Abstraction Based Data Mining

Gaurav Kothari[#1], Anup Ahuje[#2], Amish Patel[#3], Abhinandan Khilari[#4]

[#]*Department of Computer Engineering*
KJ Somaiya College of Engineering, Vidyavihar, Mumbai, India
[1]*gaurav.k@somaiya.edu*
[2]*anup.ahuje@somaiya.edu*
[3]*amish.patel@somaiya.edu*
[4]*abhinandan.k@somaiya.edu*

***Abstract*—**Data Mining is the process in which useful information is extracted from large dataset. Such huge datasets consists of typically large number of patterns and features. This consumes a lot of storage space and since all of the data cannot be stored in main memory, it has to be fetched from secondary storage as required increasing the disk I/O operations. This situation can be resolved by using abstraction in data mining. Abstraction in simple terms refers to compact representation of dataset. Such an abstraction helps in reducing the time and space requirements of the overall decision making process. It is also important that the abstraction generated from the data are generated in as minimum number of scans as possible. In this paper we implemented existing algorithms which generate compact representations of patterns in data mining operations and analysed and compared the results of implementation to determine their efficiency.

Keywords— *Abstraction, Data Mining, Pattern Count Tree, Prefix-Suffix Tree, Prefix-Postfix Tree, Pattern Range Tree, Node, Compression Rate, Compaction, Pattern, Feature*

*****

I. INTRODUCTION

Due to rapid advancement in technology, the collection and storage of data has increased highly. In addition to this today's databases are dynamic. The state of database changes due to either addition/deletion of tuples to/from the database. Further, in order to extract insights or knowledge from these large datasets, data mining operations are used. Clustering is one such data mining activity which deals with large amounts of data. Clustering has been widely applied in many areas such as pattern recognition and image processing, information processing, medicine, geographical data processing, and so on. Most of these domains deal with massive collections of data. In order to handle such huge data, methods used should be efficient in terms of memory, storage and processing time.

In data mining application, both the number of patterns and features are typically large. They can't be stored in main memory and hence needs to be transferred from secondary storage as and when required. This takes a lot of time. In order to reduce time, it is necessary to devise efficient algorithms to minimize the disk I/O operations. This situation can be resolved by using abstraction in data mining. Abstraction in simple terms refers to compact representation of dataset .Such an abstraction helps in mining the time and space requirements of the overall decision making process. It is also important that the abstraction be generated from the data in small number of scans.

Abbreviations

PC	Pattern Count tree
PS	Prefix-Suffix Tree
PPS	Prefix-Postfix Tree
PR	Pattern Range Tree
N1	Number of Nodes before compression
N2	Number of Nodes after compression
CR	Compression Rate

II. LITERATURE REVIEW

A background study is done to review the existing algorithms used for data abstraction. Four existing algorithms are chosen for implementation, analysis and comparison. [1].

A. *Pattern Count Tree*

In PC Tree Nodes of the branches of the patterns are shared if they have same prefix. [1][4][5]

B. *Prefix-Suffix Tree*

In Prefix-Suffix Structure Nodes of the branches of the patterns are shared if they have same prefix and same suffix [2].

C. *Prefix-Postfix Tree*

In Prefix-Postfix Structure Nodes of the branches of the patterns are shared if they have same postfix [2].

D. ***Pattern Range Tree***

It a data structure, which is used to store the training patterns in a compact manner .by skipping common nodes in across patterns [3]

From the survey it can be inferred that these algorithms achieve a considerable abstraction of dataset by reducing storage and performance time in mining process which is instrumental in improving data mining process.

III. DATA ABSTRACTION ALGORITHMS

Abstraction in simple terms means compact representation of data. Abstraction helps in reducing the time and space requirements of the overall decision making process in data mining operations. Detailed explanations for these algorithms are provided in this section.

