IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-2, Issue-10 E-ISSN: 2347-2693

Comparison on Different Data Mining Algorithms

Aruna J. Chamatkar[1*] and P.K. Butey [2]

[1*] *Research Scholar , Department Of Electronics & Computer Sci., RTM Nagpur University, Nagpur , India*

[2] *HOD, Computer Science Department, Kamla Nehru Mahavidyalaya ,Nagpur India*

www.ijcaonline.org

Received: 22 Sep 2014 Revised: 11 Oct 2014 Accepted: 24 Oct 2014 Published: 31 Oct 2014

Abstract — Data mining an interdisciplinary research area spanning several disciplines such as machine learning, database system, expert system, intelligent information systems and statistic. Data mining has evolved into an active and important area of research because of previously unknown and interesting knowledge from very large real-world database. Many aspects of data mining have been investigated in several related fields. A unique but important aspect of the problem lies in the significance of needs to extend their studies to include the nature of the contents of the real world database. In this paper we are going to compare the three different algorithms which are commonly used in data mining. These three algorithms are CHARM Algorithm, Top K Rules mining and CM SPAM Algorithm.

Keywords - Data Mining, CHARM algorithm, K rule mining, CM SPAM Algorithm

I. INTRODUCTION

In present day human beings are used in the different technologies to adequate in there society. Every day the human beings are using the vast data and these data are in the different fields .It may be in the form of documents, may be graphical formats ,may be the video ,may be records (varying array) .As the data are available in the different formats so that the proper action to be taken for better utilization of the available data. As and when the customer will require the data should be retrieved from the database and make the better decision.

This technique is actually we called as a data mining or Knowledge Hub or simply KDD (Knowledge Discovery Process).The important reason that attracted a great deal of attention in information technology the discovery of useful information from large collections of data industry towards field of “Data mining” is due to the perception of “we are data rich but information poor”. There is very huge amount of data but we hardly able to turn them in to useful information and knowledge for managerial decision making in different fields. To produce information it requires very huge database. It may be available in different formats like audio/video, numbers, text, figures, and Hypertext formats. To take complete advantage of data; the data retrieval is simply not enough, it requires a tool for extraction of the essence of information stored, automatic summarization of data and the discovery of patterns in raw data.

With the enormous amount of data stored in databases, files, and other repositories, it is very important to develop powerful software or tool for analysis and interpretation of such data and for the extraction of interesting knowledge that could help in decision-making. The only answer to all above is ‘Data Mining’. Data mining is the extraction of hidden predictive information from large databases; it is a powerful technology with great potential to help organizations focus on the most important information in their data warehouses [1][2][3][4]. Data mining tools predict behaviors and future trends help organizations and firms to make proactive knowledge-driven decisions [2]. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by prospective tools typical of decision support systems. Data mining tools can answer the questions that traditionally were too time consuming to resolve. They created databases for finding predictive information, finding hidden patterns that experts may miss because it lies outside their expectations.

There are many different algorithm are proposed by the different authors for the data mining. In the paper we are going to study and compare three algorithms of the data mining and they are CHARM algorithm, K rule mining and CM SPAM algorithms.

II. DIFFERENT ALGORITHMS

1. CHARM Algorithm

CHARM is an efficient algorithm for enumerating the set of all frequent closed item-sets. There are a number of innovative ideas employed in the development of CHARM; these include:

1) CHARM simultaneously explores both the item-set space and transaction space, over a novel IT-tree (item set-tides tree) search space of the database. In contrast previous algorithms exploit only the item-set search space.

2) CHARM uses a highly efficient hybrid search method that skips many levels of the IT-tree to quickly identify the

frequent closed item-sets, instead of having to enumerate many possible subsets.

3) It uses a fast hash-based approach to eliminate non-closed item-sets during subsumption checking. CHARM also able to utilize a novel vertical data representation called diffset [5], for fast frequency computations. Diffsets also keep track for differences in the tids of a candidate pattern from its prefix pattern. Diffsets drastically cut down (by orders of magnitude) the size of memory required to store intermediate results. Thus the entire working set of patterns can fit entirely in the memory, even for huge databases.

