Efficient Web Log Mining Using Enhanced Apriori Algorithm with Hash Tree and Fuzzy

S.Veeramalai [1], N.Jaisankar [2] and A.Kannan [3]

Department of Information Science and Technology, Anna University –Chennai
Chennai-600025, Tamil Nadu, India
[1] veera2000uk@gmail.com , [2] jaisasi_win@yahoo.com , [3] kannan@annauniv.edu

Abstract

Web usage mining is the type of Web mining activity that involves the automatic discovery of user access patterns from one or more Web servers. In this paper we analyze the pattern using different algorithms like Apriori, Hash tree and Fuzzy and then we used enhanced Apriori algorithm to give the solution for Crisp Boundry problem with higher optimized efficiency while comparing to other algorithms.

Keyword

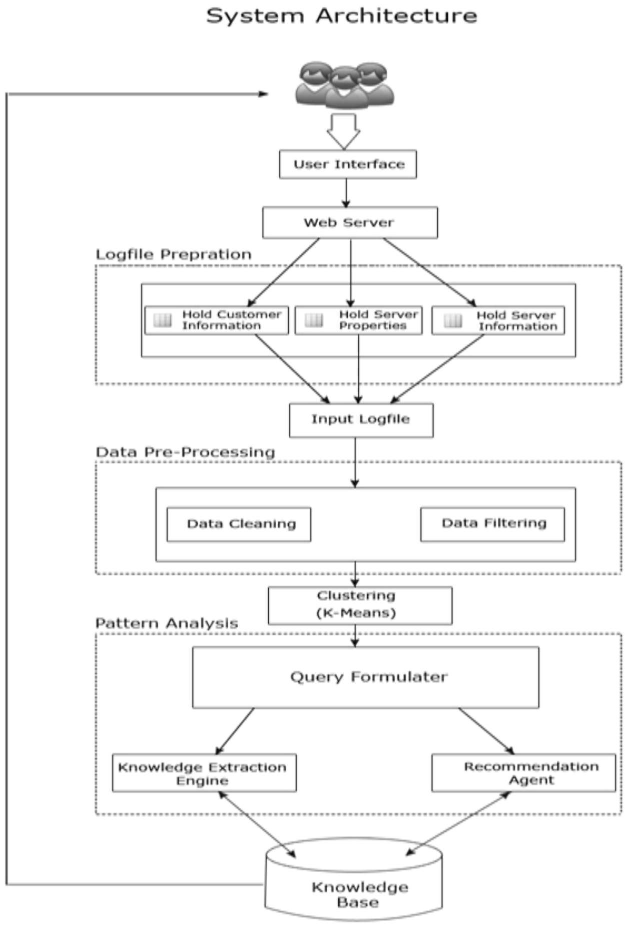
Data mining, Web mining, Web log, Association rule, Apriori, Fuzzy.

1. Introduction

The aim in web mining is to discover and retrieve useful and interesting patterns from a large dataset. In web mining, this dataset is the huge web data [7]. Web data contains different kinds of information, including, web structure data, web log data, and user profiles data [9, 10]. Web mining is the application of data mining techniques to extract knowledge from web data, where at least one of structure or usage data is used in the mining process. Web usage mining has various application areas such as web pre-fetching, link prediction, site reorganization and web personalization [1, 2, and 14]. Most important phases of web usage mining are the [2,3] reconstruction of user sessions by using heuristics techniques and discovering useful patterns from these sessions by using pattern discovery techniques like association rule mining, Apriori etc [4,3]. We propose an integrated system (Web Tool) for applying data mining [16] techniques such as association rules or sequential patterns on access log files. The fig.1 represents the System architecture diagram for our paper.

DOI : 10.5121/ijcsit.2010.2406

2. System Architecture



[Fig .1]

2.1 Web Usage Mining

Web Usage Mining consists of three phases which are named data processing, pattern discovery and pattern [2,19] analysis. This phase has two parts called data cleaning and filtering[1, 19]. Filtering is the most important task in web usage mining since the quality of mined patterns depends on this directly. In the pattern discovery phase, Special pattern discovery algorithms are applied on raw data which is output of the data processing phase [3,7]. In the pattern analysis phase interesting knowledge is extracted from frequent patterns and these results are used in various applications such as personalization, system improvement, site modification.

