IETE Technical Review

ISSN: 0256-4602 (Print) 0974-5971 (Online) Journal homepage: www.tandfonline.com/journals/titr20

An Efficient Algorithm for Extracting High Utility Itemsets from Weblog Data

Brijesh Bakariya & G.S. Thakur

To cite this article: Brijesh Bakariya & G.S. Thakur (2015) An Efficient Algorithm for Extracting High Utility Itemsets from Weblog Data, IETE Technical Review, 32:2, 151-160, DOI: 10.1080/02564602.2014.1000396

To link to this article: https://doi.org/10.1080/02564602.2014.1000396

Published online: 30 Jan 2015.

An Efficient Algorithm for Extracting High Utility Itemsets from Weblog Data

Brijesh Bakariya and G.S. Thakur

Department of Computer Applications, Maulana Azad National Institute of Technology, Bhopal, India

ABSTRACT

High utility itemset refers to those set of items which has high utility such as profit in a database. High utility of itemset plays a crucial role in real life. In recent years, various algorithms have been proposed for finding high utility itemset but unfortunately they are not completely relevant at the time and space point of view. In the data mining field, high utility itemset can be found in different categories of data like time series, categorical, etc. Log data is useful for finding behaviour of the user in different aspects. In this paper, we have proposed an algorithm named HUIM (High Utility Itemsets Mining) and construct HUI-FP (High Utility Itemsets-Frequent Pattern) Tree for efficiently mining high utility itemsets from log database. The behaviour of the user can be predicted through the high utility of every visited page. We have also proposed pattern generation technique based on cosine similarities among itemsets. These techniques generate strong patterns, and customized users profile according to that pattern. The proposed algorithm is better than the previous state of the art algorithm for high utility itemset generation.

***Keywords*:**
Complexity, Cosine similarity, Frequent pattern, High utility itemset, Weblog.

1. INTRODUCTION

Web-mining aims to discover useful information or knowledge from the web hyperlink structure, page content, and usage data [1]. Data mining plays an important role to find unknown patterns and hidden information[2]. There are different techniques in data mining such as classification, clustering, and association rule mining to extract the hidden information from a heterogeneous database. One of the most important and researched techniques of data mining is Association Rule Mining [3,4]. It aims to extract interesting correlations, frequent patterns, associations, or casual structures among sets of items in the transaction databases or other data repositories [5,6]. The relative importance of every item cannot be considered to overcome such problems, a rule called Weighted Association Rule Mining (WARM) [7–9]. In WARM, a weight is considered for every transaction. But there are certain problems that exist in WARM since some infrequent items have a heavy weight [10] and in this technique the quantity of items is not taken into consideration. Utility itemset mining is an important field of data mining; its aim to extract high utility itemset from databases and those itemsets, which have high project, are founded [11]. Here, the utility of itemset means profitability importance and interestingness of the item [12,13]. In a transactional database, the utility of an item depends on two types of utility: internal utility and external utility as we are shown in Figure 1.

Internal utility consists importance to some item in every transaction and external utility consist importance of distinct item [14,15]. An itemset is called high utility itemset, when its utility is not less than the minimum defined utility. Mining of high utility itemset from the database is a very important task. There are various applications of high utility itemset mining, such as website clickstream analysis, marketing, web usage mining, user behavior prediction, e-commerce, mobile-commerce environment, genes prediction in bioinformatics, drug-prediction in medical, etc. [16,17]. Many such techniques of frequent pattern mining such as sequential pattern mining, sequential pattern mining based on constraints [18,19], mining of clickstream data, and mining of relational data have been proposed [20]. Out of all these techniques of frequent pattern mining, a technique called association rule mining is a very important one [21].

The two very famous algorithms of association rule mining are a-priori algorithm and Frequent Pattern Growth (FP-Growth) mining [3,5,22]. The performance point of view, Apriori algorithm is not very efficient as it demands database to be scanned again and again in turn increases the time complexity. To overcome this

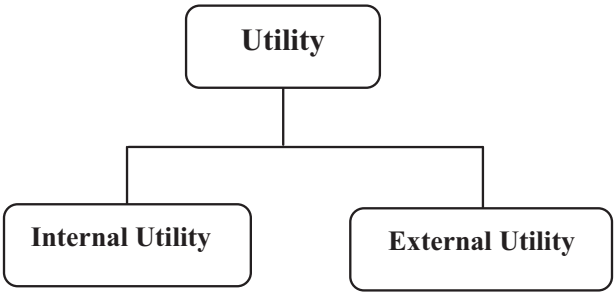

Figure 1: Taxonomy of utility.

problem, frequent pattern growth mining algorithm has been used [3,23]. Here a data structure is used to store data which is called frequent pattern tree [12,24]. In this algorithm, frequent pattern can be found out only after scanning the database twice. Hence, this algorithm is efficient in terms of complexity. Because its focus only on importance of utility not on quantity of itemsets. This efficient, high utility pattern can obtain.

