MINING THE CONCISE REPRESENTATIONS OF HIGH UTILITY ITEMSETS

***Mr.IMMANUEL.K, **Mr.E.MANOHAR, *** Dr. D.C. Joy Winnie Wise, M.E., Ph.D.**

** M.E.(CSE), Francis Xavier Engineering College, Tirunelveli, India*

*** Assistant Professor CSE department, Francis Xavier Engineering College, Tirunelveli, India*

****Head and Professor of CSE department, Francis Xavier Engineering College, Tirunelveli, India*

ABSTRACT

Mining high utility itemsets from databases is an emerging topic in data mining, which refers to the discovery of itemsets with utilities higher than a user-specified minimum utility threshold min_util. In the existing system, the compact and lossless representation of HUIs from closed high utility itemsets, three efficient algorithms are used AprioriHC (Apriori-based algorithm for mining High utility Closed itemsets), AprioriHC-D (AprioriHC algorithm with Discarding unpromising and isolated items) and CHUD (Closed High Utility Itemset Discovery) to find this representation. In this project two efficient algorithms are proposed to reduce competition and also high utilities. There are: (1) AprioriHC-MINEX (AprioriHC algorithm with MINEX) and (2) AprioriHC-NDI (AprioriHC algorithm with Non-Derivable Itemsets) to find Concise representation. In AprioriHC-MINEX algorithm explores the itemset lattice, starting from the empty set and stopping at the level of the largest frequent free-sets. At each iteration of this loop, it scans the database to find out which candidates of size i are frequent free-sets. Then, it generates candidates for the next iteration, taking every set of size i +1 such that all proper subsets are frequent free-sets. The algorithm finishes when there are no more candidates. In AprioriHC-NDI algorithm uses a level-wise algorithm to find all frequent NDIs. This algorithm use rules up to depth k if

I. INTRODUCTION

Knowledge discovery in databases (KDD) is the process of discovering useful knowledge from a collection of data. This widely used data mining technique is a process that includes data preparation and selection, data cleansing, incorporating prior knowledge on data sets and interpreting accurate solutions from the observed results. KDD includes multidisciplinary activities. This encompasses data storage and access, scaling algorithms to massive data sets and interpreting results. The data cleansing and data access process included in data warehousing facilitate the KDD process. Artificial intelligence also supports KDD by discovering empirical laws from experimentation and observations. The patterns recognized in the data must be valid on new data, and possess some degree of certainty. Data mining (the analysis step of the "Knowledge Discovery in Databases" process, or KDD), an interdisciplinary subfield of computer science, is the computational process of discovering patterns in large data sets ("big data") involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining

process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data preprocessing, model and interference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Data mining can unintentionally be misused, and can then produce results which appear to be significant; but which do not actually predict future behavior and cannot be reproduced on a new sample of data and bear little use. Often this results from investigating too many hypotheses and not performing proper statistical hypothesis testing. A simple version of this problem in machine learning is known as overfitting, but the same problem can arise at different phases of the process and thus a train/test split - when applicable at all - may not be sufficient to prevent this from happening. The final step of knowledge discovery from data is to verify that the patterns produced by the data mining algorithms occur in the wider data set. Not all patterns found by the data mining algorithms are necessarily valid. It is common for the data mining algorithms to find patterns in the training set which are not present in the general data set. This is called overfitting.

To overcome this, the evaluation uses a test set of data on which the data mining algorithm was not trained. The learned patterns are applied to this test set, and the resulting output is compared to the desired output. For example, a data mining algorithm trying to distinguish "spam" from "legitimate" emails would be trained on a training set of sample e-mails. Once trained, the learned patterns would be applied to the test set of e-mails on which it had not been trained. The accuracy of the patterns can then be measured from how many e-mails they correctly classify. A number of statistical methods may be used to evaluate the algorithm, such as ROC curves. If the learned patterns do not meet the desired standards, subsequently it is necessary to re-evaluate and change the pre-processing and data mining steps. If the learned patterns do meet the desired standards, then the final step is to interpret the learned patterns and turn them into knowledge. Intuitively, a set of items that appears in many baskets is said to be “frequent.” To be formal, we assume there is a number s, called the support threshold. If I is a set of items, the support for I is the number of baskets for which I is a subset. We say I is frequent if its support is s or more. It is a frequent itemset that is both closed and its support is greater than or equal to minsup. An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset. Frequent itemsets are the itemsets that occur frequently in the transaction data set. The goal of Frequent Itemset Mining is to identify all the frequent itemsets in a transaction dataset. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n distinct literals called items. An itemset is a non-empty set of items. An itemset $X = (i_1, i_2, \ldots, i_k)$ with k items is referred to as k-itemset, A transaction T = consists of a transaction identifier (TID) and a set of items $(i_1, i_2, \ldots, i_k)$, where ij Î I, j = 1, 2, …, k. The frequency of an itemset X is the probability of X occurring in a transaction T. A frequent itemset is the itemset having frequency support greater a minimum user specified threshold. In this paper high utility itemset mining is presented. For mining high utility itemset in this project two efficient algorithms are proposed to reduce competition and also high utilities. There are: (1) AprioriHC-MINEX (AprioriHC algorithm with MINEX) and (2) AprioriHC-NDI (AprioriHC algorithm with Non-Derivable Itemsets) to find Concise representation. In AprioriHC-MINEX algorithm explores the itemset lattice, starting from the empty set and stopping at