A. *Pattern Count Tree (PC)*

Pattern count tree is a data structure stores all transactions of the transaction database [4], DB in a compact way. In PC Tree Nodes of the branches of the patterns are shared if they have same prefix. Each node of the PC-tree consists of four fields

<u>*Feature*</u>: specifies non-zero positional value of the pattern
<u>*Count*</u>: specifies the number of patterns represented by a portion of the path reaching this node
<u>*Child pointer:*</u> represents the pointer to the following path
<u>*Sibling pointer*</u> points to the node which indicates the subsequent other paths from the node under

Feature	Count	Child Pointer	Sibling Pointer

Figure 1. PC Tree Node Structure

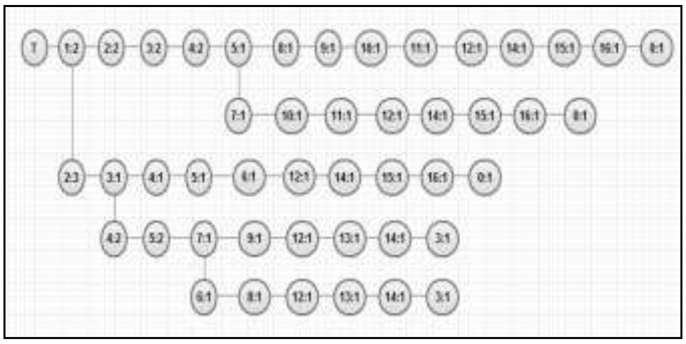


Figure 2. PC Tree Structure

Characteristics

i. PC-tree is a complete representation of the transaction database.
ii. PC-tree provides a compact representation of the transaction database.
iii. PC-tree is independent of order of the transactions in the transaction database.
iv. PC-tree is Incremental. New patterns can be added to existing structure
v. PC-tree is constructed in a single scan of database.

B. *Prefix-Suffix Tree (PS)*

Prefix-Suffix Tree is a variant of PC-tree and is similar to Prefix-Postfix structure in which the transactions having same prefix or same postfix have their branches being shared [2].The construction is similar to Prefix-Postfix structure. The advantage of this scheme is that this also generates some synthetic patterns which aids in clustering. The advantage of this scheme over PPC is that it is possible to get back the original transactions by storing a little information. Each node consists of following:

<u>*Feature*</u>: specifies non-zero positional value of the pattern
<u>*Count*</u>: specifies the number of patterns represented by a portion of the path reaching this node
<u>*Child pointer:*</u> represents the pointer to the following path
<u>*Sibling pointer*</u>: points to the node which indicates the subsequent other paths from the node under

Feature	Count	Child Pointer	Sibling Pointer

Figure 3. Prefix-Suffix Tree Node Structure

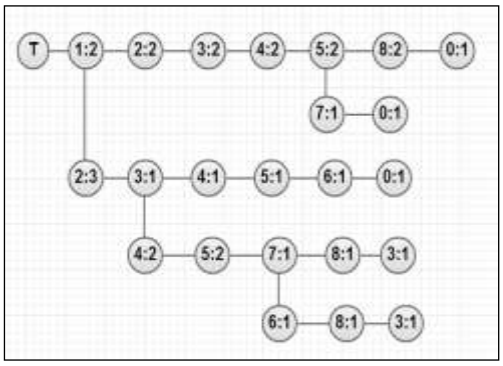


Figure 4. PS Tree Structure

Characteristics

i. PS-tree is a complete representation of the transaction database.
ii. PS-tree is a compact representation of the transaction database.
iii. PS-tree is independent of order of the transactions in the transaction database.
iv. PS-tree is Incremental. New patterns can be added to existing
v. structure

vi. It is simple to construct.
vii. It also generates a synthetic pattern which helps in clustering.
viii. It is possible to retrieve original patterns.

C. *Prefix-Postfix Tree (PP)*

In this Structure, along with the prefixes, patterns having same suffixes are also shared. The algorithm is similar to PC-tree construction [2].