2. RELATED WORKS

In this section, we will discuss research on high utility itemset mining. Tseng et al. [15] propose two algorithms named Utility Pattern Growth (UP-Growth) and UP-Growth+ for mining high utility itemsets with a set of effective strategies for pruning candidate itemsets. The information of high utility itemsets is maintained in a tree-based data structure named a Utility Pattern Tree (UP-Tree) so that candidate itemsets can be generated efficiently with only two scans of the database. The performance of UP-Growth and UP-Growth+ is compared with the FP-Growth and IHUP algorithms on many types of both real and synthetic data sets. Experimental results show that the proposed algorithms, especially the UP-Growth+ are not only reducing the number of candidates effectively, but also outperform other algorithms substantially in terms of run-time, especially when databases contain many long transactions. Ramaraju et al. [25] proposed a novel conditional high utility tree to compress transactional databases in two stages to reduce the search space and a new algorithm called HU-Mine is proposed to mine complete set of high utility itemsets. This algorithm needs only two database scans in contrast to many scans of existing algorithm. The results of the proposed work are compared with two-phase algorithms. Liu et al. [26] proposed an algorithm, called HUI-Miner (High Utility Itemset Miner) for high utility itemset mining. HUI-Miner uses a novel structure, called utility-list, to store both the utility information about an itemset and the heuristic information for pruning the search space of HUI-Miner. By avoiding the costly generation and utility computation of numerous candidate itemsets, HUI-Miner can efficiently mine high utility itemsets from the utility lists constructed from a mined database. They compared HUI-Miner with the $IHUPT_{TWU}$, UP-Growth and UP-Growth+ algorithms on various databases, and experimental results showed that HUI-Miner outperforms these algorithms in terms of both running time and memory consumption. Ahmed et al. [27] proposed two novel tree structures, called UWAS tree (utility-based web access sequence tree), and IUWAS-tree (incremental UWAS tree), for mining web access sequences in static and dynamic databases, respectively. In this approach, the structures handled both forward and backward references, static and dynamic data, avoided the label-wise candidate generation and test methodology. In addition, they did not scan databases multiple times and considered both internal and external utilities of a web page. Extensive performance analyses show that our approach is very efficient for both static and incremental mining of high utility web access sequences. Shankar et al. [14] present novel algorithm Fast Utility Mining (FUM) which finds all high utility itemsets within the given utility constraint threshold. It is faster and simpler than the original UMining algorithm. In their experimental evaluation on transaction datasets showed that our algorithm executes faster than UMining algorithm and exceptionally faster when more itemsets have identified as high utility itemsets and when the number of distinct items in the database increases. They also suggested a novel method of generating different types of itemsets such as High Utility and High Frequency itemsets (HUHF), High Utility and Low Frequency itemsets (HULF), Low Utility and High Frequency itemsets (LUHF) and Low Utility and Low Frequency itemsets (LULF) using a combination of FUM and Fast Utility Frequent mining (FUFM) algorithms. Ahmed et al. [28] propose three novel data structures to efficiently perform incremental and interactive HUP mining. In the first, the data structure – using incremental HUP Lexicographic Tree ($IHUP_L$-Tree) – is arranged with the items in lexicographic order and inserted as a branch inside the tree. In this tree data structure, tree will construct according to transaction weighted utilization of an itemset. The second data structure is the IHUP Transaction Frequency Tree ($IHUP_{TF}$-Tree); in this, tree nodes are arranged in descending order according to their transaction frequency so that items occurring in many transactions can be kept in the upper part of the tree, and therefore, higher prefix-sharing can be achieved and the third data structure is IHUP Transaction Weighted Utilization Tree ($IHUP_{TWU}$-Tree) is designed based on the transaction weighted utilization value of items in descending order. Fournier-Viger et al. [29] propose an algorithm named FHM (Fast High

Utility Miner). In this algorithm, they compare FHM to HUI-Miner in this analysis of item co-occurrences to reduce the number of join operations that need to be performed. They test FHM with four real-life datasets and reduce the number of join operations by up to 95% and it is up to six times faster than HUI-Miner algorithm. The state of the art algorithm has the following problems. In HUI Miner [26], the algorithm is not applying preprocessing techniques for the database, and it is also not applicable for weblog data. IHUP [28] generates a frequent pattern for all candidate itemset (favourable and unfavourable). The FHM [29] algorithm occupies a large number of memory space and is time-consuming because it computes all the candidates which makes it difficult to handle larger databases. We have proposed an algorithm, namely HUIM, to discover from this problem.

3. GENERAL TERMS AND DEFINITIONS

The basic terms and formal definition of high utility itemset mining based on related concepts are described below. Let $P = \{p_1, p_2, p_3 \ldots p_m\}$ be a set of visited pages by user. Each item p_i $(1 \leq i \leq m)$ has a unit profit $pft(p_i)$. An itemset X is a set of k different items $\{i_1, i_2, i_3 \ldots i_k\}$, where $i_j \in P$, $1 \leq j \leq k$, k is a length of X. An itemset with length k is called a k-itemset. A transaction database $D = \{T_1, T_2, T_3 \ldots T_n\}$ contains a set of transactions, and each transaction T_d $(1 \leq d \leq n)$ has a unique identifier d, called T_{id}. Each item p_i in transaction T_d is associated with a quantity q (p_i, T) that is, the purchased quantity of p_i in T_d.

Definition 3.1: Utility of any item p_i in transaction T_d is the measurement of quantity which is represented by $U(p_i, T_d)$ [15].

$$U(p_i, T_d) = pft(p_i) \times q(p_i, T_d) \tag{1}$$

Definition 3.2: Utility value of itemset X in T_d is denoted by $U(X, T_d)$ [15].

$$U(X, T_d) = \sum_{p_i \in X \wedge X \subseteq T_d} U(p_i, T_d) \tag{2}$$

Definition 3.3: Utility value of itemset X in D is denoted as $U(X)$ [15].

$$U(X) = \sum_{X \subseteq T_d \wedge T_d \in D} U(X, T_d) \tag{3}$$

Definition 3.4: In any transaction, T_d transaction utility is denoted by "TU" which is the sum of the total profit of items in T_d. It is defined as follows [15].

$$\mathrm{TU}(T_d) = \sum_{p_i \in T_d} U(p_i, T_d) \tag{4}$$

Definition 3.5: Transaction Weighted Utilization (TWU) of any itemset is denoted by TWU (X) [15].

$$\mathrm{TWU}(X) = \sum_{X \subseteq T_d \in D} \mathrm{TU}(T_d) \tag{5}$$

Definition 3.6: Minimum Utility Threshold (MUT) is a value which is set by the user and its value depends on the total transaction utility [15].

$$\mathrm{MUT} = \mathrm{UPP} \times \sum_{T_d \in D} \mathrm{TU}(T_d) \tag{6}$$

where UPP is a user preferred percentile.

