Fast and Parallel Mining of K High Utility Item Set

R.Kanimozhi, HOD in Computer Applications, Idhaya College for Women, Kumbakonam

Dr.K.Saravanan, Dean, Faculty of Computer Science, PRIST UNIVERSITY, Vallam, Thanjavur.

Abstract----A large number of contributions in the literature have been proposed for item set mining, exploring various measures according to the chosen relevance criteria. However, items are actually different in many aspects in a number of real applications, such as retail marketing, network log, etc. The difference between items makes a strong impact on the decision making in these applications. Therefore, traditional ARM cannot meet the demands arising from these applications. By considering the different values of individual items as utilities, parallel mining focuses on identifying the itemsets with high utilities.The parallel mining of high utility itemsets will take very less time than mining with the single system over large number of transactions. The most studied measure is probably the number of frequent item sets processed in import and export business process. While the problem has been widely studied, only few solutions scale. This is particularly the case when i) the data set is massive, calling for large-scale distribution, ii) the length k of the informative item set to be discovered is high and/or iii) the data are dynamic. In this paper, we address the problem of parallel mining of large informative k-High Utility item sets (liki) based on joint entropy. We propose advanced FPHIKS (Fast Parallel Highly Informative k-High Utility item sets) a highly scalable, parallel liki mining algorithm and forward selection algorithm. FPHIKS renders the mining process of large scale databases (up to double or treble terabytes of data) succinct and effective. Its mining process is made up of only two efficient parallel jobs. With FPHIKS, we provide a set of significant optimizations for calculating the joint entropies of liki having different sizes, which drastically reduces the execution time of the mining process.

Index Terms— Data Mining, Fast Algorithm, Association Rule, Business Process, K-Item Set, Big Data, Utility mining, high utility itemset mining.

1 INTRODUCTION

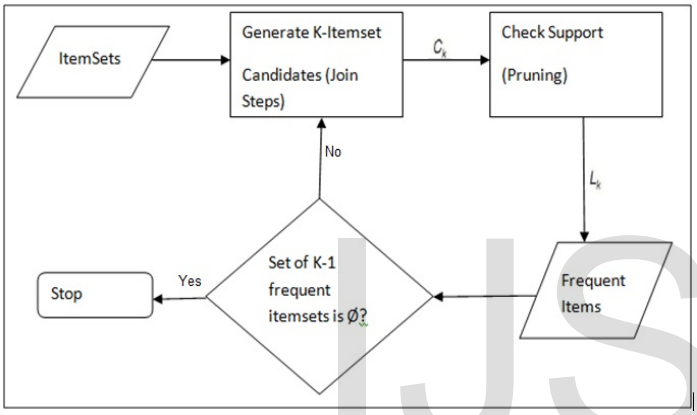
Feature set, or itemset, mining [1] is one of the fundamental building bricks for exploring informative patterns in databases. Features might be, for instance, the words occurring in a document, the score given by a user to a movie on a social network, or the characteristics of plants (growth, genotype, humidity, biomass, etc.) in a scientific study in agronomics. However, frequency does not give relevant results for a various range of applications, including information retrieval [3], since it does not give a complete overview of the hidden correlations between the itemsets in the database. This is particularly the case when the database is sparse [4]. Using other criteria to assess the informativeness of an itemset could result in discovering interesting new patterns that were not previously known. To this end, information theory [5] gives us strong supports for measuring the informativeness of itemsets. One of the most popular measures is the joint entropy of an itemset. An itemset X that has higher joint entropy brings up more information about the objects in the database.

For more efficiency, we provide PHIKS with optimizations that allow for very significant improvements of the whole process of liki mining. The first technique estimates the upper bound of a given set of candidates and allows for a dramatic reduction of data communications, by filtering unpromising itemsets without having to perform any additional scan over the data. The second technique reduces significantly the number of scans over the input database of each mapper, i.e., only one scan per step, by incrementally computing the joint entropy of candidate features. This reduces drastically the work that should be done by the mappers, and thereby the total execution time.

