International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 2, Issue 5, May 2014)

Implementation of Apriori Algorithm on Traffic Data

Kavita[1], Trilok Gabba[2]
[1]M. Tech schacolr, [2]HOD, BITS, Bhiwani

***Abstract*— Data is increasing very rapidly now days. Due to this reason it is difficult to manage and search in data is a challenge. Our researchers are going to work on management of data and make it easy to access to any one. So association rule is technique that is used in data mining for make it easy to access. In this paper we are going to represent implementation of Apriori algorithm on data of train traffic. This implementation identify that how much chance of accident occur in which condition.**

***Keywords*— Data Mining, Association Rule, Apriori Algorithm, Item set generation, Railway traffic.**

I. INTRODUCTION

One of the most popular data mining approaches is to find frequent item-sets from a transaction dataset and derive association rules. Finding frequent item-sets (item-sets with frequency larger than or equal to a user specified minimum support) is not trivial because of its combinatorial explosion. Once frequent item-sets are obtained, Association rules are generated straight forward with confidence larger than or equal to a user specified minimum confidence. Apriori is an algorithm for finding frequent item-sets using candidate generation [1]. It is characterized as a level-wise complete search algorithm using anti-monotonicity of item-sets, "if an item-set is not frequent, then no one superset can be frequent". By convention, items assume by Apriori within a transaction or item-set are sorted in lexicographic order. Size k be Fk for frequent item-set and their candidates be Ck. Apriori first scans the database and searches for frequent item-sets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. Following three steps are iterates and extracts all the frequent item-sets.

1. Generate Ck+1, candidates of frequent item-sets of size k +1, from the frequent item-sets of size k.

2. Scan the database and calculate the support of each candidate of frequent item-sets.

3. Add those item-sets that satisfy the minimum support requirement to Fk+1.

The Apriori algorithm is shown in Fig. 3. Line 3 generates Ck+1 function for Apriori from Fk in the following two step process:

a. Join step: Generate RK+1, the initial candidates of frequent item-sets of size k + 1 by taking the union of the two frequent item-sets of size k, Pk and Qk that have the first k−1 elements in common.

RK+1 = Pk ∪ Qk = {i teml, . . . , i temk−1, i temk , i temk}

Pk = {i teml , i tem2, . . . , i temk−1, i temk }

Qk = {i teml , i tem2, . . . , i temk−1, i temk}

Where, i teml < i tem2 < • • • < i temk < i temk.

b. Prune step: Check if all the item-sets of size k in Rk+1 are frequent and generate Ck+1 by removing those that do not pass this requirement from Rk+1. This is because any subset of size k of Ck+1 that is not frequent cannot be a subset of a frequent item-set of size k + 1.

Function subset in line 5 finds all the candidates of the frequent item-sets included in transaction t. Then, frequency calculates by Aprioti algorithm only for those candidates generated this way by scanning the database. Mostly kmax+1 times an Apriori algorithm scan database when the maximum size of frequent item-sets is set at kmax.

II. APRIORI ALGORITHM

A. Apriori Algorith–

Following pass can be used to generate all frequent item-set

Pass 1

1. Generate the candidate item-sets in C1
2. Save the frequent item-sets in L1

Pass k

1. Generate the candidate item-sets in Ck from the frequent item-sets in Lk-1

i. Join Lk-1 p with Lk-1q, as follows:

Insert into Ck

Select p.item1, p.item2 . . . p.itemk-1, q.itemk-1

From Lk-1 p, Lk-1q

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 2, Issue 5, May 2014)

Where p.item1 = q.item1, p.itemk-2 = q.itemk-2, p.itemk-1 < q.itemk-1

ii. Generate all (k-1)-subsets from the candidate item-sets in Ck

iii. Prune all candidate item-sets from Ck where some (k-1)-subset of the candidate item-set is not in the frequent item-set Lk-1

2. Scan the transaction database to determine the support for each candidate item-set in Ck

3. Save the frequent item-sets in Lk

B. Example of algorithm –

Minimum support is 50% from user specified its assume.

