SURVEY ON FREQUENT PATTERN MINING

Keshav Lodhi*

Abstract

In this paper we present the various elementary traversal approaches for mining frequent pattern to find out association rules. We start with a formal definition of association rule and its basic algorithm. We then discuss the association rule mining algorithms from several perspectives such as breadth first approach, depth first approach and Hybrid approach. Frequent pattern mining has been focused themes in data mining re-search for over a decade. Abundant literature has been dedicated to this research and tremendous progress has been made, ranging from efficient and scalable algorithms for frequent itemset mining in transaction databases to numerous research frontiers, such as sequential pattern mining, structured pattern mining, correlation mining , associative classification, and frequent pattern-based clustering .

Keyword- association rules,frequent pattern mining,breadth first,defth first,hybrid approach.

* S.A.T.I. (Vidisha)

1. Introduction-

Frequent itemsets play an essential role in many data mining tasks that try to find interesting patterns from databases, such as association rules, correlations, sequences, episodes, classifiers, clusters and many more of which the mining of association rules is one of the most popular problems. The original motivation for searching association rules came from the need to analyze so called supermarket transaction data, that is, to examine customer behavior in terms of the purchased products. Association rules describe how often items are purchased together.in fig.1 we capture the market-basket scene.

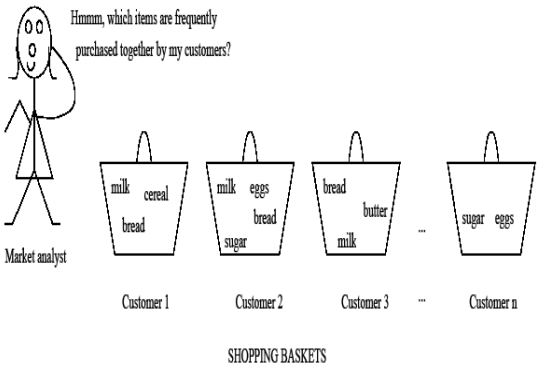

Fig.1

For example, an association rule "beer => chips (80%)" states that four out of five customers that bought beer also bought chips. Such rules can be useful for decisions concerning product pricing, promotions, store layout and many others. Since their introduction in 1993 by Argawal , the frequent itemset and association rule mining problems have received a great deal of attention. Within the past decade, hundreds of research papers have been published presenting new algorithms or improvements on existing algorithms to solve these mining problems more efficiently.

Association rule mining is as important research subject put forward by Agrawal in reference [1]. Association rule mining techniques can be used to discover unknown or hidden correlation items found in the database of transactions. The pattern of mining association rule could be decomposed into two sub problems. First, mining of frequent itemsets/patterns and generating association rule from frequent patterns is next . Finding frequent patterns becomes the main work of mining association rules [2]. In this chapter, we explain the basic frequent itemset and association rule mining problems. We describe the main techniques used to solve these problems

and give a comprehensive survey of the most influential algorithms that were proposed during the last decade.

2. Problem Disccription- Following the original definition by Agrawal et al. the problem of association rule mining is defined as: Let be a set of binary attributes called items. Let be a set of transactions called the database. Each transaction in has a unique transaction ID and contains a subset of the items in. A rule is defined as an implication of the form where the sets of items (for short itemsets) and are called antecedent (left-hand-side or LHS) and consequent (right-hand-side or RHS) of the rule respectively.

For example-The rule found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, he or she is likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as, promotional pricing or product placements. In addition to the above example from market basket analysis association rules are employed today in many application areas including Web usage mining, intrusion detection, Continuous production and bioinformatics. As opposed to sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.Means there is no restriction on data set.In one transaction there can be only two or three

Items and in another one can have any number items.

Transaction Id	MILK	Bread	Butter	Beer
1	1	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

Table 1

To illustrate the concepts, we use a small example from the supermarket domain. The set of items is and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table to the right. An example rule for the supermarket could be meaning that if butter and bread are bought, customers also buy milk.

Note: this example is extremely small. In practical applications, a rule needs a support of several hundred transactions before it can be considered statistically significant, and datasets often contain thousands or millions of transactions. To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best-known constraints are minimum thresholds on support and confidence.

