ISSN:0975-9646

A New Model for Multiparty Collaborative Distributed Mining using Bloom Filter

Deepak Joshy[1], Ms.PrinceMary[2]

[1]*Student,* [2]*Faculty of Computing,*
Sathyabama University, India

Abstract— Collaborative pattern mining is a pattern mining technique used in a distributed environment where each site participate collaboratively with one another for the discovery of patterns. Each individual site is capable of mining patterns and can communicate with other sites in the distributed environment. Currently existing work have failed to tackle various problems using this technique like effective data exchange and cross-site pattern pruning mechanisms. This paper advocates an effective method to perform pattern pruning in a distributive manner with the help of a data structure called Bloom Filter. With its help cross site pattern pruning can be performed with minimized data transfer between the sites as well as increased speed and with the capability of mining Local, Global and Inter-database Patterns.

Keywords— ***Distributed data mining, Bloom Filter, Frequent item set, Distributed associative rule mining***

I. INTRODUCTION

Pattern mining is the process of discovering patterns or relations that are present among data in very large dataset or database. The discovery of the patterns can help to understand the data better and we can make use of the knowledge discovered for future purposes. For many applications data is collected from various distributed sources and these sources may be distributed over different locations [3,4]. The amount of data will be huge and lot of processing and data transfer will be required in order to discover the patterns.

From an association rule mining point of view ,past researches have made significant efforts to discover a variety of patterns such as frequent item-sets, temporal, spatial, and/or sequential association rules, closed patterns or sequential patterns. Common challenges that are faced in this area are: (1) identification of patterns from continuous volume of data or from large databases [13] (L- Patterns) and (2) discovery new patterns by unifying multiple databases into a single view[4,5] (G-pattern mining). In distributed databases, common goal is discover G-patterns, usually done by unifying multiple databases into a single view. Collective data mining [6] is a typical research work done in this area. A common way is find promising local candidates and sending them to a central site for synthesizing [7,8]. In situations where two or more companies/organizations agreed to share data for research for mutual gain but they are not willing to disclose all their sensitive data, so there should be some mechanism to share the essential data without compromising the data security. Hence there should be some efficient way to exchange data effectively along with a mechanism to monitor the data being shared.

Patterns that can be discovered in a distributed environment are of three kind 1) Local Patterns (L-Pattern), 2) Global Patterns (G- Pattern), and 3) Inter-database Patterns (I-Pattern). Local Patterns [9] are those patterns that are local to a certain distributed site. Global patterns are those patterns that are present in all the distributed sites and inter-database patterns are the patterns that are present in more than one site but not all. Local and Global patterns can be found out with ease with already existing method but in-order to find Inter-database patterns no efficient methods currently exists.

II. RELATED WORK

Mining of data in a distributed environment can be done in 3 ways, 1) Sequentially 2) Parallel and 3) Collaboratively. In sequential way, mining is performed in a sequential manner starting from the first site, discovering patterns as we move along to other sites one after the other. When the final site is reached the global patterns will discovered. This method is incapable of finding out inter-database patterns. In the parallel approach, there will a master site which will collect data from all the distributed sites and performs mining and returns the result to the user. The disadvantage with this approach is that there is the need of central site and in case when data mining is carried out for different companies with different requirements, there must be multiple master sites controlled by each company/organization for their own purposes.

In the collaborative approach the distributive sites are capable of finding out patterns by their own. They will exchange message to communicate with other distributed sites. This is an effective technique to find out inter-database patterns and there is no need for a central site. The existing work using this technique is very less and their efficiency is also poor.

The information exchange is done in raw formats and hence the data transfer is more and cause for a bottleneck in the network. The challenges that we face using this method are absence of a system framework capable of mining 3 types of pattern, data exchange method to pass information between the distributed sites, type of mining procedure that should be used and pattern pruning mechanism. This paper proposes a method to exchange pattern information with other distributed sites effectively which is both fast and requires less data transfer also helps to perform pattern pruning between the sites easily.

III. PROPOSED WORK

In collaborative pattern mining[15], pattern discovery is performed in a distributed manner with each distributed site carrying out pattern pruning in collaboration with its peers by using some pattern pruning mechanism. The pattern pruning mechanism this paper proposes is based on a data structure called Bloom Filter [10,11,12]. In this paper, frequent pattern mining[2] is used to perform mining and the using the patterns that are obtained at a distributed site pattern pruning is performed at other sites with the help of Bloom Filter.

Bloom Filter is a probabilistic space efficient data structure that is capable of finding out whether an element is present in a set or not. It contains a m-bit array and k Hash Functions $H_1(.)$, $H_2(.)$ ... $H_k(.)$.The strength of a BF test whether a given element is a member of a set in a very effective way [7,14]. Initially each element in the m-bit array is initialized as 0. Each hash function will take an element as input and the output will be a number between 0 and m indicating the position in the m-bit bit array and the value in the corresponding bit position will be changed to 1.