2 BACKGROUNDS

Liki Discovery in a Centralized Environment an effective approach is proposed for liki discovery in a centralized environment. Their Forward Selection heuristic uses a "generating-

pruning" approach, which is similar to the principle of Apriori. $i1$, the feature having the highest entropy is selected as a seed. Then, $i1$ is combined with all the remaining features, in order to build candidates. In other words, there will be $|\mathcal{F}-1|$ candidates (i.e., ($i1$, $i2$),($i1$, $i3$),...,($i1$, $i|\mathcal{F}-1|$)). The entropy of each candidate is given by a scan over the database, and the candidate having the highest entropy, say ($i1$, $i2$), is kept. A set of $|\mathcal{F}-2|$ candidates of size 3 is generated (i.e., ($i1$, $i2$, $i3$),($i1$, $i2$, $i4$),...,($i1$, $i2$, $i|\mathcal{F}-2|$)) and their entropy is given by a new scan over the database. This process is repeated until the size of the extracted itemset is k.



Such an inadequacy calls for new distributed algorithmic principles. To the best of our knowledge, there is no previous work on distributed mining of liki. However, we may build on top of cutting edge studies in frequent itemset mining, while considering the very demanding characteristics of liki. Interestingly, in the case of frequent itemsets in MapReduce, a mere algorithm consisting of two jobs outperforms most existing solutions by using the principle of SON, a divide and conquer algorithm. Unfortunately, despite its similarities with frequent itemset mining, the discovery of liki is much more challenging. Indeed, the number of occurrences of an itemset X in a database $\mathcal{D}$ is additive and can be easily distributed (the global number of occurrences of X is simply the sum of its local numbers of occurrences on subsets of $\mathcal{D}$). Entropy is much more combinatorial since it is based on the projection counting of X in $\mathcal{D}$ and calls for efficient algorithmic advances, deeply combined with the principles of distributed environments.

3 PHIKS ALGORITHM

However, given the "generating-pruning" principle of this heuristic, it is not suited for environments like Spark or MapReduce and would lead to very bad performances. The main reason is that each scan over the data set is done through a distributed job (i.e., there will be k jobs, one for each generation of candidates that must be tested over the database). Our experiments, in Section V, give an illustration of the catastrophic response times of ForwardSelection in a straightforward implementation on MapReduce (the worst, for all of our settings). This is not surprising since most algorithms designed for a centralized itemset mining do not perform well in massively distributed environments in a direct implementation and liki don't escape that rule.

3.1 DISTRIBUTED PROJECTION COUNTING

Its need to provide tools for computing the projection of an itemset X on a database $\mathcal{D}$, when $\mathcal{D}$ is divided into subsets on different splits, in a distributed environment, and entropy has to be encoded in the key-value format. We have to count, for each projection p of X, its number of occurrences on $\mathcal{D}$. This can be solved with an association of the itemset as a key and the projection as a value. On a split, for each projection of an itemset X, X is sent to the reducer as the key coupled with its projection. The reducer then counts the number of occurrences, on all the splits, of each (key value) couple and is therefore able to calculate the entropy of each itemset. Communications may be optimized by avoiding to emit a : val couple when the projection does not appear in the transaction and is only made of '0' (on the reducer, the number of times that a projection p of X does not appear in $\mathcal{D}$ is determined by subtracting the number projections of X in D from $|\mathcal{D}|$).

3.2 DISCOVERING LIKI IN TWO ROUNDS

Our heuristic will use at most two MapReduce jobs in order to discover the *k*-itemset having the highest entropy. The goal of the first job is to extract locally, on the distributed subsets of $\mathcal{D}$, a set of candidate itemsets that are likely to have a high global entropy. To that end, we apply the principle of Forward Selection locally, on each mapper, and grow an itemset by adding a new feature at each step. After the last scan, for each candidate itemset *X* of size *k* we have the projection counting of *X* on the local data set.

4 FORWARD BACKWARD ALGORITHMS

The term *forward–backward algorithm* is also used to refer to any algorithm belonging to the general class of algorithms that operate on sequence models in a forward–backward manner. In this sense, the descriptions in the remainder of this article refer but to one specific instance of this class. forward–backward algorithm computes a set of forward probabilities which provide, for all $k \in \{1, \ldots, t\}$, the probability of ending up in any particular state given the first k observations in the sequence, i.e. $P(X_k \mid o_{1:k})$. In the second pass, the algorithm computes a set of backward probabilities which provide the probability of observing the remaining observations given any starting point k, i.e. $P(o_{k+1:t} \mid X_k)$. These two sets of probability distributions can then be combined to obtain the distribution over states at any specific point in time given the entire observation sequence.