the level of the largest frequent free-sets. At each iteration of this loop, it scans the database to find out which candidates of size i are frequent free-sets. Then, it generates candidates for the next iteration, taking every set of size i +1 such that all proper subsets are frequent free-sets. The algorithm finishes when there are no more candidates. In AprioriHC-NDI algorithm uses a level-wise algorithm to find all frequent NDIs. This algorithm use rules up to depth k if it only evaluate the rules RJ(I) for $|I - J| \leq k$.

II. RELATED WORK

Alva Erwin et al [1] have proposed high utility itemsets mining that extends frequent pattern mining to discover itemsets in a transaction database with utility values above a given threshold. However, mining high utility itemsets presents a greater challenge than frequent itemset mining, since high utility itemsets lack the anti-monotone property of frequent itemsets. Transaction Weighted Utility (TWU) proposed recently by researchers has anti-monotone property, but it is an overestimate of itemset utility and therefore leads to a larger search space. Bai-En Shie et al [2] has mining high utility mobile sequential patterns by integrating mobile data mining with utility mining. Two tree-based methods are proposed for mining high utility mobile sequential patterns. Cheng Wei Wu et al. [3] has mining closed+ high utility itemsets, which serves as a compact and lossless representation of high utility itemsets. We present an efficient algorithm called CHUD (Closed+ High Utility itemset Discovery) for mining closed+ high utility itemsets. Further, a method called DAHU (Derive All High Utility itemsets) is proposed to recover all high utility itemsets from the set of closed+ high utility itemsets without accessing the original database. Ching-Huang Yun [4] have proposed a new data mining capability for a mobile commerce environment. To better reflect the customer usage patterns in the mobile commerce environment, we propose an innovative mining model, called mining mobile sequential patterns, which takes both the moving patterns and purchase patterns of customers into consideration. Claudio Lucchese [5] have proposed a new scalable algorithm for discovering closed frequent itemsets, a lossless and condensed representation of all the frequent itemsets that can be mined from a transactional database. This algorithm exploits a divide-and-conquer approach and a bitwise vertical representation of the database and adopts a particular visit and partitioning strategy of the search space based on an original theoretical framework, which formalizes the problem of closed itemsets mining in detail.

III. GENERAL DEFINITIONS

- **Itemset:** Set of items that occur together
- **Association Rule:** Probability that particular items are purchased together.

X ® Y where X ÇY = 0

- **Support**, supp(X) of an itemset X is the ratio of transactions in which an itemset appears to the total number of transactions. The support value of X with respect to T is defined as the

proportion of transactions in the database which contains the item-set X. In the example database, the item-set {milk, bread, butter} has a support of 1/5=0.2 since it occurs in 20% of all transactions (1 out of 5 transactions). The argument of supp() is a set of preconditions, and thus becomes more restrictive as it grows (instead of more inclusive).

- **Share** of an itemset is the ratio of the count of items purchased together to the total count of items purchased in all transactions.
- **Confidence** of rule X ® Y, denoted conf(X ® Y). The confidence value of a rule, X $\Rightarrow Y$, with respect to a set of transactions T, is the proportion the transactions that contains X which also contains Y. Confidence is defined as:

$$\text{conf}(X \Rightarrow Y) = \text{supp}(X \cup Y)/\text{supp}(X)$$

For example, the rule {butter, bread} ⇨{milk} has a confidence of 0.2/0.2 =1 in the database, which means that for 100% of the transactions containing butter and bread the rule is correct (100% of the times a customer buys butter and bread, milk is bought as well). Note that supp(X $\cup Y$) means the support of the union of the items in X and Y. This is somewhat confusing since we normally think in terms of probabilities of events and not sets of items. We can rewrite supp(X $\cup Y$) as the joint probability P($E_X \cap E_Y$), where E_X and E_Y are the events that a transaction contains itemset X or Y, respectively.