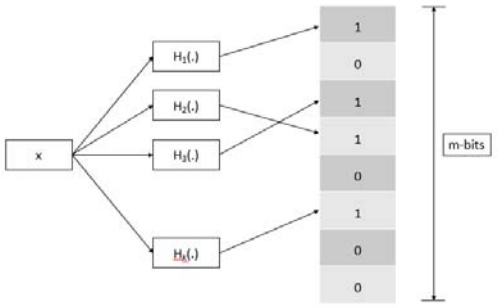

Figure 1 : Bloom Filter Architecture

If we want to add an element ‘x’ to the Bloom Filter, we use the k-Hash functions to hash the element and find k-positions in the array to be made as ‘1’. In order to search for a particular element say ‘y’ in the bloom filter, we again uses the k-hash functions to find k positions and checks in the m-bit array whether these k positions have value ‘1’. If all the k positions have the value ‘1’ then we can say that the element could be present in the set. If at least one of the k position has value as ‘0’, we can conclude that the element ‘y’ is not present in the set. False positives are possible but false negatives will not occur, hence we optimize the size of the bloom filter and the number of hash functions to reduce the false positive rate according to our application.

Another feature of Bloom filters is that there is a clear tradeoff between the size of the filter and the rate of false positives. After the insertion of *n* keys into a boom filter of size *m* making use of *k* hash functions, the probability of a bit being still 0 is :

$$p_0 = \left(1 - \frac{1}{m}\right)^{kn} \approx 1 - e^{-\frac{kn}{m}}. \qquad (1)$$

We assume that we use hash function that spread elements evenly across the space {1-m}.

The probability of a false positive is:

$$p_{err} = (1 - p_0)^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \qquad (2)$$

In (2) p_{err} is minimized for $k = \frac{m}{n} \ln 2$ hash functions. In practice however, only a small number of hash functions are used. The reason is that the computational overhead of each hash additional function is constant while the incremental benefit of adding a new hash function decreases after a certain threshold (see 2) .

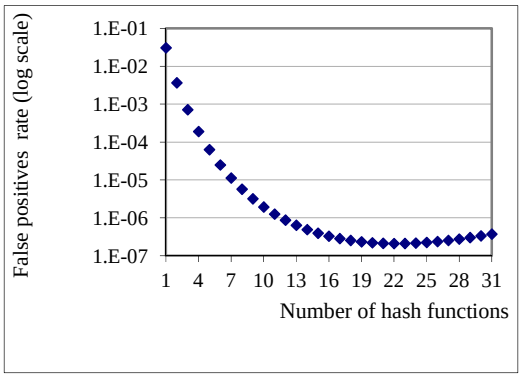

Figure 2: False positive rate as a function of the number of hash functions used. The size of the Bloom filter is 32 bits per entry (m/n=32). In this case using 22 hash functions minimizes the false positive rate. Note however that adding a hash function does

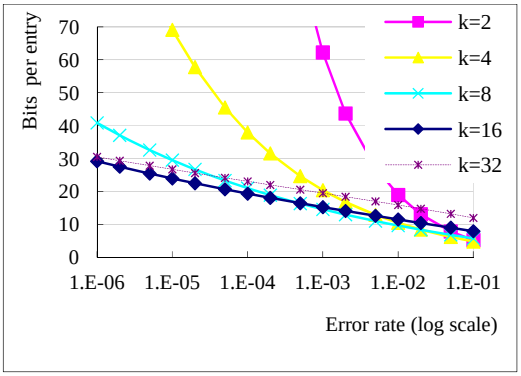

Figure 3: Size of Bloom filter (bits/entry) as a function of the error rate desired. Different lines represent different numbers of hash keys used. Note that, for the error rates considered, using 32 keys does not bring significant benefits over using on

(2) is the base formula for engineering Bloom filters.It allows, for example, computing minimal memory requirements (filter size) and number of hash functions given the maximum acceptable false positives rate and number of elements in the set (as we detail in 3).

$$\frac{m}{n} = \frac{-k}{\ln\left(1 - e^{\frac{\ln p_{err}}{k}}\right)} \text{ (bits per entry)} \quad (3)$$

The user will give a query according to which patterns are to discover. The client machine will distribute the query to each distributed site. And at each site the pattern mining takes place.

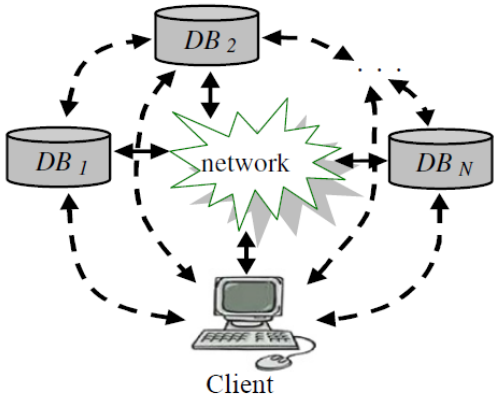

Figure 3: Mining activities are carried out in distributed sites with bloom filter exchanged between sites for cross site pruning

We make use of apriori[2] algorithm to discover the frequent patterns at the distributed site. At a given site the apriori algorithm will generate patterns with the help of data present at that distributed site, after the fining out the length-l patterns, a bloom filter will be created using those patterns and it will be broadcasted to other sites. At the receiving site, using the bloom filter pattern pruning will be carried out eliminating un-wanted patterns. In this way the sites will collaborative with one another in pruning and a finally a global pattern will be discovered at each site. Along with the global patterns relevant inter database patterns are also generated at that site.