The forward and backward steps may also be called "forward message pass" and "backward message pass" - these terms are due to the *message-passing* used in general belief propagation approaches. At each single observation in the sequence, probabilities to be used for calculations at the next observation are computed. The smoothing step can be calculated simultaneously during the backward pass. This step allows the algorithm to take into account any past observations of output for computing more accurate results.

The problem of extracting informative itemsets was not only proposed for mining static databases. There have been also interesting works in extracting informative itemsets in data streams. The authors of [8] proposed an efficient method for discovering maximally informative itemsets (i.e., highly informative itemsets) from data streams based on sliding window. Parallel mining of informative itemsets from large databases based on frequency informativeness measure has received much attention recently.

Forward(guessState, sequenceIndex):

ifsequenceIndex is past the end of the sequence, return 1

if (guessState, sequenceIndex) has been seen before, return saved result

result = 0

for each neighboring state n:

result = result + (transition probability from guessState to

n given observation element at sequenceIndex)

* Backward(n, sequenceIndex+1)

save result for (guessState, sequenceIndex)

return result

However, and to the best of our knowledge, there has been no prior work on parallel discovery of maximally informative *k*-itemsets from massive, distributed, databases.

5 CONCLUSIONS

In this paper, we proposed a reliable and efficient parallel maximally informative *k*-itemset algorithm namely PHIKS, that has shown significant efficiency in terms of runtime and scalability. PHIKS elegantly determines liki in very large databases with at most two rounds. With PHIKS, we propose a bunch of optimizing techniques that renders the liki mining process very fast. These techniques concern the architecture at a global scale, but also the computation of entropy on distributed nodes, at a local scale. The result is a fast and efficient discovery of liki with high itemset size. Such ability to use high itemset size is mandatory when dealing with Big Data.

ACKNOWLEDGMENT

We would like to thank everyone who has participated in the evaluation of the prototype system. The authors aregrateful for the constructive comments of the three referees on an earlier version of this article. This research was supported in different part by current business process at different enterprises.

REFERENCES

[1] J. Han, Data mining : concepts and techniques. Elsevier/Morgan Kaufmann, 2012.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in proceedings of International Conference on very Large Data Bases (VLDB), 1994, pp.

[3] E. Greengrass, "Information retrieval: A survey," 2000.

[4] H. Heikinheimo, E. Hinkkanen, H. Mannila, T. Mielikäinen, and J. K. Seppänen, "Finding low-entropy sets and trees from binary data," in Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2007, pp. 350–359.

[5] T. M. Cover, Elements of information theory. Hoboken, N.J: WileyInterscience, 2006.

[6] P. Fournier-Viger, C. Wu, and V. S. Tseng, "Mining top-k association rules," in Proc. Int. Conf. Can. Conf. Adv. Artif. Intell., 2012, pp. 61–73.

[7] P. Fournier-Viger, C. Wu, and V. S. Tseng, "Novel concise representations of high utility itemsets using generator patterns," in Proc. Int. Conf. Adv. Data Mining Appl. Lecture Notes Comput.Sci., 2014, vol. 8933, pp. 30–43.

[8] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in Proc. ACM SIGMOD Int. Conf. Manag. Data, 2000, pp. 1–12.

[9] J. Han, J. Wang, Y. Lu, and P. Tzvetkov, "Mining top-K frequent closed patterns without minimum support," in Proc. IEEE Int. Conf. Data Mining, 2002,

[10] S. Krishnamoorthy, "Pruning strategies for mining high utility itemsets," Expert Syst. Appl., vol. 42, no. 5, pp. 2371–2381, 2015.

[11] C. Lin, T. Hong, G. Lan, J. Wong, and W. Lin, "Efficient updating of discovered high-utility itemsets for transaction deletion in dynamic databases," Adv. Eng. Informat., vol. 29, no. 1, pp. 16–27, 2015.