ASSESSMENT ON IMPROVISING FREQUENT ITEMSET MINING IN HUGE VOLUME OF DATASET

Dr. K. Kavitha
Assistant Professor in Computer Science
Mother Teresa Women's University, Kodaikanal

ABSTRACT

Map Reduce programming model is focused for distributed large scale data processing with some constraints and properties. It is frequently used as a classification or clustering method in machine learning or data mining. Distributed data processing framework called Map Reduce proposed to simplify the parallel processing using a distributed computing platform which typically uses client/server communication model. In this process, uses a Parallel Frequent Item Sets mining algorithm for predicting the large amount of datasets using the Map reduce technique. Map reduce model is a vital research area in recent years. The previous research work in this background and the basic concept of the system requirement for this thesis is considered in this paper. This gives an objective, critical summary of published research literature relevant to this research and also summarised the concept of FiDoop. Many Fidoop incorporates the frequent items ultra-metric tree or FIU-tree achieving compressed storage and avoiding the necessity to build conditional pattern bases.

KEYWORDS: Fi-Doop, Itemset, Map reduce, Frequent item, Parallel Mining, Hadoop

[1] INTRODUCTION

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job. The major advantage of MapReduce is
that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change.

This simple scalability is what has attracted many programmers to use the MapReduce model. MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a Map() procedure (method) that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() method that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

The model is a specialization of the split-apply-combine strategy for data analysis. It is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as in their original forms. The key contributions of the MapReduce framework are not the actual map and reduce functions (which, for example, resemble the 1995 Message Passing Interface standard's reduceand scatter operations), but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine. As such, a single-threaded implementation of MapReduce will usually not be faster than a traditional (non-MapReduce) implementation; any gains are usually only seen with multi-threaded implementations. The use of this model is beneficial only when the optimized distributed shuffle operation (which reduces network communication cost) and fault tolerance features of the MapReduce framework come into play. Optimizing the communication cost is essential to a good MapReduce algorithm.

[2] REVIEW OF LITERATURE

2.1 Approximating a Collection of Frequent Sets

Foto Afrati, Aristides Gionis, Heikki Mannila addressed the issue of overwhelmingly large output size by introducing and studying the following problem: What are the k-sets that best approximate a collection of frequent item sets? This work measures the approximate collection of sets by k sets is defined to be the size of the collection, i.e., the part of the collection that is included in one of the k sets.

Traditional parallel algorithms for mining frequent itemsets aim to balance load by equally partitioning data among a group of computing nodes. This work started this study by discovering a performance measure of the existing parallel Frequent Itemset Mining algorithms. Given a large dataset, data partitioning strategies in the existing solutions suffers high communication and mining overhead induced by redundant transactions transmitted among computing nodes. It addressed this problem by developing a data partitioning approach called FiDoop-DP using the MapReduce programming model. The goal of FiDoop-DP is to boost the performance of parallel Frequent Itemset Mining on Hadoop clusters. It exploits correlations among transactions. Incorporating the similarity metric and the Locality-Sensitive Hashing technique, FiDoop-DP places highly similar transactions into a data partition to improve locality without creating an excessive number of redundant transactions.
Experimental result reveals that FiDoop-DP is conducive to reduce network and computing loads by the virtue of eliminating redundant transactions on Hadoop nodes. FiDoop-DP significantly improves the performance of the existing parallel frequent-pattern scheme by up to 31 percent with an average of 18 percent which provides simple polynomial-time approximation algorithms[1].

The collection of frequent patterns can be used in at least two different ways: first, one can be interested in the individual patterns and their occurrence frequencies; second, one can be interested in the whole collection. Even restricting the output to the border of the frequent item-set collection does not help much in alleviating the problem. The work provides better possibility but not an efficient method, since the number of terms in the formula is exponential.

2.2 Risk Assessment for Diabetes Mellitus using Association Rule Mining

Diabetes is part of the growing epidemic of non communicable diseases, with a high burden for the society on developing countries in future. For suppressing the development of diabetes mellitus and the onset of complications to manage their healthcare or personal data. The authors X.Rexeena, B.Suganya Devi, S.Saranya aims to apply association rule mining to electronic medical records to discover sets of risk factors which summarizes the high risk of diabetes. Association rules are implications that associate a set of potentially interacting conditions (e.g. high BMI and the presence of hypertension diagnosis) with elevated risk. The use of association rules is particularly beneficial because in addition to quantifying the diabetes risk, also readily provide the physician with a “justification”, namely the associated set of conditions. This set of conditions can be used to guide treatment towards a more personalized and targeted preventive care or diabetes management. A clinical application of association rule mining to identify sets of comorbid conditions that imply significantly increased risk of diabetes. Association rule mining on this extensive set of variables resulted in an exponentially large set of association rules. The main contribution is a comparative evaluation of these extended summarization techniques that provides guidance to practitioners in selecting an appropriate algorithm for a similar problem[2].