- The support of an itemset is defined as the proportion of transactions in the data set which contain the itemset. In the example database, the itemset has a support of since it occurs in 20% of all transactions (1 out of 5 transactions).
- The confidence of a rule is defined. For example, the rule has a confidence of in the database, which means that for 50% of the transactions containing milk and bread the rule is correct (50% of the times a customer buys milk and bread, butter is bought as well). Be careful when reading the expression: here supp(X∪Y) means "support for occurrences of transactions where X and Y both appear", not "support for occurrences of transactions where either X or Y appears", the latter interpretation arising because set union is equivalent to logical disjunction. The argument of is a set of preconditions, and thus becomes more restrictive as it grows (instead of more inclusive).
- Confidence can be interpreted as an estimate of the probability, the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.
- The lift of a rule is defined as or the ratio of the observed support to that expected if X and Y were independent. The rule has a lift of .

  The conviction of a rule is defined as . The rule has a conviction of, and can be interpreted as the ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions. In this example, the conviction value of 1.2 shows that the rule would be incorrect 20% more often (1.2 times as often) if the association between X and Y was purely random chance.

2. Scope for scalability in mining frequent itemset-

There are three main areas which can be focused upon to find the association rules efficiently. By efficient we mean to reduce time complexity, incur less I/O overhead as well as to efficiently use the storage space. Due to large amount of data available in data warehouses, we should be able to mine data that is scalable i.e. that can grow in size.

As mentioned above, finding frequent itemset is a time consuming process. So the first area of focus should be fast generation and pruning of candidate itemset. Thus by speeding up this step we will enhance the generation of frequent itemset. There are various approaches which have been followed such as depth first search, breadth first search, hybrid, partitioning the database as well as sampling. A large number of increasingly efficient algorithms to mine frequent itemset have been developed over the years [7].

Another area of focus is the data structure which has been used for storing the intermediate candidate itemset in the first step. It should be such that the data which is stored there can be retrieved fast. Several data structures have been used for mining frequent itemset. They can be divided into two main categories as array-based and tree-based. In array-based representation, transactions are stored as arrays of items in the memory. For example, H-Mine uses an array structure called Hstruct [4]. In tree-based representation, variants of prefix trees are used. A prefix tree can be used to organize the transactions by grouping and storing transaction data in memory. In [6], transactions are first grouped using a compressed prefix tree and mapped to an array-based data structure, which is then used for the mining process.

The third issue on which the research is going on is parallel mining. Parallelism can be implemented at various stages. However synchronization can be problem while applying parallelism. But as finding frequent itemset is most expensive, since the number of itemset grows exponentially with the number of items, time complexity can be greatly reduced if parallelism is applied while finding frequent itemset. In this paper we mainly concentrate on the first issue i.e. quickly finding frequent itemset.

In this paper we elaborate on the various traversal approaches which can be used for finding the candidate itemset. However, for mining the frequent itemset efficiently, we have to use a combination of effective traversal scheme, an effective but simple data structure as well as concepts of parallel mining.

3. Commonly used association rule mining algorithms

3.1Breadth First Approach

In Breadth First or bottom-up approach, the computation starts from frequent 1-itemsets (the minimum length frequent itemset) and continues until all maximal (length) frequent itemset are found. Because of its way of working, it is also known as hierarchical algorithm. During the execution, every frequent itemset is explicitly considered. Such algorithms perform well when all