Definition 3.7: X is a high transaction weighted utility itemset only when TWU$(X) >=$ MUT otherwise it will be a low transaction weighted itemset.

Definition 3.8: Itemset X is a high utility itemset if $U(X) >=$ MUT. High utility itemset means, determining itemset which fulfils the criteria of $U(X) >=$ MUT.

Definition 3.9: For any itemset $X = \{i_1, i_2 \ldots i_n\}$ is a K-itemset where K is number of itemset and it should be greater than or equal to 2, for all $2 \leq n \leq$ K. The value of $Y = \{i_n\} \cup X$ then cosine similarity [30] of Y calculated from Eq. (7).

$$\mathrm{Cos(Y)} = \frac{\sum_{n=1}^{K} \mathrm{Supp}(Y) \times \mathrm{Util}(Y)}{\sqrt{(\prod)^{K} \mathrm{Supp}(Y) \times \mathrm{Util}(Y)}} \tag{7}$$

In this paper, one of the parameters we are considering is time from weblog. Here in Table 2, maximum durations are five hours. So that is 24 hours in a day divided into five slots, first four slots will be five hours and the last slot will be four hours. According to timestamp, web users can visit any number of web pages at any time and all the user's activities are maintained through the server.

4. FINDING UTILITY OF TRANSACTIONAL DATASET

Weblog data is automatically created and maintained by the web server, when user hit on the website then log will be stored on the web server. Before applying

Table 1: Preprocessed web log sample

IP address	User name	Timestamp	Access request	Result status code	Bytes transferred	Referrer URL	User agent
123.456.78.2	——	25/Apr/1998:03:04:41 -0500	GET p_1.html http/1.0	200	1923	p_1	Mozilla/4.05
123.456.78.9	——	25/Apr/1998:03:05:20 -0500	GET p_2.html http/1.0	200	2828	p_2	Mozilla/4.05
123.456.78.9	——	25/Apr/1998:03:05:28 -0500	GET p_1.html http/1.0	200	1923	p_1	Mozilla/4.05
123.456.78.5	——	[25/Apr/1998:03:05:41 -0500	GET p_3.html http/1.0	200	2736	p_3	Mozilla/4.05
123.456.78.9	——	25/Apr/1998:03:05:54 -0500	GET p_4.html http/1.0	200	2828	p_4	Mozilla/4.05

data mining technique for server log [12,21,31], it needs to preprocess that server log and after applying preprocessing technique they will transform in structure form and split according to its attribute. In Table 1, we are showing a sample of the preprocessed weblog. When the user interacts with the internet than their visited page entries are written in the transactions table. In this, we have taken five transactions; by every transaction, we maintain the sequence of the pages.

The profit of an itemset depends not only on the support (total number of times an itemset occurs in a transactional database out of the total number of transactions) of the itemset but also on the prices or weight of items in that itemset [10]. Here we will have to predefine profit value of each and every item. As we are shown in Table 3, predefined profit value of p_1, p_2, p_3, p_4, p_5, p_6, and p_7 item. Utility of every transaction is found out by the given value of each transaction page and profit.

If we have to find the utility of more than one page, then we have to find a utility on the given page by summing up all the pages. The itemset having utility more than the minimum utility threshold are known as high utility itemset. Calculate $pft(p_i)$ values from Table 3 and $q(p_i, T_d)$ values from Table 4.

$$U(p_i,T_d) = pft(p_i) \times q(p_i,T_d)$$

$$U(\{p_1,T_1\}) = 1 \times 5 = 5$$

Table 2: Predefined timeslots

Timestamp(T)	Time intervals
T_1	00:01 05:00
T_2	05:01 10:00
T_3	10:01 15:00
T_4	15:01 20:00
T_5	20:01 00:00

Table 3: Profit table

Item	p_1	p_2	p_3	p_4	p_5	p_6	p_7
Unit profit	5	2	1	2	3	1	1

Utility of an itemset in a transaction T_d can be calculated from Eq. (2).

$$U(X,T_d) = \sum_{p_i \in X \wedge X \subseteq T_d} U(p_i,T_d)$$

$$U(\{p_1\ p_4\},T_1) = U(\mathrm{p}_1,T_1) + U(\mathrm{p}_4,T_1)$$
$$= 5 + 2 = 7$$

Utility of an itemset X in the database where $X = \{p_1p_4\}$.

$$U(\{p_1\ p_4\}) = U(\mathrm{p}_1\ \mathrm{p}_4,T_1) + U(\mathrm{p}_1\ \mathrm{p}_4,T_3)$$
$$= 7 + 17 = 24$$

The utility of an item in transaction depends on quantity and profit value. For example, the utility of item $\{p_1\}$ in the transaction T_1 is 5. The utility of an itemset (i.e. more than one item) X is a summation of a utility for all the items contained in X. For example, a utility of itemsets $\{p_1p_4\}$ in the transaction T_1 is 7, and utility of an itemset refers the total profits of the itemset in the database; for example, an itemset $\{p_1p_4\}$ appears in transactions T_1 and T_3. Utilities in T_1 and T_3 are 7 and 17, respectively; therefore, a utility of the itemsets $\{p_1p_4\}$ is 24. An itemset is called as a high utility itemset if and only if its utility is no less than a user-specified threshold otherwise that itemset is called a low utility itemset. For example, $\{p_2p_4\}$:30 are a high utility itemset but $\{p_2\}$ is a low utility itemset; our goal is to find all the high utility itemsets from a given transactional database under a user-specified minimum utility threshold [32].

