Computer Science and Engineering 2015, 5(4): 67-70
DOI: 10.5923/j.computer.20150504.01

Infrequent Weighted Itemset Mining

Puja Keste*, Nuzhat F. Shaikh

Department of Computer Engineering, Modern Education Society's COE, Pune, India

Abstract Mining frequent items in data mining are useful for retrieving the related data present in the dataset. Using an input dataset the weighting function is calculated. Infrequent weighted itemset minimum support value is calculated. Using the minimum support value the infrequent weighted itemset support value is calculated. Then the summation is calculated for all the systems in separately. Then combine the two systems and summate the values which are minimum. Finally, three systems are combined and summate the values which are minimum among the three. Find the threshold value for the dataset and filter the systems combination. If the summation value is greater than the threshold means, then the combination of systems are not considered. Otherwise, it is considered for the future result. Then find the equivalent weighted transaction dataset from transaction dataset. And find the infrequent weighted itemset minimum support value. Find the threshold value for the equivalent weighted itemset dataset. And get the satisfied system combinations. Then an infrequent weighted itemset miner is used to find the common systems that present in the two results. Here we can apply UP Growth algorithm to find the infrequent itemsets from the transactional database. By Using UP growth we can find the infrequent weighted itemset and the result is calculated. From the infrequent weighted itemset mining the final result is calculated.

Keywords Infrequent Itemset mining, Association Rule Mining, weight, Correlation

1. Introduction

Association rule mining is very emerging today as there is growth in finding frequent itemsets in transactional databases as well as infrequent itemsets. Association rule mining good example is If a customer buys butter and bread, customers also buy milk. The Association rule mining goal is to find correlation in different data sets from database. Itemset mining is very important in data mining used for finding the relationships in the dataset. Frequent itemset is finding frequent items in a transactional database where as infrequent itemset mining is finding rare transactions in the database. In association rule mining the first well known algorithm is Apriori technique/algorithm used to find frequent itemsets in a transactional database. Apriori algorithm takes more computational time to find frequent itemsets. Frequent itemset mining is find items in terms of support value of the item in a transactional database. There are mainly two concepts used to find frequent itemsets support and confidence.

2. Existing System

The discovery of infrequent and weighted itemsets, i.e., the Infrequent Weighted Item sets from transactional weighted datasets. The IWI-support measure is defined as a weighted frequency of occurrence of an itemset in the analyzed data. The IWI-support-min measure, which relies on a minimum cost function. That is the occurrence of an itemset in a given transaction is weighted by the weight of its least interesting item. The IWI-support-max Function relies on a maximum cost function, which is used to find maximum number of utilization of items from transactional dataset. Equivalent weighted transaction dataset is use for the process. It is generated from the transaction dataset. IWI Miner is used for find the common systems that present in the transaction dataset and the equivalent weighted transaction dataset. IWI Mining is used for get the frequent items from the transaction dataset. That is the occurrence of an itemset in a given transaction is weighted by the weight of the most interesting item.

Existing Advantages

- IWI miner can be able to discover the infrequent and weighted item set.
- IWI Mining is used for get the frequent items from the transaction dataset.
- Equivalent weighted transaction dataset is use for the process.
- IWI Miner is used for mining high utility item sets.
- Candidate item sets can be generated efficiently with only two scans of the database.

Existing Disadvantages

- Not Portable for huge number of potential high utility itemsets.

* Corresponding author:
poojaa11891@gmail.com (Puja Keste)

- Memory Requirement is more.
- Accuracy is not achieved in the final utility item set.
- Higher processing time is needed
- Too many scans are needed.

3. Proposed System

Two algorithms, named UP-Utility Pattern Growth and UP-Growth+, and a compact Catalog structure, called UP-Catalog (Utility Pattern Catalog), are proposed for discovering high utility itemsets and maintaining important information related to utility patterns within databases. Two Scans are carried out. First, generating Elements of UP Catalog. Then transaction are Reorganized. DLU (Discarding Local Unpromising Items during Constructing a Local UP-Catalog)and DLN Decreasing Local Node Utilities during Constructing a Local UP-Catalog are used for reconstructing the Catalog.

