NECLAT CLOSED: A Vertical Algorithm for Mining Frequent Closed Itemsets.

PROJECT REPORT

Course code: *CSE 503 - 156*

Course Name: *Data Mining*

Submitted by:

Name : Kasturi Tamhankar
Reg No: 21MSD7003
Name: Deepayan Roy
Reg No: 21MSD7031
Name: Koffi Azougo
Reg No: 21MSD7055
Name: Vaishnavi Tandon
Reg No: 21MSD7022
Name: Kushi
Reg No: 21MSD7011
Name: Jhansi Priya Gudipudi
Reg No: 21MSD7001
Name: Pruthvik Chandarana
Reg.No: 21MSD7040
Name: Mohammad Anas Ansari
Reg.no: 21MSD7030

Submitted to:

Faculty Name: Dr. Lalitha Kumari P

ABSTRACT:

To reduce the runtime and memory requirement of frequent item-sets mining tasks, a lossless and concise collection of all frequent itemsets is provided by frequent closed item sets. A fast-mining algorithm of frequent closed itemset called NECLAT- CLOSED, based on the concept and methods of vertical database format. On comparing with the leading algorithms, NECLAT CLOSED performance is better in terms of runtime and memory usage.

INTRODUCTION:

Frequent itemset mining was created for market basket research to detect items that are commonly purchased together in a consumer transaction database. Most early techniques to mine frequent itemset were centered on enumerating all frequent itemsets as quickly as possible. However, the number of frequently extracted item sets was usually so large that storage and post-processing procedures required unrealistic resources. As a result, various condensed representations for frequent item sets have been developed in order to significantly cut down on the number of item sets to be extracted.

FP-growth is currently one of the most prominent and efficient algorithms for frequent itemset mining. It is based on a prefix tree representation of the specified database of transactions called an FP-tree. This can save a large amount of memory for storing the transactions and memory space. Borgel [1], incorporates two variants of the middle operation of computing a projection of an FP-tree and pruned projected FP-timber by means of casting off objects which have become infrequent due to the projection. Ma and Qi [2], present two techniques that are developed in order to improve the efficiency of mining frequent closed itemsets, the strategy not only reduces the search space but also reduces the size of the FP-tree. The CFI-tree in the algorithm FP close is used, which stores all frequent closed itemsets, but we present a method of projection that can save comparison times of items in closed checking.

Unlike the horizontal database format, each itemset in a vertical database format is associated with a set of transactions. Zaki and Gouda [3], keeps track of differences in the transaction ids of a candidate pattern from its generating frequent patterns. Whereas, Shenoy et al. [4], use a compressed bit-vector representation of itemsets, called snakes, and aggressively materialize the benefits offered by the vertical data layout.

A large number of iterations is required for processing the items and more escape time is required for finding frequent itemset in Eclat. Eclat algorithm uses a bottom-up approach that is very complex.

Kaur et al [5], improved the Eclat algorithm to reduce escape time and the number of iterations by using a top-down approach and transposing the original database D. Mohapatra et al. [5], introduced two algorithms for frequent itemset mining, Apriori, it uses depth-first search and Eclat. Eclat algorithm that even after varying the minimum support the maximum and minimum support remains constant and using the Apriori algorithm that increasing the minimum lift value results in a gradual decrease in maximum support and a small increase in minimum confidence values. With the combination of these two algorithms frequent itemset mining is performed efficiently.

Manjit Kaur[6] cuts down access time using vertical dataset in eclat by alleviating the limitation of apriori that requires repeated searching of database to locate common itemsets. Trieu and Kunieda [7], used combination transaction_idset + different_set formats to represent databases in vertical layout to be more beneficial if the minimum support is high. On the other hand, Singh et al. [8] show the scalability of the proposed algorithms by increasing the number of cores and size of the dataset, outperforming the Spark-based Apriori by many times. Yu et al. [10], sorted items in descending order according to the frequencies in the transaction cache while itemset used ascending order of support during support count. Compared with the traditional Eclat algorithm, the results of experiments show that the Bi-Eclat algorithm gains better performance on several public databases given.

Eclat and its various versions have crisp applications in a real-time environment. Juliandiny et al. [11], applied the association rule using the ECLAT algorithm to determine purchasing pattern of products at the Coffee Shop. The ECLAT algorithm performs a frequent itemset search using the association rule technique. The combination of items formed can be recommended to the Coffee Shop owner to form a package menu and assist in determining future business decisions. Zheng et al. [12], applied an optimized method of frequent sets calculation-a method of parallel NEclat combined with a cloud programming model, and on-road transportation management. This method can solve the problem that the Eclat algorithm cannot be calculated by pruning, and achieve a parallel compute. The practical application showed that this method can reduce time waste by more than 40%, and it is suitable for the data mining of transport management information association rules.