maximal frequent itemset are short. However, performance drastically decreases when some of the maximal frequent itemset are relatively long. The classical Apriori algorithm proposed by Agrawal et al in 1993[8], its enhancement AprioriTid algorithm [9] are all examples of breadth first approach. Apart from this there are also hash based techniques such as DHP algorithm(Direct hashing and pruning) proposed by Park et al in 1995[10] and Tree projection algorithm[11].The partition algorithm[12] and DIC(Dynamic Itemset counting)[13] are also categorized as breadth first algorithms. The Apriori and AprioriTid algorithms generate the candidate itemset to be counted in a pass by using only the itemset found large in the previous pass {without considering the transactions in the database.} The basic concept behind this is that any subset of a large itemset must be large. The Apriori [9] algorithm has the additional property that the database is not used at all for counting the support of candidate itemset after the first pass. Rather, an encoding of the candidate itemset used in the previous pass is employed for this purpose. AprioriTid has the enhanced feature that it replaces a pass over the original dataset by a pass over the candidate set. Another algorithm DHP [10] is a hash based algorithm and is especially effective for generating candidate 2-itemsets, where the number of candidate 2-itemsets in terms of magnitude is smaller than the previous algorithms, thus resolving the performance bottleneck. The DHP algorithm (Park et al, 1995) tries to reduce the number of candidates by collecting approximate counts in the previous level. Like Apriori it requires as many database passes as the longest itemset. It also trims the transaction database at a much earlier stage of the iteration, thereby reducing the computational cost for later stages significantly. The Tree projection algorithm[11] proposed by Agrawal et al in 2001 represents frequent patterns as nodes of a lexicographic tree and uses the hierarchical structure of the lexicographic tree to successively project transactions and uses matrix counting on the reduced set of transactions for finding frequent patterns. The TreeProjection algorithm outperforms the Apriori method by more than an order of magnitude. At very low levels of support, the Apriori algorithm runs out of memory and is unable to run to completion. This behavior illustrates the memory advantages of using a lexicographic Tree projection algorithm over the hash tree implementation of the Apriori algorithm. As compares the Apriori and DHP over the various passes on a standard dataset .It is observed that DHP, because of its hashing and timely pruning technique requires less than half the amount of time as compared to classical Apriori over 5 passes of candidate itemset generation. The partition algorithm proposed in 1995 reads the

database at most two times to generate all significant association rules. Contrast this with the previous algorithms, where the database is not only scanned multiple times but the number of scans cannot even be determined in advance. All the partition algorithms such as the classic partition algorithm proposed by Saverese et al [12] and DIC(Direct Itemset counting) proposed by Brin [13] work well on large datasets. In partition algorithm [12], during the first scan, algorithm generates a set of all potentially large itemsets by scanning the database once. This set is a superset of all large item sets, i.e. it may contain false positives, but no false negatives are reported. During the second scan, counters for each of these item sets are setup and their actual support is measured in one scan of the database. The partition sizes are chosen such that each partition can be accommodated in the main memory so that the partitions are read only once in each phase. One major problem with Partition is that as the number of partitions increases, the number of locally frequent itemsets, increases. While this can be reduced by randomizing the partition selection, randomized partitions will have a large number of frequent itemsets in common. Partition can thus spend a lot of time in performing these redundant intersections.DIC [13] algorithm is also adopted by the thought of dividing the database into several partitions, each partition is marked at the beginning in the process of scanning the database, candidate items can be added in each marked partition, the support can be calculated when the itemsets are calculated.DIC separates the strict restriction between counting and generating candidates. Whenever a candidate reaches min-sup, DIC starts generating additional candidates based on it. The main bottleneck of data set partitioning algorithm is that the execution time is long and frequent itemsets generated are not very accurate. But this type of algorithm has a high degree of parallelism; just scanning the database twice, greatly reduces me operation to improve the efficiency of the algorithm. Overall, we can conclude that the breadth-first algorithm has the shortcoming of generating a large number of candidate items and need to repeatedly scan the database. This is definitely not suitable for databases with large number of itemsets.

3.2 Depth First Approach -

Depth first search has the advantage that it is not necessary to re-create the projected transactions for each level-k of a search tree. The depth first projection technique provides locality of data access, which can exploit multiple levels of cache. Some of the depth first algorithms are FP-growth Eclat [17].