Several factors make this a realistic assumption. First, CHARM breaks the search space into small independent chunks (based on prefix equivalence classes [6]). Second, diffsets lead to extremely small memory footprint. Finally, CHARM uses simple set difference operations and not requires any complex internal data structures (candidate generation and counting happens in a single step). The current trend toward large (gigabyte-sized) main memories, combined with the above features, makes CHARM a efficient and practical algorithm for reasonably large databases.

The CHARM Algorithm for the data mining is as follows:

```
CHARM (D, min sup):
1. [P] = {Xi × t(Xi) : Xi ∈ I ∧ σ (Xi) ≥ min sup}
2. CHARM-Extend ([P], C = ∅ )
3. return C //all closed sets
CHARM-Extend ([P], C):
4. for each Xi × t(Xi) in [P]
5. [Pi] = ∅ and X = Xi
6. for each Xj × t(Xj) in [P], with Xj ≥ f Xi
7. X = X ∪ Xj and Y = t(Xi) ∩ t(Xj)
8. CHARM-Property([P], [Pi])
9. if ([Pi] = ∅ ) then CHARM-Extend ([Pi], C)
10. delete [Pi]
11. C = C ∪ X //if X is not subsumed

CHARM-Property ([P], [Pi]):
12. if (σ(X) ≥ minsup) then
13. if t(Xi) = t(Xj) then //Property 1
14. Remove Xj from [P]
15. Replace all Xi with X
16. else if t(Xi) ⊂ t(Xj) then //Property 2
17. Replace all Xi with X
18. else if t(Xi) ⊃ t(Xj) then //Property 3
19. Remove Xj from [P]
20. Add X × Y to [Pi] //use ordering f
21. else if t(Xi) = t(Xj) then //Property 4
22. Add X × Y to [Pi] //use ordering f
```

CHARM simultaneously explores both the itemset space and tidset space using the IT-tree, unlike old algorithms which typically exploit only the itemset space. CHARM uses a novel search method, which is based on the IT-pair properties, that skips many levels in the IT-tree to quickly converge on the itemset closures, rather than having to enumerate many possible subsets.

2. Mining Top K Association Rule

Association rule mining [7] consists of discovering associations between items in transportation. It is the most important data mining tasks. It has been integrated in many commercial data mining software and has wide applications in several domains.

The idea of mining top-k association rules presented in this paper is analogous to the idea of mining top-k itemsets [8] and top-k sequential patterns [9] [10] [11] in the field of frequent pattern mining in database. Note that although many authors have previously used the term "top-k association rules", they did not use the actual standard definition of an association rule. KORD [12] [13] only finds rules with a single item in the consequent, whereas the algorithm of You et al. [14] consists of mining association rules from a stream instead of a transaction database.

To achieve this goal, a question is how to combine the concept of top-k pattern mining with association rules? Two thresholds are used For association rule mining. But, in practice *minsup* is much more difficult to set than *minconf* because *minsup* depends on database characteristics that are unknown to most users, whereas *minconf* represents the minimal confidence that users want in rules and is generally easy to determine. For this reason, we define "top-k" on the support rather than the confidence.

The algorithm main idea is the following. Top K Rules first sets an internal *minsup* variable to 0. Then, the algorithm starts searching for rules. As soon as a rule is found, it is added to a list of rules L ordered by the support. The list is used to maintain the top-k rules found until now. Once *k* valid rules are found, the internal *minsup* variable is raised to the support of the rule with the lowest support in L. Raising the *minsup* value is used to prune the search space when searching for more rules. Thereafter, each time a valid rule is found, the rule is inserted in L, the rules in L not respecting *minsup* anymore are removed from L, and *minsup* is raised to the value of the least interesting rule in L. The algorithm continues searching for more rules until no rule are found, which means that it has found the top-k rules.