METHODOLOGY:

The frequent itemset mining task is formally defined as follows (Agrawal et al., 1993). Consider a set of items I = {i1, i2, …, im}. A transaction database D = {T1, T2, …, Tn} is a set of transactions, where Tj⊆I. For convenience, we refer to each transaction with its index (identifier). A set Y, where

Y⊆I, is called an itemset or a pattern. An itemset Y is called a k-itemset iif (if and only if) it contains k items. A transaction T contains an itemset Y if and only if Y⊆T. A set of transactions containing an itemset Y is called the transaction-set of Y, and is denoted as TSet(Y).

For convenience, items are ordered according to the relation ≼. For any two items i and j, i≼j iif support(i)⩽support(j). We can write a k-itemset Y as Y = Xi, where i is the largest item in Y, concerning relation ≼, and X is a (k-1)-itemset. An itemset Y is frequent iif support(Y)⩾minSupport, where minSupport is a user-defined minimum frequency threshold.

In this paper the memory representations that we have used:

-Search Tree to represent the search space.

-Traverse the search Tree in a depth-first manner.

-Hashmap data structure for checking the subsumption where the hashmap associates the support for an itemset to a list of closed frequent itemsets.

The mathematical concepts used are set Theory with its properties and Closure as well as Partial Closure of an Itemset.

NEclat Closed Algorithm is implemented mainly using three procedures.

A - Procedure (Algorithm)1:

-Scan the entire database to find the 1-itemsets which are ordered according to ≤ relation.

-Insert every 1-itemset into the list of the children of the root note (By default the root node is ∅.

-Generate the set of all the frequent closed itemsets by visiting each and the child from the root. (The process of visiting each child leads to procedure two where we should check some conditions)

```
Input: A transaction database D, and a user-defined minSupport
Output: The set of all frequent closed itemsets
1 Scan D to find TSets of 1-itemsets;
2 F1 ← Frequent 1-itemsets which are ordered according to ≼;
3 root ← ∅;
4 for i ∈ F1 do
5   Insert i into the list of children of root;
6 for each child current in the children of root do
7   visit(current,I); // Algorithm 2
```

B - Procedure 2: Visiting each child and store its Partial Closure

This procedure tells whether the right siblings of a visiting child will be added to the PClosure of the visiting child or not.

Let Z be the super-items formed by the visiting child and its right siblings.

-If the support of Z is greater than the minimum support and The Negative Transaction Sets of Z (NegTSet(Z)) is equal to the empty set (∅), then We add the siblings to the PClosure of the visiting child.

-But if the support of Z is greater than the minimum support, and the NegTSet(Z) is not equal to the empty set, the Z is considered a child of the visiting node.

-If the support of the current node is equal to the support of its siblings then we remove the siblings because it is not considered siblings.

```
   Input: The current itemset Y and the level of Y in the itemset search tree
1  Procedure visit(Y,level)
2      PClosure(Y) ← Y;
3      for each SIB in the right siblings of Y do
4          Let Y = Xi and SIB = Xj;
5          Z ← Xij;
6          if level = 1 then
7              NegTSet(Z) = TSet(Y) − TSet(SIB);
8          else
9              NegTSet(Z) = NegTSet(SIB) − NegTSet(Y);
10         support(Z) = support(Y) − |NegTSet(Z)|;
11         if minSupport ≤ support(Z) then
12             if TNegSet(Z) = ∅ then
13                 Insert j into PClosure(Y);
14             else
15                 Insert Z into the list of children of Y;
16         if support(Y) = support(SIB) then
17             Remove SIB from the siblings of Y;
18     if insertIfClosed(PClosure(Y),support(Y)) then
19         Print PClosure(Y);
20     for each child current in the children of Y do
21         visit(current,level + 1);
22     return;
```

C - Procedure 3: Checking the Subsumption

This procedure tells whether an Itemset P is a closed itemset or not.

Definition: Let A and B be two itemsets.

A is subsumed by B if and only if A is a subset of B and support of A is equal to support of B : (A ⊆ B and support (A) = support (B))

For checking the subsumption:

-Let P be the itemset that we subsume. L is a list of all the itemset having the same support of P using Hashmap.

-For each item set Q in L, if | P | < | Q | and P ⊆ Q, then P is not closed.

