr. J.W. Bakal Sir and Madhu Madam for facilitating all the necessary inputs, study material and resources and guiding me with their rich experience. I would especially like to thank my parents, in-laws and my husband for their unconditional support.

IV. REFERENCES

[1] Theint Aye, "Web cleaning for mining of web usage patterns", *International Conference on Computer research and Development(ICCRD)*, pages 490-494, Vol. 2, May 2011

[2] K. R. Suneetha, Dr. K. R. Krishnamoorthy, "Identifying User Behavior by Analyzing Web Server Access Log", *International Journal of Computer Science and Network Security(IJCSNS)*,pages 327-331, VOL.9 No.4, April 2009

[3] Guangyuan Li Qin Xiao Qinbin Hu Changan Yuan, **"**An Efficient Algorithm for Mining Frequent Sequences in Dynamic Environment", *in Granular Computing, 2009, GRC '09. IEEE International Conference*, pages: 329 – 333, Aug. 2009

[4] Jiawei Han · Hong Cheng · Dong Xin · Xifeng Yan , "Frequent pattern mining: current status and future directions" ,*In Proceedings of International Conference on Data Mining Knowledge Discovery Journal(DATAMINE)*, pages 55-86, Vol. 15 No.1, March 2007

[5] Murat Ali Bayir, Ismail H. Toroslu, Ahmet Cosar," Performance Comparison of Pattern Discovery Methods on Web Log Data", *Computer Systems and Applications, IEEE International Conference*, pages 445 – 451, April 2006

[6] Osmar R. Zaıane, Mohammad ElHajj, "Pattern Lattice Traversal by Selective Jumps", *in Proc. 2005 Int'l Conf. on Knowledge Discovery and Data Mining (ACM SIGKDD*), pp 729-735, Chicago, August, 2005

[7] Xidong Wang, Yiming Ouyang, Xuegang Hu, Yan Zhang, "Discovery of User Frequent Access Patterns on Web Usage Mining", *Computer Supported Cooperative Work in Design Proceedings 8th IEEE International Conference*, pages 765 – 769, Vol 1, November 2004

[8] Jiawei Han, Jian Pei, Yiwen Yin ,Runying Mao ,"Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*", In Proceeding of International Conference on Data Mining and Knowledge Discovery*, 8, pp. 53–87, 2004

[9] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Mei-Chun Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 11, November 2004

[10] Jaideep Srivastava, Robert Cooleyz , Mukund Deshpande, Pang-Ning Tan, "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data", *In proceedings of the 9th IEEE International conference on Tools with*

Artifical Intelligence (ICTAI'97), pages 558-567, 1997

[11] R. Srikant and R. Agrawal, "Mining Sequential Patterns," *Proceedings of Fifth International Conference Extending Database Technology (EDBT '96),* pp. 3-17, Mar. 1996

[12] *Ramakrishnan Srikant, Rakesh Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements", In Proceedings of the 11th International Conference on Data Engineering, pages 3-14, 1995.*

◈◈◈