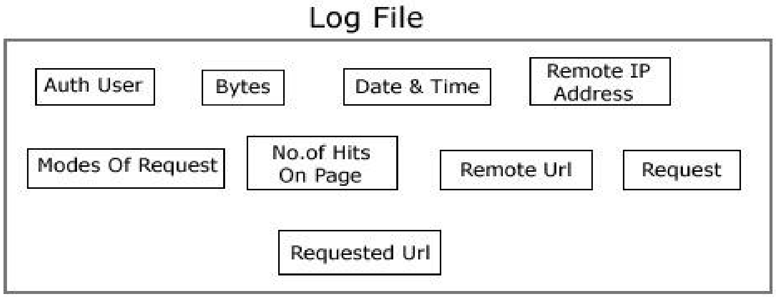
3. Generation of Log file

The quality of the patterns discovered in web usage mining process highly depends on the quality of the data used in the mining processes [11]. When the web browser traces the web pages and stores the Server log file. Web usage data contains information about the Internet addresses of web users with their navigational behavior the basic information source for web usage [17].

3.1. Web Server Data:

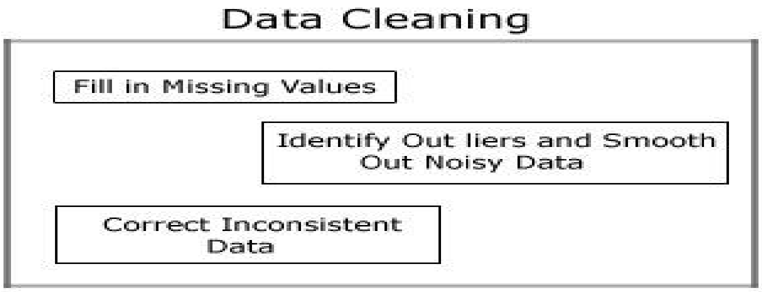
When any user agent (e.g., IE, Mozilla, Netscape, etc) hits an URL in a domain, the information related to that operation is recorded in an access log file. In the data processing task, the web log data can be [4, 13 and 17] preprocessed in order to obtain session information for all users. Access log file on the server side contains log information of user that opened a session [17, 18]. These records have seven common fields, which are:

1. User's IP address, 2. Access date and time, 3. Request method (GET or POST),
4. URL of the page accessed, 5. Transfer protocol (HTTP 1.0, HTTP 1.1,),
6. Success of return code. 7. Number of bytes transmitted.

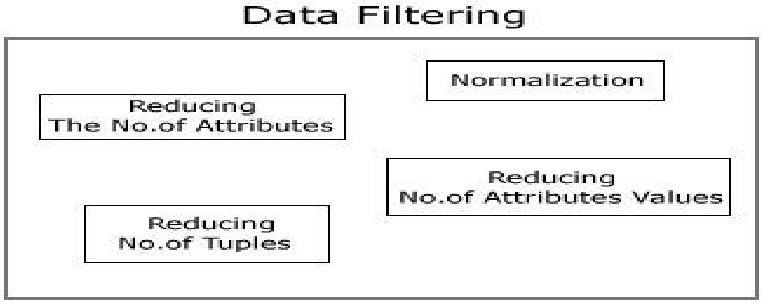


Log File Functional Diagram [Fig.2]

3.2. Preprocessing


Data Cleaning
Fill in Missing Values
Identify Out liers and Smooth Out Noisy Data
Correct Inconsistent Data

Data Cleaning Diagram [Fig.3]



Data Filtering Diagram [Fig.4]

Data Cleaning is also a customized step [2, 4], which includes integrating different usage logs, and parsing data from these usage logs. This process can be performed by detecting file types which have suffixes such as text and hyperlink. The nature of the data to be clustered plays a key role when choosing the right algorithm for clustering [5, 11and 12].

3.3. Pattern Analysis Phase

Pattern discovery is the main issue in both web usage mining and data mining [1, 2 and3]. The search space increases exponentially as the lengths of patterns to be discovered increase [6, 12 and 13]. Also, discovered patterns must be interpreted and understandable knowledge must be extracted from them. Also the comparison of Pattern Discovery on Web Logs Data [4, 5]. Commonly used pattern discovery algorithms that are also suitable for Web Usage Mining are [10, 11and 12]

4. General Rule for Proposed 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. In the first step all possible combination of items, called the candidate item set (Ck) is generated [20, 21]. In 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 [6, 4 and 3].