Proposed System Advantages

- Generation of candidates is more efficient.
- Good Tradeoffs in memory usage.
- High utility itemsets are efficiently identified.
- UP-Growth not only reduces the number of candidates effectively.
- The execution time is minimized.

Proposed System Disadvantages

- Large number of candidate item sets degrades the mining performance in terms space requirement.
- The situation may become worse when the database contains lots of long transactions or long high utility itemsets.

4. System Architecture

Frequent itemset mining is very important area area in data mining. It is used to find specific relationships between items in transactional databases. Frequent pattern mining is used to solve the variety of problems like discovering association rules, sequential patterns, correlations and much more. A transactional database conations transactional data which is collection of items called an itemset which have frequent occurrence in a database. In existing techniques there is large set of potential high utility item sets and their mining performance is degraded consequently. There is a lacking of mining performance with these huge number of potential high utility itemsets; higher processing time too. A novel algorithms as well as a compact data structure for efficiently discovering high utility itemsets are proposed. High utility itemsets is maintained in a Catalog-based data structure named UP-Catalog (Utility Pattern Catalog). So implementing a new approach in proposed system named UP-Growth (Utility Pattern Growth) algorithm, for discovering high utility item sets and maintaining important information related to utility patterns within databases. Only two scans are carried out for generating elements of UP-Catalog. Transaction are Reorganized. DLU(Discarding Local Unpromising Items during Constructing a Local UP-Catalog) and DLN Decreasing Local Node Utilities during Constructing a Local UP-Catalog are used for reconstructing the Catalog. Implementing mining process through Discarding Local Unpromising Items and Decreasing Local Node Utilities strategies, Experimental results predicts those not only reduce the number of candidates effectively but also outperform other algorithms. Using our proposed approach generation of candidates is more efficient, memory usage is low, High utility item sets are efficiently identified, UP growth not only reduces the number of candidates effectively. The execution time is minimized.

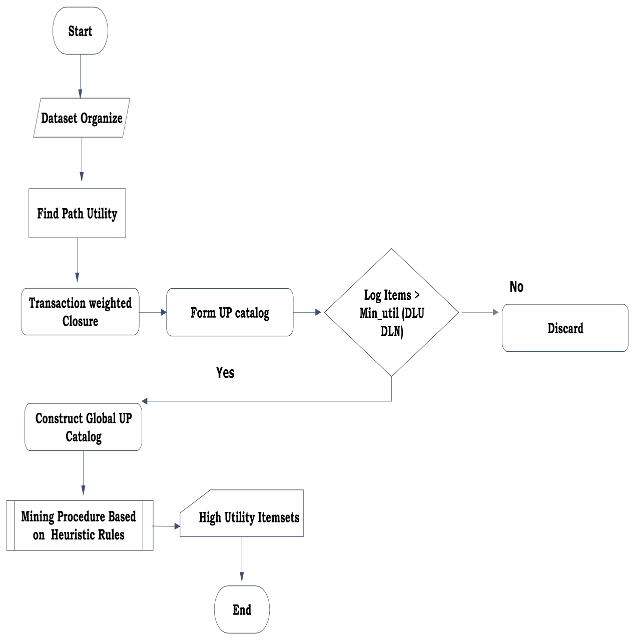
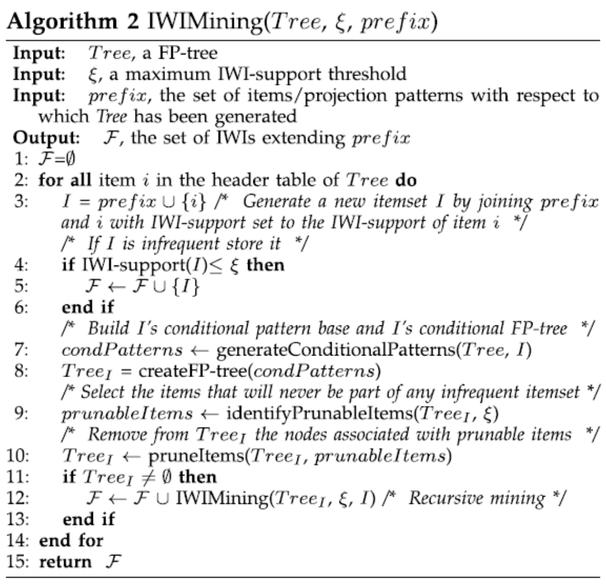