Association rule mining identifies the sets of risk factors and the corresponding patient subpopulations who are at significantly increased risk of progressing to diabetes. An excessive number of association rules were discovered impeding the clinical interpretation of the results. The number of rules is used for clinical interpretation is make feasible. Many of these rules are slight variants of each other leading to the obfuscation of the clinical patterns underlying the ruleset. One remedy to this problem, which constitutes the main focus of this work, is to summarize the ruleset into a smaller set that is easier to overview.

Author first reviewed the existing rule set and database summarization methods, then proposed a generic framework that these methods fit into and finally extend these methods to take a continuous outcome variable (the martingale residual in our case) into account. This system have a significantly higher chance of progression to diabetes than the patients who are either not hypertensive. In association rule mining, items do not play particular roles: there are no designated predictor variables or outcome variables.

2.3 Summarizing Itemset Patterns Using Probabilistic Models

The summarization proceeds in a level-wise iterative fashion. Occurrence statistics of itemsets at the lowest level are used to construct an initial MRF. Statistics of itemsets at the next level can then be inferred from the model. ChaoWang and Srinivasan Parthasarath used those patterns whose occurrence cannot be accurately inferred from the model to augment the model in an iterative manner,
repeating the procedure until all frequent itemsets can be modeled. The resulting MRF model concise and useful representation of the original collection of item-sets. Extensive empirical study on real datasets showed that the new approach can effectively summarize a large number of itemsets and significantly out performs extant approaches. The experimental results on several real datasets showed that the proposed approach compares favorably with role patterns on the axes of accuracy, space and performance. Specifically, It finds on real datasets which is much more accurate given the same space budget, and more often than not significantly faster than profile patterns[3].

The idea of using frequent itemsets to construct an MRF was first proposed by Pavlov et al. A k-itemset and its support represent a k–way statistic and can be viewed as a constraint on the true underlying distribution that generates the data. Given a set of itemset constraints, a maximum entropy distribution satisfying all these constraints is selected as the estimate for the true underlying distribution. The restoration error of all frequent patterns is slightly higher than that of non-derivable patterns. It is worth pointing out that the restoration error on all frequent itemsets is also very small, supporting claim that non-derivable patterns play a key role in representing the whole collection of frequent itemsets. Occurrence statistics of itemsets at the lowest level are used to construct an initial MRF. The Patterns used in this strategy whose occurrence cannot be accurately inferred from the model to augment the model in an iterative manner, repeating the procedure until all frequent itemsets can be modeled.

2.4 Extracting Redundancy-Aware Top-K Patterns

The significance is usually defined by the context of applications. Previous studies have been concentrating on how to compute top-k significant patterns or how to remove redundancy among patterns separately. There is limited work on finding those top-k patterns which demonstrate high-significance and low-redundancy simultaneously. Finding top-k patterns demonstrate high significance and low redundancy simultaneously which is limited. Dong Xin Hong Cheng Xifeng Yan Jiawei Han studied the problem of extracting redundancy-aware top-k patterns from a large collection of frequent patterns. It first examines the evaluation function for measuring the combined significance of a pattern set and propose the MMS(Maximal Marginal Significance) as the problem formulation. The goal is to extract the frequent patterns of term occurrence, called themes, buried in a large set of documents. Given a document set, the top-k frequent patterns returned by a mining algorithm are not necessarily the best k themes one can find. Many frequent term sets could overlap significantly with each other. Such overlapping may render top-k important themes very redundant[4].

In this paper, it formulates the redundancy-aware top-k pattern extraction problem through a general ranking model which integrates two measures, significance and redundancy, into one objective function. The MMS problem is equivalent to searching a constrained rooted minimum spanning tree on the directed redundancy graph such that the overall weights on the root node and on the edges in the tree are maximized. The constraint specifies that the root must be the most significant pattern in the tree.

To extract redundancy-aware top-k patterns, we examined two problem formulations: MAS and MMS. We studied a unified greedy approach to compare these two functions and show that MMS is a reasonable formulation to our problem. It also further present an improved algorithm for MMS and show that the performance is bounded by O(logk). Both MMS algorithms are able to find high-significant and low-redundant top-k patterns. This study opens a new direction on
finding both diverse and significant top-k answers to querying, searching, and mining, which may lead
to promising further studies.

This method can be used to get the redundancy-aware top-k ranking as well as commonly used
objective measures include support, confidence, lift, coherence, and tf-idf for text patterns and attribute
values for database tuples. This method leads frequency restoration error and does not provide
redundancy aware pattern extraction.