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. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them [20,21 and 22]. Those processes iterated until the antecedent becomes empty. Since the second sub problem is quite straight forward, most of the researches focus on the first sub problem. The Apriori algorithm finds the frequent sets L In Database D .A k-item set is an item set with exactly k items in it [19, 18].

An association rule is about the relationship between two disjoint item sets, X and Y. It is denoted as $X \Rightarrow Y$. It presents the pattern → When X occurs, Y also occurs.

Association rules do not represent any sort of causality or correlation between the two items sets.

$X \Rightarrow Y$ does not mean that X causes Y. There is no causality.

$X \Rightarrow Y$ can imply different meaning than $Y \Rightarrow X$, unlike correlation.

Support for an item set X in a transactional database D is defined as count(X) / |D|.

For an association rule $X \Rightarrow Y$, we can calculate

Support($X \Rightarrow Y$) = support (XY) = support(X union Y).

Confidence($X \Rightarrow Y$) = support (XY) / support(X).

Support (S) and Confidence (C) can also be related to joint probabilities and conditional probabilities as follows

Support($X \Rightarrow Y$) = P(XY). Confidence ($X \Rightarrow Y$) = P(Y/X).

SET k = 1;

Find frequent item set, from the set of all candidate item sets;

(C_k : Candidate item set of size k) , (L_k : frequent item set of size k)

Scan D and count each item set in C_k, if the count is greater than minSupp, and then add that itemset to L_k

For k = 1, C_1 = all item sets of length = 1

For k > 1, generate C_k from L_{k-1} as follows:

The join step:

C_k = k-2 way join of L_{k-1} with itself.

If both $\{a_1,..,a_{k-2}, a_{k-1}\}$ & $\{a_1,.., a_{k-2}, a_k\}$ are in L_{k-1}, then add $\{a_1,..,a_{k-2}, a_{k-1}, a_k\}$ to C_k

Remove $\{a_1, \ldots, a_{k-2}, a_{k-1}, a_k\}$, if it contains a non-frequent (k-1) subset.

For every non-empty subset A of X

Let B = X - A.

$A \Rightarrow B$ is an association rule if Confidence $(A \Rightarrow B) \geq$ minConf.

Where, confidence $(A \Rightarrow B)$ = support (AB) / support (A), and

Support $(A \Rightarrow)B .)$=Support (AB)

To overcome boundary problem, Find out the min support, Scan D and count each itemset in C_k, if the count is greater than minSupp, then add that itemset to L_k

For k = 1, C_1 = all item sets of length = 1, For k > 1, generate C_k from L_{k-1} as follows:

The join step:

C_k = k-2 way join of L_{k-1} with itself.

If both $\{a_1,..,a_{k-2}, a_{k-1}\}$ & $\{a_1,.., a_{k-2}, a_k\}$ are in L_{k-1}, then add $\{a_1,..,a_{k-2}, a_{k-1}, a_k\}$ to C_k

The items are always stored in the sorted order.

The prune step:

Remove $\{a_1, \ldots, a_{k-2}, a_{k-1}, a_k\}$, if it contains a non-frequent (k-1) subset. For every non-empty subset A of X

Let B = X - A.

$A \Rightarrow B$ is an association rule if Confidence $(A \Rightarrow B) \geq$ minConf.

Where, confidence $(A \Rightarrow B)$ = support (AB) / support (A), and Support $(A \Rightarrow)B .)$ = 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, $L_{k-1.}$ 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, $L_{k-1.}$

LINKS	**VALUES**
.COM	**1**
.EDU	**2**
.ORG	**3**
.IN	**4**
.NET	**5**

C **L**

TID	LINKS
001	1 3 4
002	2 3 5
003	1 2 3 5
004	2 5

Links in a particular log file

C_1 **L_1**

ITEMSET	LINKS
{1}	**2**
{2}	**3**
{3}	**3**
{4}	**1**
{5}	**3**

Links in a particular Item SET

C_{1S} **L_{1S}**

ITEMSET	LINKS
{1}	**2**
{2}	**3**
{3}	**3**
{5}	**3**

C_2

ITEMSET
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

C_2 **L_2**

ITEMSET	LINK
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

C_{2s} **L_{2s}**

ITEMSET	LINK
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3 **L_2**

ITEMSET	LINK
{2 3 5 }	2

C_4

ITEMSET
{2 3 5}

4.1. Rule for an Efficiency Improvement

One way to improve efficiency of the APRIORI would be to

1. Prune without checking all k-1 subsets.
2. Join without looping over the entire set, $L_{k-1.}$
3. *Speed up searching and matching*.
4. *Reduce the number of transactions* (a kind of instance selection).
5. Reduce the number of passes over data on disk. E.g. *Reducing scans via Partition.*
6. Reduce number of subsets per transaction that must be considered.
7. Reduce number of candidates (a kind of feature selection).