To search for rules, Top K Rules does not rely on the classical two steps approach to generate rules because it would not be efficient as a top-k algorithm (as explained in the introduction). The strategy used by Top K Rules instead consists of generating rules containing a single item in the antecedent and a single item in the consequent. Then, each rule is recursively grown by adding items to the antecedent or consequent. To select the items that are added to a rule to grow it, it scans the transactions containing the rule to find single items that could expand its right or left part. The name of the two processes for expanding rules in Top K Rules is right expansion and left expansion. These processes can be applied recursively to explore the search space of association rules.

Another idea incorporated in Top K Rules is to try to generate the most promising rules first. This is because if rules with high support are found earlier, Top K Rules can raise its internal *minsup* variable faster to prune the search space. To perform this, Top K Rules uses an internal variable R to store all the rules that can be expanded to have a chance of finding more valid rules. Top K Rules uses this set to determine the rules that are the most likely to produce valid rules with a high support to raise *minsup* more quickly and prune a larger part of the search space.

The Top K Rule algorithm is as follows:

```
TOPKRULES(T, k, minconf) R := Ø. L := Ø. minsup := 0.
1. Scan the database T once to record the tidset of each item.
2. FOR each pairs of items i, j such that $|tids(i)| \times |T| \geq minsup$ and $|tids(j)| \times |T| \geq minsup$:
3. $sup(\{i\}\rightarrow\{j\}) := |tids(i) \cap tids(j)| / |T|$.
4. $sup(\{j\}\rightarrow\{i\}) := |tids(i) \cap tids(j)| / |T|$.
5. $conf(\{i\}\rightarrow\{j\}) := |tids(i) \cap tids(j)| / |tids(i)|$.
6. $conf(\{j\}\rightarrow\{i\}) := |tids(i) \cap tids(j)| / |tids(j)|$.
7. IF $sup(\{i\}\rightarrow\{j\}) \geq$ minsup THEN
8. IF $conf(\{i\}\rightarrow\{j\}) \geq minconf$ THEN SAVE($\{i\}\rightarrow\{j\}$, L, k, minsup).
9. IF $conf(\{j\}\rightarrow\{i\}) \geq minconf$ THEN SAVE($\{j\}\rightarrow\{i\}$, L, k, minsup).
10. Set flag expandLR of $\{i\}\rightarrow\{j\}$ to true.
11. Set flag expandLR of $\{j\}\rightarrow\{i\}$ to true.
12. R := R∪{$\{i\}\rightarrow\{j\}$, $\{j\}\rightarrow\{i\}$}.
13. END IF
14. END FOR
15. WHILE $\exists r \in$ R AND $sup(r) \geq minsup$ DO
16. Select the rule rule having the highest support in R
17. IF rule.expandLR = true THEN
18. EXPAND-L(rule, L, R, k, minsup, minconf).
19. EXPAND-R(rule, L, R, k, minsup, minconf).
20. ELSE EXPAND-R(rule, L, R, k, minsup, minconf).
21. REMOVE rule from R.
22. REMOVE from R all rules $r \in$ R | $sup(r) < minsup$.
23. END WHILE
```

