Interactive Representation of Closed+ High Utility Itemsets using Hadoop

Shaikh Hadiya Mohammad Ejaz[1] Asst. Prof. P. N. Kathavate[2]
[1]PG Student [2]Assistant Professor
[1,2]Department of Computer Science & Engineering
[1,2]WIT, Solapur University, Solapur India

Abstract— In the past few years a large numbers of algorithms have been design to mining a high utility itemsets from a database. The problem of all those algorithms is to produce a large numbers of high utility itemsets which downgrades the performance of mining process. To reach the high performance for mining the task we design a novel framework in this paper for provides the mining Closed+ high utility mining itemsets (CHUI), which gives the meaningful representation of HUIs. The already existing designed algorithm named CHUD (Closed+ High Utility itemsets Discovery) to find concise representation is further enhanced by implementing the CHUD based on pattern generated from FP algorithm in Hadoop, so system with less data set which required more memory can be analyzed in low memory based system with the help of distributed file system. A DAHU (Drive All High Utility Itemstes) method is introduce to retrieve all HUIs from set of CHUIs without accessing original databases.

Key words: CHUD, CHUIs, DAHU

I. INTRODUCTION

Frequent itemset mining (abbreviated as FIM)[2],[3], is a Major research topic in data mining. One of its famous applications is market basket analysis, which refers to the discovery of sets of items (itemsets) that are frequently picked up or purchased together by customers. However, in this application, the traditional model of FIM may invent a large amount of frequent or constant itemsets with low profit and drop the information on valuable itemsets having low selling frequencies. There are two problem in FIM such as (1) FIM treats all items as having the same importance/unit profit/weight and (2) it consider that every item in a transaction appears in a binary form, i.e., an item can be either present or absent in a transaction, which doesn't express its purchase quantity in the transaction. Hence, FIM cannot satisfy the requirement of users who need to discover itemsets with high utilities such as high profits. The utility of an itemset show its importance, which can be measured in term of profit, cost, quantity or other information depending on the user choice. An itemset is called a high utility itemset (abbreviated as HUI) if its utility is no less than a user specified minimum utility threshold. Utility mining has a large field of applications such as website click stream analysis cross-marketing analysis and biomedical domains. However, HUIs mining is not an easy task since the downward closure property [1, 2, and 10] in FIM does not hold in utility mining. The search space cannot be directly pruned or cut backed to find HUIs as in FIM since a superset of a low utility itemset can be a high utility itemset. Many studies [4, 5, 6, 7, and 8] were suggested for mining HUIs, but they often present a large number of high utility itemsets to users such that awareness of the results becomes difficult. Meantime, the algorithms become inefficient in terms of time and memory need. In particular, the work of the mining task decreases greatly under low minimum utility thresholds or dense databases.

To diminish the computational cost in FIM while presenting less and more important patterns to users, many studies advanced the concise representations, such as free sets [9], non-derivable sets [10], maximal itemsets [28] and closed itemsets [11, 12-13 14, 15].These representations successfully reduce the set of itemsets form, but they were developed for frequent itemset mining rather than the high utility itemset mining. Therefore, a big research question is "Is it possible to accept a tight and lossless representation of high utility itemsets encouraged by these representations to address the preceding problems in HUI mining?"

Responding this question positively tough. Developing a concise and complete representation of HUIs poses several challenges:

1) Combining the concepts of concise or terse representations from FIM into HUI mining, which may not useful to the users.
2) The representation may not provide a powerful reduction in terms of the number of extracted patterns to explain using this representation.
3) Algorithms for obtaining the representation may not be efficient. Algorithms may be slower than the perfect algorithms for mining all HUIs.
4) It may be tough to design and develop an efficient method for regaining all HUIs from the representation.

"In this paper, we intention all of these challenges by introducing a concise and useful representation of HUIs named Closed+ High Utility Itemsets (Closed+ HUIs), which combine the concept of closed itemset into HUI mining. The improvements are four-fold in correspondence to answering the four challenges already mentioned:

1) The introduce representation is lossless by using a new structure or framework named utility unit array that allows regaing all HUIs and their utilities efficiently.
2) The proposed representation is also tight or compact.
3) We recommend an efficient or adequate algorithm, named CHUD (Closed+ High Utility itemset Discovery) to find concise representation based on FP-Growth algorithm in Hadoop framework that obtain CHUIs.So system with less memory or data set which required more memory can be analyzed in low memory based system with the help of distributed file system.
4) We introduce a top-down mechanism named DAHU (Derive All High Utility itemsets) for efficiently restoring all HUIs from the set of Closed+ HUIs. The combination of CHUD and DAHU gives a new way to obtain all HUIs and it exceed UP Growth [16], the state-of the- art algorithm for mining HUIs.