In 2000 Han et al proposed frequent pattern tree (FP-tree) structure, which is an extended prefix tree structure, used for mining the complete set of frequent patterns by pattern fragment growth. It encodes the dataset using a compact data structure FP-tree and extracts frequent itemsets directly from this structure. This algorithm is about an order of magnitude faster than the Apriori algorithm. However, the number of conditional FP-trees is in the same order of magnitude as number of frequent itemsets. The algorithm is not scalable to sparse and very large databases. The OIP algorithm [15] Opportune Project mines complete set of frequent item sets, which is efficient on both sparse and dense databases at all levels of support threshold, and scalable to very large databases. It opportunistically chooses between two different data structures, array based or tree-based, to represent projected transaction subsets, and heuristically decides to build unfiltered pseudo projection or to make a filtered copy according to features of the subsets. DepthProject [16] employs a selective projection and uses the horizontal bit string representation for projected transaction subsets. the comparison of Tree Projection(Breadth first) and FP-Tree(Depth first) approach of runtime on T25:I10:D10K with 1K items as dataset. It shows that FP-growth is better than Tree Projection when support threshold is very low and database is quite large. The Eclat algorithm [17] proposed by Zaki et al is based on a parallel approach and partitions the database. It incorporates some features of clustering as well as parallel mining. The algorithm uses a scheme to cluster related frequent itemsets together, and to partition them among the processors. At the same time it also uses a different database layout which clusters related transactions together, and selectively replicates the database so that the portion of the database needed for the computation of association rules is local to each processor. After the initial set-up phase, the algorithm eliminates the need for further communication or synchronization thus eliminating the problem of synchronization faced in parallel mining. The algorithm further scans the local database partition only three times, thus minimizing I/O overheads.The gist is that depth first approach works well for large amount of data as compared with bredath first approach and so it can be applied for scalable datasets.

3.3 Pattern Growth Algorithms-

Two major costs of Apriori based algorithms are the cost to generate candidate frequent itemsets and the I/O cost. Data mining community has addressed the issues related I/O, but the issues

related to candidate frequent itemsets generation remain. First, the cost required to generate candidate k

itemsets especially when there are a lot of k –1 frequent itemsets. For example, if there are n frequent 1 itemsets, Apriori based algorithms would require to generate approximately n2/2 candidate frequent itemsets. Secondly, the memory

required to hold the candidate frequent itemsets and their supports could be substantial. For example, when n equals 10,000, there would be more than 108 length 2 candidate frequent itemsets. Assuming it requires 4 bytes to hold the support and 4 bytes to hold the itemsets, it would require close to 0.5 gigabytes of main memory to store the information. Furthermore, the memory required does not include the overhead from the data structure. Thirdly, the cost required to counting the support of candidate itemsets may not be trivial. As observed in run time behaviour of Apriori based algorithms, the run time increases as the support decreases. Therefore, the cost of candidate frequent itemsets generation of Apriori based algorithms could easily overshadow the cost of I/O. Han et al. proposed a data structure called frequent pattern tree or FPTree . FP-growth mines frequent itemsets from FP-Tree without generating candidate frequent itemsets. FP-Tree is an extension of prefix tree structure. Only frequent items have nodes in the tree. Each node contains the item's label and its frequency. The paths from the root to the leaves are arranged according to the support of the items with the frequency of each parent is greater than or equal to the sum of its children's frequency. The construction of FP-Tree requires two data scans. In the first scan, the support of each item is found. In the second scan, items within transactions are sorted in descending order according to the support of items. If two transactions share a common prefix, the shared portion is merged and the frequencies of the nodes are incremented accordingly. Nodes with the same label are connected with an item link. The item link is used to facilitate frequent pattern mining. In addition, each FP-Tree has a header that contains all frequent items and pointers to the beginning of their respective item links. FP-growth partitions the FP-Tree based on the prefixes. FP-growth traverses the paths of FP-Tree recursively to generate frequent itemsets. Pattern fragments are concatenated to ensure all frequent itemsets are generated properly. In this way, FP-growth avoids the costly operations of generating and testing of candidate itemsets. As pointed out by the authors of FP-Tree, no algorithm works in all situations. The fact holds true for FP-Tree when the dataset is sparse. When the data is sparse, the compression achieved by the FP-Tree is small and the FP Tree

is bushy. As a result, FP-growth would spend a lot of effort to concatenate fragmented patterns with no frequent itemsets being found. A new data structure called H-struct is introduced in. Transactions are sorted with an arbitrary ordering scheme. Only frequent items are projected in the H-struct. H-struct consists of projected transactions and each node in the projected transactions contains item label and a hyper link pointing to the next occurrence of the item. A header table is created for H-struct. The header contains frequencies of all items, their supports and hyper link to the first transaction containing given item. H-mine mines the H-struct recursively by

building a new header table for each item in the original header with subsequent headers omitting items that have been mined previously. For each sub-header,H-mine traverses the H-struct according to the hyper links and finds frequent itemsets for the local header. At the same time, H-mine builds links for items that have not been mined in the local header. Those links are used to find conditional

frequent patterns within the local header. The process is repeated until all frequent itemsets have been mined. In case of a dense dataset, H-struct is not as efficient as FP-Tree because FP-Tree allows compression. H-mine would dynamically switch to FP-Tree when the dataset is found to be dense.