This can be done by using *hash trees*

A *Hash tree* stores all candidate k-item sets and their counts. The 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 C_2 = 12, 13, 15, 23, 25, 35 is shown below

```
          {}               **root
/1          |2     \3      **edge+label
/2 |3 \5 /3 \5     /5
[12:][13:][15:] [23:][25:] [35:]                     **leaves
```

An internal node v at level m contains, bucket pointers. These tell which branch is the next one to be traversed. The hash of the m^{th} item is used to decide this.

Join step using Hash Tree

Only the frequent k-1 item sets, which have common parents, should be considered for the joining step. So checking all k-1 item sets in $L_{k-1}$is avoided.

Prune step using Hash Tree

To determine if a k-1 itemset is frequent, we have to look only for those item sets that have common parents, and thus avoid going through all k-1 item sets in L_{k-1}.To overcome crisp boundary problem , find out the min support ,Scan D and count each itemset in C_k, if the count is greater than min Supp, then add that itemset to L_k

For k = 1, C_1 = all item sets of length = 1, For k > 1, generate C_k from L_{k-1} as follows:

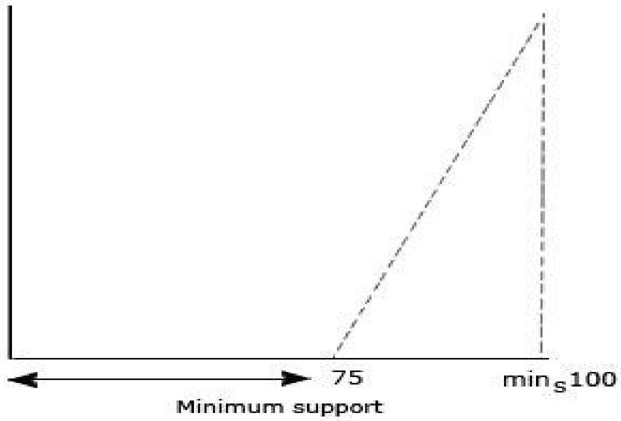
The join step:

C_k = k-2 way join of L_{k-1} with itself. If both $\{a_1,..,a_{k-2}, a_{k-1}\}$ & $\{a_1,.., a_{k-2}, a_k\}$ are in L_{k-1}, then add $\{a_1,..,a_{k-2}, a_{k-1}, a_k\}$ to C_k The items are always stored in the sorted order.

The prune step:

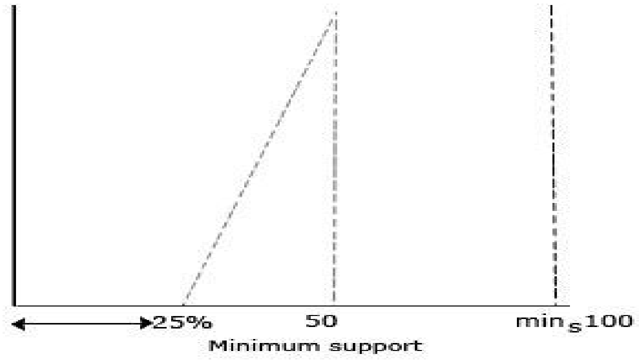
Remove $\{a_1, \ldots,a_{k-2}, a_{k-1}, a_k\}$, if it contains a non-frequent (k-1) subset.For every non-empty subset A of X Let B = X - A.

$A \Rightarrow B$ is an association rule if Confidence ($A \Rightarrow B$) $\geq$ minConf. Where, confidence ($A \Rightarrow B$) = support (AB) / support (A), and Support (A $\Rightarrow$)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 et, $L_{k-1.}$



Crisp Boundry Analysis
Using Fuzzy

Now the minimum support value will have the crisp boundry 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 suppot by 50% of the total item set, sice we calculate the min support from Apriori hash tree the result will be in ascending so the optimized result will not be behind the 50% region , **Mtc=(lk:no of item set)/(1/2 of total itemset)**