- Given: Database of transaction shown below

Table- 1
Database

TID	*A*	*B*	*C*	*D*	*E*	*F*
T_1	1	0	1	1	0	0
T_2	0	1	0	1	0	0
T_3	1	1	1	0	1	0
T_4	0	1	0	1	0	1

- The Candidate Item-Sets In C2 Are Shown Below

TABLE- 2
ITEM-SET IN C2

Itemset *X*	supp(X)
{A,B}	25%
{A,C}	50%
{A,D}	25%
{B,C}	25%
{B,D}	50%
{C,D}	25%

- The frequent item-sets in L2 are shown below

Table – 3
Item-set in L2

Itemset *X*	supp(X)
{A,C}	50%
{B,D}	50%

III. RELATED WORK

Since Apriori algorithm was first introduced and as experience was accumulated, frequent item-set mining can be devise using efficient algorithm. Many of them share the same idea with Apriori in that they generate candidates. The size of candidate item-sets can be reduced using Hash-based technique. Each item-set is hashed into a corresponding bucket by using an appropriate hash function. Since a bucket can contain different item-sets, if its count is less than a minimum support, these item-sets in the bucket can be removed from the candidate sets. The entire mining problem divided into n smaller problems using partitioning. There are n non-overlapping partitions used to divide dataset such that each partition fits into main memory and each partition is mined separately. Since any item-set that is potentially frequent with respect to the entire dataset must occur as a frequent item-set in at least one of the partitions, all the frequent item-sets found this way are candidates. A random sampled small subset of the entire data is simply to mine by using sampling. We can find all the frequent item-sets there is no guarantee for this, normal practice is to use a lower support threshold. Between accuracy and efficiency a trade-off has generated. A horizontal data format is used in Apriori, i.e. frequent item-sets are associated with each transaction. In transaction IDs (TIDs) different format is used for vertical data are associated with each item-set. With this format, Intersection of TIDs is taken to perform mining process. The support count is simply the length of the TID set for the item-set. There is no need to scan the database because TID set carries the complete information required for computing support. FP-growth (frequent pattern growth) method is an improvement in Apriori algorithm that succeeded in eliminating candidate generation [2]. It adopts a divide and conquer strategy by (1) compressing the database representing frequent items into a structure called FP-tree (frequent pattern tree) that retains all the essential information and (2) dividing the compressed database into a set of conditional databases, each associated with one frequent item-set and mining each one separately. It scans the database two times. At first scan, all the frequent items and their support counts (frequencies) are derived and they are sorted in the order of descending support count in each transaction. When come to second scan item belongs to each transaction is merged into a prefix tree and items (nodes) that appear in common in different transactions are counted. An item is associated with each node and its count.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 2, Issue 5, May 2014)

Since descending order of frequency to short items, Near to root of prefix tree some nodes closer are shared by more transactions, Very compact representation is resulting that stores all the necessary information. By choosing an item in the order of increasing frequency, pattern growth algorithm works on FP-tree and extracting frequent item-sets that contain the chosen item by recursively calling itself on the conditional FP-tree. Original Apriori algorithm is slower than FP-algorithm in order of magnitude.

There are several other dimensions regarding the extensions of frequent mining for pattern. Mostly The following included: (1) incorporating taxonomy in items [3]: Use of taxonomy makes it possible to extract frequent item-sets that are expressed by higher concepts even when use of the base level concepts produces only infrequent item-sets. (2) Incremental mining: Database is not stationary that is assumed and a new instance of transaction keeps added. The algorithm in [4] updates the frequent item-sets without restarting from scratch. (3) For item we use numeric value: When the item corresponds to a continuous numeric value, current frequent item-set mining algorithm is not applicable unless the values are discretised. Subspace method of clustering can be used to obtain an optimal value interval for each item in each item-set [5]. (4) Information gain or $\chi 2$ value is also use for measure like frequency: These measures are useful in finding discriminative patterns but unfortunately do not satisfy anti-monotonicity property. However, these measures have a nice property of being convex with respect to their arguments and it is possible to estimate their upper bound for supersets of a pattern and thus prune unpromising patterns efficiently. Apriori SMP uses this principle [6]. (5) Using richer expressions than item-set: To enable mining from more complex data structure enable by tree, sequence etc [7, 8]. (6) Closed item-sets: A frequent item-set is closed if it is not included in any other frequent item-sets. Thus, once the closed item-sets are found, all the frequent item-sets can be derived from them. To find the closed item-sets we use LCM is most efficient algorithm [9].

IV. Implementation Results

As we have choose Train traffic data for implementation of apriori algorithm. step by step process shown in figures as following:

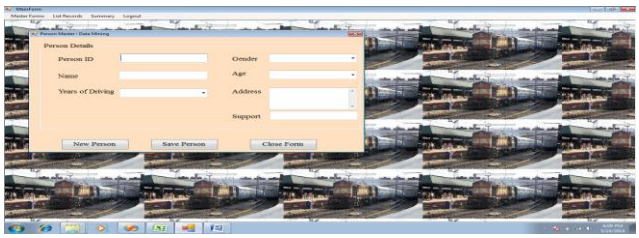

Figure 1 Form to fill personal detail of passangers

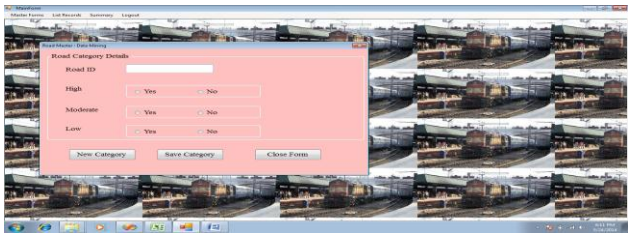

Figure 2 Detail of road category

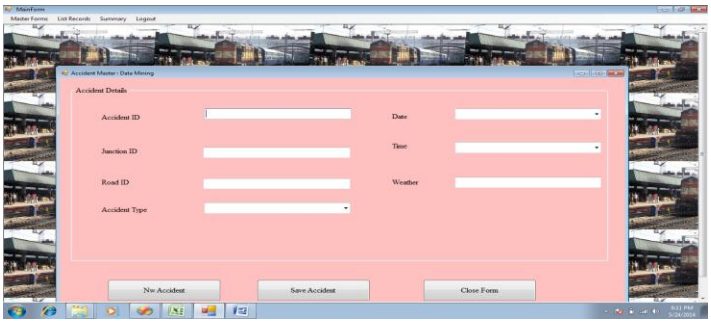

Figure 3 Form of accident detail

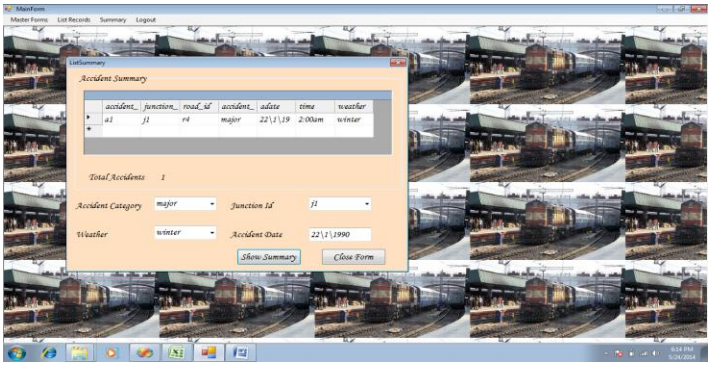

Figure 4 Accident Summary

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 2, Issue 5, May 2014)

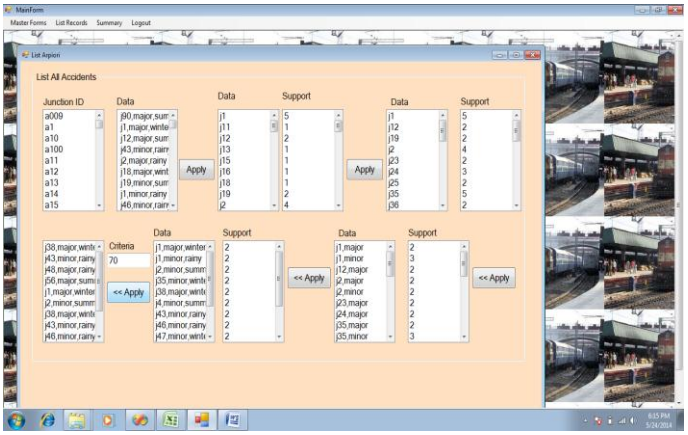

Figure 5 List of accident on basis of apriori algorithm

V. CONCLUSION

This implementation is focused on how to solve the efficient problems of Apriori algorithm and raise another association rules mining algorithm. This has certain reference value to research and solve the issues of data expiation and information lacking. It hopes to dig out more useful information. So data managed in proper way that is easy for anyone to access related data.

REFERENCES

[1] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th VLDB conference, pp 487–499.

[2] Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of ACM SIGMOD international conference on management of data, pp 1–12

[3] Srikant R, Agrawal R (1995) Mining generalized association rules. In: Proceedings of the 21st VLDB conference. pp. 407–419.

[4] CheungDW,Han J,NgV,WongCY(1996) Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proceedings of the ACM SIGMOD international conference on management of data, pp. 13–23

[5] Washio T, Nakanishi K, Motoda H (2005) Association rules based on levelwise subspace clustering. In: Proceedings. of 9th European conference on principles and practice of knowledge discovery in databases. LNAI, vol 3721, pp. 692–700 Springer, Heidelberg.

[6] Morishita S, Sese J (2000) Traversing lattice itemset with statistical metric pruning. In: Proceedings of PODS'00, pp 226–236.

[7] Yan X, Han J (2002) gSpan: Graph-based substructure pattern mining. In: Proceedings of ICDM'02, pp 721–724.

[8] Inokuchi A,Washio T, Motoda H (2005) General framework for mining frequent subgraphs from labeled graphs. Fundament Inform 66(1-2):53–82.

[9] Uno T, Asai T, Uchida Y, Arimura H (2004) An efficient algorithm for enumerating frequent closed patterns in transaction databases. In: Proc. of the 7th international conference on discovery science. LNAI vol 3245, Springe, Heidelberg, pp 16–30.