3.4 Incremental Update with Apriori-based Algorithms-

A general incremental update data-mining algorithm is highly desirable in frequent pattern mining. It is because the complete dataset is normally huge and the incremental portion is relatively small compared to the complete dataset. In many cases, it is not feasible to perform a complete data mining process while transactions are being added continuously. Therefore, incremental data mining algorithms have to reuse the existing information as much as possible, so that either computational cost and/or I/O cost can be reduced. A general incremental mining algorithm called Fast Update 2, FUP2 that allows both addition and deletion of transactions was proposed . The major idea of FUP2 is to reduce the cost of candidate frequent itemsets generation.

Incremental portion of the dataset is scanned; frequent patterns in the incremental data are compared with the existing frequent itemsets in the original dataset. Previous frequent itemsets are removed if they are no longer frequent after the incremental portion of the data is added or

removed. The supports of previous frequent itemsets that are still frequent are updated to reflect the changes. In those ways, previous frequent itemsets that are still frequent are not required to be checked for their supports again. New k +1 candidate frequent itemsets are generated from frequent k itemsets. The entire updated dataset is

scanned to verify those newly added candidate itemsets if they are indeedfrequent. The process is repeated until the set of candidate frequent itemset

becomes empty. FUP2 offers some benefits over the original Apriori; however, it still requires multiple scans of the dataset. Another incremental Apriori based algorithm is called Sliding Window

Filtering, SWF for short. SWF incorporates the main idea of Partition algorithm with Apriori to allow incremental mining. SWF divides the dataset into several partitions. During the scan of partitions, a filtering threshold is employed in each partition to generate candidate frequent 2 itemsets. When a candidate 2 itemset is found to be frequent in the newly scanned partition, the partition number and the frequency of the itemset are stored. Cumulative information about candidate frequent 2 itemsets is selectively carried over toward subsequence partition scans. Cumulative frequencies of previous generated candidate frequent 2 itemsets are maintained as new partitions are being scanned. False candidate frequent itemsets are pruned when the cumulative

support of the candidate frequent itemsets fall below required proportional support since they have become frequent. Once incremental portion of the dataset is scanned, scan reduction techniques are used to generate all subsequence candidate frequent itemsets . Another data scan over the whole dataset is required to confirm the frequent itemsets. In the case of data removal, the partition to be removed are scanned, the cumulative count and the start partition number of candidate length 2 itemsets are modified accordingly. Although SWF achieves better performance than pervious algorithms, the performance of SWF is still depending on the selection of partition size and

removal of data can only be done at partition level.

3.5 Hybrid Approach -

There is also a hybrid approach wherein at some point breadth first approach is used and at some point depth first approach is applied in the same process. Some of the algorithms which work on

hybrid approach are AprioriHybrid, Pincer Search below shows an example of the two approaches .

Algorithm	Traversals Technique	datastructure	Number of database scan
Apriory	BFT	HASH TREE	K
Dhp	BFT	HASH TREE	K
Partition	BFT	None	2
DIC	BFT	Prefix tree	<=K
FP tree	DFT	Prefix tree	2
Eclat	DFT	none	<=3
Pincer search	Hybrid	Link list	<=4

Table 2: Comparison of some major algorithms shows the current scenario of research done on frequent itemset mining [22] in context to year of publication and its number of citations. As the data continue to increase in complexity (which includes size, type, and location of data), parallel computing will be essential in delivering fast interactive solutions for the overall KDD process. Applying parallel mining in collaboration with an effective data structure for storing candidate sets can be exploited for a scalable approach in mining association rules.