Table 4: Transactional table

T_{id}	p_1	p_2	p_3	p_4	p_5	p_6	p_7	TU
T_1	1	0	1	1	0	0	0	8
T_2	2	0	6	0	2	0	5	27
T_3	1	2	1	6	1	5	0	30
T_4	0	4	3	3	1	0	0	20
T_5	0	2	2	0	1	0	2	11
Profit	5	2	1	2	3	1	1	
TWU	65	61	96	58	88	30	38	

4.1 High Utility Itemset

An itemset X is called high utility itemset if U $(X) > =$ MUT otherwise those itemsets are discarded. TU and TWU can be calculated using support and profit value of an item as shown in Table 4. From Eq. (6), for example, we have preset Minimum Utility Threshold (MUT) = 30.

There are following high utility itemsets, having its minimum utility 30 or more. The values of an itemsets are calculated using Eqs. (2) and (3), $\{p_2p_4\}$: **30**, $\{p_2p_3p_5\}$: **31**, $\{p_1p_3p_5\}$: **31**, $\{p_2p_3p_4\}$: **34**, $\{p_2p_4p_5\}$: **36**, $\{p_2p_3p_4p_5\}$: **40** and $\{p_1p_2p_3p_4p_5p_6\}$: **30**.

The main challenge in utility mining is that the downward closure property (superset of a low utility itemset may be a high utility itemset) [3] cannot be applied. For example, $\{p_2\}$ is a low utility itemset but its superset $\{p_2p_4\}$ is a high utility itemset because item utility of $\{p_2\}$ is 16 (low utility itemset because $16 \leq 30$) but the superset of $\{p_2\}$ itemset is $\{p_2p_4\}$ and utility of $\{p_2p_4\}$ is 30 (high utility itemset because $30 \geq 30$). Therefore, pruning search space is difficult in utility mining.

4.2 High Utility Weblog (HUWL)

In Table 5, Transactional Utility (TU) in every transaction has been calculated. We compute the transaction utility for each transaction

$$\mathrm{TU}(T_d) = U(T_d, T_d).$$

$$\text{i.e } \mathrm{TU}(T_1) = U(T_1, T_1) = (\{p_1p_3p_4\}, T_1) = 8.$$

Similarly we can get all transaction utility for all transactions. HUIM algorithm computes a utility for each transaction. For example, the utility of the transaction T_1 is 8. Then it finds all the items and their TWUs. The TWU of an item is the summation of the utilities of all the transactions containing the item. For example, the TWU of item $\{p_1\}$ is 65, since item $\{p_1\}$ appears in T_1, T_2, and T_3. Their transaction utilities are 8, 27, and 30, the summation is 65.

$$\mathrm{TWU}(X) = \sum_{x \subseteq T_d \in D} \mathrm{TU}(T_d)$$

Table 5: Transactional table and utility

T_{id}	Transactional path	TU	Items(pages)	TWU
T_1	$(p_3,1),(p_1,1),(p_4,1)$	8	p_1	65
T_2	$(p_3,6),(p_5,2),(p_1,2)$	27	p_2	61
T_3	$(p_3,1),(p5,1),(p1,1),(p_2,2),(p_4,6)$	30	p_3	96
T_4	$(p_3,3),(p_5,1),(p_2,4),(p_4,3)$	20	p_4	58
T_5	$(p_3,2),(p_5,1),(p_2,2)$	11	p_5	88

$$\text{i.e } \mathrm{TWU}(p_1) = U(T_1, T_1) + U(T_2, T_2) + U(T_3, T_3)$$
$$= 8 + 27 + 30 = 65$$

$$\text{and } \mathrm{TWU}(p_2) = U(T_3, T_3) + U(T_4, T_4) + U(T_5, T_5)$$
$$= 30 + 20 + 11 = 61$$

5. PROPOSED APPROACH

In this section, we introduce the proposed algorithm for mining maximal high utility itemsets from data streams. The flow of an approach is shown in Figure 2.

5.1 Algorithm:

High Utility Itemsets Mining (HUIM)

Terms using HUIM:

Transaction Utility TU, Transactional Weighted Utilization TWU, High Utility Itemset- Frequent Pattern HUI-FP, Frequent Pattern FP, Cosine Similarity Pattern CSPT.

Input:

Transaction Table Td, Database DB, Profit Table PT and Minimum Utility Threshold (MUT), MCSV: Minimum

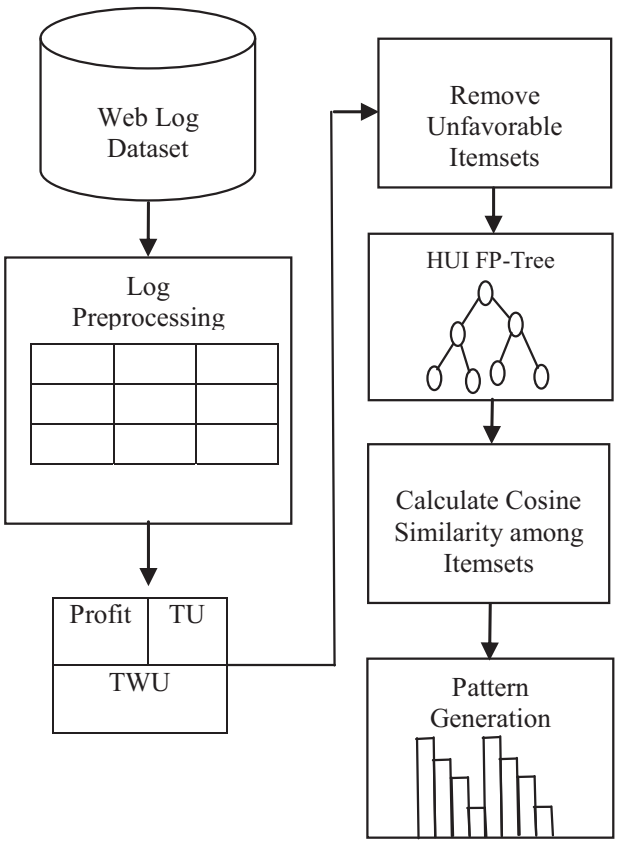

Figure 2: Flow of proposed approach.