Crisp Boundry Analysis
Using Modified Algorithm

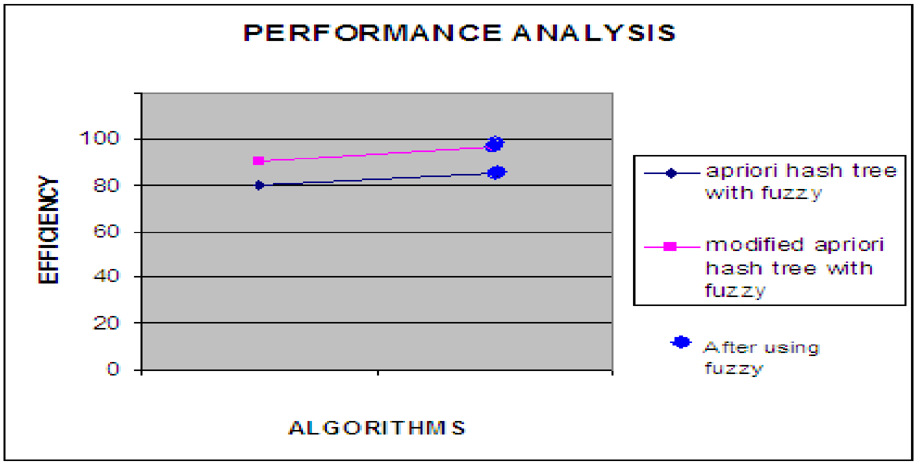
Now we have obtained the optimized value compare to the previous Apriori algorithm Thus we can overcome the crisp boundary problem by our modified algorithm and we have improved the efficiency by our algorithm.

5. Result and Discussion

There are many major contributions that are involved in this work with respect to Information retrieval from the web. 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.

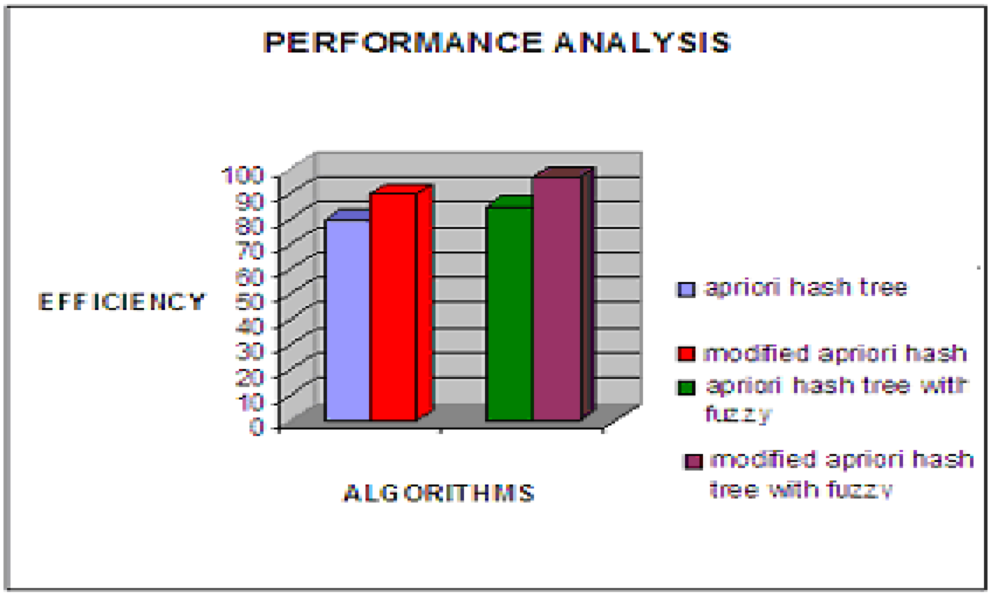
ALGORITHMS	**EFFICIENCY**
Apriori Hash Tree	80
Apriori Hash Tree with Fuzzy	85
modified Apriori Hash Tree	90
modified Apriori Hash Tree with Fuzzy	97