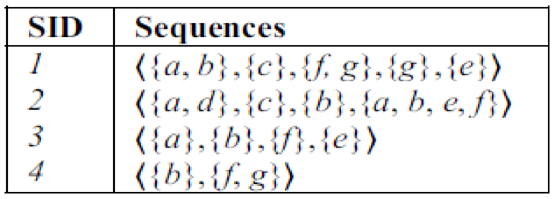
The main procedure of Top K Rules is shown above. The algorithm first scans the database once to calculate $tids(\{c\})$ for each single item c in the database (*line 1*). Then, the algorithm generates all valid rules of size 1*1 by considering each pair of items i, j, where i and j each have at least $minsup \times |T|$ tids (if this condition is not met, clearly, no rule having at least the minimum support can be created with i, j) (line 2). The supports of the rules $\{i\}\rightarrow\{j\}$ and $\{j\}\rightarrow\{i\}$ are simply obtained by dividing $|tids(i\rightarrow j)|$ by $|T|$ and $|tids(j\rightarrow i)|$ by $|T|$ (*line 3 and 4*). The confidence of the rules $\{i\}\rightarrow\{j\}$ and $\{j\}\rightarrow\{i\}$ is obtained by dividing $|tids(i\rightarrow j)|$ by $|tids(i)|$ and $|tids(j\rightarrow i)|$ by $|tids(j)|$ (line *5 and 6*). Then, for each rule $\{i\}\rightarrow\{j\}$ or $\{j\}\rightarrow\{i\}$ that is valid, the procedure SAVE is called with the rule and L as parameters so that the rule is recorded in the set L of the current top-k rules found (line *7 to 9*). Also, each rule $\{i\}\rightarrow\{j\}$ or $\{j\}\rightarrow\{i\}$ that is frequent is added to the set R, to be later considered for expansion and a special flag named *expandLR* is set to true for each such rule (line *10 to 12*).

After that, a loop is performed to recursively select the rule r with the highest support in R such that $sup(r) \geq minsup$ and expand it (line *15 to 23*). The idea is to always expand the rule having the highest support because it is more likely to generate rules having a high support and thus to allow to raise *minsup* more quickly for pruning the search space. When there is no more rule in R with a support higher than *minsup* the loop terminated. For each rule, a flag *expandLR* indicates if the rule should be left and right expanded by calling the procedure EXPAND-L and EXPAND-R or just left expanded by calling EXPAND-L. For all rules of size 1*1, this flag is set to true.

3. CM SPAM Algorithm

Mining useful patterns in sequential data is a challenging task. Many studies have been proposed for mining interesting patterns in sequence databases [15]. Sequential pattern mining is probably the most popular research topic among them. A subsequence is called sequential pattern or frequent sequence if it frequently appears in a sequence database and its frequency is no less than a user-specified minimum support threshold minsup [16]. Sequential pattern mining plays an important role in data mining and is essential to a wide range of applications such as the analysis of web medical data, program executions, click-streams, e-learning data and biological data [15]. Several efficient algorithms have been proposed for sequential data mining and one of them is CM SPAM Algorithm.

Problem with sequential data mining, Let $I = \{i1, i2, ..., il\}$ be a set of items (symbols). An *itemset* $Ix = \{i1, i2, ..., im\} \subseteq I$ is an unordered set of distinct items. The *lexicographical order* $>lex$ is defined as any total order on I. Without loss of generality, it is assumed in the following that all itemsets are ordered according to $>lex$. A *sequence* is an ordered list of itemsets $s = <I1, I2, ..., In>$ such that $Ik \subseteq I$ ($1 \leq k \leq n$). A *sequence database SDB* is a list of sequences $SDB = >s1, s2, ..., sp\_$ having sequence identifiers (SIDs) 1, 2...*p*. Example**.** A sequence database is shown in Fig. 1 (left). It contains four sequences having the SIDs 1, 2, 3 and 4. Each single letter represents an item. Items between curly brackets represent an itemset. The first sequence $<\{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\}>$contains five itemsets. It indicates that items *a* and *b* occurred at the same time, were followed by *c*, then *f*, *g* and lastly *e*.



SID	Sequences
1	$\langle\{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\}\rangle$
2	$\langle\{a, d\}, \{c\}, \{b\}, \{a, b, e, f\}\rangle$
3	$\langle\{a\}, \{b\}, \{f\}, \{e\}\rangle$
4	$\langle\{b\}, \{f, g\}\rangle$

(a)

ID	Pattern	Support
p1	⟨{*a*},{*f*}⟩	3
p2	⟨{*a*},{c} {*f*}⟩	2
p3	⟨{*b*},{f,g}⟩	2
p4	⟨{*g*},{e}⟩	2
p5	⟨{*c*},{f}⟩	2
p6…	⟨{*b*}⟩	4

(b)

