Retrieving Frequent Item Sets from Distributed Data Base

Hamideh Hajiabadi, Saeideh Kabiri Rad

Abstract- With fast growing of the network and the data storage, large scale data are rapidly expanded and collected on the physically distributed storage, consequently traditional data mining approaches are not appropriate for information retrieval purpose. Distributed data mining techniques are developed in order to examine distributed data by parallel algorithms. Distributed data mining algorithms based on finding frequent itemsets are widely used for this purpose. The itemsets retrieved are numerous. In this paper proposed a tree based mining approach contributing to user such that reducing number of retrieved itremsets. The algorithm is implemented and the results are demonstrated.

Index Term: Frequent itemsets, Non-Derivable itemsets, Distributed database.

I. INTRODUCTION

ARM algorithms are practiced on a distributed database consisting transactions illustrated by grouped objects. ARM algorithms are aimed to discover objects which are associated with each other. By considering A and B as objects indicating two itemsets, a rule $A \Rightarrow B$ exists if the appearance of A means the appearance of B and it indicates that A and B are associated. With numerous rules can be extracted from database, the rules their frequencies are more than a *minimum support* are notified as frequent rules. The value of minimum *support* is usually determined by user depending to the conditions. As it is discussed earlier it is not possible to centralize raw data due to the high communication costs; hence, the distributed association rule mining which is known as DARM should be used. Run time and communication costs are critical issues which are used to compare DARM algorithms. The DARM algorithm intend to decrease communication cost so that generating global association rules from geographically distributed dataset costs less than combining and centralizing the distributed datasets. Traditional DARM algorithms are based on Apriori algorithm which not meaningfully improve the performance of finding frequent itemsets. There are two major task included in association rule mining approach which are finding frequent itemsets and the generating rules. The first step is more important because finding frequent itemsets automatically leads to generating rules. Association rule mining in centralize site are well researched earlier. [1] large data always are saved in a distributed sites and it is not possible to move data in a center site because of the communication cost and the size of large data. Consequently a distributed data mining algorithm is needed which several researchers studied in which needs some communication time. [9-12]

Manuscript Received on September 2014.
Hamideh Hajiabadi, Birjand University of Technology, Iran.
Saeideh Kabiri Rad, Birjand University of Technology, Iran.

DARM performance is regarded as run time and communication time between distributed sites. Some itemsets which are named derivable in central ARM algorithm, can be derived from another itemsets. In this paper an approach proposed to distinguish between derivable itemsets and non-derivable one in a distributed data base. The paper is organized as follows. Several related works are explained in the next section and the advantages and drawbacks of each are clarified. Section 3 contains the proposed algorithm. The proposed algorithm is examined on some databases and the experimental results are contained in section 4. Conclusion is included in section 5.

II. BACKGROUND

In this section several researches are studied and organized.

Association rule mining in certain database

The idea of finding frequent itemsets was first presented by Agrawal in 1993 named Apriori [1]. In the Apriori algorithm all the transactions which are precisely specified are mined and the frequent itemsets are retrieved. [1]

Count Distribution (CD) algorithm

CD algorithm are based on Apriori algorithm in a distributed database horizontally partitioned. The CD algorithm is performed iteratively. In each iteration the candidate itemsets are selected by applying Apriori algorithm on the frequent itemsets which are retrieved in the previous iteration. The local support are calculated for each itemsets locally an announced to other sites in order to calculate global support. [2,3]

Fast Distributed Mining (FDM) algorithm

FDM algorithm is much like CD, the only difference between them is that in FDM a pruning technique is used in order to minimize the communications between sites. A polling site is selected by several policy and the local supports are sent to the polling site. [4]

Association rule mining in uncertain database

The Apriori algorithm is not work for the probabilistic data bases including uncertain data. The uncertain data bases are included transactions containing items with the probabilistic existence. $P(x, t_i)$ denotes as the probability of x included in t_i. W_1 denotes as the probability of existence x in t_i and W_2 indicates the probability of non existance of x in t_i.