6. Performance Evaluation Chart

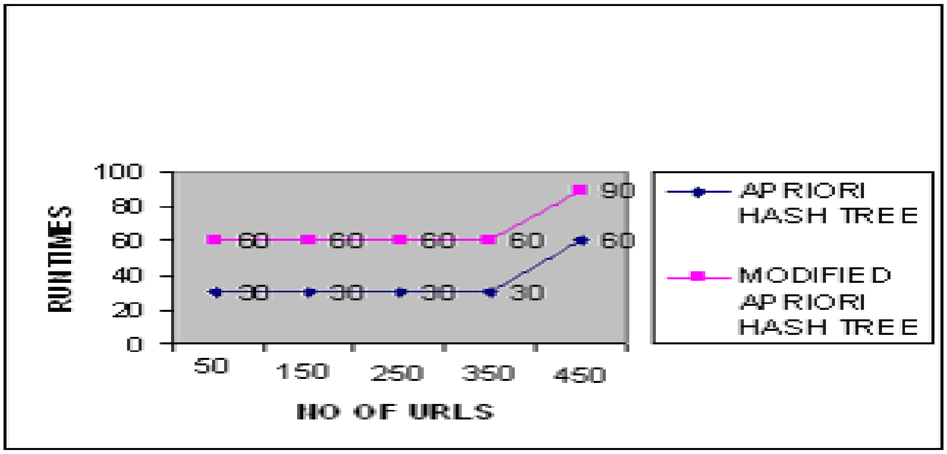


Efficiency of two algorithms [Fig.5]

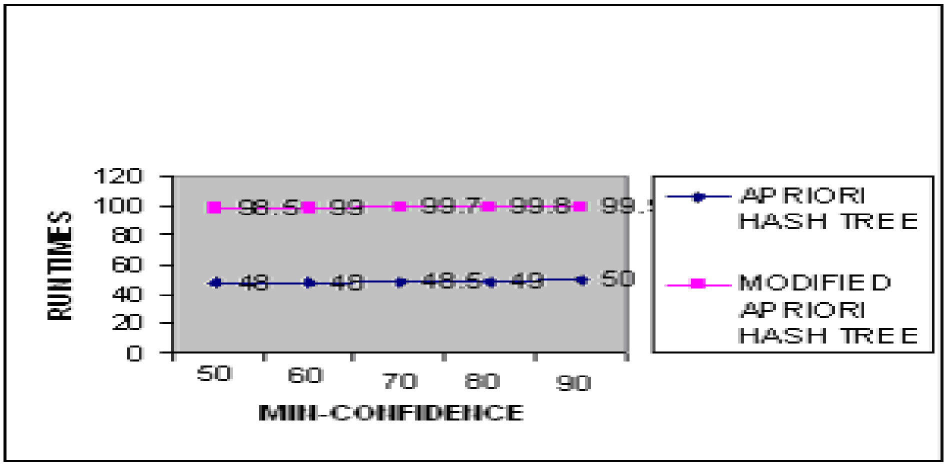
The efficiency of the apriori hash tree algorithms and modified apriori hash tree with fuzzy algorithm comparison is given in [fig5] [fig6]. [fig7]. From the given bar chart, we can understand that the modified Apriori hash tree with fuzzy is more effective.



Efficiency of two algorithms Fig.6]



Performance Analysis with URL [Fig .7]



Performance Analysis with confidence [Fig.8]

7. Conclusion & Future Work

With modified Apriori algorithm the structure is formulated with the help of hash tree algorithm. Our design tool allows experimenting with the concepts of fuzzy modify association rules. It finally we analyzed the crisp boundary problem in the combined algorithm and it is overcome by our modified association Apriori hash tree fuzzy algorithm and the efficiency is increased in it. In future, our work will be enhanced to develop the optimizing search application system.

8. References

[1] Diamanto Oikonomopoulou, Maria Rigou, Spiros Sirmakessis, Athanasios Tsakalidis,, "Full-Coverage Web Prediction based on Web Usage Mining and Site Topology". Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04) 0-7695-2100-2/04, April 20, 2009.

[2] Junjie Chen and Wei Liu, "Research for Web Usage Mining Model", International Conference on Computational Intelligence for Modelling Control and Automation,and International Conference on Intelligent Agents,Web Technologies and Internet Commerce (CIMCA-IAWTIC'06) 0-7695-2731-0/06 © 2006 IEEE

[3] Olfa Nasraoui, Maha SolimanEsin Saka, Antonio Badia, Memberand Richard Germain "A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites", IEEE Transactions On Knowledge And Data Engineering, Vol. 20, No. 2, February 2008

[4] F. Masseglia, P. Poncelet, and M. Teisseire, "Using data mining techniques on web access logs to dynamically improve hypertext structure". In ACM SigWeb Letters, 8(3): 13-19, 1999. Web Site Link: http://portal.acm.org/citation.cfm?id=951440.951443.