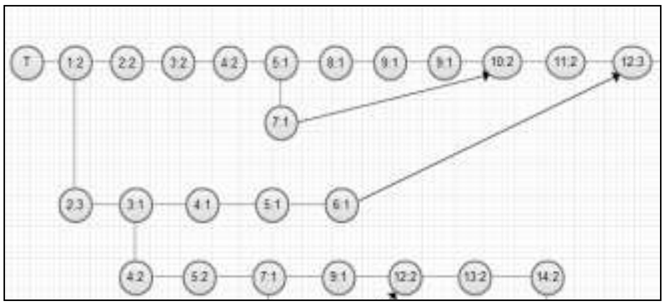


Figure 5. PP Tree Structure

Characteristics

i. It gives compact representation of transactions of database.
ii. It gives complete representation of transaction database.
iii. Construction of the structure is complex since it searches the already constructed structure to check whether current pattern's postfix is already present.
iv. Multiple scans are required to construct this structure

D. *Pattern Range Tree (PR)*

PR-tree is a data structure, which is used to store the training patterns in a compact manner [3]. Each node of the tree consists of the fields shown in Figure below.

<u>Feature</u>: field specifies the position of nonzero value of the pattern.
<u>Child</u>: field represents the pointer to the following path.
<u>Sibling</u>: field represents pointer to the node which indicates subsequent other paths from the parent of the node under consideration.
<u>Flag</u>: field gives the information about the existence of the feature between the FEATURE field value of current node and the FEATURE field value of its immediate parent.
A negative value of FLAG field signifies the absence of all the features between the FEATURE field value of current node and FEATURE field value of its immediate parent.
A positive FLAG field value signifies the presence of all features between the FEATURE field value of current node and the FEATURE field value of its immediate parent.

Feature	Child	Sibling	Flag

Figure 6. PR Tree Node Structure

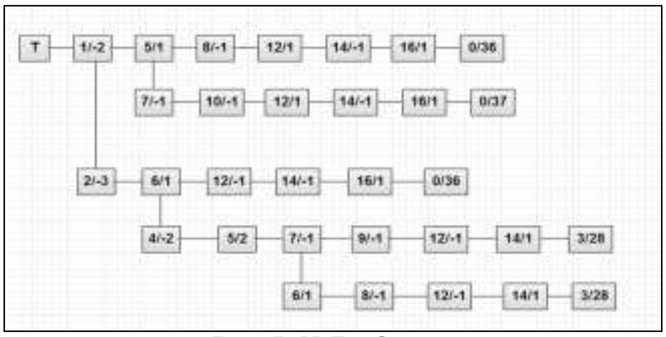


Figure 7. PR Tree Structure

Characteristic

i. In PR-Tree, there is no need to maintain a node for every feature in the pattern.
ii. PR-tree is a complete representation of the transaction database.
iii. PR-tree is a compact representation of the transaction database.
iv. PS-tree is Incremental. New patterns can be added to existing structure.
v. It is possible to retrieve original patterns.

IV. COMPARISON AND ANALYSIS OF ALGORITHMS ON TEST DATASET

We have considered a test data set and have applied the four algorithms discussed in the previous section on test transaction dataset to get the respective representations, then comparison between these algorithms is made with respect to number of nodes and compression rate. Programming language used for implementing algorithms is JAVA.

Test Dataset

Consider an example of small test transaction dataset .Each pattern has unique ID (In this case it's 1, 2, 3, 4, 5 and 6) and every pattern has set of features. Patterns having same label can be merged depending on the algorithm used and they can share same features. On the following dataset, all the above discussed four algorithms are applied and corresponding reduction in number of nodes is noted as well as diagrammatic representation of the tree generated is shown.

Pattern	Features	Label
1	1,2,3,4,5,8,9,10,11,12,14,15,16	0
2	1,2,4,5,7,10,11,12,14,15,16	0
3	2,3,4,5,6,12,14,15,16	0
4	2,4,5,7,9,12,13,14	3