-Else P is subsumed by Q, hence insert P just before Q.

```
Input: The itemset P and its support value
Output: A boolean value that indicates whether P is a closed itemset or not.
1  Procedure insertIfClosed(P,support(P))
2      L ← Hash[support(P)];
3      isClosed ← true ; // By default, P is a closed itemset.
4      if L is empty then
5          Insert P into L;
6      else
7          for each itemset Q in L do
8              if |P| < |Q| then
9                  if P ⊆ Q then
10                     isClosed ← false ; // Change the default assumption about P.
11                     break;
12             else
                   // P can not be subset of any remaining itemsets.
13                 Insert P into L, just befor Q;
14                 break;
15     return isClosed;
```

CONCLUSION AND FUTURE WORK:

This paper proposed an efficient algorithm for mining frequent closed itemsets using the vertical database and Negative Transaction set by validating the subsumption and boosting the runtime and memory consumption. Initially, the concepts and techniques underlying NECLATCLOSED were based on the vertical database representation and Transaction Sets.

Using these concepts and techniques, NECLATCLOSED was proposed, followed by a simple bitmap-based structure.

For fast subsumption checking, a set representation was developed finally to demonstrate the effectiveness of the proposed algorithm in terms of Experiments carried out on runtime and memory consumption.

The results of the experiments revealed that in terms of performance, NECLATCLOSED outperforms the other leading algorithms

Limitations of NEclat algorithm

Although the memory consumption of the NECLATCLOSED is better than that of the leading algorithms, memory consumption for large datasets can still be high.

Be enhanced the most serious problems with frequent itemset mining the tasks that lead to algorithm performance impediments are a large search space, as well as massive amounts of input and output data.

The problem's vast operational and data-related spaces limit the serial algorithm enhancements Only to a limited extent can serial algorithms improve mining process performance.

In the same space, improvement is not possible. This is also true for our algorithm. Working with big data often necessitates the Parallel programming architectures to be used. As an example, the proposed algorithm, as one of the future works, has the potential to lay the groundwork for novel algorithms.

Based on parallel programming architectures, particularly the Map-Reduce Framework Other future projects include the creation of similar algorithms for top-rank-k frequent closed itemset mining, frequent maximal item sets, and top-rank-k frequently occurring maximal itemsets

REFERENCE:

[1] An Implementation of the FP-growth Algorithm, Christian Borgelt, Department of Knowledge Processing and Language Engineering, School of Computer Science, Otto-von-Guericke-University of Magdeburg, Universitatsplatz 2, 39106 Magdeburg, Germany

[2] L. Ma and Y. Qi, "An Efficient Algorithm for Frequent Closed Itemsets Mining," 2008 International Conference on Computer Science and Software Engineering, 2008, pp. 259-262, doi: 10.1109/CSSE.2008.1042.l

[3] Fast Vertical Mining using Diffusers

Zaki, M.J. and Gouda, K., 2003, August. Fast vertical mining using diff sets. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 326-335).

[4] Turbo-charging vertical mining of large databases

Shenoy, P., Haritsa, J.R., Sudarshan, S., Bhalotia, G., Bawa, M., and Shah, D., 2000. Turbo-charging vertical mining of large databases. ACM Sigmod Record, 29(2), pp.22-33.

[5] TITLE: Advanced Eclat Algorithm for Frequent Itemset Generation

Author: Kaur et al

Year: 2015

[6] D. Mohapatra, J. Tripathy, K. K. Mohanty, and D. S. K. Nayak, "Interpretation of Optimized Hyper Parameters in Associative Rule Learning using Eclat and Apriori," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 879-882, doi: 10.1109/ICCMC51019.2021.9418049.

[7] International Journal of Computer Systems (ISSN: 2394-1065), Volume 01– Issue 03, December 2014 ECLAT Algorithm for Frequent Itemsets Generation Manjit Kaur, Urvashi Grag Computer Science and Technology, Lovely Professional University Phagwara, Punjab, India.

[8] Trieu, T.A. and Kunieda, Y., 2012, February. An improvement for the dEclat algorithm. In Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication (pp. 1-6)

[9] Singh, P., Singh, S., Mishra, P.K. and Garg, R., 2019, May. RDD-Eclat: approaches to parallelize Eclat algorithm on spark RDD framework. In International Conference on Computer Networks and Inventive Communication Technologies (pp. 755-768). Springer, Cham.

[10] TITLE: Improvement of Eclat Algorithm Based on Support in Frequent Itemset Mining
Author: Yu et al
Year: 2014

[11] S. Juliandiny, P. Palupiningsih, H. Bedi Agtriadi, B. Prayitno, and E. Putra, "Implementation of ECLAT Algorithm to Determine Product Purchasing Pattern at Coffee Shop," *2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*, 2022, pp. 219-222, doi: 10.1109/ISMODE53584.2022.9742812.
https://ieeexplore.ieee.org/abstract/document/9742812

[12] Zheng X, Wang S. Study on the method of road transport management information data mining based on pruning eclat algorithm and MapReduce. Procedia-Social and Behavioral Sciences. 2014 Jul 14;138:757-66.
https://www.sciencedirect.com/science/article/pii/S1877042814041743