$$W_1 = P(x, t_i) \quad (1)$$

$$W_2 = 1 - P(x, t_i) \quad (2)$$

There are numerous items in transactions of real data bases, if they are independent, the expected support of an itemset X can be written as the equation (3)

$$supp(X) = sum_i(\prod_{x \in X} P(x, t_i)) \quad (3)$$

The itemset X is candidate as frequent itemset if the support of X exceed the

user defined threshold. These kinds of algorithms are known as semi supervised one in which constrains are defined as SQL constrains by user in order to conduct the mining process to retrieve itemsets satisfying constrains. [5,6] for example consider the constrains $\max(X.price) \leq \$30$ in an e-shopping application, the user is interested in findings itemsets X which the sum of prices of each items included in X is less than $30.

III. PROPOSED APPROACH

One of the time consuming step in the distributed association rule mining algorithms, is the frequent itemsets transmission. With reducing in the number of candidate itemsets, a lot of time is savied leading to a better performance. Several paper are worked on discovering relationships between itemsets's support in order to minimize number of candidates [7,8]. Giving I and J are itemsets where $I \subseteq J$, the upper and lower bound of support for an itemset I can be calculated as the equation (4) and (5). $|I - J|$ denotes the number of items included in I minus the number of items included in J.

$$l(I) = \sum_{J \subseteq X \subseteq I} (-1)^{|I-X|+1} \, supp(X)$$

$$if \ |I - J| is \ odd \quad (4)$$

$$u(I) = \sum_{J \subseteq X \subseteq I} (-1)^{|I-X|+1} \, supp(X)$$

$$if \ |I - J| is \ even \quad (5)$$

So the upper and lower bound of the Itemset I can be calculated by the support of the itemset X where $J \subseteq X \subseteq I$. If the lower and upper bound of I are equals, then the following equation can be deduced.

$$supp(I) = l(I) = u(I) \quad (6)$$

These kinds of itemsets are called as derivable itemsets. Consequently in the Apriori algorithm in each iteration the derivable itemsets are eliminated result in minimizing in the number of candidate itemsets. Non derivable itemset is named as NDI.

The proposed algorithm is performed in order to retrieve frequent NDIs in each site.

$NDI = \emptyset, DI = \emptyset, l = 0, u = |D|$
$for \ each \ itemset \ i \in I \ do$
    $compute \ the \ i.supp$
    $compute \ I.u$
    $compute \ I.l$
    $if \ i.u \neq i.l$
        $add \ i \ to \ set \ NDI$
    $else$
        $add \ i \ to \ set \ DI$
end

Fig. 1 Algorithm is Done in Each Site

The set of all non-derivable frequent itemsets are denotes as NDI and the derivable ones as DI. The number of all transaction included in the site is denotes as |D|.

After discovering DI and NDI for each site, they are transmitted to pulling site. The global lower and upper bound of itemset I is evaluated by equation (7) and (8).

$$I.u = I.u_1 + I.u_2 + \cdots + I.u_n \quad (7)$$

$$I.l = I.l_1 + I.l_2 + \cdots + I.l_n \quad (8)$$

The algorithm explained in fig 2 is done in the pulling site and generate the new candidate itemsets.

$CI = \emptyset$
$for \ each \ itemset \ i \in I \ do$
    $if \ i.l = i.u$
        $prune \ i \ from \ NDI \ to \ DI$
    $else$
        $if \ i.supp \geq t$
            $add \ i \ to \ CI$
end

Fig. 2 Algorithm which is Done at the Pulling Site

The candidate frequent itemsets are denotes as *CI*. *t* is a user-defined threshold that the support of candidate itemsets must be upper than *t*.

IV. EXPERIMENTAL RESULT

The proposed algorithm is implemented and evaluated by using real database included in UCI. All experiments are performed on the workstation containing 6 pc with 2.4 Giga Hz CPU and 1GB internal memory. The data set using for this reason is named *mushroom*. The run time of the proposed algorithm is compared with run time of standard CD algorithm and the results are demonstrated in fig 3.