II. LITERATURE REVIEW

In the previous research, different efficient algorithms used for mining a closed high utility itemsets mining.

C.-W Wu, P. Fournier-Viger, P. S. Yu. and V. S. Tseng, [17] this paper provides mining closed+ high utility itemsets, which handle as a concise and lossless representation of high utility itemsets. They present an algorithm named CHUD (Closed+ High Utility itemset Discovery) for mining closed+ high utility itemsets. A method named DAHU (Derive All High Utility itemsets) is construct to repair complete high utility itemsets from the set of closed+ high utility itemsets without achieving the original database. Further increase the performance of CHUD algorithm they includes three effective and powerful strategies named REG, RML and DCM.

B.-E. Shie, V. S. Tseng, and P. S. Yu, [18] this paper, they proposed a novel algorithm, namely GUIDE, for efficiently mining temporal maximal utility itemsets from the landmark time to the present in data streams. They also proposed a new data structure, namely TMUI-tree, for storing information in the processes of mining utility patterns from data streams. The main contributions of GUIDE and TMUI-tree are that GUIDE is the first one-pass algorithm for mining maximal utility itemsets in data streams and TMUI-tree is easy to maintain and it can help GUIDE finding TMUIs efficiently.

V.S. Tseng et al,[19] in this paper focuses on temporal high utility itemset mining(THUI).Which discovered the temporal high utility itemsets with less candidate itemsets and higher performance. THUI-Mine employs a filtering threshold in each partition to generate a progressive set of itemsets. They are two problem with THUI-mine algorithm first it require huge memory and second it's generate a lot of false candidate itemsets.

C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong and Y.-K. Lee, [20] authors propose three novel tree structures to efficiently to perform interactive and incremental HUP mining.1) tree structure provides Incremental HUP Lexicographic Tree (IHUPL-Tree), is arranged according to an item's lexicographic order. The incremental data can be capture without any reconstructing operation. 2)tree structure is the IHUP Transaction Frequency Tree (IHUPTF-Tree), which give a solid size by arranging items according to their transaction frequency (descending order).3)tree,IHUP-Transaction-Weighted Utilization Tree (IHUPTWU-Tree) is designed based on the TWU value of items in descending order to diminish the mining time.

B. Vo, H. Nguyen, T. B. Ho, and B. Le,[21] they designed the parallel method for mining HUIs from vertically partitioned distributed databases, and the efficient algorithm is also proposed. The mining algorithm in distributed databases is more efficient than that in centralized database. The algorithm scans only local databases once and only item that its two satisfies minutil must be sent to MasterSite by using WIT-tree technique. Therefore, it spends a limited time for communication between MasterSite and SlaverSites.

C.-W. Wu, B.-E. Shie, V. S. Tseng and P. S. Yu, [22] they address problem by proposing a new framework named top-k high utility itemset mining, where k is represent the number of high utility itemsets to be mined. They designed an efficient algorithm named TKU (Top-K Utility itemsets mining) for mining such itemsets without setting min_util.

III. METHODOLOGY

A. Association Rules Mining

It is intended to identify strong rules discovered in databases using two different measures of interestingness. The first one is support which generates frequent item set from the provided database and the other one is confidence which is focuses on rule generation.

- Frequent item sets- A set of attributes is termed as frequent item set if the occurrence of the set within the database is more than a user given threshold.
- Support- Support determines how often a given rule is applicable to a given data set.
- Confidence- Confidence determines how frequently items in Y appear in transactions that contain X.

We calculate support and confidence by using following formulas:

$$\text{Support, } s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

$$\text{Confidence, } c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Where, X and Y disjoint item set.

B. FP Growth Algorithm Description in Hadoop

To break the two drawbacks [23] of Apriori algorithm FP-growth algorithm is used. FP-growth requires constructing FP-tree. For that, it requires two passes. FPgrowth uses divide and conquer strategy. It requires two scans on the database. It first computes a list of frequent items sorted by frequency in descending order (F-List) and during its first database scan. In the second search, the database is compressed into a FP-tree [24]. This algorithm performs mining on FP-tree recursively. There is a problem of finding frequent itemsets which is converted to searching and constructing trees recursively. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. There are two sub processes of frequent patterns generation process which includes: construction of the FP-tree, and generation of the frequent patterns from the FP-tree.