2.5 Mining Compressed Frequent-Pattern Sets

ChaoWang and Srinivasan Parthasarath[5] studied the problem of compressing frequent-
pattern sets. Typically, frequent patterns can be clustered with a tightness measure and a representative
pattern can be selected for each cluster. Unfortunately, finding a minimum set of representative patterns
is NP-Hard. They developed two greedy methods, RP-global and RP-local. The former has the
guaranteed compression bound but higher computational complexity. The latter sacrifices the
theoretical bounds but is far more efficient. The performance study shows that the compression quality
using RP-local is very close to RP-global, and both can reduce the number of closed frequent patterns
by almost two orders of magnitude. Furthermore, RP-local mines even faster than FP-Close, a very fast
closed frequent-pattern mining method. It showed that RP-global and RP-local can be combined
together to balance the quality and efficiency. The objective of the clustering is to minimize the number
of clusters (hence the number of representative patterns). Finally, show the problem is equivalent to
set-covering problem, and it is NP-hard w.r.t. the number of the frequent patterns to be compressed.
The real bottleneck of the problem is not efficient but usability. The closed itemsets cannot get any
compression on this subset.

2.6 Improving Mining of Frequent Itemsets

The Apriori algorithm is a classic way of mining frequent item sets in a database. A variety of
Apriori algorithms aims to shorten database scanning time by reducing candidate item sets. For
example, Park et al. proposed the direct hashing and pruning algorithm to control the number of can-
didate two-item sets and prune the database size using a hash technique. In the inverted hashing and
pruning algorithm every k-item set within each transaction is hashed into a hash table. Bezel et al.
designed the tree-based association rule algorithm, which employs an effective data-tree structure to
store all item sets to reduce the time required for scanning databases.

To improve the performance of Apriori algorithms, Han et al. proposed a novel approach
called FP-growth to avoid generating an excessive number of candidate itemsets[6]. The main idea of
FP-growth is projecting database into a compact data structure, and then using the divide-and-conquer
method to extract frequent item sets. The main bottlenecks of FP-growth are: 1) the construction of a
large number of conditional FP trees residing in the main memory and 2) the recursive traverse of FP
trees. To address this problem, stay et al. proposed a new method called FIUT, which relies on frequent
items ultra-metric trees to avoid recursively traversing FP trees.

Zhang et al. proposed a concept of constrained frequent pattern trees to substantially improve
the efficiency of mining association rules[7]. Parallel frequent item sets mining algorithms based on
Apriori can be classified into two camps, namely, count distribution (e.g., count distribution (CD)
fast parallel mining, and parallel data mining (PDM) and data distribution (e.g., data distribution (DD)
and intelligent data distribution). In the count distribution camp, each processor of a parallel system
calculates the local support counts of all candidate item sets. Then, all processors compute the total
support counts of the candidates by exchanging the local support counts. The CD and PDM algorithms have simple communication patterns, because in every iteration each processor requires only one round of communication. In the data distribution camp, each processor only keeps the support counts of a subset of all candidates. Each processor is responsible for sending its local database partition to all the other processors to compute support counts. In general, DD has higher communication overhead than CD, because shipping transaction data demands more communication bandwidth than sending support counts. The cascade running mode in existing Apriori-based parallel mining algorithms leads to high communication and synchronization overheads.

To reduce time required for scanning databases and exchanging candidate itemsets, FP-growth-based parallel algorithms were proposed as a replacement of the Apriori-based parallel algorithms. A few parallel FP-growth-based parallel algorithms were implemented using multithreading on multicore processors. A major limitation of these parallel mining algorithms lies in the infeasibility to construct main memory based FP trees when databases are very large. This problem becomes pronounced when it comes to massive and multidimensional databases.

[3] CONCLUSION

Traditional parallel algorithms for mining frequent itemset aim to balance load by equally partitioning data among a group of computing nodes. This study discovered performance measure of the existing parallel Frequent Itemset Mining algorithms. Given a large dataset, data partitioning strategies in the existing solutions suffer high communication and mining overhead induced by redundant transactions transmitted among computing nodes. This paper addressed this problem by developing a data partitioning approach called FiDoop-DP using the MapReduce programming model. The overarching goal of FiDoop-DP is to boost the performance of parallel Frequent Itemset Mining on Hadoop clusters. Incorporating the similarity metric and the Locality-Sensitive Hashing technique, FiDoop-DP places high similar transactions into a data partition to improve locality without creating an excessive number of redundant transactions.

REFERENCE

4. Dong Xin Hong Cheng Xifeng Yan Jiawei Han, Extracting Redundancy-Aware Top-K Patterns KDD’06,
August 20–23, 2006, Philadelphia, Pennsylvania, USA.

5. ChaoWang and Srinivasan Parthasarath, Mining Compressed Frequent-Pattern Sets, Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005