[5] Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W., & Musen, M.A. (2001), "Creating Semantic Web Contents with Protégé-2000". IEEE Intelligent Systems 16(2), 60-71.

[6] Lassila, O. (1998), "Web Metadata: A Matter of Semantics ". IEEE Internet Computing 2(4), 30-37.

[7] Tak Woon Yan , Matthew Jacobsen , Hector Garcia-Molina , Umeshwar Dayal, From user access patterns to dynamic hypertext linking, Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems, p.1007-1014, May 1996, Paris, France

[8] ZHANG Xuewu, SU Fenzhen, DU Yunyan Institute of Geographic Sciences and Natural Resources, Research, CAS Beijing, China, "Association Rule Mining on Spatio-temporal Processes "2008 IEEE.

[9] Jike Ge Yuhui Qiu Zuqin Chen Shiqun Yin Faculty of Computer and Information Science. Southwest University, Chongqing, China gjkid@swu.edu.cn "Technology of Information Push Based on Weighted Association Rules Mining".

[10] Agrawal R, Imielinski T, Swami A. (1993): Mining association rules between sets of items in large databases. Proceeding of ACM SIGMOD Interation Confidence. Management of Data, Washington: 207-216.

[11] Ai-Bo Song , Zuo-Peng Liang, Mao-Xian Zhao, Yi-Sheng Dong, "Mining Web Log data based on Key path", Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 November 2002.

[12] Mike Perkowitz, Oren Etzioni, "Adaptive Web Sites: Automatically Synthesizing Web Pages", Department of Computer Science and Engineering, Box 352350 University of Washington, Seattle, WA 98195, 1998, American Association for Artificial Intelligence (www.aaai.org).

[13] V elasquez, Bassi J D, YasudaA. "Mining Web data to create online navigation recommendations". Data Mining, 2004:166-172. Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04) 0-7695-2142-8/04 IEEE.

[14] Sutheera Puntheeranurak , Hidekazu Tsuji, "Mining Web logs for a Personalized Recommender System", 0-7803-8932-S/05/ 2005 IEEE

[15] Junzhong Ji, Zhiqiang Sha, Chunnian Liu ,“Online Recommendation Based on Customer Shopping Model in E-Commerce”, Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI’03) 0-7695-1932-6/03 © 2003 IEEE.

[16] Fan Wang, Gagan Agrawal, Ruoming Jint, Helen Piontkivska, ”SNPMiner: A Domain-Specific Deep Web Mining Tool”, 1-4244-1509-8/07/ @ 2007 IEEE

[17] R. Vaarandi, “A Data Clustering Algorithm for Mining Patterns from Event Logs,” in Proceedings of the 3rd IEEE Workshop on IP Operations and Management. Kansas City, MO, USA: IEEE Press, October 2003, pp. 119 – 126.

[18] James H. Andrews, Member, IEEE, and Yingjun Zhang, “General Test Result Checking with Log File Analysis ”, 0098-5589/03/ @ 2003 IEEE Published by the IEEE Computer Society.

[19] Pawan Lingras ,Saint Mary’s University, “Rough Set Clustering for Web mining”,
0-7803-7280-8/02/ @ 2002 IEEE

[20] Li-Juan Huang, “Apriori Algorithm’s Application In The Design For Individualized Virtual Shop On The Internet ”, Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007, 1-4244-0973-X/07/ ©2007 IEEE.

[21] Tomasz Str, Kowski Henryk, Rybiski1. A distributed version of Apriori rule generation based on rough set theory. Workshop "Concurrency, Specification and Programming 2004”.

[22] Frans Coenen, Paul Leng, and Shakil Ahmed, ‘IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 16, NO. 6, JUNE 2004’, “Data Structure for Association Rule Mining-Trees and P-Trees”.

Authors

S.Veeramalai received the Master Degree in Computer Science and Engineering from SASTRA University in 2002. He is currently a PhD candidate in the Department of Information Science and Technology at Anna University-Chennai, Tamil Nadu. His research interest includes the data mining and web mining.

N.Jaisankar is Master Degree holder in Computer Science and Engineering. He is currently a PhD candidate in the Department of Information Science and Technology at Anna University-Chennai, Tamil Nadu. His research interest includes the network security and automata theory.

Dr.A.Kannan is working as a Professor in the Department of Information Science and Technology at Anna University-Chennai. He has published sixty technical papers in various journals and conferences. His research interest includes Database Management System, Artificial Intelligence, Data mining and Network Security.