5	2,4,5,6,8,12,13,14	3
	Total Nodes : 54	

Pattern Count Tree (PC) Representation

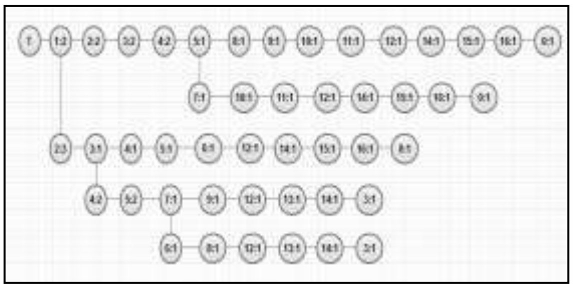


Figure 8. PC Tree for Dataset

No of Nodes: 47

Prefix Suffix Tree (PS) Representation

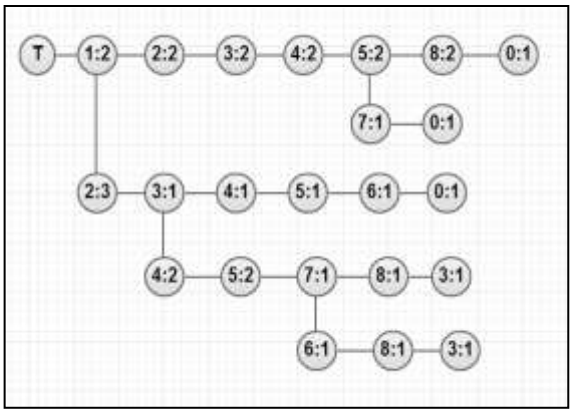
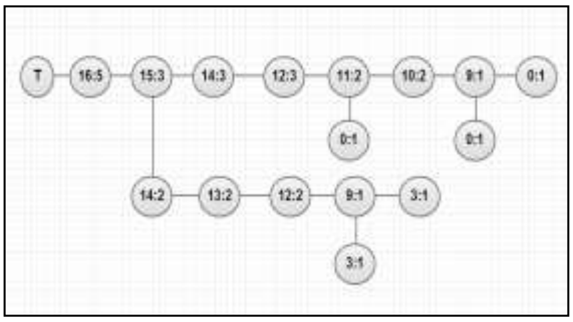


Figure 9. Prefix Tree for Dataset

No of Nodes: 24



No of Nodes: 17

Total Nodes = 24+17=41

Prefix Postfix Structure (PP) Representation

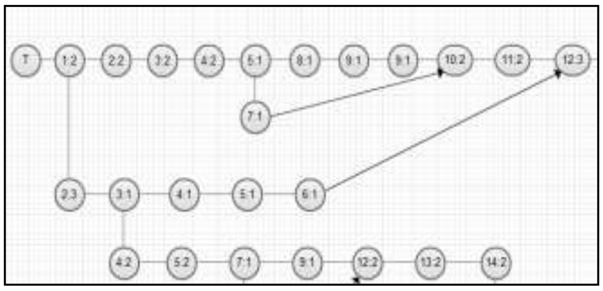


Figure 10. PPS Tree for Dataset

No of Nodes: 32

Pattern Range Tree (PR) Representation

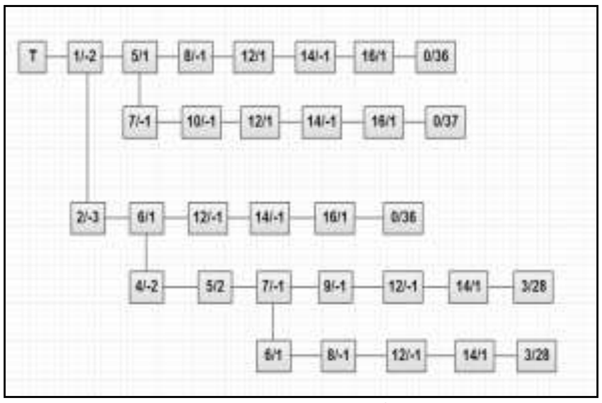


Figure 11. PR Tree for Dataset

No of Nodes: 32

Comparison

Results obtained from implementing the algorithms on dataset are analysed and comparison is established.

Parameters used for comparison are

i. Number of Nodes
ii. Compression Rate

N1 - Number of Nodes before compression
N2 - Number of Nodes after compression
CR - Compression Rate

$$CR\ (\%) = \frac{|N1 - N2|}{N1} \times 100$$

Figure 12. Compression Rate Calculation

Following table shows the analysis and comparisons made on the results obtained.