- **Transaction Database** stores transaction data. Transaction data may also be stored in some other form than a m x n database.
- **High Utility Itemset Mining**
  Let I={i_1,i_2,....,i_M} has a set of items, and D={T_1,T_2,.....,T_N} has a set of transactions database. Each transaction T_R has belongs to transaction database D and number of itemsets I to purchase items i because all items are stored in database D, where Quantity of itemsets is represented as QT(I). To identify each transaction unique identifier is used. The internal utility is defined as the items i can count in the transactions database D for each transactions T_R. The external utility is defined as the utility values are added in the database D for every transactions. High utility itemset is defined as set of items i no less than user specified minimum threshold; other than high utility itemset is called as low utility itemset. The length of the itemset is denoted as M number of item in a set. i.e. $1 \leq I \leq M$.

IV. PROPOSED SYSTEM

The traditional ARM approaches consider the utility of the items by its presence in the transaction set. The frequency of itemset is not sufficient to reflect the actual utility of an itemset. For example, the sales manager may not be interested in frequent itemsets that do not generate significant

profit. Recently, one of the most challenging data mining tasks is the mining of high utility itemsets efficiently. Identification of the itemsets with high utilities is called as Utility Mining. The utility can be measured in terms of cost, profit or other expressions of user preferences. For example, a computer system may be more profitable than a telephone in terms of profit. Utility mining model was proposed in [19] to define the utility of itemset. The utility is a measure of how useful or profitable an itemset X is. The utility of an itemset X, i.e., u(X), is the sum of the utilities of itemset X in all the transactions containing X. An itemset X is called a high utility itemset if and only if u(X) >= min_utility, where min_utility is a user defined minimum utility threshold.

Fig 1 System Architecture

The main objective of high-utility itemset mining is to find all those itemsets having utility greater or equal to user-defined minimum utility threshold. Several data mining tasks are based on the evaluation of frequency queries to determine how often a particular pattern occurs in a large data set. This project considers the problem of frequency query evaluation, when patterns are itemsets or conjunctions of properties, in dense data sets1 like, for instance in the context of census data analysis or log analysis. In these important but difficult cases, there is a combinatorial explosion of the number of frequent itemsets and computing the frequency of all of them turns out to be intractable. This project presents an efficient technique to approximate closely the result of the frequency queries, and formalize it within the ϵ -adequate representation framework a new ϵ-adequate representation for the frequency queries. This representation, called free-sets, is more condensed than the ϵ -adequate representation based on itemsets. In the representation proposed in this paper, we call free-set an itemset Y such that the items in Y cannot be used to form a nearly exact rule. Fig 1 shows the system architecture of high utility mining. There are two modules presents in high utility mining. One is admin module and other one is customer module. The admin module consists of adding the items, viewing the items, detect de-duplication, creating transactional database for all available items, calculating the freesets using AprioriHC-MINEX algorithm and calculating the Non derivable sets using AprioriHC-NDI algorithm. The customer module consists the viewing the items and buying the items

A. Freesets

Let r be a binary database over R, X $\subseteq$ R is a $-free-set w.r.t. r if and only if there is no δ -strong rule based on X in r. The set of all $-free-sets w.r.t. r is noted Free(r, δ). Since δ is supposed to be rather small, informally, a free-set is a set of items such that its subsets (seen as conjunction of properties) are not related by any very strong positive correlation. One of the most interesting properties of freeness is its anti-monotonicity w.r.t. itemset inclusion. A property ρ is anti-monotone if and only if for all itemsets X and Y, ρ (X) and Y $\subseteq$ X implies ρ (Y). The anti-monotonicity has been identified as a key property for efficient pattern mining, since it is the formal basis of a safe pruning criterion. Indeed, efficient frequent set mining algorithms like apriori make use of the (anti-monotone) property “is frequent” for pruning. The anti-monotonicity of freeness follows directly from the definition of free-set and is stated by the following theorem. Let X be an itemset. For all Y $\subseteq$ X if X $\in$ Free(r, δ) then Y $\in$ Free(r, δ).