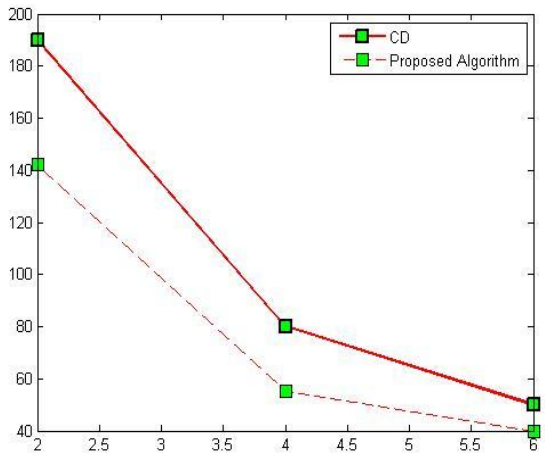

Fig. 3 Comparing Run-Time of Standard CD and Proposed Algorithm

As fig 3 demonstrates the run-time is to some extend improved in the proposed algorithm.

V. CONCLUSION

In this paper a new approach proposed in order to mine distributed data bases and retrieve frequent itemsets. Large data have got different properties which make it more difficult to retrieve information from distributed raw data. The parallel algorithms which are developed for this purpose are contained three major steps which are done iteratively: Discovering local frequent itemsets, transmission to a pulling site and updating candidate frequent itemsets.

In the proposed algorithm the itemsets are divided into derivable itemsets and non-derivable ones. *Support* of the derivable itemsets can be calculated using other itemsets's *support*, consequently derivable itemsets can be eliminated. The proposed approach is implemented and performed on a real life application and compared with standard CD algorithm.

REFERENCES

1. R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the ACM SIGMOD 1993, pp. 207–216.
2. D.W. Cheung, et al.,"A Fast Distributed Algorithm for Mining Association Rules", Proc. Parallel and Distributed Information Systems, IEEE CS Press,1996, pp. 31-42.
3. D.W. Cheung, et al.,"Efficient Mining of Association Rules in Distributed Databases", IEEE Trans. Knowledge and Data Eng., Vol. 8, No. 6, 1996, pp. 911-922.
4. Roger S. Pressman,"Software Engineering", USA, 2005, Lee,"Introduction to System Analysis and Design" , Galgotia Book Source Publications.
5. R.T. Ng, L.V.S. Lakshmanan, J. Han, A. Pang, Exploratory mining and pruning optimizations of constrained associations rules, in: Proceedings of the ACM SIGMOD 1998, pp. 13–24.
6. L.V.S. Lakshmanan, C.K.-S. Leung, R.T. Ng, Efficient dynamic mining of constrained frequent sets, ACM Trans. Database Syst. 28 (4) (2003) 337–389.
7. Calders, T. (2004). Deducing bounds on the support of itemsets. *Database Technologies for Data Mining- Discovering Knowledge with Inductive Queries, Vol. 2682 of LNCS*, pages 214-233. Springer-Verlag.
8. Calders, T. & Goethals, B. (2002). Mining all non derivable frequent itemsets. *Proc. Principles and Practice of Knowledge Discovery in Databases PKDD'02*, Vol. 2431 of *LNAI*, pp. 74-85, Helsinki, FIN, Springer-Verlag.
9. J.J. Cameron, A. Cuzzocrea, F. Jiang, C.K.-S. Leung, Mining frequent itemsets from sparse data streams in limited memory environments, in: Proceedings of the WAIM 2013, Springer, pp. 51–57.
10. E. Kem O.Ozkasap, ProFID: practical frequent items discovery in peer-to-peer networks, Future Gener. Comput. Syst. 29 (6) (2013) 1544–1560.
11. C.K.-S. Leung, C.L. Carmichael, Exploring social networks: a frequent pattern visualization approach, in: Proceedings of the SocialCom 2010, IEEE Computer Society, pp. 419–424.
12. R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the ACM SIGMOD 1993, pp. 207–216.