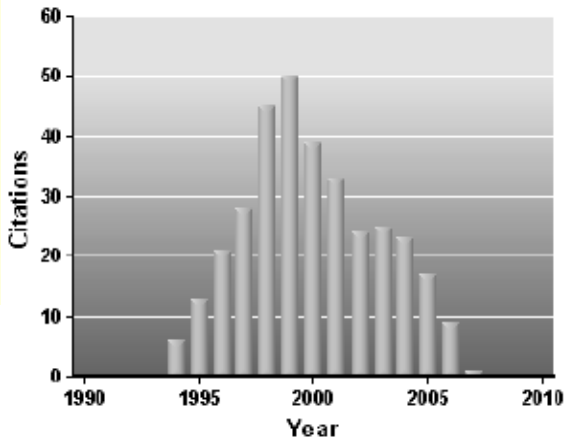

Fig 1: Current status of research

4. Conclusions

Throughout the last decade, a lot of people have implemented and compared several algorithms that try to solve the frequent itemset mining problem as efficiently as possible. Unfortunately, only a very small selection of researchers put the source codes of their algorithms publicly available such that fair empirical evaluations and comparisons of their algorithms become very difficult. Moreover, we experienced that different implementations of the same algorithms could still result in significantly different performance results. As a consequence.

REFERENCES

[1] R. Agrawal, H.Mannila, R. Srikant, H. Toivonen, and A. I.Verkamo, 1996,Fast discovery of association rules, In Advances in Knowledge Discovery and Data Mining, MIT Press.

[2] R. Agrawal, T. Imielinski, and A.Swami,1993,Mining association rules between sets of items in large databases, In ACM SIGMOD Intl. Conf. Management of Data.

[3] J. Han, J. Pei, and Y. Yin,2000,Mining Frequent Patterns without Candidate Generation, Proc. of the ACM SIGMOD, Dallas, TX.

[4] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, 2001,HMine:Hyper-Structure Mining of Frequent Patterns in Large Databases, Proc. of IEEE ICDM, San Jose, California.

[5] R. Agrawal and R. Srikant,1994,Fast Algorithms for Mining Association Rules, Proc. of the 20th Int. Conf. on VLDB, Santiago, Chile.

[6] Y. G. Sucahyo and R. P. Gopalan,2003,CT-ITL: Efficient Frequent Item Set Mining Using a Compressed Prefix Tree with Pattern Growth, Proc. of 14th Australasian Database Conference, Adelaide, Australia.

[7] M. J. Zaki,2000,Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering, (12, May/June 2000) 372-390.

[8] Agrawal R, Imielinski T, Swami A, 1993,Mining Association Rules between Sets of Items in Large Databases, Proc. of the 1993 ACMSIGMOD Conference. Washington D C, 207-216.

[9] Agrawal R, Srikant R,1994,Fast Algorithm for Mining Association Rules, Proc of the 20th Very Large Data Bases (VLDB 94) Conference. Santiago, Chile.

[10] Park J S, Chen M S, Yu P S, 1995, An effective hash based algorithm for mining association rules, In Proc. Of 1995 ACM SIGMOD. San Jose, 175-186.

[11] Agarwal R, Aggarwal C, Prased V V V,2001,A tree projection algorithm for generation of frequent itemsets, Parallel and distributed Computing, 61(3): 350-371.

[12] Savasere A, Omieeinski E, Navathe S,1995,An efficient algorithm for mining association rules in large databases, Proc. of the 21st International Conference on Very Large Databases. Zurich, Switzerland, 432-443.

[13] Brin S, Motwani R, Ullman J D,1997,Dynamic Itemset counting and implication rules for market basket data, Proc. of the 1997 ACM SIGMOD International Conference on Management of Data.

[14] Han J, Pei J, Yin Y, 2000, Mining frequent patterns without candidate generation, In Proc. of the 2000 ACM SIGM0D Conference on Management of Data. Dallas, TX.

[15] Liu J, Pan Y, Wang K, et al, 2002,Mining frequent item sets by opportunistic projection,Proc Of the Eighth ACM SIGKDD Intl. Conf on Knowledge Discovery and Data Ming. Alberta, Canada, 229-238.