C. FP Growth algorithm steps

1) Step1: Equally split the transaction database into several sub-transaction databases and then assign them to different nodes in Hadoop cluster. This step is naturally operated by HDFS, when necessary we can use the balance command enabling its file system to achieve load balancing.
2) Step2: Calculate support count of each item in the transaction database by MapReduce, and then collect the set of I_list from support count in descending order.[25]
3) Step3: Divide I_list into M groups, denoted as Group_list (abbreviated as G_list), and assign group_id for each group sequentially and each G_list contains a set of items.[2]
4) Step4: Complete the parallel computing of FP-Growth algorithm by MapReduce. The Map function compares the item of each transaction in the sub-transaction database with the item in G_list. If they are same, then distribute the corresponding transaction to the machine associated with G_list. Otherwise, compares with the next item in G_list. Finally, the separate sub-transaction databases corresponded to G list will be produced. The

Reduce function recursively computes the independent sub-transaction databases generated in step, and then constructs the FP-tree. This step is similar to the process of traditional FP-tree generation, but the difference is a size K max-heap HP which stores frequent pattern of each item.

5) Step5: Aggregate the local frequent itemsets generated from each node in the cluster by MapReduce, and finally get the global frequent itemsets.

After obtain the frequent itemsets .Check the utility of itemsets. Itemsets that have low utility is discarded.

D. CHUD (Closed High Utility Itemsets from Discovery) Algorithm and DAHU method

An algorithm called CHUD and DAHU method Based on the concept of closed pattern. CHUD handling mining Closed+ High Utility Itemsets based on pattern generated from FP algorithm in Hadoop.

1) CHUD Algorithm

– Input D: the database, abs_min_utility, Support Count
– Output: Complete sets of CHUIs
  1) InitialDatabaseScan(D)
  2) FP Growth algorithm (D, Support Count) in hadoop.

DAHU method recovers all high utility itemsets from the Closed+ High Utility Itemsets.

2) DAHU Algorithm

– Input: ML: the maximum length of itemset in HC; abs_min_utility;
– HC = {HC1, HC2… HCML}: the complete set of CHUIs;
– Output: H: the complete set of HUIs

1) HML := HCML
2) for (k: = ML – 1; k > 0; k – –) do
3) {for each k-itemset X = {a1, a2… ak}∈HC_k do
4) { if (au(X) < abs_min_utility) then delete X from HCk
5) else add X and its absolute utility au(X) to H.
6) { for each item ai∈X do
7) { Y := X – {ai}
8) au(Y):= au(X) – V(X, ai)
9) if (au(Y) ≥ abs_min_utility) then
10) {if Y∈HCk-1 and SC(X) > SC(Y) then
11) {SC(Y):= SC(X) }
12) else if Y HCk-1 then
13) { HCk-1 := HCk-1 ∪Y
14) SC(Y):= SC(X) }}}}}}

Although it can perform mining high utility itemsets faster than existing CHUD algorithm (Wu and Yu, 2011) based on the concept of closed pattern were proposed.

The pseudo code of DAHU algorithm show above. It takes as input an absolute minimum utility threshold abs_min_utility, a set of CHUIs HC and ML the maximum length of itemsets in HC. DAHU outputs the complete set of high utility itemsets H =$U^k_{i=1}$ H_k respecting abs_min_utility, where H_k denotes the set of HUIs of length k. To derive all HUIs, DAHU proceeds as follows. First, the set H_{ML} is initialized to HC_{ML}, where the notation HC_k represents the set of k-itemsets in HC. During Line 2 to Line 14 in Fig. 14, each set Hk is constructed from k = (ML – 1) to k = 1. In each iteration, H_{k-1} is recovered by using HCk. For each itemset X = {a1, a2,…, ak} in HCk, if the absolute utility of X is no less than abs_min_utility, the algorithm outputs the high utility itemset X with its abso-lute utility and then generates all (k–1)-subsets of X. The latter are obtained by removing each item ai∈X from X one at a time to obtain subsets of the form Y = X – {ai}. If Y is not present in H_k or Y is present in H_k with SC(X) > SC(Y), Y is added to H_{k-1}, its support count is set to the support count of X (Property 4), i.e., SC(Y) = SC(X), and the absolute utility of Y is set to the absolute utility of X minus the i-th value in V(X), i.e., au(Y) = au(X) – V(X, ai) (Property 6-8). Besides, the utility unit array of V(Y) is set to V(X) with the value V(X, ai) removed (Property 8). This process is repeated until H has been completely recovered.