Discovering all frequent free-sets

This section describes an algorithm, called AprioriHC-MINEX that generates all frequent free-sets. For clarity, this method omits the fact that it outputs their supports as well. AprioriHC-MINEX can be seen as an instance of the levelwise search algorithm presented in Mannila and Toivonen. It explores the itemset lattice (w.r.t. set inclusion) levelwise, starting from the empty set and stopping at the level of the largest frequent free-sets. More precisely, the collection of candidates is initialized with the empty set as single member (the only set of size 0) and then the algorithm

iterates on candidate evaluation and larger candidate generation. At each iteration of this loop, it scans the database to find out which candidates of size *i* are frequent free-sets. Then, it generates candidates for the next iteration, taking every set of size *i* + 1 such that all proper subsets are frequent free-sets. The algorithm finishes when there is no more candidate. The algorithm is given below as Algorithm 3.1.

Algorithm (AprioriHC-MINEX)
Input: r a binary database over a set of items R, σ is the threshold.

Output: FreqFree(r, σ)

1. C_0 := {Φ};
2. i := 0;
3. **while** C_i .= Φ **do**
4. FreqFreei := {X | X Є C_i and X is a " σ- frequent set in r };
5. C_{i+1} := {X | X ⊆ R and ∀ Y ⊆ X, Y Є $\bigcup_{j\leq i} Freqfree_j$}\$\bigcup_{j\leq i} C_j$;
6. i := i + 1;
7. **output** $\bigcup_{j<i} Freqfree_j$;

B. Non derivable Itemsets

The main goal of this algorithm is to present several new methods to identify redundancies in the set of all frequent itemsets and to exploit these redundancies, resulting in a concise representation of all frequent itemsets and significant performance improvements of a mining operation. Based on the deduction rules, it is possible to generate a summary of the set of frequent itemsets. Indeed, suppose that the deduction rules allow for deducing the support of a frequent itemset I exactly, based on the supports of its subsets. Then there is no need to explicitly count the support of I requiring a complete database scan; if we need the support of I, we can always simply derive it using the deduction rules. Such a set I, of which we can perfectly derive the support, will be called a Derivable Itemset (DI), all other itemsets are called Non-Derivable Itemsets (NDIs). We will show in this section that the set of frequent NDIs allows for computing the supports of all other frequent itemsets, and as such, forms a concise representation of the frequent itemsets. To prove this result, we first need to show that when a set I is non-derivable, then also all its subsets are non-derivable. For each set I, let l_I (u_I) denote the lower (upper) bound we can derive using the deduction rules. The AprioriHC-NDI Algorithm Based on the results in the previous section, we propose a level-wise algorithm to find all frequent NDIs. Since derivability is monotone, we can prune an itemset if it is derivable. This gives the AprioriHC-NDI algorithm as shown below. The correctness of the algorithm follows from the results.

Algorithm

AprioriHC-NDI(D,s)
i := 1; NDI := {}; C1 := {{i} | i Є I};
for all I in C_1 do I.l := 0; I.u := |D|;

while C_i not empty do
Count the supports of all candidates in C_i in one pass over D;
$F_i := \{I \in C_i \mid support(I,D) \geq s\};$.
$NDI := NDI \cup F_i;$
Gen := {};
for all $I \in F_i$ do
if $support(I) \neq I:l$ and $support(I) \neq I:u$ then
$Gen := Gen \cup \{I\};$
$PreC_{i+1} := AprioriGenerate(Gen);$
$C_{i+1} := \{\};$
for all $J \in PreC_{i+1}$ do
Compute bounds [l, u] on support of J;
if $l \neq u$ then $J,l := l; J.u := u; C_{i+1} := C_{i+1} \cup \{J\};$
i := i + 1
end while
return NDI

Since evaluating all rules can be very cumbersome, in the experiments we show what the effect is of only using a couple of rules. We will say that we use rules up to depth k if we only evaluate the rules R_J (I) for $|I - J| \leq k$. The experiments show that in most cases, the gain of evaluating rules up to depth k instead of up to depth k - 1 typically quickly decreases if k increases. Therefore, we can conclude that in practice most pruning is done by the rules of limited depth.