Start
Dataset Organize
Find Path Utility
Transaction weighted Closure
Form UP catalog
Log Items > Min_util (DLU DLN)
No
Discard
Yes
Construct Global UP Catalog
Mining Procedure Based on Heuristic Rules
High Utility Itemsets
End

Figure 1. System Architecture of Infrequent Weighted itemset

5. Algorithms/Methodology

5.1. IWI Miner and MIWI Miner Algorithms

```
Algorithm 1 IWI-Miner($T$, $\xi$)
Input:  $T$, a weighted transactional dataset
Input:  $\xi$, a maximum IWI-support threshold
Output: $\mathcal{F}$, the set of IWIs satisfying $\xi$
1: $\mathcal{F}=\emptyset$ /* Initialization */
   /* Scan $T$ and count the IWI-support of each item */
2: countItemIWI-support($T$)
3: $Tree$ ← a new empty FP-tree; /* Create the initial FP-tree from $T$ */
4: for all weighted transaction $t_q$ in $T$ do
5:     $TE_q$ ← equivalentTransactionSet($t_q$)
6:     for all transaction $te_j$ in $TE_q$ do
7:         insert $te_j$ in $Tree$
8:     end for
9: end for
10: $\mathcal{F}$ ← IWIMining($Tree$, $\xi$, null)
11: return $\mathcal{F}$
```



Algorithm 2 IWIMining($Tree$, ξ, $prefix$)

```
Input:  Tree, a FP-tree
Input:  ξ, a maximum IWI-support threshold
Input:  prefix, the set of items/projection patterns with respect to
   which Tree has been generated
Output:  F, the set of IWIs extending prefix
 1: F=∅
 2: for all item i in the header table of Tree do
 3:     I = prefix ∪ {i} /* Generate a new itemset I by joining prefix
        and i with IWI-support set to the IWI-support of item i */
        /* If I is infrequent store it */
 4:     if IWI-support(I)≤ ξ then
 5:         F ← F ∪ {I}
 6:     end if
        /* Build I's conditional pattern base and I's conditional FP-tree */
 7:     condPatterns ← generateConditionalPatterns(Tree, I)
 8:     Tree_I = createFP-tree(condPatterns)
        /* Select the items that will never be part of any infrequent itemset */
 9:     prunableItems ← identifyPrunableItems(Tree_I, ξ)
        /* Remove from Tree_I the nodes associated with prunable items */
10:     Tree_I ← pruneItems(Tree_I, prunableItems)
11:     if Tree_I ≠ ∅ then
12:         F ← F ∪ IWIMining(Tree_I, ξ, I) /* Recursive mining */
13:     end if
14: end for
15: return F
```

Algorithm 1 is IWI Miner pseudo code is represented. The first steps i.e. lines 2-9 of Algorithm 1 is used to generate the FP-tree associated with the input weighted data set T. Then, at line 10 the recursive mining process is invoked on the constructed FP-tree. The FP-tree is initially occupied with the set of equivalent transactions generated from T. For each weighted transaction, the equivalent set is generated by applying function equivalent transaction Set at line 5, which implements the transactional data set equivalence transformation. Once a compact FP-tree representation of the weighted data set T has been created, the recursive itemset mining process is executed (Algorithm 1, line 10). In Algorithm second we can see pseudo code of the IWI mining procedure. IWI Miner depend on a projection based approach [10], items belonging to the header table associated with the input FP-tree are iteratively considered given in lines 2-14. Initially, at line 3 each item is combined with the current prefix to generate a new itemset I. in line 4-6 If I is infrequent, then it is stored in the output IWI set F. Then, at line 7-8 the FP-tree projected with respect to I is generated and the IWI Mining procedure is recursively applied on the projected tree to mine all infrequent extensions of I at line 12. Unlike conventional FP-Growth-like algorithms [10], IWI Miner adopts a different pruning strategy see lines 9-10 for detail. According to Property, identify Prunable Items procedure visits the FP-tree and identifies items that are only included in paths whose leaves have an IWI-support above since in line 10, they cannot be part of any IWI, they are pruned.