E. System Architecture

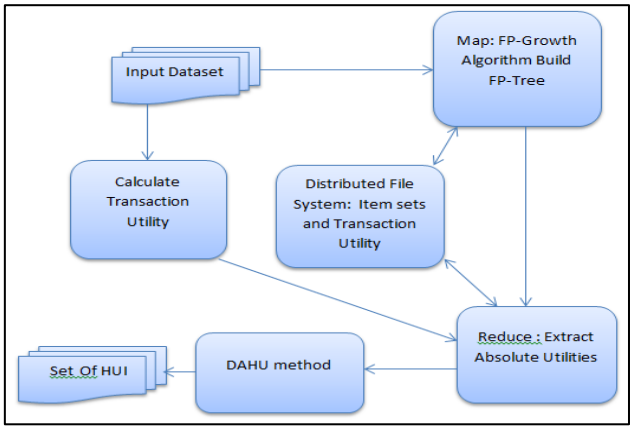

Fig. 1: Flow of Proposed Work

As shown in Fig I input is transactional database this input is given to the Mapper FP-Growth algorithm build the FP-Tree. After this Build FP-Tree is given to Reducer and the itemsets store in Distributed File System. Reduce to extract absolute utilities using Absolute_min_Theshold and store the transaction utility to Distributed File system. Extract Absolute utility given the input to DAHU method that generates set Of HUI.

IV. EXPERIMENTAL RESULT

The system was developed using Java platform and Hadoop technology. The high utility itemsets mining an algorithm called CHUD and DAHU method based on the concept of closed pattern. CHUD conduct mining Closed+ High Utility Itemsets based on pattern generated from FP algorithm in Hadoop.

Testing was performed on the Mall datasets with different User Utility Threshold and result obtained overall better than existing CHUD algorithm. The graph of different user utility threshold with existing CHUD algorithm vs. improve CHUD algorithm which is our implemented algorithm in hadoop is shown in Fig.II

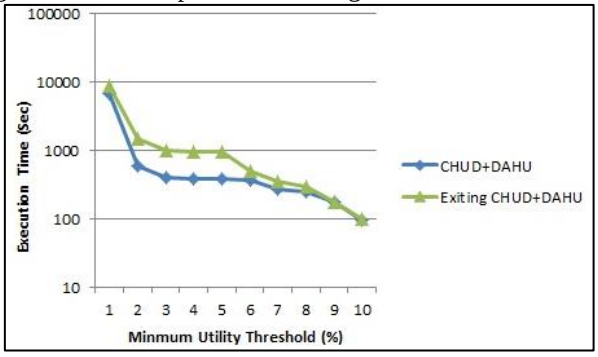

Fig. 2: Result Analysis

On the basis of above graph, we can say that Improve CHUD Algorithm perform better.

V. CONCLUSION

We are discussed the data mining algorithms for mining High Utility Itemsets. Mainly efficient algorithm named as CHUD (Closed High Utility Itemsets Discovery) based on FP-Growth algorithm in Hadoop. The TU-Table (Transaction Utility Table) [26] data structure is used for storing the transaction utilities of transactions. The DAHU (Derive All High Utility itemsets) method is proposed to regain all HUIs from the set of CHUIs without accessing the original database. The experimental result shows that, the proposed model gives better result overall than the existing CHUD (Closed High Utility Itemsets Discovery) algorithm.

REFERENCE

[1] Cheng-Wei Wu, Philippe Fournier-Viger, Philip S. Yu, Fellow, IEEE, Vincent S. Tseng " Efficient Algorithms for Mining the Concise and Lossless Representation of Closed High Utility Itemsets " DOI 10.1109/TKDE. 2014.2345377, IEEE Transactions on Knowledge and Data Engineering

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in Proc. of the 20th Int'l Conf. on Very Large Data Bases, pp. 487-499, 1994.

[3] C.-W.Wu, B.-E. Shie, V. S. Tseng, and P. S. Yu, "Mining top-k high utility itemsets," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2012, pp. 78–86.

[4] C. Lucchese, S. Orlando, and R. Perego, "Fast and memory efficient mining of frequent closed itemsets", IEEE Trans. Knowl. Data Eng., vol. 18, no. 1, pp. 21–36, Jan. 2006.

[5] T. Calders and B. Goethals, "Mining All Non-derivable Frequent Itemsets," in Proc. of the Int'l Conf. on European Conference on Principles of Data Mining and Knowledge Discovery, pp. 74-85, 2002.