Cosine Similarity Value, X: Suffix of a tree internally is φ *MSV: Minimum Support Value.*

Output:

All High Utility Itemset

1. Read *Td*, where *Td* ∈ DB.
2. Calculate TU for every item (p_i), where $p_i \in$ Td.
3. Calculate TWU of every itemset (*X*), where $X \subseteq T_d \in$ DB.
4. If TWU (*X*) ≥ MUT then those items are a favourable item (Fi), where Fi∈ pi.
5. Otherwise Unfavourable item (Ui) then Discarded Ui, where Ui∈ pi.
6. TWUs wise rearrange all itemset (*X*) and generate HUI FP-Tree.
7. For each item i_n from bottom top on the tree's head table and generate $Y = i_n \cup X$; *Y*: Suffix pattern for the tree.
8. Assign TWU for all i_n and also calculate cosine similarity Cos(*Y*).
9. If Cos(*Y*) > MCSV then CSPT = CSPT ∪ MCSV.
10. Create *Y*'s Conditional FP (YCFP) Tree with MSV.
11. If YCFP ≠ ø (Null) then
12. Repeat step 9 for different i_n.

Before creating a tree data structure, we have to make a table with individual items which is shown in Figure 3. Here we preset Minimum Cosine Similarity Value (MCSV) is 0.60 and Minimum Utility Threshold (MUT) is 30. For finding the strong pattern, following conditions should be satisfied. First, we calculate Minimum Support (MS) and TWU, it should be greater or equal to 30. Here, we discarded two items p_6 and p_7 because of minimum utility threshold. Only those items are considered as a favourable item, which value is more than preset minimum utility threshold otherwise called as a unfavourable items. We calculate cosine similarity for all the favourable items and generate strong patterns among them, as we shown in Figure 3 and Table 5, after removing the external unfavourable item and node utility then tree will be prune and utility is also decreased, as we shown in Figure 3.

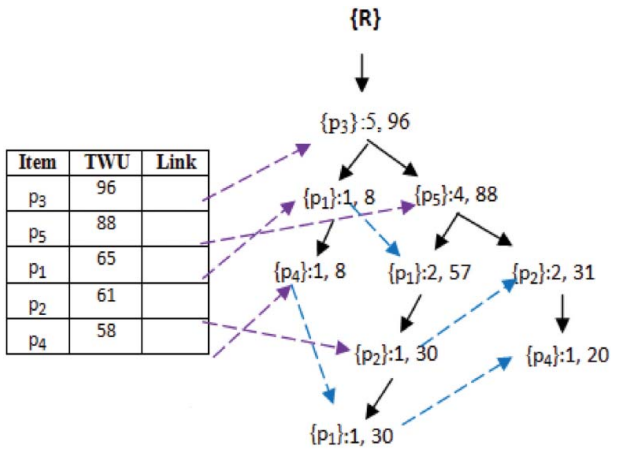

Figure 3: HUI-FP Tree when MUT = 30.

We have constructed a tree-based structure called High Utility Itemset-Frequent Pattern (HUI-FP). It is used to maintain the information about itemsets and their utilities. Each node of an HUI-FP tree consists of an item name, a TWU value, and a support count. The purpose of HUI-FP tree construction is to reduce search and space time because the candidate itemsets can be generated efficiently with only two scans of the database. FP-Tree generates all the patterns on the basis of support count [3,22]. Moreover, it is used only single parameter for generating patters but here, we have used two parameters such as MUT and TWU in HUI-FP tree and this generates only favourable itemset. We can generate entire navigation pattern from the root of HUI-FP tree. It also generates the common navigation patterns among all items or web pages. We can also extract the behaviour of users (like common interests of web users, highest number of frequent pages, most popular page, etc.) from weblog data.

Here we calculate the cosine similarity value and generate interesting pattern for all possible combinations of an itemset.

According to Eq. (7), all similar patterns can be calculated.

$$\text{Cos}(Y) = \frac{\sum_{n=1}^{K} \text{Supp}(Y) \times \text{Util}(Y)}{\sqrt{(\prod)^{K} \text{Supp}(Y) \times \text{Util}(Y)}}$$

$$i_n = p_4, X = \text{ø} Y = \{i_n\} \cup X = \{p_4\}$$

$$\text{Cos(Y)} = 1$$

1. $i_n = p_1, X = \{p_4\}, Y = \{i_n\} \cup X = \{p_2 p_4\}$

$$\text{Cos}(Y) = \frac{\sum_{n=1}^{2} \text{Supp}(Y) \times \text{Util}(Y)}{\sqrt{(3.14)^2 \times \text{Supp}(Y) \times \text{Util}(Y)}}$$

$$\text{Cos}(Y) = \frac{3 \times 65 + 3 \times 58}{\sqrt{(3.14)^2 \times (3 \times 3) \times (65 \times 58)}}$$

$$\text{Cos(Y)} = 0.639 (\text{pattern accepted})$$

2. $i_n = p_2, X = \{p_4\}, Y = \{i_n\} \cup X = \{p_2p_4\}$

$$\text{Cos } (Y) = \frac{3\times61+3\times58}{\sqrt{(3.14)^2\times(3\times3)\times(61\times58)}}$$

$\text{Cos}(Y) = 0.632(\text{pattern accepted})$

3. $i_n = p_3, X = \{p_4\}, Y = \{i_n\} \cup X = \{p_3p_4\}$

$$\text{Cos}(Y) = \frac{5\times96+3\times58}{\sqrt{(3.14)^2\times(5\times3)\times(96\times58)}}$$

$\text{Cos}(Y) = 0.722(\text{pattern accepted})$

4. $i_n = p_5, X = \{p_4\}, Y = \{i_n\} \cup X = \{p_5p_4\}$

$$\text{Cos}(Y) = \frac{4\times88+3\times58}{\sqrt{(3.14)^2\times(4\times3)\times(88\times58)}}$$

$\text{Cos}(Y) = 0.675(\text{pattern accepted})$

Similarly for three itemsets, the Cosine similarities are

5. $i_n = p_4, X = \{p_1p_2\}, Y = \{i_n\} \cup X = \{p_1p_2p_4\}$

$$\text{Cos}(Y) = \frac{3\times61+3\times65+3\times58}{\sqrt{(3.14)^3\times(3\times3\times3)\times(61\times65\times58)}}$$

$\text{Cos}(Y) = 0.124$ (pattern rejected)

Similarly we can calculate all possible patterns.