V. SCREENSHOTS

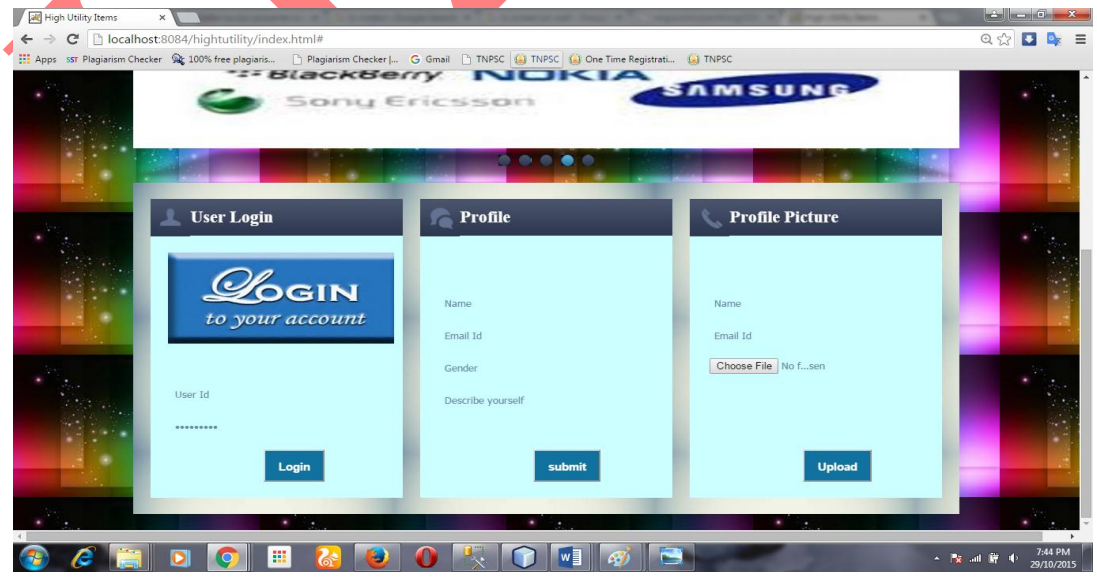

Fig 2.Home Page

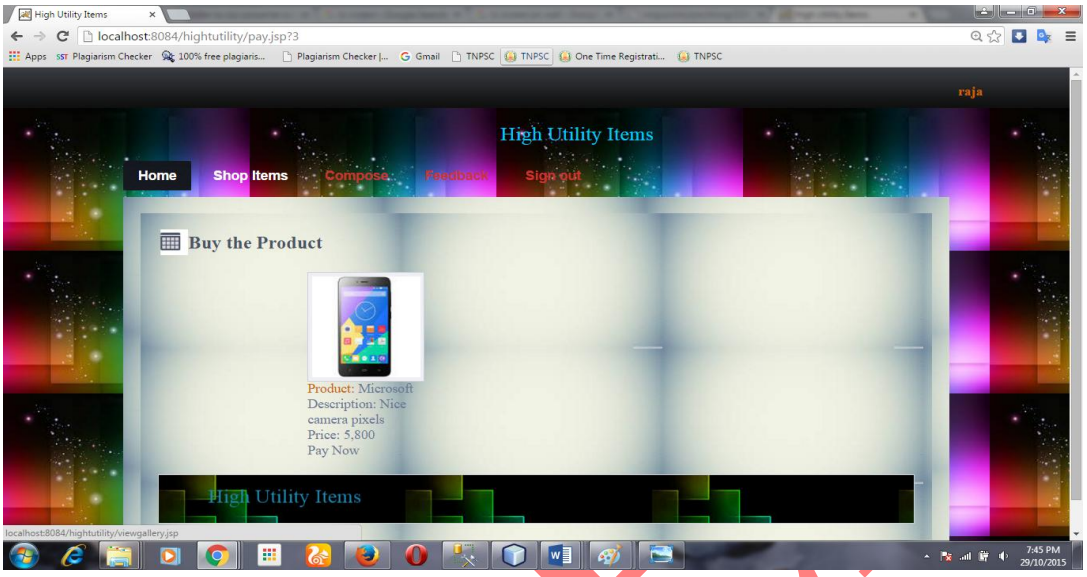

Fig 3.Customer Page

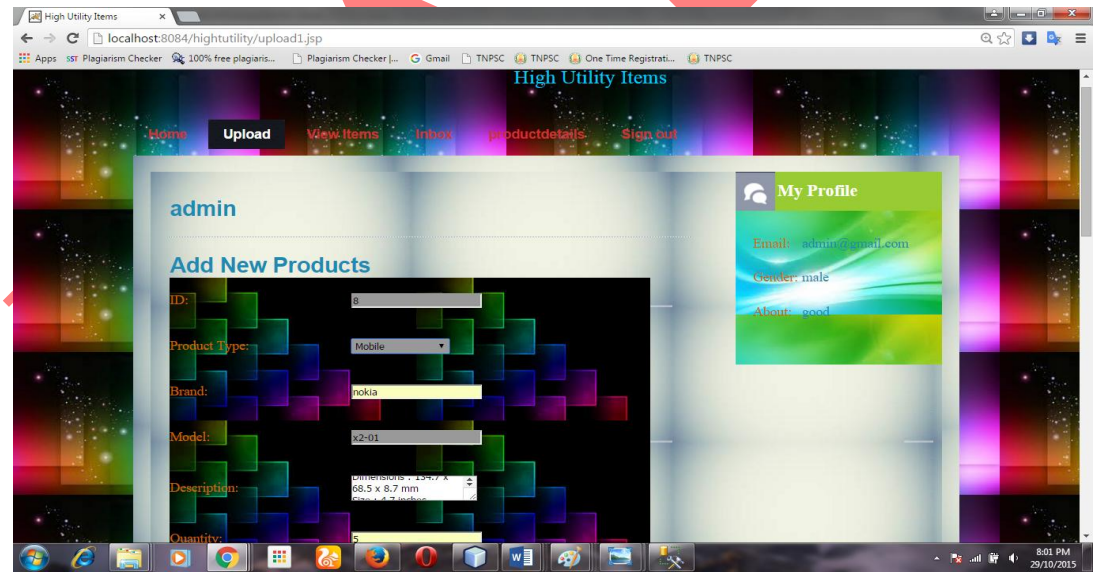

Fig 4.Admin Page

Fig 5.AprioriHC-MINEX

Fig 6.AprioriHC-NDI

VI. CONCLUSION

In this project for mining high utility itemsets two algorithms are presented namely AprioriHC-MINEX (AprioriHC algorithm with MINEX) and AprioriHC-NDI (AprioriHC algorithm with Non-Derivable Itemsets). These two algorithms are uses AprioriHC which perform a breadth-first search for calculating high utility itemset mining. The results shows that these two algorithms efficiently calculating the high utility itemsets.

VI. REFERENCES

[1] Alva Erwin, Raj P. Gopalan, and N.R. Achuthan (2011), "Efficient Mining of High Utility Itemsets from Large Datasets", Springer-Verlag Berlin Heidelberg

[2] Bai-En Shie, Hui-Fang Hsiao, Vincent S. Tseng, and Philip S. Yu (2008) "Mining High Utility Mobile Sequential Patterns in Mobile Commerce Environments", ACM transaction systems

[3] Cheng Wei Wu, Philippe Fournier-Viger, Philip S. Yu, Vincent S. Tseng (2011) "Efficient Mining of a Concise and Lossless Representation of High Utility Itemsets", 11th IEEE International Conference on Data Mining