Algorithm	PC Tree	PP Tree	PS Tree	PR Tree
N1	55	55	55	55
N2	47	32	41	32
CR	14.54%	41.81%	25.45%	41.81%

Figure 13. Results after implementation

Results are further illustrated more clearly using graph.

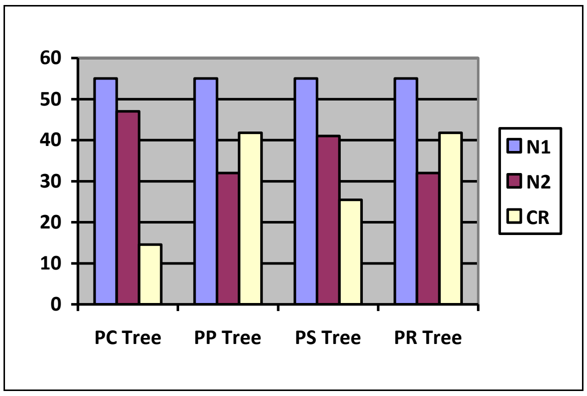


Figure 14. Performance graph of algorithms

Analysis

From the comparison made between PC-tree, PSS Structure, PP Structure & PR tree, following conclusions can be established.

i. Pattern Range (PR) Tree & Prefix-Postfix (PP) Tree achieves maximum compaction of dataset by 41.81%.
ii. PC Tree Showed minimum compaction of 14.54%.
iii. Prefix-Suffix showed compaction of 25.45%.

V. FUTURE WORK

In future, these algorithms can be applied to real world commercial databases and their operational efficiency and compression rate can be analysed to view their success rate.

VI. CONCLUSION

After implementing these four data abstraction algorithms and analysing results of implementation, we would like to conclude that Prefix-Postfix Structure (PPS) does maximum compaction of dataset but it is complex to implement and requires multiple scans of database so it is not efficient to use it on practical scale.

Other algorithms like pattern count tree (PC), prefix-suffix tree (PS) and pattern range tree (PR) are quite simple to implement and provides satisfactory results with a future scope and can be implemented for a large commercial datasets to get compact representation of dataset so as to reduce storage and processing time.

VII. ACKNOWLEDGEMENT

This work is project on “Abstraction Based Data Mining” under guidance of Department of computer Engineering, K.J Somaiya College of Engineering, Vidyavihar, Mumbai.

VIII. REFERENCES

[1] V.S. Ananthanarayana, M.N. Murty*, D Subramanian Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India Received 22 September 2000; accepted 20 October 2000 An incremental data mining algorithm for compact realization of prototypes

[2] Radhika M. Pail and V.S Ananthanarayanaz 1 Manipal Institute of Technology, Manipal 2 National Institute of Technology Karnataka, Surathkal A. Ghosh, R.K. De, and s.|<. Pal (Eda): PReMI 2007, LNCS 4815, pp. 3164323, 2001. © Springer-Verlag Berlin Heidelberg 2007Prefix-Sufiix Trees: A Novel Scheme for Compact Representation of Large Datasets.

[3] 10th International Conference on Information Technology-2013 hreerangP.R.shreerangakamath@gmail.com Dept. of information Technology, Akshat Vigakshatvig19@gmail.com national Institute of Technology Karnataka , Dr. V.S. AN ananathvs1967@gmail.com Surathkal, India .An Efficient Classification Algorithm based on Pattern Range Tree Prototypes.

[4] 1 Geetha M, 2 R. J. D’Souza .1Member IAENG, Department of Computer Science, M.I.T., Manipal, Karnataka, India.2Professor, Department of MACS, N.I.T.K., Surathkal,Karnataka, India An Efficient Reduced Pattern Count TreeMethod for Discovering Most Accurate Set of Frequent itemsets.

[5] V.S. Ananthanarayana, M. Narasimha Murty *, D.K.SubramanianDepartmentofComputerScience andAutomation,Indian Institute of Science, Bangalore 560 012,IIndiaTree structure for efficient data mining using rough sets.