6. $i_n = p_4, X = \{p_1p_3\}, Y = \{i_n\} \cup X = \{p_1p_3p_4\}$

$\text{Cos}(Y) = 0.12(\text{pattern rejected})$

7. $i_n = p_4, X = \{p_1p_5\}, Y = \{i_n\} \cup X = \{p_1p_5p_4\}$

$\text{Cos}(Y) = 0.14(\text{pattern rejected})$

6. EXPERIMENTAL RESULTS

The performance of the proposed algorithms is evaluated in this section. The experiments were performed on a 2.20 GHz Intel i3 processor with 4 GB main memory. The operating system is Microsoft Windows 7. The algorithms are implemented in MATLAB. The experiment was performed on 10,000 transactions and this dataset is downloaded from NASA-HTTP [31]. It is having 10,000 web records, which consist IP address, User name, Timestamp, Access request, Result status code, Bytes transferred, Referrer URL, and User agent attributes, and preprocessed data is shown in Table 1. It is necessary preprocessing to obtain a dataset that suits our need in the desired form. In this experiment, we replace a token value for each URL of weblog dataset because integer computation is much easier to other data type. The experiments were conducted by

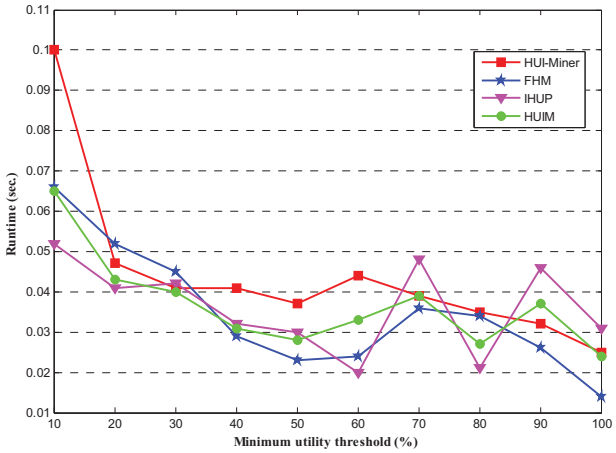

Figure 4: Run-time comparisons on different minimum utility threshold.

varying the minimum utility threshold from 10% to 100% on 10,000 web transactions. To show the performance of the proposed algorithms, we compared with HUI-Miner [26], IHUP [28], and FHM [29] algorithms. Figure 4 shows an execution time with different minimum utility threshold on 10,000 web transactions. In Figure 4, we demonstrate the run time may be increased when minimum utility threshold is decreased and it is depending on the number of itemset is being generated.

Figure 5 shows an execution time with different web transactions. In this figure, we fixed 30 as a minimum utility threshold value and found execution time for all existing algorithms. As we know, if we perform any operation on a large number of data then it will consume some time. The proposed approach takes less time for computation because it removes unfavourable

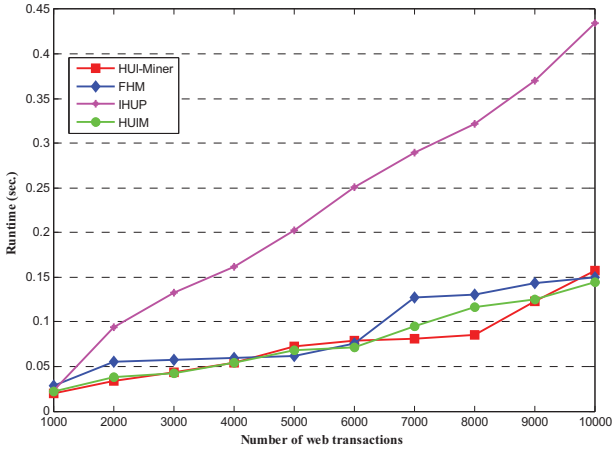

Figure 5: Run-time comparisons against number of web transactions.

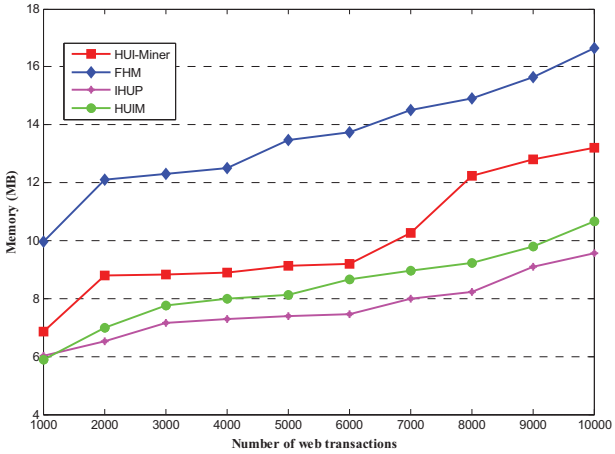

Figure 6: Comparison of memory utilization.

items based on utility first. Then it performs operation only on favourable items.