Figure.1 A sequence database (a) and some sequential patterns found (b)

The pseudocode of SPAM is shown below. SPAM take as input a sequence database *SDB* and the *minsup* threshold. SPAM first scans the input database *SDB* once to construct the vertical representation of the database *V* (*SDB*) and the set of frequent items *F*1. For each frequent item *s* ∈*F*1, SPAM calls the SEARCH procedure with <*s*>, *F*1, *{e* ∈ *F*1|*e >lex s}*, and *minsup*. The SEARCH procedure outputs the pattern <*{s}*> and recursively explore candidate patterns starting with the prefix <*{s}*>. The SEARCH procedure takes as parameters a sequential pattern *pat* and two sets of items to be appended to *pat* to generate candidates. The first set *Sn* represents items to be appended to *pat* by *s*-extension. The *s*-extension of a sequential pattern <*I*1*, I*2*, ...Ih*> with an item *x* is defined as <*I*1*, I*2*, ...Ih*>*, {x}*>. The second set *Si* represents items to be appended to *pat* by *i*-extension. The *i*-extension of a sequential pattern <*I*1*, I*2*, ...Ih*> with an item *x* is defined as <*I*1*, I*2*, ...Ih* ∪*{x}*>. For each candidate *pat* generated by an extension, SPAM calculate its support to determine if it is frequent. This is done by making a join operation (see [3] for details) and counting the number of sequences where the pattern appears. The IdList representation used by SPAM is based on bitmaps to get faster operations
[3]. If the pattern *pat* is frequent, it is then used in a recursive call to SEARCH to generate patterns starting with the prefix *pat*. Note that in the recursive call, only items that resulted in a frequent pattern by extension of *pat* are considered for extending pat. SPAM prunes the search space by not extending infrequent patterns. This can be done due to the property that an infrequent sequential pattern cannot be extended to form a frequent pattern [1].

```
SPAM(SDB, minsup)
1. Scan SDB to create V(SDB) and identify F1, the list of frequent items.
2. FOR each item s ∈F1,
3. SEARCH(<s>, F1, {e > F1 | e >lex s}, minsup).

SEARCH(pat, Sn, In, minsup)
1. Output pattern pat.
2. Stemp := Itemp := Ø
3. FOR each item j ∈ Sn,
4. IF the s-extension of pat is frequent THEN Stemp := Stemp ∪{i}.
5. FOR each item j ∈ Stemp,
6. SEARCH(the s-extension of pat with j, Stemp , {e ∈ Stemp | e >lex j}, minsup).
7. FOR each item j ∈ In,
8. IF the i-extension of pat is frequent THEN Itemp := Itemp ∪{i}.
9. FOR each item j ∈ Itemp,
10. SEARCH(i-extension of pat with j, Stemp , {e ∈ Itemp | e >lex j}, minsup).
```

III. ALGORITHMS RESULT EVALUATION

In this paper we study different algorithms for the data mining the result evaluation of those algorithms are shown below. This helps to evaluate all the studies algorithm and compare them with each other.

Database name	Item	Avg. Length	Time(s)	Max Pattern (%)
Chess	76	37	20	20
Connect	130	43	40	10
Mushrooms	120	23	9	0.075
Gazelle	498	2.5	10	0.01

Table 1 Performance Evaluations of CHARM

Database name	Item	Avg. Length	Time(s)	Max Pattern (%)
Chess	75	37	8	1.49
Connect	129	43	283	25.51
Mushrooms	128	23	20	3.46
Gazelle	498	2.5	368	46.39

Table 2. Performance Evaluations of Top K rule

Database name	Item	Avg. Length	Time(s)	Max Pattern (%)
Chess	76	37	15	18.81
Connect	130	43	40	12.3
Mushrooms	120	23	60	0.59
Gazelle	498	2.5	80	24.08

Table 3 Performance Evaluations of CM SPAM

IV. CONCLUSION

This paper has attempted to review the three data mining algorithms i.e. CHARM algorithm, K rule mining and CM SPAM algorithms. We also see all the aspects of these three algorithms. We compare the Performance of the all three algorithms and study their results.We observe that the time complexity of Charm algorithm is better as compare to Top

K rule and CM-Spam, While the maximum pattern matching percentages is best for TopK rule.