1. At site S_i, use the Apriori mining approach to generate a complete set of length-l patterns.

2. Using the frequent length-l patterns in site $Site_i$ construct a bloom filter (BF_i-l), and broadcast BF_i-l to other distributed sites.

3. After a distributed site S_i receives the bloom filters BF_j-l from other sites, it can query BF_j-l and prune out length-l patterns in S_i and then grow length-(l+1) patterns.

4. Set l←l+1 and repeat Steps 2 to 4 until no more frequent patterns can be discovered from any sites

The patterns that are discovered at the distributed sites will be send to the client site where the user can view the patterns. This method is asynchronous and hence each distributed site can work without synchronizing with other sites. Notice that bloom filter cannot encode support values of the patterns.

IV. CONCLUSION

In this paper, our aim is to mine patterns in a distributive environment and to discover Local, Global and Inter-database patterns. We have seen that the current researches focuses only on mining L- and G- Patterns. There is no effective method to mine all three types of patterns. Using the bloom filter technique we can effectively mine three types of patterns (L, G and I-patterns) in a collaborative manner with minimized data transfer between the sites. This mining technique has very little privacy concerns and requires low computational costs and memory consumption. Experimental comparisons demonstrated that this outperforms other simple methods. The problem addressed in this paper mainly focuses on frequent item-set mining. However, this method can be extended to handle other types of patterns as well such as constrained frequent item-sets, closed frequent patterns and sequential patterns.

REFERENCES

[1] R. Agrawal, J.C. Shafer, Parallel mining of association rules, IEEE Transactions on Knowledge and Data Engineering 8 (6) (December 1996) 962–969.

[2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, Proc. of VLDB Conference, 1994.
[3] F. Provost, Distributed data mining: scaling up and beyond. In Kargupta, H., Chan, P., eds.: Advances in Distributed and Parallel Knowledge Discovery, MIT/ AAAI Press, 2000.
[4] S. Zhang, C. Zhang, X. Wu, Knowledge discovery in multiple database, Springer,2004.
[5] M. Ashrafi, D. Taniar, K. Smith, ODAM: an optimized distributed association rule mining algorithm, IEEE Distributed Systems Online 5 (3) (2004).
[6] H. Kargupta, B.H. Park, D. Hershberger, E. Johnson, Collective data mining: a new perspective toward distributed data mining, Advances in Distributed and Parallel Knowledge Discovery, MIT/AAAI Press, Cambridge, MA, 1999.
[7] D. Cheung, V. Ng, A. Fu, Y. Fu, Efficient mining of association rules in distributed databases, IEEE Trans. on Knowledge and Data Engineering 8 (1996).
[8] X. Wu, S. Zhang, Synthesizing high-frequency rules from different data sources, IEEE Transactions on Knowledge and Data Engineering 15 (2) (2003) 353–367.
[9] X. Zhu, R. Jin, Y. Breitbart, G. Agrawal, MMIS-07, 08: mining multiple information sources workshop report, ACM SIGKDD Explorations 10 (2) (2008) 61–65.
[10] A. Border, M. Mitzenmacher, Network applications of bloom filters: a survey, Proc.of the 40th Annual Allerton Conf. on Communication, Control, and Computing, Urbana-Champaign, Illinois, 2002, pp. 636–646.
[11] B. Chazelle, J. Kilian, R. Rubinfeld, A. Tal, The Bloomier filter: an efficient data structure for static support lookup tables, Proc. of the 5th ACM-SIAM Symposium on Discrete Algorithms, 2004, pp. 30–39.
[12] S. Cohen, Y. Matias, Spectral bloom filters, Proc. of SIGMOD Conference, 2003, pp. 241–252.
[13]S. Zhang, M. Zaki, Mining multiple data sources: local pattern analysis, Data Mining and Knowledge Discovery 12 (2–3) (2006) 121–125.
[14] X. Gong, W. Qian, Y. Yan, A. Zhou, Bloom filter-based XML packets filtering for millions of path queries, Proc. of ICDE Conference, 2005, pp. 890–901.
[15] Xingquan Zhu, Bin Li, Xindong Wu, Dan He, Chengqi Zhang, CLAP: Collaborative pattern mining for distributed information systems, Decision Support Systems, Volume 52, Issue 1, December 2011, pp. 40–51