[6] K. Chuang, J. Huang, M. Chen, "Mining Top-K Frequent Patterns in the Presence of the Memory Constraint," VLDB Journal, Vol. 17, pp. 1321-1344, 2008.

[7] R. Chan, Q. Yang, and Y. Shen, "Mining High Utility Itemsets," in Proc. of the IEEE Int'l Conf. on Data Mining, pp. 19-26, 2003.

[8] A. Erwin, R. P. Gopalan, and N. R. Achuthan, "Efficient Mining of High utility Itemsets from Large Datasets," in Proc. of the Int'l Conf. on Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 554-561, 2008.

[9] K. Gouda and M. J. Zaki, "Efficiently Mining Maximal Frequent Itemsets," in Proc. of the IEEE Int'l Conf. on Data Mining, pp. 163-170, 2001.

[10] T. Hamrouni, "Key Roles of Closed Sets and Minimal Generators in Concise Representations of Frequent Patterns," Intelligent Data Analysis.Vol. 16, Issue 4, pp. 581-631, 2012.

[11] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," in Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.

[12] T. Hamrouni, S. Yahia, E. M. Nguifo, "Sweeping the Disjunctive Search Space Towards Mining New Exact Concise Representations of Frequent Itemsets," Data & Knowledge Engineering, Vol. 68, Issue 10, pp. 1091-1111, 2009.

[13] H.-F. Li, H.-Y.Huang, Y.-C.Chen, Y.-J.Liu and S.-Y.Lee, "Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams," in Proc. of the IEEE Int'l Conf. on Data Mining, pp. 881-886, 2008.

[14] C.-W. Lin, T.-P.Hong, W.-H.Lu, "An Effective Tree Structure for Mining High Utility Itemsets," Expert Systems with Applications, Vol. 38 Issue 6, pp. 7419-7424, 2011.

[15] C.-W Wu, P. Fournier-Viger, P. S. Yu. and V. S. Tseng, "Efficient Mining of a Concise and Lossless Representation of High Utility Itemsets," in Proc. of the IEEE Int'l Conf. on Data Mining, pp. 824-833, 2011.

[16] H. Li, J. Li, L. Wong, M. Feng, Y. Tan, "Relative risk and odds ratio: a data mining perspective," in Proc. of the ACM SIGACT-SIGMOD-SIGART Sympo-sium on Principles of Database Systems, pp. 368-377, 2005.

[17] B. Le, H. Nguyen, T. A. Cao, and B. Vo, "A Novel Algorithm for Mining High utility Itemsets," in Proc. of the First Asian Conference on Intelligent Infor-mation and Database Systems, pp.13-17, 2009.

[18] Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining Algorithm," in Proc. of the Utility-Based Data Mining Workshop, pp. 90-99, 2005.

[19] C. Lucchese, S. Orlando and R. Perego, "Fast and Memory Efficient Mining of Frequent Closed Itemsets," IEEE Transactions on Knowledge and Data Engineering, Vol. 18, Issue 1, pp. 21-36, 2006.

[20] Y.-C. Li, J.-S.Yeh, and C.-C. Chang, "Isolated Items Discarding Strategy for Discovering High utility Itemsets," Data & Knowledge Engineering, Vol. 64, Issue 1, pp. 198-217, 2008.

[21] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient Mining of Associ-ation Rules using Closed Itemset lattice," Journal of Information Systems, Vol 24, Issue 1, pp. 25–46, 1999.

[22] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient Tree Struc-tures for High utility Pattern Mining in Incremental Databases," IEEE Trans-actions on Knowledge and Data Engineering, Vol. 21, Issue 12, pp. 1708-1721, 2009.

[23] Sotiris Kotsiantis, Dimitris Kanellopoulos, Association Rules Mining: A Recent Overview, GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 71-82.

[24] Gagandeep Kaur, Shruti Aggarwal , Performance Analysis of Association Rule Mining Algorithms, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 8, August 2013, ISSN: 2277 128X.

[25] White, T. (2012) Hadoop: The Definitive Guide, 3rd ed., O'Reilly Media Inc, Sebastopol,CA.

[26] B.-E. Shie, H.-F. Hsiao, V. S. Tseng and P. S. Yu, "Mining High Utility Mobile Sequential Patterns in Mobile Commerce Environments," in Proc. of the Intl. Conf. on Database Systems for Advanced Applications and Lecture Notes in Com-puter Science (LNCS), Vol. 6587/2011, pp. 224-238, 2011.