[4] Ching-Huang Yun and Ming-Syan Chen, (2007) Mining Mobile Sequential Patterns in a Mobile Commerce Environment IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, VOL. 37, NO. 2.

[5] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego (2006), "Fast and Memory Efficient Mining of Frequent Closed Itemsets", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 18, NO. 1

[6] Guo-Cheng, Lan Tzung-Pei Hong, Vincent S. Tseng (2011), "Projection-Based Utility Mining with an Efficient Indexing Mechanism", International conferences

[7] T. Hamrouni a,b, S. Ben Yahia a,*, E. Mephu Nguifo (2010) "Sweeping the disjunctive search space towards mining new exact 3 concise representations of frequent itemsets" ARTICLE in DATA & KNOWLEDGE ENGINEERING

[8] Jean-Fran_cois Boulicaut, Artur Bykowski, and Christophe Rigotti (2010) Approximation of Frequency Queries by Means of Free-Sets Springer-Verlag Berlin Heidelberg

[9] Kun-Ta Chuang, Jiun-Long Huang† and Ming-Syan Chen (2011), Mining Top-k Frequent Patterns in the Presence of the Memory Constraint IEEE TRANSACTIONS ON SYSTEM

[10] Ying Liu, Wei-keng Liao, Alok Choudhary (2009), "A Fast High Utility Itemsets Mining Algorithm", ACM transaction