5.2. UP Growth

Input: A UP-Tree, a header table H for UP Tree, item set X, Transactional Database D, User Defined Threshold value
Output: Infrequent Weighted Item sets

```
Begin
Scan Database for Transactions Td->D
Determine transaction utility of Td and TWU of itemset(X)
Compute min_support
if(TW U (X) <=min_sup) then remove items from transaction database
Else insert into header table H and to keep the items in the descending order.
Repeat step 4 & 5 until end of the D.
Insert Td into global UP-Tree
Apply DGU and DGN strategies on global UP-Tree
Re-construct the UP-tree
For each item ai in H do
Generate a HUI = X U ai
Estimate utility of Y is set as ai's utility value in H
put local promising items in CPB into H
Apply strategy DLU to reduce path utilities of the paths
Apply strategy DLN and insert paths into Td
if Td! = null then call for loop
End for
End
```

6. Experimental Results

We are expecting the results by using UP-Growth Algorithm; get the infrequent itemset and minimum infrequent itemset.

1. End user or developer can extract all infrequent itemset from the transactional dataset.
2. End user obtain reorganized transaction utility to get potential high utility itemset.
3. Accuracy is achieved in the final utility item set.
4. By selecting number of utility items it gets high utility itemset.

Table 1. Expected result of System

Input	Existing IWI Algorithm	Proposed UP-Growth Algorithm
Dataset	120(milliseconds)	70(Milliseconds)

Table 1 show the expected results while execution of existing algorithm that is IWI-Miner and proposed algorithm that is UP Growth Algorithm.

7. Conclusions

Discovering the frequent item sets by using the weight. It is for differentiate the relevant items. UP Growth and UP Growth++ mining are proposed efficiently. The discovered patterns are used in real time. For user query the relevant items are discovered by using existing methods take more time. By using our methodology the relevant items for the query is find out between the short duration. Discovering the frequent items are the output for this project. We are discovering the infrequent items on the basis of the frequent weight of the items. The frequent items are omitted. And display the infrequent items in the dataset.

ACKNOWLEDGMENTS

The authors would like to thanks the Department of Computer Engineering Wadia College of Engineering & Research, Pune, India for the guidance and cooperation.

REFERENCES

[1] Agrawal R, Imielinski T, & Swami, A. "Mining association rules between sets of items in large Databases". In proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington, DC, 1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases (VLDB '94), pp. 487-499, 1994. Copyright to IJARCCE www.ijarcce.com 7673.

[3] Luca Cagliero and Paolo Garza "Infrequent Weighted Itemset Mining using Frequent Pattern Growth", IEEE Transactions on Knowledge and Data Engineering, pp. 1- 14, 2013.

[4] Liu, G., Lu, H., Yu, J. X., Wang, W., & Xiao, X.. "AFOPT: An Efficient Implementation of Pattern Growth Approach", In Proc. IEEE ICDM'03 Workshop FIMI'03, 2003.

[5] Balazes Racz, "nonordfp: An FP-Growth Variation without Rebuilding the FP-Tree", 2nd Int'l Workshop on Frequent Itemset Mining Implementations FIMI2004

[6] Grahne O. and Zhu J. "Efficiently Using Prefix-trees in Mining Frequent Itemsets", In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining, 2004.

[7] Cornelia Gyorodi, Robert Gyorodi, T. Cofeey & S. Holban –"Mining association rules using Dynamic FP-trees" – ȋn Proceedings of The Irish Signal and Systems Conference, University of Limerick, Limerick, Ireland, 30th June-2nd July 2003, ISBN 0-9542973-1-8, pag. 76-82.

[8] Grahne G. and Zhu J., "Efficiently Using Prefix-Trees in mining Frequent Item sets," Proc. ICDM 2003Workshop Frequent Item set Mining Implementations, (2003).

[9] A. Gupta, A. Mittal, and A. Bhattacharya, "Minimally Infrequent Itemset Mining Using Pattern-Growth Paradigm and Residual Trees," Proc. Int'l Conf. Management of Data (COMAD), pp. 57- 68, 2011.

[10] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," Procedings of ACM SIGMOD Intnational Conference on Management of Data, ACM Press, Dallas, Texas, pp. 1-12, May 2000.

[11] Han, J., Pei, J., & Yin, Y. "Mining frequent patterns without candidate generation". In Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD '96), Page 205-216, 2000.