Our algorithms also outperform HUI-Miner, FHM, and IHUP in memory usage since they avoid handling a large number of data. Figure 6 shows memory consumption with different number of web transactions and we fixed 30 as a minimum utility threshold value. This figure clearly demonstrates that HUI-Miner [26], IHUP [28], and FHM [29] algorithm need 16.5, 11.5, and 7.5 MB on 10,000 web transaction dataset. HUIM algorithm needs 10.5 MB which is less amount of memory consumed compared to FHM and HUI-Miner but IHUP is quite efficient in terms of storage.

Figure 7 shows a number of high utility itemsets generated on different number of web transactions. For getting a different number of high utility, we had to fix 30 as a minimum utility threshold. Moreover, those

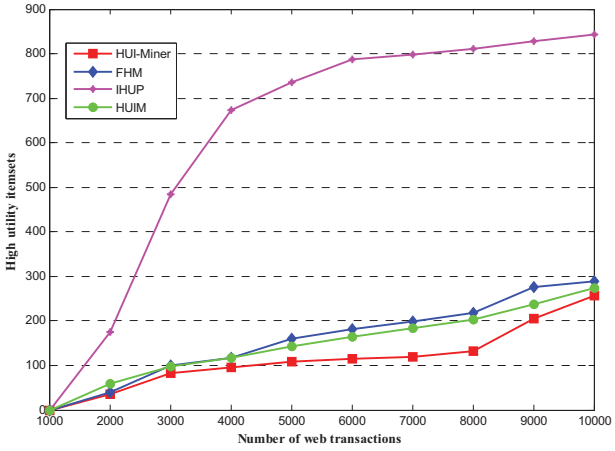

Figure 7: Number of high utility itemsets on different web transactions.

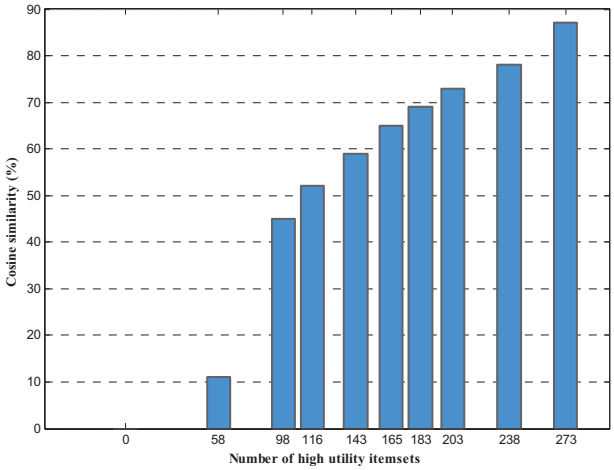

Figure 8: Cosine similarity among high utility itemsets.

itemsets having high utility value is 30 or more, than those itemsets are favourable itemsets otherwise unfavourable. We have to select only favourable itemset and find a cosine similarity value on it.

Cosine similarity is a measure of similarities among web transaction sequences. Figure 8 shows numbers of high utility itemsets and their cosine similarity of our HUIM algorithm. Here it considers the Minimum Cosine Similarity Value (MCSV) = 0.60 or 60%. Those itemsets are much similar to each other when MCSV is 60% or more.

7. CONCLUSIONS

In this paper, we have proposed an efficient algorithm named High Utility Itemset Mining (HUIM) for mining and generating high utility itemsets from transactional database. We have proposed one of the data structures named HUI-FP tree because it can be efficiently generated with two database scans. Moreover, we develop a pattern generation technique with cosine similarity value and enhance performance of utility mining. In the experiment, preprocessed weblog dataset was used to perform a thorough performance evaluation in terms of execution time, memory usage, number of high utility itemset generation, and cosine similarity among high utility itemsets. We are showing a result, according to the dataset, threshold value and high utility value to improve performance and reduce time and space complexity. The proposed algorithm is useful in e-commerce for different perspectives. It finds the web pages as itemsets that are useful for advertisement management, tourist guidance, and product value analysis and also useful for finding user behaviour similarity. The proposed algorithm may be

used by the business people for improving their business in e-commerce.