ACKNOWLEDGMENT

Ms. Aruna J. Chamatkar is MCA from Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur. Currently pursuing PhD from RTM Nagpur University under the guidance of Dr. Pradeep K. Butey. Her research area is Data Mining and Neural Network.

Dr. Pradeep K. Butey is research Supervisor for Computer Science at RTM Nagpur university. He is the Head Of the Department (Computer Science) at Kamla Nehru Mahavidyalaya, Nagpur . His area of interest includes Fuzzy Logic, Neural Network and Data Mining.

REFERENCES

[1] Neelamadhab Padhy, Dr. Pragnyaban Mishra, Rasmita Panigrahi "The Survey of Data Mining Applications And Feature Scope" at International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol.2, No.3, June 2012**.**

[2] Introduction to Data Mining and Knowledge Discovery, Third Edition ISBN: 1-892095-02-5, Two Crows Corporation, 10500 Falls Road, Potomac, MD 20854 (U.S.A.), 1999.

[3] Larose, D. T., "Discovering Knowledge in Data: An Introduction to Data Mining", ISBN 0-471-66657-2, ohn Wiley & Sons, Inc, 2005.

[4] Dunham, M. H., Sridhar S., "Data Mining: Introductory and Advanced Topics", Pearson Education, New Delhi, ISBN: 81-7758-785-4, 1st Edition, 2006

[5] M. J. Zaki and K. Gouda. Fast vertical mining using Diffsets. Technical Report 01-1, Computer Science Dept., Rensselaer Polytechnic Institute, March 2001.

[6] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions* on Knowledge and Data Engineering, 12(3):372-390, May-June 2000.

[7] R. Agrawal, T. Imielminski and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," Proc. ACM Intern. Conf. on Management of Data, ACM Press, June 1993, pp. 207-216.

[8] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-k Closed Sequential Patterns", Knowledge and Information Systems, vol. 7, no. 4, 2005, pp. 438-457.

[9] C. Kun Ta, J.-L. Huang and M.-S. Chen, "Mining Top-k Frequent Patterns in the Presence of the Memory Constraint," VLDB Journal, vol. 17, no. 5, 2008, pp. 1321-1344.

[10] J. Wang, Y. Lu and P. Tzvetkov, "Mining Top-k Frequent Closed Itemsets," IEEE Trans. Knowledge and Data Engineering, vol. 17, no. 5, 2005, pp. 652-664.

[11] A. Pietracaprina and F. Vandin, "Efficient Incremental Mining of Top-k Frequent Closed Itemsets," Proc. Tenth. Intern. Conf. Discovery Science, Oct. 2004, Springer, pp. 275-280.

[12] G. I. Webb and S. Zhang, "k-Optimal-Rule-Discovery," Data Mining and Knowledge Discovery, vol. 10, no. 1, 2005, pp. 39-79.

[13] G. I. Webb, "Filtered top-k association discovery," WIREs Data Mining and Knowledge Discovery, vol.1, 2011, pp. 183-192.

[14] Y. You, J. Zhang, Z. Yang and G. Liu, "Mining Top-k Fault Tolerant Association Rules by Redundant Pattern Disambiguation in Data Streams," Proc. 2010 Intern. Conf. Intelligent Computing and Cognitive Informatics, March 2010, IEEE Press, pp. 470-473.

[15] Mabroukeh, N.R., Ezeife, C.I.: A taxonomy of sequential pattern mining algorithms. ACM Computing Surveys 43(1), 1–41 (2010).

[16] Agrawal, R., Ramakrishnan, S.: Mining sequential patterns. In: Proc. 11th Intern. Conf. Data Engineering, pp. 3–14. IEEE (1995)