REFERENCES

1. B. Liu, *Web Data Mining,* Berlin, Heidelberg: Springer, 2007, pp. 449–79.
2. H. Xiong, P. Tan, and V. Kumar, "Hyperclique pattern discovery," *Data Mining Knowledge Discov*, Vol. 13, pp. 219–42, May. 2006.
3. R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," in *Proceeding 20th VLDB Conference,* San Francisco, 1994, pp. 487–99.
4. R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceeding of the ACM SIGMOD Conference,* New York, 1993, pp. 207–16.
5. J. Wang, J. Han, Y. Lu, and P. Tzvetkov, "Algorithm for mining Top-K frequent closed itemsets," *IEEE Trans. Knowledge Data Eng.*, Vol. 17, pp. 652–64, May 2005.
6. R. Karim, C. F. Ahmed, B.Jeong, and H.Choi, "An efficient distributed programming model for mining useful patterns in big datasets," *IETE Tech. Rev.*, Vol. 30, pp. 53–63, Feb. 2013.
7. C.H. Cai, A.W.C. Fu, C.H.Cheng, and W.W. Kwong, "Mining association rules with weighted items," in Proc*eedings of* IEEE Int*ernationa*l Database Eng*ineering* and Applications Symp*osium*, Cardiff, 1998, pp. 68–77.
8. S. Yen, Y. Lee, C. Wang, C. Wu, and L. Ouyang, "The studies of mining frequent patterns based on frequent pattern tree," in *Proceeding of 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining,* Heidelberg, 2009, pp. 232–41.
9. K. Sun, and F. Bai, "Mining weighted association rules without preassigned weights," *IEEE Trans. Knowledge Data Eng.*, Vol. 20, pp. 489–95, Apr. 2008.
10. U. Yun, and J. J. Leggett, "WIP: Mining weighted interesting patterns with a strong weight and/or support affinity," *Inf. Sci.*, Vol. 177, pp. 3477–99, Sept. 2007.
11. R. Chan, Q. Yang, and Y. Shen, "Mining high utility itemsets," in Proc*eedings* of *3rd* IEEE International Conference on Data Mining, Washington, 2003, pp. 19–26.
12. B. Bakariya, K. K. Mohbey, and G. S. Thakur, "An inclusive survey on data preprocessing method used in web usage mining," in *Proceedings of 7th International Conference on Bio-Inspired Computing: Theories and Application*, Gwalior, 2012, pp. 407–16.
13. X. Zhu, J. Guo, X. Cheng, Y. Lan, and W. Nejdl, "Recommending high utility query via session-flow graph," *Adv. Inf. Retrieval*, Vol. 7814, pp. 642–55, Mar. 2013.
14. S. Shankar, T. Purusothaman, S. Jayanthi, and N. Babu, "A fast algorithm for mining high utility itemsets," in *Proceeding of IEEE International Advance Computing Conference*, Patiala, 2009, pp. 1459–64.
15. V. Tseng, B. Shie, C. Wu, and P. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Trans. Knowledge Data Eng.*, Vol. 25, pp. 1772–86, Aug. 2013.
16. J. Pillai, and. O. P. Vyas, "Overview of itemset utility mining and its applications," *International Journal of Computer Applications*, Vol. 5, pp. 9–13, Aug. 2010.
17. S. Yen, and Y. Lee, "Mining high utility quantitative association rules," in *Proceeding of 9th International Conference. Data Warehousing and Knowledge Discovery*, Regensburg, 2007, pp. 283–92.
18. B. Mallick, D. Garg, and P. Grover, "Constraint-based sequential pattern mining: A pattern growth algorithm incorporating compactness, length and monetary," *Int. Arab J. Inf. Technol.*, Vol. 11, pp. 33–42, May. 2014.
19. D. Guan, and W. Yuan, "A survey of mislabeled training data detection techniques for pattern classification," *IETE Tech. Rev.*, Vol. 30, pp. 524–30, Dec. 2013.
20. V. Dhanasekar, and T. Angamuthu, "An efficient approach for effectual mining of relational patterns from multi-relational database," *Int. Arab J. Inf. Technol.*, Vol. 10, pp. 260–8, May 2013.
21. B. Bakariya, and G. S. Thakur, "Effectuation of web log preprocessing and page access frequency using web usage mining," *Int. J. Comput. Sci. Eng.*, Vol. 1, pp. 1–5, Sept. 2013.
22. J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," *Data Mining Knowledge Discov.*, Vol. 8, pp. 53–87, Jan. 2004.
23. A. Erwin, R. P. Gopalan, and N. R. Achuthan, "CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach," in *Proceeding of 7th IEEE International Conference on Computer and Information Technology*, Fukushima, 2007, pp. 71–6.
24. H. Xiong, S. Shekhar, P. Tan, and V. Kumar, "Exploiting a support based upper bound of Pearson's correlation coefficient for efficiently identifying strongly correlated pairs," in *Proceeding of 10th International Conference on Knowledge Discovery and Data Mining*, New York, 2004, pp. 334–43.
25. C. Ramaraju, and N. Savarimuthu, "A conditional tree based novel algorithm for high utility itemset mining," in *Proceeding of International Conference on Recent Trends in Information Technology*, Chennai, 2011. pp. 701–6.
26. M. Liu, and J. Qu, "Mining high utility itemsets without candidate generation," in *Proceeding of 21st ACM International Conference on Information and Knowledge Management*, New York, 2012, pp. 55–64.
27. C.F. Ahmed, S.K. Tanbeer, and J. Byeong-Soo, "Mining high utility web access sequences in dynamic web log data," in *Proceeding of 11th ACIS International Conference on Software Engineering Artificial Intelligence, Networking and Parallel/Distributed Computing*, London, 2010, pp. 76–81.
28. C. F Ahmed, S. K.Tanbeer, J. Byeong-Soo, and L. Young-Koo, "Efficient tree structures for high utility pattern mining in incremental databases," *IEEE Trans. Knowledge Data Eng.*, Vol. 21, pp. 1708–21, Dec. 2009.
29. P. Fournier-Viger, C. Wu, S. Zida, and V.S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning," *Foundations Intell. Syst.*, Vol. 8502, pp 83–92, Jun. 2014.
30. J. Cao, Z. Wu, and J. Wu, "Up cosine interesting pattern discovery: A depth first method," *Int. J. Inf. Sci. Elsevier*, Vol. 266, pp. 31–46, May 2014.
31. Weblog dataset downloaded from http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html Retrieved on March 12th, 2013.
32. K.K. Mohbey, and G.S. Thakur, "Interesting user behaviour prediction in mobile e-commerce environment *using constraints*," *IETE Tech. Rev.*, doi:10.1080/02564602.2014.968224.

Authors

Brijesh Bakariya received his graduation degree from Barkatullah University, Bhopal, MP in 2005, and his post-graduation degree in Computer Applications from Devi Ahilya Vishwavidyalaya, Indore, MP in the year 2009. He is currently pursuing a PhD degree in the Department of Computer Applications, Maulana Azad National Institute of Technology, Bhopal, MP. His research interests include web mining and clustering.

E-mail: brijesh_scs@yahoo.co.in

Ghanshyam Singh Thakur has received BSc degree from Dr. Harisingh Gour University, Sagar, MP in 2000. He has received MCA degree in 2003 from Pt. RaviShankar Shukla University, Raipur, CG and PhD degree from Barkhatullah University, Bhopal, MP in the year 2009. He is assistant professor in the Department of Computer Applications, Maulana Azad National Institute of Technology, Bhopal, MP, India. He has eight years of teaching and research experience. He has 26 published articles in national and international journals. His research interests include text mining, document clustering, information retrieval, and data warehousing. He is a member of the CSI, IAENG, and IACSIT.

E-mail: ghanshyamthakur@gmail.com

DOI: 10.1080/02564602.2014.1000396;